SESSION

CLOUD COMPUTING, CLOUD SERVICES AND SECURITY + MOBILE COMPUTING

Chair(s)

TBA

Providing Lifetime Service-Level-Agreements for Cloud Spot Instances

Alexander Pucher, Rich Wolski, Chandra Krintz
Department of Computer Science
University of California, Santa Barbara
{pucher, rich, ckrintz}@cs.ucsb.edu

***Abstract*—Spot instances are commonly offered by IaaS cloud providers to opportunistically utilize spare capacity and meet temporary user demand for additional resources. Although the availability of service SLAs is a core paradigm of cloud computing, spot instances in practice still come without any service quality guarantees. We aim to extend the spot instance service to provide a probabilistic SLA for eviction probability, based on the user estimate of the maximum expected instance lifetime. This probabilistic foundation simplifies reasoning about the spot instance service and enables providers to construct higher-level SLAs from them. For this to be possible, however, the statistical guarantees must be adhered to strictly, for a wide range of real-world workloads, at cloud scale. We propose a new approach to providing SLAs on the time-until-eviction for spot instances by employing Monte-Carlo simulation to compute the distribution of future spot instance lifetimes at current cloud utilization levels. We then show that an IaaS cloud scheduler can use the quantiles of such conditioned distributions to safely provision spot instance requests and maintain an SLA with a specific target eviction rate.**

I. Introduction

Cloud computing, in the form of Infrastructure as a Service (IaaS), has emerged as a new paradigm for Information Technology (IT) management of data center infrastructure. Under the IaaS cloud model, users request that data center resources be *provisioned* for their exclusive use via network-facing web service interfaces (APIs). "The Cloud" services these requests in a way analogous to the way in which e-commerce services operate: automatically and transactionally. Users interact only with the automated cloud services and requests are either fulfilled or denied immediately (possibly due to error) so that the user may retry if he or she desires to do so. The user-requested services are then typically delivered until cancellation subject to a *Service Level Agreement* (SLA).

In this paper, we examine the feasibility of using two classes of resources requests – *on-demand* and *spot* – to achieve greater resource utilization while providing statistical service level guarantees for *both* classes. We borrow this terminology from Amazon's AWS [1] where *spot instances* are pre-emptable requests that may be terminated without warning if the "spot market" conditions warrant (i.e. the current market price exceeds the user's bid), and *on-demand instances* are never pre-empted, but incur a higher per-minute occupancy cost than spot instances.

Our goal is to understand whether it is possible to use the spot instance service in a cloud to accept workload subject to a statistical guarantee on minimum time until pre-emption. In particular we

- demonstrate that it is possible to provide statistical guarantees on minimum spot instance lifetimes using production private cloud workload traces, and
- detail the effectiveness of co-scheduling on-demand workloads with spot workloads with these guarantees to utilize otherwise unused resource capacity.

This work complements previous investigations of public cloud spot instance services from the user perspective. Prior work develops bidding schemes based on spot instance price history of public clouds and looks at revenue optimization for hypothetical web services built on top of spot instances [2], [3], [4]. While these works demonstrate the economic viability of using spot instances in public clouds, they rely on use-case specific utility functions and assumptions about the spot instance workload, such as the ability to checkpoint.

In contrast, our work is motivated by production enterprise, research, and high-performance computing environments where a private cloud offers scalable, automated, SLA-governed service to its users. In these settings, resource utilization must be maximized, although resources are often over-provisioned to ensure an acceptable user experience in terms of response time and available capacity. Furthermore, unlike public cloud, enterprise private clouds do not typically charge a fee for usage. Rather, each user is given a a usage quota and must maximize resource utilization subject to quota limits.

In this setting, spot-instances can allow enterprise users to exceed their respective quotas by taking advantage of otherwise unused capacity. Because a spot-instance will be terminated if the capacity is needed to run an on-demand instance, users can run spot instances without a charge to their respective quotas. That is, the capacity for spot-instances is "scavenged" and then reclaimed when it is needed for on-demand instances.

From a cloud perspective, the uncertain duration of time that a spot-instance will run before it is terminated makes their use difficult. Our work uses on-line simulation to *predict* spot-instance lifetimes based on the recent history of cloud activity. In particular, we use a Monte-Carlo style [5] simulation (run every few minutes) to estimate the distribution of the time-until-eviction of spot instances from requests in the recent past. We use this non-parametric approach to compute the quantiles (i.e. percentiles) of the empirical distributions of spot instance lifetimes that are conditioned on the capacity currently available in the cloud. These quantiles then serve as a probabilistic lower bound on the future time-until-eviction which the user can interpret as a statistical "guarantee" of spot instance lifetime. For example, the lower 0.95 quantile indicates the minimum spot-instance lifetime that as user can expect with probability 0.95.

We evaluate our method based on trace-driven simulation with synthetic and recorded commercial traces from Eucalyptus [6], [7] IaaS clusters deployed in production systems [8]. We use the synthetic workload traces to demonstrate the theoretical efficacy of our approach and give an in-depth look at the steps required to produce accurate time-until-eviction estimates via Monte-Carlo simulation. We then apply this method in a cloud scheduler that is capable of maintaining an arbitrary SLA for maximum eviction probability of spot instances in an IaaS cloud even when faced with the adverse conditions of commercial production environments.

II. Methodology

The goals of our methodology are to define a method

- for predicting minimum lifetime of spot instances with configurable confidence bounds using historical observations of previous instance behavior, and
- for using these predictions in scheduler-level admission control to ensure that all accepted spot requests meet their target lifetime.

This latter requirement is consistent with current cloud abstractions in that requests are either accepted by the cloud (and thus subject to the advertised SLA) or rejected because the SLA cannot be met.

To predict minimum lifetime, we have developed a prediction utility that uses historical data from IaaS system logs of instance types (core counts) and instance start/stop events to construct empirical distributions of instance lifetimes conditioned on available cloud capacity via periodic Monte-Carlo simulation. From these lifetime distributions, the utility extracts the quantile associated with the SLA offered by the IaaS cloud for spot instances (e.g. a 95% or 99% confidence bound on the likelihood that the instance will not be evicted) to predict minimum lifetime for each level of available capacity.

For admission control, we assume that spot requests are accompanied by a *user-specified* lifetime (maximum) when submitted. Our IaaS scheduler uses (i) the quantile estimates for the SLA generated by the prediction utility, (ii) the instance size (also specified per request) and maximum lifetime from the user, and (iii) the currently available capacity of the system, to decide whether to admit a spot request. The scheduler evicts spot instances if/when an on-demand instance request is made and the cloud has insufficient capacity to service the request.

A. Scheduling Model

Instance requests (to either start or stop an instance) are routed to a scheduler (as implemented by IaaS infrastructures such as Eucalyptus [6], Open Stack [9], and Cloud Stack [10]) which handles admission control and placement of instances on physical resources in a cluster of "nodes." IaaS clouds typically define "instance types" that describe the resources that an instance will consume (CPU cores, memory, ephemeral disk storage, etc.). In the Eucalyptus systems (production and research) that we investigate in this work, we observe that the memory footprint associated with each instance type is such that the instance placement decision by the scheduler can be made strictly on core count.

When an instance is admitted, the scheduler makes a placement decision by selecting a node on which the instance will run. In this study, we use simple first-fit placement in favor of more complex approaches to highlight the impact SLA-aware admission control. If an on-demand instance is requested, and the scheduler cannot find a node with available capacity, the scheduler selects one or more spot instances to terminate (evict) so that the on-demand instance can be scheduled.

Further, our scheduler (like other Eucalyptus schedulers) assumes that the instance type definitions nest with respect to their core counts. For example, an empty 4-core node node is seen by the scheduler as having 1x 4-core slot, 2x 2-core slots, 4x 1-core slots, or 1x 2-core, 2x 1-core slots. The distinction between available cores and *available slots* is important when generating time-until-eviction estimates for different instance sizes and different cluster load levels.

B. Prediction

Past work has shown that cloud workloads can be highly variable and may not be easily described by single well-known distributions [11]. To address this problem we run a Monte-Carlo-style simulation on-line to generate the empirical distribution of expected spot instance lifetimes. However, we note that the time-until-eviction is affected by the capacity of the cloud that is occupied by un-evictable on-demand workload and other spot instances. Intuitively, if the cloud is relatively "empty", a spot-instance that is introduced will likely live longer than if the cloud is close to "full" capacity. Thus, our Monte-Carlo simulation produces a *set* of empirical distributions, one conditioned on each level of possible occupancy.

For example, a cloud with 100 cores has 101 possible occupancy levels: from 0 cores occupied to 100 cores occupied and each level of occupancy corresponds to a different distribution of spot-instance lifetimes. We use quantiles of these distributions to quote the expected lifetime to the scheduler during the admission control phase based on the current occupancy level at the time the spot-instance request is made. If the instance (based on its maximum lifetime specified by its user) is expected to be evicted with a higher probability than specified by the target probability (quoted as an SLA) for the cloud, it is rejected (not admitted). The cloud administrator is responsible for setting the SLA on eviction probability that is advertised to all cloud users.

The Monte-Carlo simulator generates a sample of "fictitious" spot-instance requests that using the recent cloud load history. It repeatedly chooses a random point in the history and simulates the arrival and eviction of a spot-instance, recording the occupancy level at he time the spot-instance starts and its time-until-eviction. Running faster than real time, it generates 10000 such samples and divides them into empirical distributions based on occupancy level.

C. Eviction Policy

The eviction policy affects the conditional spot-instance lifetime distributions generated by the simulation (but not the correctness of the method). Many policies are possible but each has an impact on user experience. In this paper we chose a simple "Youngest-Job-First" (YJF) eviction policy. Choosing the "youngest" (i.e. the spot instance that has started most recently) to evict among the candidate spot instances is an attempt to minimize the "regret" associated with an eviction in this online

TABLE I: Parameters of synthetic log-normal on-demand and spot instance workloads

	VM arrival	VM duration	VM cores	mean util.
on-demand	$\mu = 4, \sigma = 1$	$\mu = 6, \sigma = 1.5$	1	21.77
spot	$\mu = 4, \sigma = 1$	$\mu = 6, \sigma = 1.5$	1	21.95

decision making problem [12]. That is, the amount of work that is lost because of an eviction is minimized.

D. Evaluation Metrics

To evaluate the system we use trace-based simulation with both synthetic and production traces taken from private Eucalyptus IaaS clouds. We replay each trace in its entirety and we log each individual state change in the simulated system. In each case, the simulator uses separate traces for spot-instance and on-demand instance requests. We then generate summary statistics and evaluate our solution using two metrics:

- eviction ratio of spot instances
  $evicted = evictions/admissions$
- admission ratio of spot instances
  $admitted = admissions/requests$

The enforcement of the target SLA probability has highest priority. After the SLA is fulfilled, a high number of completed spot instance requests is desirable to maximize utilization.

III. Results

Our experiments are run in simulation, based on our previous work on validated simulation of private IaaS clouds. We use both, synthetic traces and anonymized production traces obtained from Eucalyptus IaaS cloud installations. For reproducibility we assume instant start and stop of instances in the traces and rely on a publicly available set of anonymized commercial production traces [8].

A. Prediction with synthetic traces

To outline our approach and show its basic behavior we compare a scenario with an SLA-unaware scheduler and the SLA-aware scheduler using multiple different SLA levels using 10-day synthetic traces (Parameters in Table I). Our initial setup uses a single platform (IaaS cluster configuration) and synthetic on-demand and spot request traces. The platform contains 8 nodes with 4 cores each, for a total of 32 cores. As a rough estimate based on mean utilization the platform should be able to support the on-demand trace plus half the spot instance trace. We use a log-normal distribution to approximate the long-tailed empirical distribution of instance life times.

Note that there is a trade-off between the probabilistic guarantee given to the user and the fraction of spot-instance requests that can be accepted by the scheduler. Greater "certainty" associated with a spot-instance SLA (in the form of a lower eviction probability) implies that fewer spot-instance requests can be accepted (to decrease the possibility that an eviction will be necessary).

To illustrate this trade-off, show the ratio of admitted spot instances, as well as the eviction ratio of spot instances in Figure 1. The x-axis shows different SLA probabilities, starting with the no-guarantees baseline on the left and then increasingly stringent SLAs of 0.25, 0.10, 0.05 and 0.01. The y-axis shows the fraction

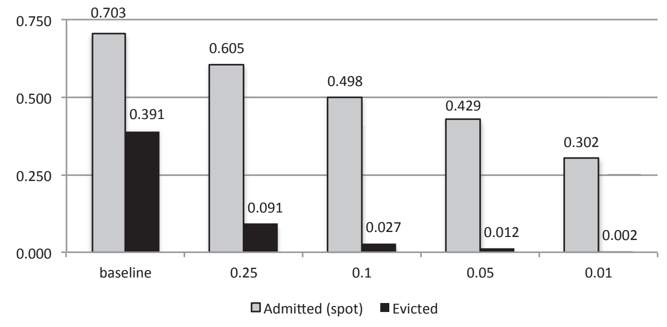


Fig. 1: Ratio of admitted and evicted spot instances with synthetic log-normal traces.

of admitted spot-instance requests instances in gray and the fraction of evicted spot instances in black. The SLA-aware scheduler meets the SLA in all cases (the eviction fraction is less than the advertised guarantee level), at the cost of preemptively rejecting an higher fraction of spot instances for stricter SLAs. The measured SLA probabilities are in fact stricter than the target SLA probabilities, showing that the predictions of time-until-eviction made by the simulation are conservative.

The most visible improvement is the step from the no-guarantees baseline to the 0.25 eviction ratio SLA. While the baseline admits 70% of all requested spot instances, 39% of the admitted spot instances are evicted before completion. The 0.25 SLA in contrast admits 60% of all requested spot instances, but only 9% of the admitted spot instances are evicted. Subsequent decreases in the demanded maximum eviction rate of spot instances decrease the number of admitted spot instances as well, but consistently (conservatively) achieve the SLA eviction probabilities.

This experiment outlines the setup of our simulation driven approach to enforce guaranteed levels of eviction probabilities in a controlled environment. In the next section we discuss the simulation method in-depth.

B. Conditional Distributions and Sample Size

The scheduler computes conditional distributions for all possible core counts on a fixed duty schedule (every 6 hours of trace time in the previous experiments) based on the history of on-demand and spot instance behavior it has observed so far. This gives rise to the property that the sample sizes for "rarely" occurring conditions may be small. For example, if the cloud is moderately loaded, the number of examples where all but one of the cores is busy might occur infrequently or not at all.

To provide an in-depth insight in the behavior of the Monte-Carlo simulation, we provide an exemplary intermediate results at the 9 day mark of our synthetic trace experiment for single-core instance slots. Figure 2 shows the number of samples generated on the y-axis for each condition on the x-axis (available slot count). In our specific example, 2 to 4 open slots are encountered the most frequently, with about 10000 samples each. High open slot counts, which correspond to low cluster utilization, are increasingly uncommon. Based on the number of samples we expect predictions for common cases to be highly accurate, while infrequently occurring cases will be based on empirical distributions estimated from small samples.

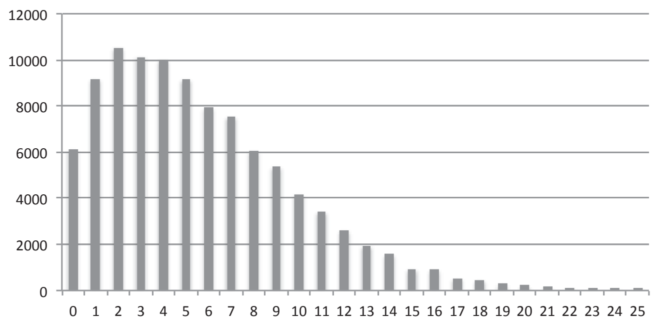


Fig. 2: Number of time-until-eviction samples per available-slots bucket for a synthetic log-normal simulation run. Frequently encountered load levels (left) have many samples, corner cases (right) have few. The shape of the histogram depends on the historic workload.

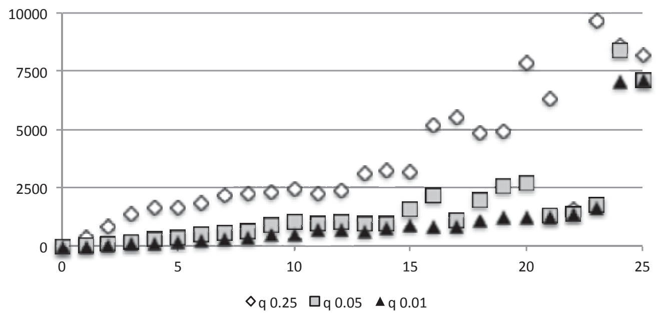


Fig. 3: Quantiles of time-until-eviction per available-slots bucket for a synthetic log-normal simulation run. Predictions for common load levels (left) can be made with high confidence, predictions for infrequent ones (center) are rough estimates that become increasingly erratic for corner cases (right).

Figure 3 shows the quantiles of the conditional distribution of times-until-eviction. The x-axis again shows the condition, while the y-axis indicates the time-until-eviction as estimated by a quantile. The estimates to the left correspond to the buckets with high sample count in Figure 2, whereas the estimates to the right decrease in sample count. The 0.25 quantile lies above the 0.10 quantile, followed by the 0.01 quantile. These quantiles estimate the minimum time a spot instance is expected to survive with the corresponding probability. For example, from the figure, 0.01 of the instances that are started when there are 15 free slots run for 900 seconds or less before being evicted. In the same column (for 15 free slots), 0.05 of the time-until-eviction samples are 1500 seconds or less, and 0.25 of them are 3200 seconds or less.

A combined look on the counts per bucket in Figure 2 and the corresponding quantiles in Figure 3 also provides an insight into the reliability of conditional estimates. Buckets 0 to 14 each have over 1000 samples each to determine quantiles from. This is generally enough for low SLA probabilities, such as 0.05 or 0.01. With increasing slot count (decreasing cluster utilization) a smoothly changing, and mostly increasing estimate of the time-until-eviction can be observed. Buckets 15 to 20 still have over 200 samples each, which is enough for rough estimates, but a look back at the quantiles shows that changes from bucket to bucket already become erratic. Estimates for 21 available slots and over appear extremely infrequently in our synthetic trace. Their samples are mostly artifacts from the initial warm-up period and as such, their estimated quantiles are not reliable (but also hardly used).

TABLE II: Mapping of recorded commercial production traces and their original hardware platforms from the data set collection [8] to experiments in this paper.

Name	Source	Organization	Workload	Nodes
A	DS2	Medium	bursts	7 x 8 cores
B	DS3	Medium	bursts	7 x 12 cores
C	DS5	Large	variable	31 x 32 cores
D	DS6	Large	constant	31 x 32 cores

Since we are using a synthetic trace based on log-normal distributions for arrival time and instance lifetime, this specific example could be described analytically as well. However, for arbitrary traces, as found in production environments, this is challenging to impossible depending on the typical usage of the cluster. Monte-Carlo simulation offers a way to estimate arbitrary empirical distributions and can be tailored to achieve the desired degree of prediction accuracy.

C. Prediction with production traces

To study the utility of Monte-Carlo-based SLA enforcement in a more realistic setting, we use four different traces obtained from independent Eucalyptus IaaS production installations for our experiments. The origin of these traces is documented in [13], [11], [14]. and the traces themselves are available as part of a collection from [8]. Table II shows the mapping of data sets from the collection to experiments in this paper, together with a short description of their workload and platform properties.

Compared to synthetic traces there are a number of important differences. First, instance starts show temporal auto-correlation. These "bursts" of instance starts are more extreme than ones observed in synthetic log-normal traces. Second, the behavior of users changes over time and causes change points which the empirical distribution derived via Monte-Carlo simulation only picks up over longer time frames. Third, instance sizes are no longer uniform and traces contain instances with slot sizes between 1 and 30 cores.

To facilitate the experiments with real world traces, two modifications are made to the Monte-Carlo simulation. First, we expect that our randomization approach may not generate starting points needed for all conditional core-utilizations needed, especially in the beginning of the experiment where data samples are scarce. To avoid rejecting spot instances unnecessarily due to a perceived lack of information, we linearly approximate quantiles of unobserved conditional distributions between observed "neighboring" distributions.

For example, if the empirical distributions conditioned over 20 slots and 18 slots are available, while there are no samples for 19 slots, the quantiles for 19 available slots are generated by linear approximation between the the matching quantiles of the neighbors. For example, the 0.01 quantile for 19 slots would then be calculated as $q(0.01|19) = (q(0.01|18)+q(0.01|20))/2$. In the case where multiple conditions are missing, we fit a line to the two endpoints in the range of missing values and use it to approximate the quantiles between. Additionally, the extreme

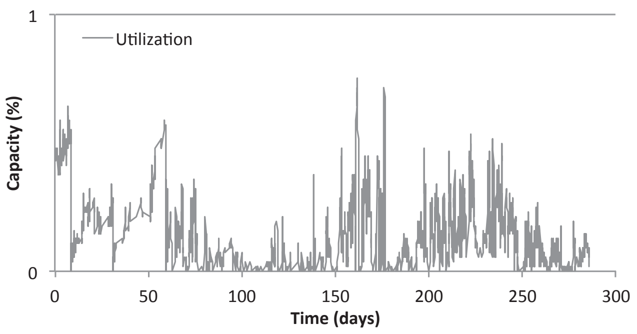


Fig. 4: Production trace B as executed on its native platform shows highly variable load and bursts of large requests as well.

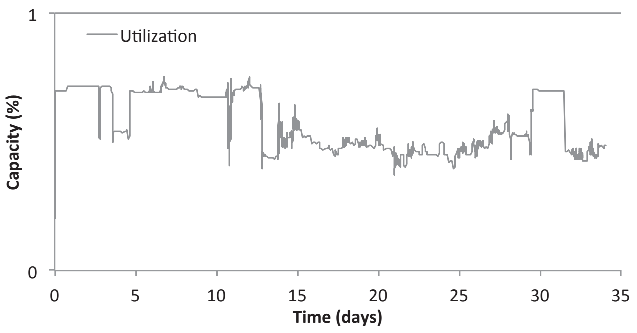


Fig. 5: Production trace C as executed on its native platform shows a mixed pattern of load with constant plateaus and periods with higher variability.

points of zero and full utilization need to be populated with useful data. We chose an impossible value for expected life time before eviction if there are no slots available for a given capacity and conservatively use the quantiles for the lowest known cluster utilization as values for zero utilization as well.

Second, we start the real world traces after a delay of 24 as we do for the synthetic traces. Such a delay allows the scheduler to "warm up."

A visual inspection of the real-world traces shown in Figures 4 and 5 shows significant spikes at irregular intervals. If an on-demand spike in load appears in an environment already loaded with spot instances, we expect to see a high number of correlated evictions, possibly leading to a violation of the SLA in the short-term. If these correlated evictions are not compensated for in the long-term by conservatively maintaining a capacity buffer, these short-term violations will sum up to an SLA violation over the course of the whole trace. We try to capture this auto-correlation by replaying the actual observed trace in our Monte-Carlo simulation rather than re-sampling the input distribution. That is, we choose random locations in the trace, but then replay the trace from those periods to include auto-correlation effects.

We take the same approach to handling change points in the production time series traces. The Monte-Carlo simulation that computes the empirical conditional distributions is re-run every 6 hours of trace time to capture changes that may have occurred in the underlying dynamics. The the Monte Carlo simulation with production traces takes no more than 300 seconds (5 minutes) to generate the empirical distributions. Thus, in a production implementation, it would be possible to make these estimates every 6 hours "on-the-fly" as part of the cloud's typical operation.

The third difference of real-world traces over to our synthetic ones are non-uniform instance capacities. This has two major implications: first, Monte-Carlo simulation must consider different instance sizes and second, placement decisions for on-demand instances made at any time may have consequences later in the trace. Because the scheduler attempts to find space for an on-demand instance and only evicts when there is insufficient capacity, the presence of spot-instances can change where the scheduler places on-demand instances. As a result, because an instance cannot span nodes, it could be that the introduction of spot instances increases the "fragmentation" of the available core capacity and, hence, affects the ability to run on-demand instances. However, while spot-instances <u>might</u> cause the scheduler to reject an on-demand instance it would have otherwise accepted (due to fragmentation effects) all of the on-demand instances that are accepted receive the SLA guarantees that they would have without spot instances present. This effect (detailed in Subsection III-E) is small for the production workloads we study but grows as the cloud runs closer to capacity.

The conditional distribution of expected lifetimes therefore effectively becomes conditioned over instance capacity (taking into account fragmentation effects) in addition to available slot count. The conditioning over instance capacity does not increase the amount of data required for accurate estimates as we can re-run the same recorded trace with different virtual instance sizes. An increasingly diverse population of instance types therefore leads to a linear increase in computational effort for Monte-Carlo simulations, but not to a relative reduction of estimation accuracy. In practice, we do not expect this to be a severe problem due to the embarrassingly parallel nature of Monte-Carlo simulation.

D. SLA-aware co-scheduling of production traces

Having addressed the issues associated with generating predictions for production traces the question remains whether these modifications allow effective admission control for varying types of production workloads. In this section we investigate the efficacy of our approach for co-scheduling different production workloads while maintaining an SLA for both, on-demand and spot instances.

We perform the evaluation with production traces in two parts and pair up our production traces based on similar platform sizes. The first combination uses highly variable workloads, "A" as on-demand trace and "B" as spot instance trace. The specifications of the physical cloud platform are taken from "A", which contains 7 nodes with 12 cores each. We refer to this configuration as "A-B". We use the inverse notation "B-A" to describe co-scheduling of "A" as spot instances in addition to "B" as on-demand trace and "B"'s physical platform, which contains 7 nodes with 8 cores each. In both cases, we set the SLA to 0.01 eviction rate and we compare the results of the SLA-aware scheduler ("sla") with the SLA-unaware baseline scheduler ("base").

The second combination investigates the co-scheduling of the more constant workloads "C" and "D" with larger platforms

TABLE III: Results of co-scheduled workloads with production traces without SLA enforcement. In all cases the eviction ratio is greater than 0.01.

Baseline	A-B	B-A	C-D	D-C
admitted (on-demand)	0.977	1.000	1.000	0.997
admitted (spot)	1.000	0.850	0.943	0.963
evicted	0.013	0.024	0.016	0.013

TABLE IV: Results of co-scheduled workload with production workloads with SLA-aware scheduler, fulfilling the 0.01 eviction SLA (equivalent to a 0.99 survival ratio)

SLA-aware	A-B	B-A	C-D	D-C
admitted (on-demand)	0.977	1.000	1.000	0.999
admitted (spot)	0.884	0.757	0.491	0.278
evicted	0.009	0.000	0.002	0.006

of 31 nodes each. The experiments are defined analogously to the first part and we refer to them as “C-D” and “D-C”.

An important side-note is that A contains a number of instances requiring 12 cores each, while the platform of B only provides a maximum of 8 cores per node. This practically lowers the load impact of A as spot trace over its impact as on-demand trace on its native platform, as high-core-count instances are rejected by the scheduler due to the physical limits of the platform.

The results are summarized in Table III for the baseline, while the results for the SLA-aware scheduler are presented in Table IV. The SLA-aware scheduler meets the threshold, while the baseline scheduler misses in all cases. The modifications discussed in the previous section allow the SLA-aware scheduler to successfully handle production traces. The results are, however, close due to low overall utilization of the underlying cluster hardware. In fact, the mean utilization of on-demand and spot traces combined is 26.62 cores. This compares to a platform capacity of 84 cores for A and 56 cores for B. This degree of under-utilization is typical for clouds over-provisioned to meet peak demand. Reducing the under-utilization is a prime goal of co-scheduling. In order to demonstrate the efficacy of our approach in more resource constrained scenarios, we perform a platform down-scaling experiment in simulation in the next section.

E. SLA-aware co-scheduling with platform scaling

In this section we stress-test our approach to computing the conditional quantiles for spot-instance lifetimes in increasingly resource constrained environments. We use setups “A-B” and “C-D” again, but vary the size of the underlying platform from N to $N-3$ nodes for “A-B” and from N to $N-15$ in steps of 5 nodes for “C-D”. This corresponds to a reduction in node count by about half while the request rate stays constant and target SLA on eviction remains at 0.01. The inverse experiments “B-A” and “D-C” show similar results and are skipped for brevity.

Figures 6 and 7 show the results for scaled-down platforms of A-B and C-D, respectively. This experiment demonstrates the robustness of the approach in fulfilling its target SLA. While the baseline scheduler does not meet the SLA in any single case, the SLA-aware approach works consistently with visible differences in behavior.

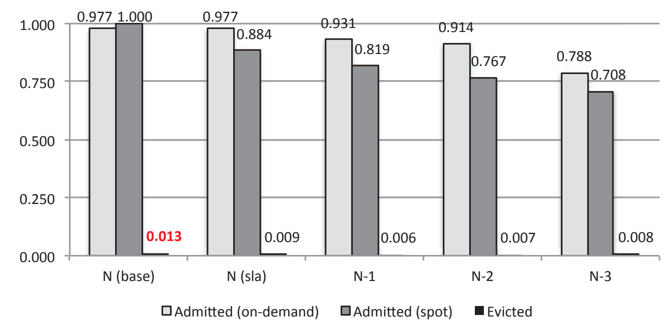


Fig. 6: Admission and eviction ratios of on-demand and spot instances for A-B down-scaled. Non-SLA base, marked ‘N (base)’ for $N = 7$ nodes in the first column compared with 0.01 SLA with full and reduced node counts $N = [7, 6, 5, 4]$ in the other columns.

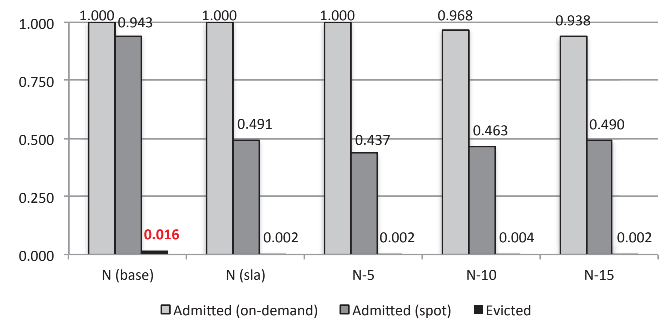


Fig. 7: Admission and eviction ratios of on-demand and spot instances for C-D down-scaled. Non-SLA base, marked ‘N (base)’ for $N = 31$ nodes in the first column compared with 0.01 SLA with full and reduced node counts $N = [31, 26, 21, 16]$ in the other columns.

Thus, while on-demand instance requests cannot completely be fulfilled in increasingly constrained environments, the on-demand rejection fractions for the production traces are small. In each figure, the column labeled N represents a replay of the production workload using the number of nodes and cores that were present when the trace was gathered (i.e. the production scenario). In the cases where our methodology offers an SLA on spot-instance lifetime in the same environment, the fraction of admitted on-demand instances is equal to the baseline.

As a result, we conclude that the success of the predictions for the real-world production traces is not due to a lack of utilization (i.e. an abundance of extra capacity) in over provisioned production clouds. Shrinking these clouds does cause some of the observed production workload to be rejected, but the generated predictions of the time-until-eviction remain valid.

An additional observation is that for the down-scaling experiments the ratio of admitted spot instances may increase as the cluster size decreases (e.g. “C-D $N-3$). An in-depth look at the simulation reveals that the rejected on-demand instances come in batches and with high core counts per instance. Their rejection due to capacity constraints leaves some additional capacity for spot instances. Furthermore, the inopportune placement of a spot instance does indeed lead to “fragmentation” and at times blocks the placement of large on-demand instances later on. While in our synthetic workloads the on-demand trace was completely unaffected by the spot trace, real-world traces are measurably impacted by the presence of spot instances.

IV. Related Work

Spot instances were first employed in 2009 as part of Amazon Web Services (AWS) [15]. Spot instances in public clouds are typically available at a rate significantly lower than that of on-demand instances as they allow providers to opportunistically utilize spare capacity. They do not, however, provide a guarantee (SLA) on their lifetime: spot instances can be terminated (evicted) at any time, whereas on-demand instances provide a 99.95% SLA on their availability once started.

Due to their unreliability but low cost they are typically used as opportunistic accelerators [16]. Previous research has also studied pricing models and user experience (Quality of Service) for services built entirely on spot instances. The authors in [17] model pricing as a mixture of multiple Gaussian distributions and reveal the challenges with modeling analytically, empirically observed phenomena in the cloud. Andrzejak et al. [2] model the trade-offs between spot instance bids and realized execution time to achieve probabilistic deadline guarantees for long-running, check-pointable jobs. In [3] the authors investigate a hypothetical service provider running a QoS-sensitive web service purely on spot instances, with a focus on revenue maximization. Similarly, [4] investigates a service running purely over spot instances and finds that existing SLAs capture only part of the observed variation typical in cloud environments.

In our work, we also service complex commercial workloads with spot instances, but side-step manual analytical modeling via Monte-Carlo simulation to provide a powerful new type of SLA on spot instance eviction probability for arbitrary jobs with bounded lifetime. While revenue and user experience depend on the specifics of the end-application, guarantees on eviction probability simplify reasoning about the system as a whole and allow providers to use it as foundation for custom SLA models.

V. Conclusions

Core economic drivers of cloud computing are the simplification of infrastructure management for clients, and increased utilization of hardware for providers by consolidation of different workloads. For private enterprise clouds, workload consolidation has its limitations due to smaller and more specialized user bases. A spot instance service model is promising, but comes with the challenge of reasoning about the implications of using evictable instances.

In this paper, we present a novel approach to providing spot instances with probabilistic SLAs on their minimum lifetime, which offers a generic solution to estimating costs and availability guarantees. We base the SLAs on estimating the time-until-eviction of spot instances from historical workload traces via Monte-Carlo simulation, which effectively enables cloud operators to offer an SLA on spot instance eviction probability. Users can request spot instances for a fixed lifetime with quantitative bounds on eviction probability and receive an immediate response from the cloud provide of whether it can provide the desired quality of service, or not. We demonstrate the robustness of our approach to providing guarantees with commercial workload traces and evaluate its efficiency for increasing utilization via workload co-scheduling.

Acknowledgments

This work was funded by NSF (CNS-0905237, CNS-1218808, and ACI-0751315) and NIH (1R01EB014877-01).

References

[1] “Amazon Web Services home page,” http://aws.amazon.com/.

[2] A. Andrzejak, D. Kondo, and S. Yi, “Decision model for cloud computing under sla constraints,” in Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on, Aug 2010, pp. 257–266.

[3] M. Mazzucco and M. Dumas, “Achieving performance and availability guarantees with spot instances,” in High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on, Sept 2011, pp. 296–303.

[4] J. Chen, C. Wang, B. B. Zhou, L. Sun, Y. C. Lee, and A. Y. Zomaya, “Tradeoffs between profit and customer satisfaction for service provisioning in the cloud,” in Proceedings of the 20th International Symposium on High Performance Distributed Computing, ser. HPDC ’11. New York, NY, USA: ACM, 2011, pp. 229–238. [Online]. Available: http://doi.acm.org/10.1145/1996130.1996161

[5] “Monte Carlo Method,” http://en.wikipedia.org/wiki/Monte_Carlo_method.

[6] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, “The eucalyptus open-source cloud-computing system,” in Cluster Computing and the Grid, 2009. CCGRID’09. 9th IEEE/ACM International Symposium on. IEEE, 2009, pp. 124–131.

[7] Eucalyptus Systems Inc., “http://www.eucalyptus.com,” Jun. 2013. [Online]. Available: http://www.eucalyptus.com

[8] R. Wolski and J. Brevik, “http://www.cs.ucsb.edu/~rich/workload,” Jun. 2013. [Online]. Available: http://www.cs.ucsb.edu/~rich/workload

[9] “OpenStack,” [Online; accessed Aug-2014] ”http://www.openstack.org/”.

[10] “CloudStack,” [Online; accessed Aug-2014] ”http://cloudstack.apache.org/”.

[11] R. Wolski and J. Brevik, “Using parametric models to represent private cloud workloads,” University of California, Santa Barbara, Tech. Rep. UCSB-CS-2013-05, August 2013, http://128.111.41.26/research/tech_reports/reports/2013-05.pdf.

[12] Y. Mansour, “Regret minimization and job scheduling,” in SOFSEM 2010: Theory and Practice of Computer Science, ser. Lecture Notes in Computer Science, J. van Leeuwen, A. Muscholl, D. Peleg, J. Pokorn, and B. Rumpe, Eds. Springer Berlin Heidelberg, 2010, vol. 5901, pp. 71–76. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-11266-9_6

[13] R. Wolski and J. Brevik, “Using Parametric Models to Represent Private Cloud Workloads,” IEEE Transcations on Services Computing, vol. 4, no. 7, pp. 714–725, October 2014.

[14] A. Pucher, E. Gul, C. Krintz, and R. Wolski, “Using Trustworthy Simulation to Engineer Cloud Schedulers,” in Cloud Engineering (IC2E), 2015 IEEE International Conference on, March 2015.

[15] “Announcing Amazon EC2 Spot Instances,” [Online; accessed Aug-2014] ”http://aws.amazon.com/about-aws/whats-new/2009/12/14/announcing-amazon-ec2-spot-instances/”.

[16] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, “See spot run: Using spot instances for mapreduce workflows,” in Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 7–7. [Online]. Available: http://dl.acm.org/citation.cfm?id=1863103.1863110

[17] B. Javadi, R. Thulasiram, and R. Buyya, “Statistical modeling of spot instance prices in public cloud environments,” in Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, Dec 2011, pp. 219–228.

Scheduling of Composite Services in Multi-Cloud Environment

Faisal Ahmad
Tata Consultancy Services Ltd
Newark, NJ, USA
faisal.nitdgp@gmail.com

Anirban Sarkar
Department of Computer Applications
National Institute of Technology, Durgapur
Durgapur, WB, India
Sarkar.anirban@gmail.com

***Abstract*—In cloud computing, resource allocation and scheduling of composite web services is very challenging and have many constraints. In a multi cloud environment, nodes can host more than one web service instances. Each web service instance can accept more than one request concurrently. Each node have different load (number of requests being served) at different time. Further, in composite web service, outputs of one service are input to another service which are required to be transferred through the cloud networks, which adds additional substantial data transfer time. The data transfer time among clouds and nodes is very important to depict the practical scenario. This paper presents an efficient scheduling mechanism which can efficiently schedule abstract services of the composite web services considering all above constrains and dynamism of the cloud. In this approach composite services are schedule at cloud, node and then at instance level. In this work, a multi-level chromosome based genetic algorithm is presented to schedule composite web services and to study its effectiveness. The paper also studies the importance of the cloud locality (inter cloud distance) and node locality (inter node distance) in the proposed scheduling of composite services in multi-cloud environment.**

Keywords- web-service composition;Scheduling; Algorithm; Cloud computing;Genetic algorithm.

I. INTRODUCTION

“Cloud computing” [1] is an internet-based computing model whereby a pool of computation resources, usually deployed as web services, are provided in a manner similar to those of public utilities. This model provides an economic way for an enterprise to build new composite web services. With the advancement of cloud and service oriented architecture (SOA) technology, large numbers of web services are available in the cloud with diverse objectives. These web services are glued together (orchestration or Choreography) to form composite services. Composite services are designed and represented with the help of abstract web services, rather than concrete services. These abstract services need to be mapped or scheduled to its appropriate concrete service instances running on a node of the cloud. For a given functional requirement, there are very large numbers of candidate web services available in the cloud. There could be many instances of a candidate service, hosted on different nodes spread across more than one cloud. Each of those hosted web service instances can have different processing speed, further at any given time, each running service instance can have different load (i.e. number of requests being served). The complete scenario is presented in the figure 1. In such a dynamic cloud environment there is a need for an efficient scheduling approach, which considers dynamism of the cloud while scheduling composite web services. It is also equally important in a multi cloud environment to study the effect of cloud locality under different load scenarios.

In order to execute efficiently any composite service in a multi cloud environment, each of the participating abstract service needs to be allocated and scheduled to an appropriate service instance running on clouds. This allocation ensures that all participating abstract services of the composite services are assigned to appropriate service instance and the scheduling ensures that the composite services are executed in the minimum time. Some of the important challenges in the scheduling of composite services in multi cloud environment are as follows.

1 The scheduling in the cloud is challenging as it is NP hard problem because of the very large number of nodes present in participating clouds. Further, for each abstract service of the composite service has many candidate services and each candidate services can have many instance services hosted on different nodes spread across different participating clouds.
2 The scheduling of the composite services in a multi cloud environment is multi-level in nature. First, appropriate cloud needs to be selected based on the criteria of cloud locality (i.e. data transfer cost from the previous cloud to current node needs to be considered). Second, appropriate nodes need to be selected based on the criteria of node locality (i.e. data transfer cost from the previous node to current node within a cloud needs to be considered) and node processing speed. Finally, appropriate running service instance needs to be selected based on the processing speed and the load (i.e. number of request currently being processed by the instance) on the nodes.
3 The composite service can have many participating services which can be scheduled in parallel. The scheduler should preserve the precedence constraints of services within composite services at the same time should exploit the parallelism to its maximum.

Figure 1 represents the multi cloud environment for scheduling of the composite service. The composite service CS_1 consists of 4 abstract service (AS_1, AS_2, AS_3, AS_4). There are 2 clouds C_1 and C_2. In Cloud C_1 there are 2 nodes (n_{11}, n_{12},) and in the cloud C_2 have 3 nodes (n_{23}, n_{24}, n_{25},). Each node in the clouds host different number of service instances. For example node n_{12} has 2 service instances while n_{25} have only 2 instances. What is required, then, is to allocate the each abstract service of the composite services to the appropriate service instance. The scheduling determines the appropriate service instance with respect to execution time so that overall execution times of composite services are minimized. Data from one service are transferred to another service instance. This is represented with dashed line. In the multi cloud environment this is very important factor, which should be considered while scheduling. Further, the transfer of data from one node to another node within clouds also adds substantial delay in the execution; simply because there is a possibility that the routers may separate the nodes within a cloud.

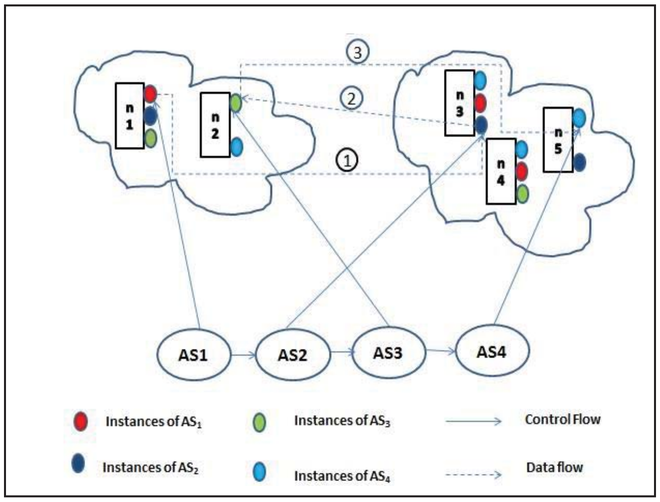


Figure 1 – Multi cloud environment

This paper present a genetic algorithm based scheduling of composite web services with its chromosome designed as a multi-level structure to accommodate the clouds, nodes and instances solution space. Scheduling composite services on multi cloud can be seen as a problem emerging from the allocation of each abstract service to the appropriate node across different clouds, which functionally matches the service's requirements while exploiting the locality of clouds, nodes and parallelism of the services to its full extent. The objective of the scheduling is to selects service instances such that the total execution times of composite services are minimized. The paper compares the results of scheduling of composite services in the individual cloud and scheduling in multi cloud. This paper also further analyzes the impact of cloud locality on the scheduling in terms of the execution time by varying load on nodes across the clouds.

II. RELATED WORK

Reference [1] maps web service calls to potential servers using linear programming. Their work is concerned with mapping single workflows Using different business metrics and a search heuristic. Reference [2] solved a new multiple composite web services resource allocation and scheduling problem in a hybrid cloud scenario where there may be limited local resources from private clouds and multiple available resources from public clouds. They have presented a random-key genetic algorithm for the resource allocation and scheduling problem. The algorithm handles both problems simultaneously. They tested empirically and the experimental results have demonstrated good scalability and effectiveness. Reference [3] presents a dynamic provisioning approach that uses both predictive and reactive techniques for multi-tiered Internet application delivery. However, the provisioning techniques do not consider the challenges faced when there are alternative query execution plans and replicated data sources. Reference [4] presents a feedback-based scheduling mechanism for multi-tiered systems with back-end databases, it assumes a tighter coupling between the various components of the system.

Typically, schedulers exist at two levels [8]. The first level is that of the local scheduler. At this level, decisions are made for each individual resource. The second level is that of the meta-scheduler. At the meta-scheduler level, several parallel applications are scheduled and undesirable interaction between these applications eliminated [13]. In order to schedule a parallel program, two pieces of information are required: information about resources and information about the application. Considerable work has been done in obtaining information about an application and making it of manageable proportions. There have been many efforts to characterize an application's structure by means of compile-time and run-time analysis of the program [9]. Information about resources has been a subject of interest and there are many existing systems that collect this information. There are tools like NWS (Network Weather Service.) that monitor and report information about network traffic. The Globus [9] project has developed the MDS protocol to store dynamic information about resources (processors and network) and GIIS to provide a query interface to locate resources of interest. [10, 11] defines "*superscheduling*" to consist of three phases: resource discovery, resource mapping and job startup. Our mechanism for scheduling fits into the resource mapping. For a given requirement, there is large number of candidate web services. Each web services have many instances running on nodes of different clouds. For a given composite web service, we attempt to map all abstract services in the incoming composite service to its appropriate service instances running on different nodes of different clouds such that the execution time of the given composite service is minimum.

Reference [5] proposed a concept of web service families, which are created to capture and manage the dynamic nature of web services in groups. Each web service family corresponds to one or many web service instances with a common business objective and same input /output structures. Reference [6] also designed an abstract execution machine, called *Web Service Dynamic Execution Machine* (*WSDE* machine). The WSDE machine is capable of executing composite web services having both deterministic and non-deterministic flows, where the composite web services are represented using a graph-based semantic based formal model named *WSDE* graph. The composite web services are represented with help of web service families as a Web-Service Dynamic Abstract Model (*WSDAM)* graph [6]. The *WSDAM* graph is executed in the *WSDE* machine, which uses the *dynamic business web service (DBWS) tree* for web service discovery. *DBWS tree,* which organized all web service families in a tree structure. The *WSDE* machine lacks the capability of efficient scheduling of web services. This paper proposes a suitable scheduling approach, which can also be used by *WSDE* machine. It should also be noted that what we propose in this paper could also be used as a general web service mechanism. A quite similar study has been done on computational grid [12], where computational nodes are scheduled for computing but this paper schedule web service instances running on computational nodes.

A lot of similar work has been done in scheduling jobs in grid and cloud but the scheduling of abstract web services of the composite web services in cloud are little different. In job or task scheduling, the nodes are selected whereby a task is executed. It is more related to computational computing. Though the web service scheduling also involved computational but prior to scheduling of abstract web services, web services are hosted on the different cloud nodes dynamically by using some task or job based computational scheduling. Once web services are up and running in different nodes across the cloud, appropriate running instances across the clouds needs to be selected for each abstract service based on performance.

Most of the previous work did not considered the data transfer time between previous service and current service during scheduling, which is very pertinent in multi cloud environment. Further, the previous works lacks the practical approach of the way in which the cloud based web services are hosted. In the practical environment, single node can host multiple service instances and each service instance have different load (i.e. maximum number of concurrent requests a node can handle and number of request being currently served). To best of our knowledge, scheduling in such environment is not considered in previous works.

III. PROBLEM FORMULATION

A composite web service (*CWS*) is represented by directed acyclic graphs (*DAG*). Each composite service consists of more than one abstract web service. Consider (AS_1, AS_2, AS_3,…..AS_n), are abstract web services for the i^{th} composite web service CWS_i, where *AS* represent abstract web service and *n* is the number of abstract web services participating in the composite service CWS_i.

A set of candidate services for an abstract web service AS_j is represented as $\{CS_{j1}, CS_{j2}, CS_{j3}\ldots\ldots CS_{jm}\}$, where *m* is the number of candidates services of AS_i.
A set of web service instances running for each candidate service CS_{jk} is represented as $\{ ICS_{ji1r}, ICSj_{i2r}, ICS_{ji3r}, \ldots\ldots ICS_{jkpr}\}$, where *p* is the number of instance services and *r* represents the cloud in which it is hosted where 1<= r<= t.
A set of clouds is represented as $\{C_1, C_2, C_3, C_4 \ldots . C_t\}$, where *t* is the total number of clouds participating. A set of nodes in the cloud C_r is represented as $\{n_{r1}, n_{r2}, n_{r3}, \ldots . n_{ry},\}$ where *y* is the total number of nodes in the C_r. The set of running instances in the s^{th} node n_{rs} in r^{th} cloud is represented as $\{in_{rs1}, in_{rs2}, in_{rs3}, \ldots..in_{rsz}\}$, where *z* is the number of instances in the node n_{rs}.

The allocation and scheduling plan for a composite web service CWS_i is represented as $X = \{Xij \mid i = 1; 2; : : : ; n\}$ such that the total cost *Cost(X)* is minimal. Let *Xi* denote an allocation and scheduling plan for each of the abstract service of the *CWS*, such that Xi = ((M$_{rij1}$; F$_{rij1}$); (M$_{rij2}$; F$_{rij2}$); : : : ; (M$_{rijk}$; F$_{rijk}$)), where M_{rijk} represents the i^{th} selected web service instance running on the j^{th} node N_j on the cloud C_r for abstract web service AS_k of the *CWS* and F_{rijk} represents the execution time of an instance of abstract web service AS_k on the i^{th} web service instance of the n^{th} node of cloud C_k . Equation *1* gives the definition of total cost for a composite service.

The cost constraint problem-
An allocation and scheduling plan X = (Xi | i = 1; 2; : : : ; n), such that the total response time Time(X) is minimal.

$$\text{Time}(X) = \sum (F_{rijk}) \qquad (1)$$

Where $k = 1$ *to* n, *n* is the number of abstract service.

Cloud Locality- In a multi cloud environment, the selected nodes on which services are hosted can be in different clouds. This involves costly data transfer across the clouds over the networks. For the scalability of composite services, it looks it is not always preferable to execute in a single cloud. If the nodes in one cloud are over loaded then it might be preferable to select nodes from other clouds. There is a tradeoff between executing in a single cloud and executing in multiple clouds. For the purpose of executing in a single cloud case, scheduling algorithm can be executed for individual cloud to estimate the execution time of each cloud and then select the cloud, which gives the minimum execution cost. This approach looks theoretically not scalable for very big composite services. To make the scheduling more scalable, it looks multi cloud environment should be used, where nodes of all clouds should be explored to find the optimum scheduling. Therefor it is very important to analyze and study whether the scheduling in multi cloud collectively will be preferable as compared to executing in the most efficient cloud. The cloud locality in a multi cloud environment is the selection of appropriate cloud based on the data transfer time so that overall execution time is minimize as compared to scheduling in a single cloud.

Maximum number of request which can be handled by a node n_{ri} in cloud C_r is represented as MR_{ir} and number of request being currently served by a node n_i in cloud C_r is represented as CR_{ir}. The load on the node n_j in the cloud C_r is represented as $Load_{ir}$ which is equal as the difference between MR_{ir} and CR_{ir}.

$$Load_{ir} = MR_{ir} - CR_{ir}$$

The load of the rth cloud *CLoad* is defined as the average of all loads of its nodes.
$CLoad_{ir} = AVERAGE\ (Load_{1r}, Load_{2r}, Load_{3r}, \ldots\ldots\ Load_{ir})$.

1- The $CLoad_{ir}$ is High i.e. when $20\% <= CLoad_{ir} <= 50\ \%$
2- The $CLoad_{ir}$ is Low i.e. when $50\% <= CLoad_{ir} <= 90\ \%$

With changing load of the clouds, the impact of the cloud locality looks more important. To analyze the cloud locality under different load we run our genetic scheduling algorithm under two loads scenarios. First when load was low and next when load was high.

IV. MULTI-LEVEL CHROMOSOME BASED GENETIC ALGORITHM

We use a multi-level chromosome based Genetic algorithm (*GA*) to solve the cloud allocation and scheduling problem in a multi cloud environment. Its main features include:

1) Multi-level chromosome is used to present the multi-level solution space requires for selecting cloud, node and service instances. A single value of the gene is not very efficient to represent the solution space in multi cloud environment.
2) The fitness function is based not only on the processing speed of the node but also on the load on the node and the communication cost.
3) The logic for selection of appropriate cloud based on cloud locality.
4) The scheduling algorithm exploits the parallelism of abstract services in composite web services to its full extent.

A. Multi-Layer Genetic Encoding

Computer simulation of *GA* makes a population of abstract representations (chromosomes) of candidate solutions (individuals) of an optimization problem evolve toward better solutions based on it parametric instances of mutation and crossover. In multi cloud environment, each cloud has many nodes, which host more than one web services instances. A single abstract service can have many candidates services, each candidate service are hosted on different instances of different nodes, which are spread across the clouds. The scheduling in such multi cloud environment needs to selects not only clouds but also the appropriate node instances. The single value chromosome is not very suitable to represent the solution space. A multi-level chromosome is proposed to represent the solution space. The higher level chromosome is used to present the clouds and within each higher level chromosome, nodes instances are represented. The composite web service allocation has the structure as $(AS_1\ (C_i\ , n_j\ , in_k), AS_2\ (C_i\ , n_j\ , in_k),\ \ldots\ ASn\ (C_i\ , n_j\ , in_k),)$ where AS_n represents n^{th} abstract service of a composite service. C_i represent the cloud selected for the abstract service, n_j represent the node selected within cloud C_i and ink represent the instance selected with the selected node n_j.

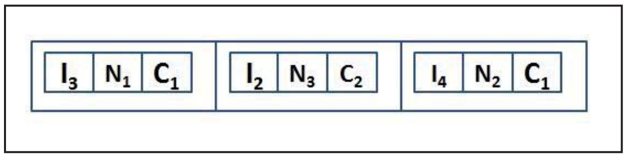


Figure 2 – Multi level Chromosome

Figure 2 represents a chromosome for a composite service having 3 abstract services. It is represented using a chromosome of size 3. Each gene of the chromosome has an inner chromosome of size 3 to represent the cloud, node and instance solution space. In this sample chromosome, the value of gene AS_1 is represented as (I_3, N_1, C_1) representing the fact that the first abstract service is scheduled in the 3rd instance of the 1st node of 1st cloud. Similarly it has values for other *2* abstracts services.

B. Fitness Function

The fitness function is very important in the GA. The fitness function is represented as equation (2). The fitness function used in this *GA* considered not only the execution time on the node based on the processing speed but also the load and cloud locality factor. The cost in transferring the data among the cloud is considered in the fitness function, which helps in selecting appropriate cloud based on the network.

$$\text{Fitness(X)} = \text{IET}_{\text{ijklm}} + \text{IDT}_{\text{ijklm}} + \text{ICT}_{\text{ijklm}} \qquad (2)$$

Where

$$\text{IET}_{\text{ijklm}} = \left(\text{I}_\text{i} * \left(\frac{1}{\text{f}_\text{k}} \right) \right)$$

$$\text{IDT}_{\text{ijklm}} = \left(\frac{\omega}{\text{MSR}_{\text{lm}} - \text{CSR}_{\text{lm}}} \right) \text{iff } (\text{MSR}_{\text{lm}} - \text{CSR}_{\text{lm}}) \neq 0$$

Where $1 \leq \omega \leq 2$

And

$$\mathrm{ICT}_{\mathrm{ijklm}} = \propto * \mathrm{B}_{\mathrm{ij}}$$

IET_{ijklm} = Instance execution time of an i^{th} instance web service of j^{th} candidate service of a k^{th} abstract service running on node *l* of cloud *m*.

ICT_{ijklm} = Instance communication time from its previous execution node of an i^{th} instance web service of j^{th} candidate service of a k^{th} abstract service running on node *l* of cloud *m*.

IDT_{ijklm} = Instance delay time of an i^{th} instance web service of j^{th} candidate service of a k^{th} abstract service running on node *l* of cloud *m*. This delay is due to the internal scheduling delay of the node.

MSR_{lm} = Maximum service request possible on node n_{lm}.

CSR_{lm} = Current number of request on node n_{lm}.

C. Mutation and Cross Over

The algorithm adopts the one point crossover. The mating takes place by swapping the portion of the chromosomes after a randomly selected point known as crossover point. Thus for the crossover of any two chromosomes, it results in offspring which resembles with one parent till the crossover point and with another one after the crossover point till the length of the chromosome.

. The top 2% best chromosome where migrated to next generation without any changes. 83% of new individuals in the next generation are generated by crossover operator. 15% of new chromosomes in the next generation are randomly generated.

V. EXPERIMENTS

Simulation experiments were conducted in matlab *2009a* to evaluate the effectiveness and scalability of our algorithm. The experimental settings for our multi-level GA are as follows: The initial population size and the maximum generations were 100 each. The cloud information was stored in data structure named as *cloud table* which we assume to hold all real time data of cloud. Cloud Table (CT) = {*node_number, clock frequency, max_request, curr_number_req_served*}. The inter cloud and inter nodes distances were also captured in a data structures. The paper executed two set of experiments.

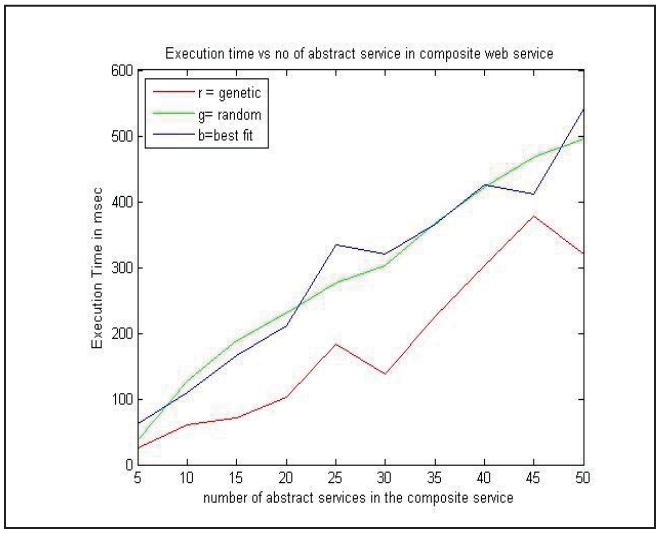
In the first experiments the multi-level chromosome based genetic algorithm is compared with random approach (*RA*) and best processing node (*BPN*) based approach to study the effectiveness of the proposed GA approached. In the (*RA*), for each abstract service, instances are randomly selected and its execution cost is calculated. In the *BPN*, among the all possible feasible candidate nodes hosting services instances, the nodes with high processing speed is selected. This set was executed repetitively by increasing the number of abstract services in the composite service and number of nodes in each participating clouds in the multiple of 5. The number of cloud was also increased in multiple of 2.

In the second experiment, the analysis of cloud locality was carried out. It is assumed that each cloud has 50 nodes and maximum numbers of instances available across all clouds are 100. For the given composite service, the Genetic algorithm was executed one by one for each participating clouds and then it is executed for all cloud collectively. This experiment set up was executed under two scenarios. In the first scenario, the load on each cloud was less < 50% and in the second scenario the load on each cloud was more than 50 %. The objective was to study execution time of each cloud individually as compared to executing in multi cloud collectively under different loads on the nodes.

VI. RESULTS AND ANALYSIS

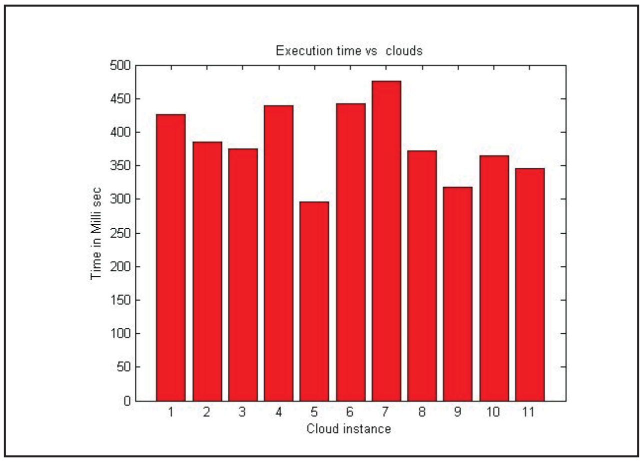
The Figure 3 represents the output for the first experiment to determine the efficiency of the genetic algorithm based approach as compared to *RA* and *BPN*. The result re-stablished the facts in context of the scheduling of composite services in multi-cloud environments that the genetic algorithm based approach gives far better result as compared to two other approaches. The *RA* and *BPN* results are not consistent. They outperform each other randomly depending on the cloud environments and the loads on nodes.

Figure 3- Execution time vs number of abstract services.



The Figure 4 represents the study of the cloud locality. The outcome established the fact that the cloud locality has impacts on the overall execution time. The communication cost is considerably high as compared to the processing time. Figure 4 shows the execution time when the composite service was executed individually in each cloud using the GA. The *11th* cloud instance on the x axis displays the execution time when it was scheduled and executed in all cloud collectively. In Figure 4, the 5th cloud instance has the minimum execution time when the composite service was executed on it.

Figure 4- Execution time vs cloud instances



This second experiments was repeated by increasing the number of clouds in multiple of 5.The table 1 represents the results for all iteration which was executed under low load and high load scenarios. The number of clouds was increased in multiple of 5 keeping the number of abstract service in the composite service as 100 and number of clouds 50 in each cloud. The column *individual cloud* represent whether the minimum execution time was achieved when it was scheduled in a single cloud or when scheduled on the *multi clouds*. All results shows that the scheduling on an individual cloud has always better result as compared to scheduling it on the multi cloud collectively.

TABLE I. ANALYSIS OF CLOUD LOCALITY

No of Clouds	**low load**		**high load**	
	Individual Cloud	***Multi Cloud***	***Individual Cloud***	***Multi Cloud***
5	True	False	True	False
10	True	False	True	False
15	True	False	True	False
20	True	False	True	False
25	True	False	True	False
30	True	False	True	False
35	True	False	True	False

VII. CONCLUSION

The paper initially presented an approach for scheduling composite web services in multi cloud environment using multi-level chromosome based genetic algorithm. Not only did the algorithm consider the processing speed of nodes but also the loads on the nodes in terms of number of request being processing by the nodes. The algorithm preserves the precedence constraints of services within composite services and exploits the parallelism to its full extend. The paper through experiment reestablished the facts in context of the web service scheduling that the genetic algorithm based approach gives far better result as compared to two other approaches based on random and best processing nodes approaches. Further, the paper analyzed and studied the impact and importance of cloud locality in the scheduling of composite services. It established the fact that the scheduling on individual cloud has always better result as compared to scheduling it on the multi cloud collectively. Therefor in a multi cloud environment, it is better to select best performing clouds under the given loads rather than scheduling composite services on multi clouds collectively.

The future study shall include extending the present work for the scheduling of multiple composite services in multi-cloud environment.

REFERENCES

[1] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam, and Q.Sheng. “Quality driven web services composition,” In Proc. of the 12th WWW Conference, 2003, pp. 411 –421.

[2] Lifeng Ai, Maolin Tang, Colin Fidge., Resource allocation and scheduling of multiple composite web services in cloud computing using cooperative coevolution genetic algorithm, Neurak Information Processing, Lectures Notes in Computer Science, 2011, Volume 7063, pp. 258 –267.

[3] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. “Dynamic provisioning of multi-tier internet applications,” In Proc. of the IEEE Int’l Conference on Autonomic Computing, 2005, pp. 217 –228.

[4] G. Soundararajan, K. Manassiev, J. Chen, A. Goel, and C. Amza. “Back-end databases in shared dynamic content server clusters,” In Proc. of the IEEE Int’l Conference on Autonomic Computing, E-Services (EEE), 2005,pp. 551–558 .

[5] Faisal Ahmad, Anirban Sarkar, Narayan C Debnath, Analysis of dynamic web services, IEEE International Conference on Computing, Management and Telecommunications (ComManTel 2014), Vietnam, pp. 2014, pp. 275 – 279.

[6] Faisal Ahmad, Suvamoy Changder, Anirban Sarkar, Web service execution model for cloud environment, ACM SIGSOFT Software Engineering Notes. Nov 2013,Vol. 38, No. 6, pp. 1–13.

[7] Faisal Ahmed, Suvamoy Changder, Anirban Sarkar, Architectural framework for web service dynamics: A Layered Approach, 22nd International Conference on Software Engineering and Data Engineering, Los Angeles, USA, Sept 2013, pp. 73–78.

[8] Weissman, J. B., Grimshaw, A. S, A Federated model for scheduling in wide Area systems, Proceedings of the 5th IEEE International Symposium on High Performance Distributed Computing , 1996, pp.542.

[9] Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S., SNAP: A Protocol for negotiating service level agreements and coordinating resource management in distributed systems, 8th Workshop on Job Scheduling Strategies for Parallel Processing, 2002, Volume 2537, pp. 153–183.

[10] Schopf, J. M., A General Architecture for Scheduling on the Grid, Argonne National Laboratory preprint, ANL/MCS-P1000-1002, 2002.

[11] Schopf, J. M., Ten Actions When Frod Scheduling, Book - Grid resource management,Publisher- Springer,2004, pp 15 – 23.

[12] Shahid Mohammad, Zahid Raza, Level -based batch scheduling strategies for computational grid, Inderscience Publisher, Vol. 5, No. 2, 2014, pp. 135-148..

[13] Rajesh Raman, Miron Livny, and Marvin Solomon, "Resource Management through Multilateral Matchmaking", *Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9)*, Pittsburgh, Pennsylvania, August 2000, pp. 290-291.

A Survey: Securing Cloud Infrastructure against EDoS Attack

A. Sukhada Bhingarkar[1], **B. Deven Shah**[2]
[1] Ph.D. Student, Information Technology , Terna Engineering College, Navi Mumbai, Maharashtra, India
[2] Professor, Information Technology, Terna Engineering College, Navi Mumbai, Maharashtra, India

Abstract - *The Distributed Denial-of-service (DDoS) attack is considered one of the largest threats to the availability of cloud computing services which is used to deny access for legitimate users of an online service. But, Economic Denial of Sustainability (EDoS) attack is a special breed of DDoS attack that targets cloud's pay-as-you-go model. EDoS attack exploits auto scaling feature of cloud. The attacker generates malicious HTTP requests for web application. The Cloud Service Provider (CSP) scales the architecture automatically to service those requests for which cloud consumer is charged. This causes a sustainable decline in the economy of the consumer. The malicious HTTP traffic mimics to be legitimate and hence go undetected. As EDoS attack is carried over extended period of time, the security mechanisms against DDoS attack are not applicable to overcome EDoS attack. This paper presents an overview of detection and mitigation methodologies implemented so far against EDoS attack and it also points out research challenges in this field.*

Keywords: Cloud Computing, DDoS attack, EDoS attack.

1 Introduction

Cloud computing refers to delivery of computing resources over the internet. Cloud computing provides shared pool of resources (example: networks, memory, computer processing, user applications) that can be rapidly provisioned and can be put out with minimal exertion. There are several benefits of cloud computing, such as cost savings, scalability, reliability, maintenance, mobile accessible etc. Besides all these benefits, Cloud Computing does come at the cost of increased security risks which is currently one of the biggest challenges this technology is facing today, limiting the number of organizations willing to embrace it wholeheartedly. DDoS is one type of aggressive attack which causes serious impact on cloud servers. According to [1], in the past year, there has been a 22% increase in total DDoS attacks, and a whopping 72% increase in average attack bandwidth.

1.1 Distributed Denial of Service (DDoS) Attack

A Distributed Denial of Service (DDoS) attack is a malicious attempt to make a server or a network resource unavailable to legitimate users by overloading the server with large number of requests. In DDoS attack, attacker begins by gaining the control of initially one computer and treats it as DDoS master. Then, it gains illegal access to as many computers on Intenet as possible and DDoS master instructs these compromised machines to send a flood of requests to the target server. The target server eventually gets overwhelmed and starts denying the requests of legitimate users [2].

1.1.1 DDoS attack on Web 1.0 applications

Web 1.0 is the first generation of the web which can be considered as read only web which involves limited user interactions or content contributions. It consists of static web pages and only allows searching the information and reading it [3]. Example of web 1.0 is shopping cart application. If such application is hosted over the cloud, then two types of DDoS attack are carried out over such website. The first takes place at the network layer (Layer 3 and 4) and the second at the application layer (Layer 7). At the network layer, attack brings down a website by overwhelming network and server resources, causing downtime and blocking responses to legitimate traffic e.g. UDP Flood, ICMP Flood and Ping of Death. Application layer DDoS attacks mimic legitimate user traffic and crash the web server by searching for content on the site or clicking the "add to cart" button. e.g HTTP flood.

1.1.2 DDoS attack on Web 2.0 applications

Web 2.0 is the second generation of World Wide Web that is focused on the ability for people to collaborate and share information online. Web 2.0 basically refers to the transition from static HTML Web pages to a more dynamic Web that is more organized and is based on serving web applications to users. Examples of Web 2.0 include social networking sites, blogs, wikis, video sharing sites, hosted services, Web applications, and web mashups [4].

A mashup is a web application that uses content from more than one source to create a single new service displayed in a single graphical interface. The architecture of web mashup is shown in figure 1 [5].

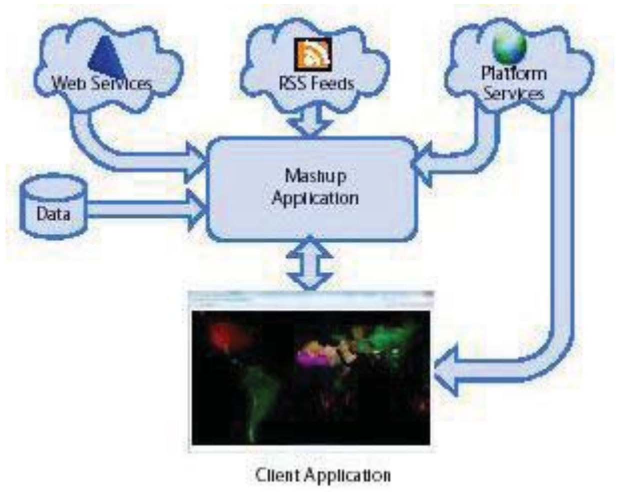


Figure 1: Web Mashup Architecture

When web mashup application is hosted over cloud, it is threatened by many attacks, one of which is DDoS attack. The mashup application may become a target of DDoS attack as follows:

i. Multiple client applications may be a botnet that mimics a legitimate web browser and tries to overwhelm the bandwidth of mashup application by a flood of HTTP requests.

ii. A DDoS attack is possible whenever third party Javascript is executed within client's browser. Third party Javascript logic can include a loop that repeatedly requests resources from targeted mashup application.

Cloud Computing follows utility model where users are charged based on the usage of the cloud's resources. This pricing model has transformed the Distributed Denial of Service (DDoS) attack problem in the cloud to a financial one known as Economic Denial of Sustainability (EDoS) attack [6].

In this paper, we describe the EDoS attack and the methodologies implemented so far for the detection and mitigation of EDoS attack.

The rest of the paper is structured as follows: section 2 briefs about EDoS attack and the difference between DDoS and EDoS attack. Section 3 discusses the detection methodologies applied to differentiate botnet and legitimate users. Section 4 presents mitigation techniques implemented to lessen the effect of an attack. Section 5 gives brief summary and analysis of all the techniques. Section 6 focuses on research challenges and concludes the paper.

2 Economic Denial of Sustainability (EDoS) Attack

DDoS attacks in traditional networked (non-Cloud) environment usually disrupt the service which hurts reputation and incurs economic loss. In Cloud environments, disrupting a service is not so easy due to its inherent capability of auto-scalability and service level agreements (SLA).

However, DDoS attempts on Cloud environments have another more alarming repercussion in that it does the consumption of more Cloud resources to provide auto-scalability, which normally exceeds the economic bounds for service delivery, thereby incurring Economic Denial of Sustainability (EDoS) for the organization whose service or Virtual Machine (VM) is targeted.

EDoS is a new breed of DDoS attack specific to Cloud environments. In this kind of attack, the Cloud service provider activates more and more resources to meet the SLA for the availability of the service for the customer, which eventually adds extra billing cost leading to EDoS. Cloud resources are metered on resource billing. Hence, the fraudulent consumption of bandwidth and computational resources of Web based cloud services incurs financial burden on the Cloud consumer and thus exploits cloud utility model.

2.1 EDoS Attack Threat Model

The target of EDoS attack is a public-facing web application or website hosted in a public Cloud Service Provider environment that is governed by a utility compute pricing model.

In this kind of attack, the attacker's intention is not to make the cloud service unavailable but to put financial burden over cloud consumer by consuming metered bandwidth of web application hosted over cloud.

The following actors are involved in EDoS attack:

1. *Cloud Service Provider (CSP)*: rents its resources and performs billing
2. *Cloud Consumer*: uses cloud resources to host its web application
3. *Legitimate Client*: accesses the services provided by cloud consumer.
4. *Attacker:* intentionally generates fraudulent traffic to hit the economy of cloud consumer.

The EDoS attack network model is shown in figure 2.

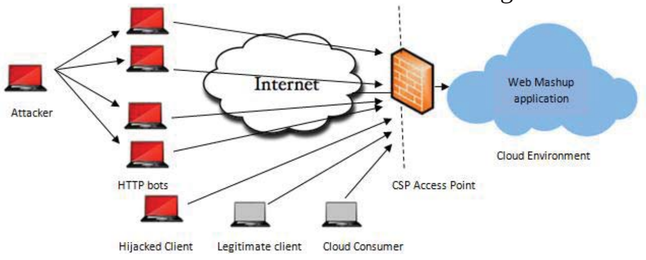


Figure 2. EDoS Attack Network Model

Now a days, the web sites are based on web 2.0 architecture. Hence, assuming that web mashup application is hosted over the cloud, EDoS attack can be performed as follows:

1. The attacker generates HTTP requests by forming a distributed botnet on the internet. These requests have a heavy workload effect on the hosting web application e.g. requesting large files, making

frequent searches on entire product range, which results in large queries on backend databases such as involving the joining of tables. The workload can be on any of the resources; bandwidth, processing, memory etc. In this case, the CSP activates more and more resources to meet the SLA for the availability of the service for the customer, which eventually adds extra billing cost leading to EDoS.

2. In case of E-commerce applications, the cloud consumer earns profit if the client makes a purchase. But if the client only browses the site without making a purchase, then the consumer earns no profit but in turn pays to the CSP.
3. EDoS attack is also possible through hijacked client browsers specifically in web 2.0 architecture wherein malicious code can be injected in the browser by the hacker that generates repeated requests to targeted mashup application resulting in overwhelming the bandwidth of mashup application.

2.2 Difference between DDoS and EDoS

There are two major differences between EDoS and DDoS attacks. First, EDoS attacks aim to make cloud resources economically unsustainable for the victim, whereas DDoS attack aims to degrade or block cloud services.
Second, DDoS attacks are carried out in a short time period whereas EDoS attacks are more subtle and carried out over a long period of time.
Third, EDoS attack occurs just above the normal activity threshold and below the DDoS attack threshold [7].
Therefore, it may be unlikely to be detected by traditional intrusion detection systems and also the methodologies used to overcome application layer DDoS attacks are not applicable to EDoS attack.

3 EDoS Detection Methodology

Detecting EDoS attack is very difficult because the way an attacker requests web resources is like that of any legitimate client and the only differentiating attribute is their intention [7]. Thus, the purpose of detection methodology is to differentiate bot behavior from human behavior.

3.1 Zipf's Law Distribution

Zipf's law was originally introduced in the context of natural languages and is performed by calculating the frequency of occurrence *F* of each word in a given text. By sorting out the words according to their frequency, a rank *R* can be assigned to each word, with for the most frequent one [8].
The methodology discussed in [9] applies the properties of Zipf's law in the analysis of aggregated user consumption patterns. The web server log contains request record for the web pages. Let *fi* be the frequency of requests and i be the rank assigned to the page. The page which is referred most is assigned rank one and so on. Thus, if Zipf's law holds, then the frequency *fi* is inversely proportional to the rank of the page. A typical Zipf's Law rank distribution is shown in Figure 3. The y-axis represents occurrence frequency, and the x-axis represents rank (highest at the left) [8].

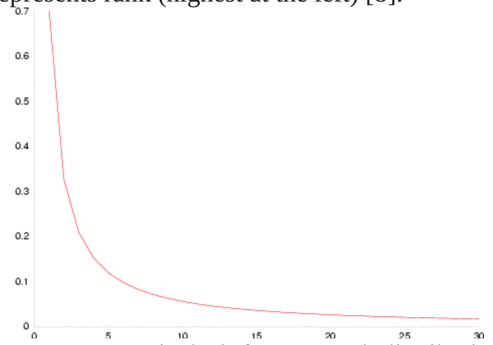


Figure 3. A typical Zipf's Law rank distribution

The detection methodology discussed in [9] determines Zipf's distribution for the training and test data sets and then computes linear regression line for each distribution. The slopes of the respective linear regressions are compared with the statistical hypothesis that the slopes are the same. If analysis indicated that the slopes were significantly different, then, it concludes that fraud motivated access patterns are observed in Web request logs.

3.2 Entropy Detection

In order to detect EDoS/ FRC attack, individual user behavior is modeled in [9] by analyzing the entropy of session lengths generated by an individual over a fixed duration of time. The session length is defined as the number of web documents requested during a session. The entropy of session length for session j is Hj composed of the n events is defined as:

$$H_j = - \sum p_i log_2(p_i)$$

The hypothesis in [9] is that, randomly generated session lengths, deviate sufficiently from a profile of normal user behavior. The proposed detection methodology in [9] computes a standard of entropy of normal session lengths based on a Web request log and then calculates entropy of session lengths for each unique user. Then, it compares the entropy result to the standard. If user's session length entropy is outside the standard, the user is designated as malicious.

3.3 Time Spent on a Page (TSP) based Detection

The methodology in [10] considers Time Spent on a Web Page(TSP) as a request dynamic to detect the attack traffic. As the bot traffic is automatically generated and its intention is to create heavy workload by browsing the web pages, a large traffic having small TSP values can be considered as malicious.

Firstly, TSP is calculated for each page as the difference between timestamps of two consecutive requests for web pages. Then, the mean of TSP is calculated from the data set Xi which represents the page requests for a page i. The requests generated by each cloud user are collected in form of logs at cloud controller. Each user has a separate log. The deviation of each log request from mean TSP is calculated. Lastly, Mean Absolute Deviation (MAD) is determined by taking the average of these deviations.
For each user, MAD plot is drawn showing deviation vs page visited and compared it with the mean of all pages. If the curve of any user deviates more from the mean curve, then that user is identified as malicious.

3.4 Web Usage Features based Detection

The application layer DDoS attacks focus on request rates of clients to differentiate between legitimate user and attacker. But, this technique is not applicable to detect attackers in EDoS attack as EDoS attack sustains over a long period of time. Hence, the attribution methodology presented in [11] targets four aspects of client web browsing behavior i.e. request volume, session volume, average session length and chi-square statistic.
The quantity of primary requests invoked by a client within an observation time period is called as Request Volume. Primary request is explicit request from a client to whereas secondary request originates from primary web document to retrieve certain image, video etc. The minimum threshold considered is 5 requests per client according to Cumulative Distribution Function (CDF) calculated over training data set. As the intention of attacker in FRC attack is to consume as much bandwidth as possible, the attacker generates number of requests more than the threshold.
Session Volume is the quantity of web sessions attributed to a single client within an observation period. The attacker in FRC attack distributes his/her resource consumption over the course of many days by launching multiple fraudulent web sessions. According to CDF for training data set, the average session volume per client is three. Thus, the client requesting slightly more than three sessions is flagged as malicious.
The number of primary requests in a web session is termed as session length and the mean of these lengths is called as average session length. According to CDF, the average session length is considered to be less than that of 5 requests. The attacker usually tries to keep average session length minimal, but for that the attacker is forced to initiate more web sessions which increases his/her session volume score and hence can be identified as fraudulent.
The Zipf like distribution for training data set broadly states that 10% of the requested documents are requested 90% of the time i.e. a significant fraction of normal client behavior is reasonably self-similar to the overall client population. Thus, chi-square statistic is used as a relative measure of similarity or dissimilarity between individual client request distributions and the overall population distribution.
The web pages are ranked and are grouped into discrete bins, each bin having probability π. The expectation for each bin is $E_i = n\pi_i$. For each client in the test dataset, a chi-square statistic is computed as:

$$\chi 2 = \sum (n_i - E_i)^2 / E_i$$

Then, the overall CDF is constructed. The client having high chi-square statistic score is considered as legitimate.
The scores for all above four metrics are summed together and compared against threshold to identify client as legitimate or malicious.

4 EDoS Mitigation Methodology

Mitigation techniques are applied to lessen the effect of an attack. This section describes various frameworks proposed so far to mitigate EDoS attack.

4.1 EDoS Armor

The mitigation technique called as EDoS Armor in [12] concentrates on protecting E-commerce applications with the assumption that attacker performs EDoS attack by not following regular workflow of E-commerce applications by purchasing the item but, by idle surfing of the web sites for entertainment or price checks.

EDoS Armor proposes muti-layered defense system which includes two modules:
(i) admission control: Challenge Server identifies whether the request is from legitimate client or bot by means of challenge which can be either of image based or any cryptographic challenge. Thus, it authenticates number of users in the system. Then, it allows limited no. of valid clients to send the requests simultaneously through port hiding. This avoids over burdening.
(ii) Congestion control: This module takes care that maximum resources are available to the good clients. The client is categorized as good or bad client depending upon his browsing behavior. Decision tree algorithm J48 is used over access log to classify the clients. The parameters used for classification are like Purchasing History, CPU Processing time, Session information, Resources Access Pattern. The priority is assigned to the clients based on types of resources they visit and the types of activities they perform.

4.2 In-cloud Scrubber Service

In-cloud scrubber service is an on-demand service proposed in [13] to mitigate network layer and application layer EDoS attack based on puzzle approach. The cloud service is switched between two modes: normal and suspected based on server load and network bandwidth. If the resource depletion level goes beyond the limit and bandwidth traffic is also very high, then the service provider suspects high rate attack. The system switches to suspected mode and called scrubber service.

The primary function of scrubber service is to generate a puzzle to check legitimacy of the client. The service generates partial hash input and hash output and transmits both pieces of information to the client.

i.e. $H(X \| k) = Y$

where, X and Y are the puzzle parameters provided by service and k is a puzzle solution.

The client is supposed to apply brute force method to find out the value of k. Here, as the puzzle generation and verification is done by third party i.e. scrubber service, the burden on Cloud Service Provider is reduced.

4.3 sPoW (Self Verifying Proof of Work)

The idea behind mitigation technique presented in [14] is to grant access to only those clients who are willing to pay for the service. The client has to contact sPoW Name server for name resolution when it wants to establish communication with the server. The client defines crypto puzzle difficulty level, k and subsequently makes a request. The server in turn generates a puzzle of required difficulty level which client is supposed to solve.

If an initial connection request is not successfully made during a given frame of time, the client may request for a more difficult puzzle. Upon successfully solving the puzzle of given difficulty level, the server establishes a secure communication channel for message exchange. This method transforms network level EDoS into traffic which can be distinguished through basic packet pattern matching. It helps to discard attack traffic before billing is triggered.

Also, application level EDoS is addressed by prioritizing the traffic. The existing connection requests have given priority over initial connection requests. The existing connection which carries purchase transaction traffic is assigned more priority than casual browsing traffic carrying connection. The initial connection requests are prioritized using sPoW scheme in which priority is based on the resources expended by client in solving the puzzle because this reflects the urge of client to establish the connection. The sPoW is self verifying because crypto-puzzle consists of both server channel connectivity details and partial encryption key. By brute forcing k bits, client discovers server channel and can place initial connection request which is queued at server end based on difficulty level of puzzle. Thus, this technique removes the requirement of separate verifier and ensures legitimate client request as client had expended enough resources to establish the connection.

4.4 DDoS Mitigation System (DDoS-MS)

The mitigation mechanism proposed in [15] is applicable to those enterprises which allow their employees to bring their own mobile device at the workplace to access enterprise database. This policy termed as Bring Your Own Device (BYOD) results into the threat of EDoS attack as the devices used by the employees are not configured by the organization.

The idea behind DDoS-MS framework is to test first two packets of each session in two successive stages instead of testing all the packets. First packet is tested by verifier node using Graphical Turing Test. Second packet is tested by client puzzle server which uses crypto puzzle to verify the source of packets.

The firewall adds the source IP address of incoming packet in either white list or black list depending on the result of verification process. The green nodes hide location of the server. The server receives only those packets which come through the green nodes. DDoS-MS enhances EDoS shield framework by decreasing end-to-end latency.

4.5 Cloud Trace Back Model (CTB)

Mary et.al. [16] have proposed a combined approach to protect the cloud against DDoS and EDoS attack. Cloud Trace Back architecture applies SOA to trace back methodology to identify true source of DDoS attack. CTB is based on Deterministic Packet Marking Algorithm wherein the attacker sends SOAP request message for web service to CTB. CTB places Cloud Trace Back Mark (CTM) within the header. Then, the SOAP message is sent to the web server. Upon discovering of an attack, the mark can be extracted to reconstruct the path.

CTB does not eliminate DDoS attack. Hence, trained back propagation neural network model is used called as Cloud Protector.

EDoS attack is detected using Verifier nodes which are a pool of virtual machine nodes. V-nodes verify the legitimate requests at application level using unique Turing test e.g. unique Question Testing. Depending upon the result of verification, the source IP address of the request is added to the white list or black list which is maintained by Virtual Firewall. Here, the architecture assumes that the system is protected against IP spoofing attacks.

5 Review of Countermeasures

A survey carried out above on detection and mitigation methodologies against EDoS attack show that there are still opportunities present to carry out further research work. Table I and II shows the summary of detection and mitigation techniques along with their limitations which can lead to further research.

Table I. Analysis of Detection Techniques

Sr. No.	**Approach**	**Methodology**	**Advantages**	**Limitations**
1	Exploiting Cloud Utility Models for Profit and Ruin	- Zipf's law Distribution - Entropy Detection of Session Lengths	Effectively detects anomalous behavior from web request logs.	- Supports only SaaS kind of service. - Does not consider Web 2.0 architecture
2	Detection of Economic Denial of Sustainability using Time Spent on a Web Page in Cloud	-Calculation of Mean Absolute Deviation of Time Spent on Web Page	Simple method to differentiate legitimate traffic from attack.	- Supports only SaaS kind of service. - Does not consider Web 2.0 architecture
3	Attribution of Fraudulent Resource Consumption in the Cloud	Analysis of web browsing behavior. - Request volume - Session Volume - Avg. Session Length - Chi-square statistic.	Minimum False Positive and False Negative Rate.	- Attacker can learn normal request patterns. - Does not consider Web 2.0 architecture

Table II. Analysis of Mitigation Techniques

Sr. No.	**Approach**	**Methodology**	**Advantages**	**Limitations**
1	EDoS Armor: A Cost Effective Economic Denial of Sustainability Attack Mitigation Framework for E-Commerce Applications in Cloud Environments	- Authentication through Crypto-puzzle - Port hiding - Decision Tree algorithm	Classifies users effectively. Supports dynamic web applications	- Provides defense only for E-commerce applications. - Does not consider Web 2.0 architecture
2	Mitigating Economic Denial of Sustainability in Cloud Computing using In-Cloud Scrubber Service	- Authentication through Crypto-puzzle	On-demand Scrubber service which removes the burden from CSP.	- Legitimate user is unwilling to solve such puzzles. - Prevents only network level EDoS attacks.
3	sPoW: On-Demand Cloud based eDDoS Mitigation Mechanism	- Crypto puzzle - Packet Filtering	Prevents EDoS traffic from using costly cloud resources due to the provision of self verification.	- Prevents only network level EDoS attacks.
4	A New Method to Mitigate the Impacts of Economic Denial of Sustainability Attacks against the Cloud	- Graphical Turing Test - Crypto puzzle	Tests only first two packets rather than testing all the packets. Decreases end-to-end latency.	- Does not deal with IP packet fragmentation. - Does not deal with dynamic IP addresses.
5	Secure Cloud Computing Environment against DDoS and EDoS attacks	- SOA applied to Cloud Trace Back Model - Neural Network - Unique Turing Test - Packet Filtering	Combined approach against DDoS and EDoS attack	- Does not deal with IP spoofing.

6 Research Challenges

There are following issues in present detection and mitigation approaches :

- All present approaches concentrate only on differentiating user as legitimate and or malicious (i.e.botnet). However, today's websites are based on web 2.0 architecture. Hence, if website hosted over a cloud is using mashup technology, then it may receive requests not only from users but also from another website. For example, if the website is of travelling website, it in turns calls API of another web site like hotel booking. So current approaches lack to distinguish whether the request is generated from bot or it is from another website.
- Web mashup application can be targeted by hijacked client's browsers that execute malicious code to generate repetitive HTTP requests. This threat is not considered in the current approaches.
- Most of the current approaches use CAPTCHA test to differentiate between human and bot. But, legitimate users are unwilling to solve CAPTCHA test. Also, bots are able to solve CAPTCHA test.
- As a part of mitigation strategy, most of the approaches deny the request once the user is classified as malicious but this is less elegant solution.

7 Conclusion

EDoS attack is more subtle attack than DDoS that seeks to disrupt long-term financial viability of operating in cloud by exploiting utility pricing model.

Unlike short lived DDoS attacks, EDoS attacks span over a long period of time. Hence, traditional DDoS defense techniques are not applicable to defend EDoS attack. Until recently, this attack is not much addressed. The current approaches discuss threat model and defense strategies for web contents hosted on cloud which follow web 1.0 architecture. But in an age of browser-based botnets and web 2.0 applications, it is necessary to build up multi layered framework which can do traffic profiling and visitor classification effectively.

8 References

[1] DDoS Attacks Evolve into Sophistication, http://www.infosecurity-magazine.com/news/ddos-attacks-evolve-into/ [Online; accessed 22-Jul-2014].

[2] Shui Yu et.al., "Can We Beat DDoS Attacks in Clouds?", IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 9, September 2014, pp. 2245-2254

[3] [Online]. Available: Web 1.0 http://computer.howstuffworks.com/web-101.htm.

[4] James Governor, Dion Hinchcliffe, Duane Nickull, "Web 2.0 Architectures", O'Reilly, 2009.

[5] [Online] Available: Enterprise Mashups, https://msdn.microsoft.com/en-us/library/bb906060.aspx

[6] J. Idziorek, M. Tannian, and D. Jacobson, ''Insecurity of Cloud Utility Models,'' IT Prof., vol. 15, no. 2, pp. 22-27, Mar./Apr. 2012.

[7] Saeed Shafieian et.al. "Attacks in Public Clouds: Can They Hinder the Rise of the Cloud", Cloud Computing: Challenges, Limitations and R& D Solutions, Springer, 2014.

[8] Shi-Ming Huang et.al. "An Investigation of Zipf's Law for Fraud Detection", Decision Support Systems, Vol. 46, Issue 1, Dec. 2008, pp-70-83.

[9] J. Idziorek and M. Tannian, "Exploiting Cloud Utility Models for Profit and Ruin," *Proc. 2011 IEEE 4th Int'l Conf. Cloud Computing (Cloud 11),* pp. 33–40.

[10] Anusha K. et. al. "Detection of Economic Denial of Sustainability using Time Spent on a Web Page in Cloud", IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2013

[11] J. Idziorek, M. Tannian, and D. Jacobson, "Attribution of Fraudulent Resource Consumption in the Cloud," *Proc. 2012 IEEE 5th Int'l Conf. Cloud Computing* (Cloud 12), IEEE, 2012, pp. 99–106.

[12] Muddassar Masood et.al "EDoS Armor: A Cost Effective Economic Denial of Sustainability Attack Mitigation Framework for E-Commerce Applications in Cloud Environments", IEEE International Multi Topic Conference (INMIC)Conference, 2013

[13] M. Naresh Kumar, et. al.. "Mitigating economic denial of sustainability (edos) in cloud computing using in-cloud scrubber service", In *In Proc. of the Fourth Intl' Conf. on Computational Intelligence and Communication Networks (CICN),* 2012.

[14] Soon Hin Khor and Aki Nakao. "spow: On-demand cloud-based eddos mitigation mechanism", In *In Proc. of the Fifth Workshop on Hot Topics in System Dependability*, 2009.

[15] Wael Alosaimi and Khalid Al-Begain, "A New Method to Mitigate the Impacts of Economic Denial of Sustainability Attacks Against the Cloud", 2013

[16] I. Mettildha Mary et. al. "Secure Cloud Computing Environment against DDoS and EDoS attacks", International Journal of Computer Science and Information Technologies, Vol. 5, 2014

Enabling Migration Decisions Through Policy Services in SOA Mobile Cloud Computing Systems

J. Medellin, O. Barack, Q. Zhang, and L. Huang
Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX, USA

Abstract—*According to some estimates, mobile applications will drive over half of the volume on the internet by the second half of the 2010s. This powerful technology is constrained by energy sources (battery), mobile spectrums and other physical characteristics. Mobile Cloud Computing (MCC) complements these devices by bringing the vast resources available in Clouds to them. Applications can be modularized and shipped for processing to these centers of greater capability freeing up computation, data requirements and ultimately energy.*

Our approach in this study is to first use SOA/BPEL/services design principles to modularize the application. Once the application is modularized we propose an architecture for a service that drives the decision of what to migrate. The application is scaled and migrated with the AppleHandoff (iOS 8.1.3) facility to provide objective results to the theory. The service architecture designed for the policy service is based on SOA principles and includes Quality of Service (QoS) and Quality of Experience (QoE) drivers identified by the user.

Keywords: Mobile Cloud Computing, Business Process Execution Language, BPEL, Services Oriented Architecture, SOA Services, Cloud Process Migration

1. Introduction

Mobile and Cloud Computing continue their march towards becoming pervasive players in all aspects of life. Cisco estimates that by 2016 more than 60% of world IP traffic will be generated by mobile devices [1] and the number of mobile multimedia users will top 800 Million by 2015 according to other estimates [2]. The global mobile application market is estimated at \$9.4 BUSD in 2014 [3][4] and will grow at a compound annual growth rate of close to 38% per year to top out at almost \$47 BUSD in 2019 [4].

Cai et al [5] declare that the essential cloud computing characteristics include on-demand service, broadband network access, resource pooling and measured service. Those characteristics can be delivered to devices such as laptops and desktops that have reliable network connections, access to electrical power and sufficient computing capacity. Correspondingly, Zhang et al [6] identify battery life, network connectivity and computational power as being the three largest challenges to mobile computing environments.

A first critical limitation to these environments is battery life. While computing capabilities for mobile devices have been significantly improving over time, battery life has not kept up and has only been improving at a rate of 5% per year, far below the improvements in computing power [4]. In order to preserve resources and battery life, execution must be off-loaded from the hand held device [1].

This paper proposes the usage of a centralized service on the mobile device to make the decision to migrate (or not) and what to migrate to the Cloud.

2. Motivation

The Cloud revolution has the ability to unleash the vast resources to supplement the smaller capabilities of mobile devices. Processes can be migrated as entire applications, methods or threads [7] and tradeoffs have to be evaluated on what to migrate.

2.1 Current approaches to modularization

Several methods exist for modularizing mobile applications for migration to the cloud. They range from entire application migration to replication with state transfer and finally some propose to modularize based on Services Oriented Architecture/BPEL concepts.

Modularization at the application level has been widely used by most Cloud providers. In this pattern, the function call is issued to a remote URL with some parameters passed to it. The cloud executes the required functionality and returns a payload to the device. The UI logic of the application displays the output to the user. In this particular case, most of the application is resident in the cloud and very little decision making is left to the device itself.

A second pattern replicates the application on both the device and the cloud and annotates the code at certain points where, according to developer judgment, an exchange of state is feasible. The compiler exposes those particular partitions and exchange of data happens at those junctures. This exchange does not have to happen on the entire state, some significant optimizations have been proposed by Yang et al [8].

Our approach proposes the use of Services Oriented Architecture Business Process Execution Language (BPEL). BPEL is a formal semantic language that allows the analyst to express business process concepts in both sequential and parallel tasks. It assumes the invocation of Services by means of contracts requiring only what is needed by the service to execute and what it will return as output. This "natural" granularity forms the basis for a smaller set of data transfer between functions being executed. BPEL is based on business process decomposition. In this technique, macro business processes (e.g., invoicing) are decomposed into sub-processes until they can be encapsulated into standardized services (e.g., vendor master data maintenance). BPEL is a recommended tool for implementation of Services Oriented Architecture (SOA) design patterns and has become more prevalent since introduction of the 2.0 standard in 2007[9].

2.2 The decision to migrate is complex

The decision to migrate an application or portion of it to the Cloud for execution is complex because some of the factors are not known. Known factors include the amount of battery remaining and the connectivity scheme (WiFi or mobile). Some of the more important unknown factors include:

1) Whether the entire application needs to be migrated or if certain portions can be migrated and what those portions are.
2) The amount of CPU cycles (and therefore battery) that is required to encapsulate all or portions of the application.
3) The mobile data/roaming plan economics of the user and the current level of consumption of base charges.
4) The encapsulation and round trip latency of the application under the current spectral scenario.
5) The amount of CPU cycles the application will consume in computation and data intensive tasks if left to execute on the device.

The objectives of this document are to investigate the implications on partitioning the application at the service contract level and to propose an architecture to enable decisions on what portions (if any) to migrate. We begin this discussion by presenting prior work applicable to modularizing and migrating of objects in MCC. The key areas reviewed are research on SOA and the services contract, MCC's heterogeneous environments, various methods for offloading while conserving battery, and tradeoff evaluations for offloading.

Next we build a hypothetical MCC business application that is created under a SOA/BPEL script and contains four services. The data and computation service contracts are analyzed versus the service operations themselves; scenarios for partitioning at the contract and other areas of the application are analyzed.

Finally, a services architecture is presented that accumulates historical, user preference and empirical data in order to facilitate the decision. The application and service architecture operates in a mix of handheld, laptop and cloud resources and is scaled in order to determine the feasibility of making service contract based partition decisions.

The two contributions that are made by this research are answers to the following questions:

- Where are the most efficient points to partition applications that have been designed using SOA/BPEL/services principles?
- What is a potential service architecture that could be used to accumulate data parameters relevant to making the migration decisions in the same application?

3. Related Work

Most of the research has been focused on potential approaches to offload computation under the constraints of unstable network connectivity and energy usage.

3.1 Heterogeneity and its problems

The MCC domain is fertile with issues related to heterogeneity, very little has been created to enhance attributes such as portability or transferability in these applications. In very similar discussions Shiraz et al [7] and Sanaei et al [10] identify heterogeneity in MCC as a major stumbling block in the vision of delivery to the mobile device as delivered to the desktop.

Both Shiraz and Saneti discuss a variety of networks available to overcome the issues of heterogeneity. MAUI, CloneCloud and COMET are discussed as three tools that are able to implement computing migration in different ways. MAUI focuses on application migration, CloneCloud on method level migration and finally, COMET's focus in on thread-based migration. All are very dependent on the Android hand held device and intel-based proprietary Clouds.

3.2 Offloading Tradeoff Analysis

A key decision to be made is if the application/method-/thread should be migrated to the Cloud. Yang et al [8] propose models for offloading based on acyclic graphs. This approach assumes that an application/method/tread either can or cannot be modularized to be migrated. If it cannot, then the migration decision is binary, it either is or is not migrated and certain formulas based on historic prediction are provided. If the application can be modularized, then the analysis focuses on which portions to be migrated based on the acyclic graph (sequential, hierarchic or mesh). The offloading algorithm measures the tradeoff between analysis and migrating versus executing the process on the device.

3.3 Modularizing and Efficient Transfer

Yang et al [8] provide a study on migration using CloneCloud. In that paper, they identify modularizing approaches as static and dynamic. Static application modularizing requires the developer to annotate the application code. The compiler uses these annotations to direct where the code should be executed (mobile or cloud) as the application is built. The approach is viable where all aspects of the mobile app can be predicted which is typically not the case in MCC applications without a SOA architecture. In the dynamic approach, the decision of where to locate the execution those partitions is made as the application is processing. They indicate that due to the nature of MCC and the lack of predictability in the environment, this is perhaps the best approach to take in modularizing. Shiraz et al [7] further indicate that the dynamic approach seems to be the prevailing approach to modularizing.

Yang et al also proceed to propose algorithms to reduce the transfer of data to achieve a 97% reduction. This is impressive and a very good example on how certain frameworks can be further optimized. We believe these types of algorithms and approaches will be implemented by the frameworks as they further mature into this space.

3.4 Corporate Mobile Applications

Cox et al [11] and Privat and Warner [12] have developed several methods for "industrializing" Apple iPhone "Apps". Their frameworks provide a certain level of resiliency and application survivability that is necessary in commercial applications. They base their methodology on developer analysis and intuition on how to modularize and how to migrate.

3.5 Cloud Vendor-specific MCC Tools

Significant Cloud vendors also provide toolkits for development and optimization of MCC. We do not address those particular frameworks due to potential Cloud vendor lock-in.

3.6 SOA, BPEL and services contracts

Erl [13] defines SOA as: "Service-oriented architecture represents an archtiectural modelâĂę.It accomplishes this by positioning services as the primary means through which solution logic is presented"

Many organizations have also adopted a language called BPEL to enable orchestration of services[9]. The objective of BPEL is to provide a structured language where business process operations are decomposed until they can be syntactically and semantically executed by services. The services are orchestrated by sequencing them in a prescribed manner. These services can also be reused in other business processes provided their context requirement is the same.

BPEL has its own structure and hierarchy that enable transactions in processes to be executed. This document uses the BPEL 2.0 for defining the process to be modeled by the MCC application[9].

As discussed above, a major component of the SOA architecture are services. Services have two fundamental components; a contract and a set of logic to perform their tasks. The contract is exposed to those needing to use it while the logic of the operations being performed is hidden. The vision for SOA is a reusable set of services that can assist in composing or extending existing functionality required by the enterprise [14].

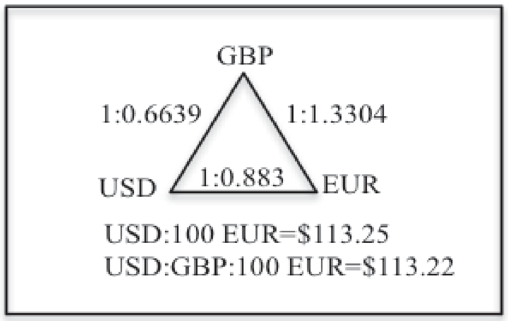


Fig. 1: Triangulated vs Direct Currency Analysis.

4. Experiment Setup

The objective of this experiment is to provide the feasibility for modularizing applications based on BPEL and services analysis.

4.1 Business Process Being Modeled

World currency exchange rates are set on a variety of parameters. Sometimes the currencies are in parity (meaning that if one was to exchange US Dollars (USD) for Euros (EUR) at a given rate or instead bought first British Pounds (GBP) and then EUR with them the operation would yield the same monetary effects; with fees held constant). In practice this is not the case and the disparity between the currencies is a daily aspect of world markets and connerstones of trading desks, currency hedging services and other firms.

The scenario that is being modeled is one in which a company has to pay an invoice in a currency that is not its own. The company is interested if it should go do a direct exchange (e.g. USD to EUR) or if there is economic advantage by triangulating (e.g. USD to GBP to EUR). The user keys in the amount, selects the target currency, the source currency and the triangulating currency. The mobile application outputs how much it would cost for a direct exchange vs a triangulated exchange as in Figure 1.

In the above example, it is more expensive to pay a direct exchange of US Dollars to Euros ($113.25USD * 0.883=100EUR) versus triangulating through the British Pound ($113.22USD * 0.6639 GBP *1.13304=100EUR). This scenario assumes no currency fees or other factors affecting the currency exchange.

4.2 Application Modeling with BPEL, SOA and services

A best practice for modeling BPEL is through the UML Activity Diagram [15]. The following activity diagram would represent the BPEL process/method for the currency analysis described above and is an adaptation of the UML Activity Diagram.

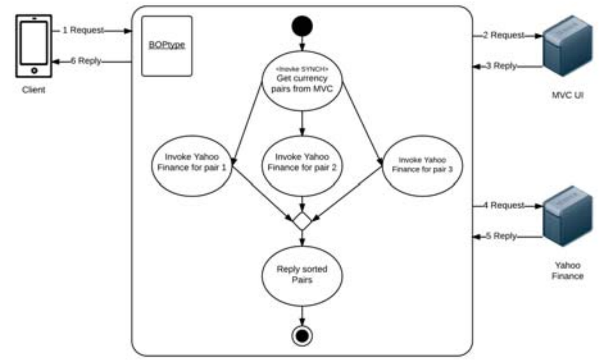

1 Request
6 Reply
Client
BOPtype
«Invoke SYNCH»
Get currency pairs from MVC
Invoke Yahoo Finance for pair 1
Invoke Yahoo Finance for pair 2
Invoke Yahoo Finance for pair 3
Reply sorted Pairs
2 Request
3 Reply
MVC UI
4 Request
5 Reply
Yahoo Finance

Fig. 2: UML Activity Diagram for Currency Analysis.

An excerpt of the corresponding BPEL for that process is in Listing 1 as follows:

```
L#   BPEL Statements
——  ——————————————————————————————
<process>
<process name="CurrencyCalculation"
  >Start identification of service<
<partnerLinks>
      <partnerLink name="ratePair"
      partnerLinkType="rt:ratePair"
      myRole="rateService"
      partnerRole="rateServiceExchange"
 </partnerLinks>
  >End identification of service<
  >Start contract variables<
<variables>
<!—input currency source —>
      <variable name="currencySource"
      messageType="fromMessage"/>
      <!—input currency target —>
      <variable name="currencyTarget"
      messageType="toMessage"/>
      <!—input pair exchange rate —>
      <variable name="exchangeRate"
      messageType="rt:pairExchange"/>
      <variable name="invoiceAmount"
      messageType="invAmt">
 </variables>
  >End contract variables<

<sequence>
  >Start invocation of retrieval service<
      <!—retrieve currency pairs —>
      <!—First module to encapsulate —>
      <receive partnerLink="ratePair"
      portType="rt:retrieveRate"
      operation="pairExchangeRate"
      variable="currencySource"
      variable="currencyTarget"
      variable="exchangeRate"
      createInstance="yes"/>
  >End invocation of retrieval service<
(. . . . )
</sequence>
(. . . . )
</process>
```

The BPEL declares the process, identifies the service to be invoked, contract variables and invokes the service to retrieve rates. The additional BPEL statements for the other services and contracts are omitted.

4.3 Cloud/Application Architecture

An application using the Apple SWIFT [16] coding environment was constructed to execute the triangulation analysis. The application uses a source currency, a target currency, and a target payable amount as inputs. The application triangulation currencies are driven from an array that is populated by the user. The Apple infrastructure allows the ability to handoff an activity from one AppleApp to another device operating the same AppleApp through their iCloud infrastructure [17]. Apple also provides the facility to notify the user if a pattern of CPU, Energy or Data Plan usage is detected so it can choose a hybrid mode of execution (with offloading). That algorithm can be created and implemented through the analysis of patterns in the App trace file with the Apple Instruments package. The following figures illustrate the application architecture of the system that was constructed (the iPad has an electrical connection removing the battery consumption constraint from that machine). They are presented as a static view (allocation) and sequence (dynamic) views. The allocation view illustrates the residence of key components on devices/services while the sequence view illustrates the key steps in execution of the application scope.

Four activities were created inside the application. Two of them were modeled to work with the AppleHandoff utility[18].

The activities modeled were (web queries and calculations can be handed off to a remote iPad or a Mac connected through iCloud):

- Parameter setup
- Web queries
- Calculations
- Return result

Figure 4 depicts the execution sequence and the update of the state token after each method is executed.

4.4 Technical Infrastructure

The application was developed with the following technical environment:

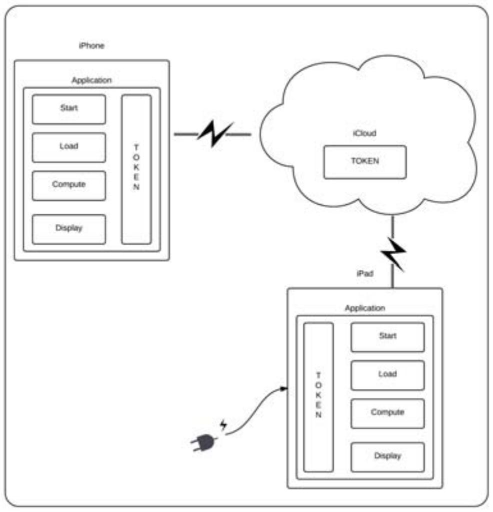


Fig. 3: AppleApp System Architecture: Allocation View.

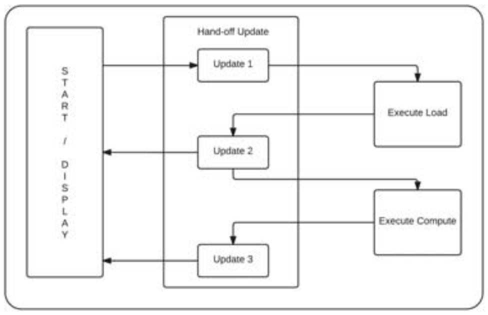


Fig. 4: AppleApp System Architecture: Dynamic View.

- Apple Macintosh working on OSX 10.10.2 Yosemite Operating System blue tooth 4.0
- Apple iPhone 6, operating on iOS 8.1.3 Mobile Operating System blue tooth 4.0
- Apple iPadMini 3, operating on iOS 8.1.3 Mobile Operating System blue tooth 4.0
- Apple XCODE 6.1.1 with SWIFT version LLVM compiler, Apple Instrument Package
- Synchronization via iCloud WiFi Ethernet

5. Experiment Execution and Results

The application was scaled to 100 currencies to identify the differences in data requirements.

5.1 Estimating volume of operations by each service contract and service

The service contracts remain constant during the execution of the applications. One array with input values for the currency pairs to be retrieved is passed to the retrieval service contract and one array with the currency pair exchange rate results is passed to the calculation service. The size of these arrays increases by three characters for the initial array and by 4 characters for the second array for every additional currency to be evaluated. This is a modest increase when compared of the incremental call to one web function, passing of parameters, receiving reply, parsing for values and storing the currency pair values in the retrieval service. It is also modest when compared to the triangulated calculation to compute the computation of load components is given in Table 1.

Table 1: Computation of load by component of AppleApp.

Item	Assumption	Type	Value
Input Currencies	1 array passed	Message	1
Contract 1	1 array passed	Message	1
Retrieve	603 query/parse	Mixed	603
Contract 2	1 array passed	Message	1
Compute 3	computations/point	Computation	603
Contract 3	1 array passed	Message	1
Output Results	1 array passed	Message	1

Figure 5 identifies the contracts as lowest points of transfer between the services. These points were used to partition the application.

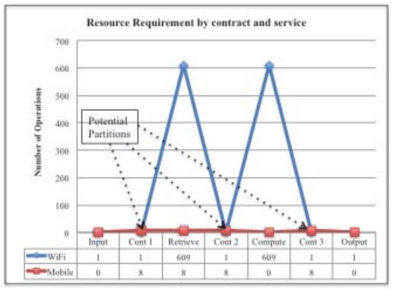


Fig. 5: Hypothetical distribution of load in AppleApp.

5.2 Experiment output when triangulated to 100 currencies

The experiment was scaled to evaluate 100 currencies for triangulation. The application was able to be modularized and each module measured according to the attributes of energy, CPU and IO. The IO and computational characteristics of the application volumes and metrics were captured through the use of the Apple Instruments and key results are shown in Figure 6.

The above instruments quantify impacts as follows:

1) IO Activity: captures information about I/O events.
2) Time Profiler: usage of CPU with time-based sampling of the application.
3) Activity Monitor: traces the overall system load.

IO activity is significantly impacted when the queries to the web are executed by the retrieve service and also when the calculations are executed by the calculate service. The

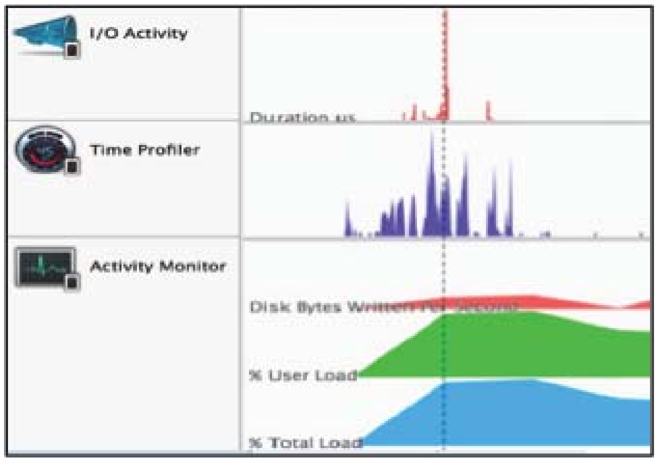


Fig. 6: Key Execution Characteristics.

activity monitor reveals that most of the load in the application is being driven by the application and no extraneous factors. The dotted line is the flag where the initialization of application finished and it also identifies the peaks among all three of the measurement instruments.

6. Architecture of Policy Services to assist in Migration Decision

The proposed architecture gathers user, contract, current state, forecasted usage and historical metrics to drive a decision of what to migrate.

6.1 Proposed services architecture

A services architecture was created to facilitate the decision of what to migrate. This architecture is illustrated in Figure 7 and consists of atomic services that assist in providing key decision variables to the centralized policy service. The policy service implements the migration decision and if it does allow for migration of all or parts, will allocate future capacity. Figure 7 also includes the MCC application.

6.2 Atomic services context

Table 2 illustrates the context of services that are proposed and the operations each will perform. The user preferences and provider contracts services are populated from input by the user. The device status is populated by querying the status ("general") App on the mobile. Finally, the allocated capacity and usage history services are maintained by the services themselves from prior decisions and trace files of the apps on the device.

These variables are used in the decision making process of the compound migration service.

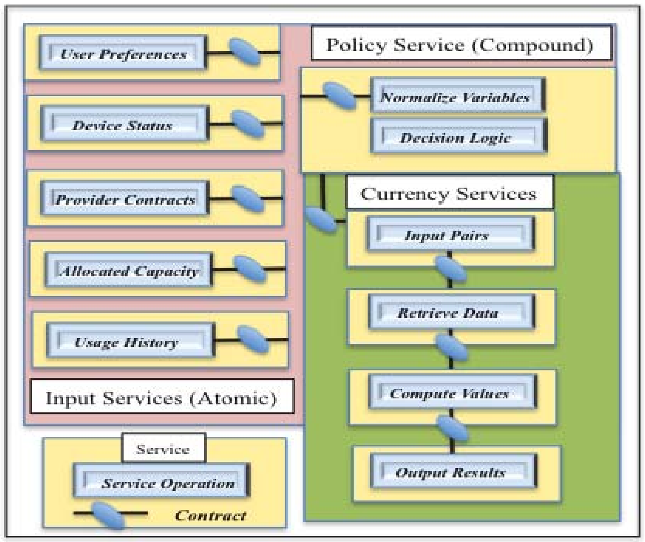


Fig. 7: Proposed Migration Service Architecture.

Table 2: Services and contexts.

Service	Context Provided
User Preferences	Tradeoffs and tolerances
Device Status	Usage, remaining resources
Provider Contracts	Mobile contract and Cloud contract
Allocated Capacity	Estimated usage for current tasks
Usage History	Prior executions and characteristics

6.3 Migration decision logic

The migration decision service logic operates on the following three equations:

$$\{\forall p \& \& \forall h : \exists w, \exists c \equiv u\} \quad (1)$$
$$\{\forall p \& \& \forall h : \exists m, \exists c \equiv u\} \quad (2)$$
$$\{\forall p : \nexists h\} \quad (3)$$

Where:
p=partition, h=history, w=wifi, u=user preferences, c=capacity, m=mobile spectrum

The decision to migrate the service is true in any of the above three equations. In equation 1 there exists a valid partition with prior migration history, there is also a WiFi spectrum and enough capacity (latency, battery/cycles) on the device to transfer to the Cloud for a round trip that is equal to the ranges defined in the user preferences. In equation 2 there is also a valid partition with prior migration history but there is a mobile spectrum with enough capacity (cost, latency, battery/cycles) on the device plans to transfer to the Cloud within tolerable ranges established in the user preferences. Finally in equation 3, there is a partition but no history and the decision is to migrate in order to begin the history creation process (assumes at least one execution will be needed to create the equation variables). In all other cases, the partition is executed on the device.

7. Discussion / Potential Risks

Perhaps the two most complex aspects of migration are what to migrate and if the migration should occur. The objective is always to minimize impact and maximize usage of low cost computing capacity. This document has offered a method for partitioning based on services contracts and a method for evaluation of the migration decision based on historical and environmental factors.

7.1 Potential risk: dynamic binding.

In most SOA environments, binding of services to processes is performed as the execution occurs. Commercial vendor offerings allow for modification of execution scripts on-the-fly while they are being executed. In our particular architecture, the services are bound and analyzed by the instruments and are not able to be modified without re-compilation. This potential limitation may inhibit flexibility similar to that offered by the traditional SOA Web Services which allows for changes up to the point of dynamic binding at execution.

7.2 Potential risk: broad applicability of BPEL.

The BPEL 2.0 standard has now been in existence almost 8 years; however, this is far from universal in adoption. The language is complete for the most common processes but still might need additional maturity in some of the less frequently used. In addition, some methodologies challenge its usage and contend that such a rigorous process may constrain delivery of code and functionality on a timely basis (Agile development).

Our objective in using BPEL was to provide an alternative to acyclic and semantic code analysis. Certainly other languages and methods exist to analyze required functionality and determine where the least amount of state data is required for transfer. The objective is to minimize that state transfer and potential latency while conserving energy on the device.

8. CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

This document explored the usage of BPEL in partitioning applications for migration and executed an experiment to review if the service contract was a potentially efficient segment to do so. The data from the services in the experiment support that conclusion.

A second contribution was the definition of an architecture to store preferences, history and state information so that the decisions to migrate can be made under certain variables. That architecture is itself a service and is being modeled in apps on the environment described above. At the point of this writing, those applications are under development.

8.2 Future work

This study identifies the service contract as a potentially optimum location for partitioning applications. The migration decision however is not as clear-cut. The proposed service architecture for the composite service is in process of being implemented in the Apple SWIFT environment.

ACKNOWLEGEMENTS

This research was supported by the U.S. National Science Foundation (NSF CNS Award 1126747).

References

[1] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *Wireless Communications, IEEE*, vol. 20, no. 3, pp. 14–22, 2013.

[2] M. Felemban, S. Basalamah, and A. Ghafoor, "A distributed cloud architecture for mobile multimedia services," *Network, IEEE*, vol. 27, no. 5, pp. 20–27, 2013.

[3] www.finance.yahoo.com/news/research-markets-global-mobile cloud.

[4] X. Ma, Y. Zhao, L. Zhang, H. Wang, and L. Peng, "When mobile terminals meet the cloud: computation offloading as the bridge," *Network, IEEE*, vol. 27, no. 5, pp. 28–33, 2013.

[5] Y. Cai, F. R. Yu, and S. Bu, "Cloud computing meets mobile wireless communications in next generation cellular networks," *Network, IEEE*, vol. 28, no. 6, pp. 54–59, 2014.

[6] W. Zhang, Y. Wen, J. Wu, and H. Li, "Toward a unified elastic computing platform for smartphones with cloud support," *Network, IEEE*, vol. 27, no. 5, pp. 34–40, 2013.

[7] M. Shiraz, A. Gani, R. H. Khokhar, and R. Buyya, "A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 3, pp. 1294–1313, 2013.

[8] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek, "Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing," 2014.

[9] http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0 OS.html.

[10] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: taxonomy and open challenges," *Communications Surveys & Tutorials, IEEE*, vol. 16, no. 1, pp. 369–392, 2014.

[11] J. Cox, N. Jones, and J. Szumski, *Professional IOS Network Programming: Connecting the Enterprise to the IPhone and IPad*. John Wiley & Sons, 2012.

[12] M. Privat and R. Warner, *Pro Core Data for IOS: Data Access and Persistence Engine for IPhone, IPad, and IPod Touch*. Apress, 2011.

[13] T. Erl, *SOA Design Patterns with Foreword by Grady Booch; The Prentice Hall Service-Oriented Computing Series from Thomas Erl*, 6th ed. SOA Systems, Inc., Prentice Hall Publisher, 2009.

[14] T. Erl, C. Gee, P. R. Chelliah, J. Kress, H. Normann, B. Maier, L. Shuster, B. Trops, T. Winterberg, C. Utschig, *et al.*, *Next Generation SOA: A Concise Introduction to Service Technology & Service-Orientation*. Pearson Education, 2014.

[15] www.oracle.com/technetwork/articles/matjaz-bpel1 090575.html.

[16] B. G. Pitre, *Swift for Beginners: Develop and Design*. Pearson Education, 2014.

[17] A. Freeman, "Pro design patterns in swift," 2015.

[18] S. Azarpour, R. R. Cepeda, T. Coron, S. Davies, J. Gundersen, M. Katz, C. Lowe, V. Ngo, R. Nystrom, C. Rocchi, A. Tam, C. Wagner, N. Waynik, and J. Wu, *iOS 8 by Tutorials; Learning the New iOS8 APIs wih SWIFT*. Razerware, LLC, raywenderlich.com, 2014.

Level of Policy for Cloud Computing Adoption in Australian Regional Municipal Government: An Exploratory Study

Omar Ali
School of Management and Enterprise
University of Southern Queensland
Toowoomba-Australia
Omar.Ali@usq.edu.au

Jeffrey Soar
School of Management and Enterprise
University of Southern Queensland
Toowoomba-Australia
Jeffrey.Soar@usq.edu.au

Jianming Yong
School of Management and Enterprise
University of Southern Queensland
Toowoomba-Australia
Jianming.Yong@usq.edu.au

Hoda McClymont
School of Management and Enterprise
University of Southern Queensland
Toowoomba-Australia
Hoda.McClymont@usq.edu.au

***Abstract* - Technology plays an important role in helping organizations to maintain control and take advantage of opportunities in a highly competitive and increasingly complex business environment. Cloud computing offers access to computing power, storage, software, storage, and other services for customers and provides remote data centres through the web. This research aims to investigate the policy that required for the adoption of cloud computing. The research employed in-depth interviews with IT managers from selected local government councils in Australia. The research aims to contribute to specific policy in the adoption of cloud computing.**

Keywords: cloud computing; policies; adoption; local governments.

I. INTRODUCTION

Cloud computing can be defined as a modern method of combining previous technologies to produce a standard prototype to serve as a platform for users trying to access shared and configurable computing resources through the internet (Kuyoro et al. 2012; Nicho & Hendy 2013). Networks, storage, servers, application and software are some of the included resources that can be established relatively faster and requires minimum management or supervision from the services providers (Kuyoro et al. 2012). The advent of cloud computing offers the potential of reducing expensive IT infrastructure (Nicho & Hendy 2013; Subashini & Kavitha 2011) and the high penetration of mobile technology and broadband Internet networks increases the extent of its availability (Gupta et al. 2013).

The government IT strategy identifies the importance of cloud computing for its readily accessible processes, storage, and communication (Paquette et al. 2010). Cloud services are now offered to individuals, companies, and government agencies (Paquette et al. 2010). Cloud provides computing resources as a service and can include software, hardware and platform (Paquette et al. 2010; Son & Lee 2011). Cloud has changed computer services from a product owned to a service offered to customers via the internet from large-scale data centres or clouds. It allows consumers to take advantage of these services at any time or place. An ever-present need for faster service delivery is expected to encourage organizations to increase demands for greater IT agility (Oliveira & Martins 2010).

As with any previous ICT innovation, businesses tend to consider guidelines and special issues from the research community regarding these new innovations. Issues such as security and privacy are the main concern for organisations regarding cloud computing services. Marston et al. (2011) highlighted the special issues in which these firms would be interested: (1) administrating and implementing a constant ICT policy across different cloud computing suppliers; (2) handling the relocation of a software subscription, which usually is the domain of the IT department, to individual departments that need the cloud application instances; and (3) initiating IT audit policies that fit with local, regional, national and international policies. According to Marston et al. (2011) there are many key players in the cloud industry, e.g. Google, Microsoft, IBM and AT&T; technology providers such as Apache and AMC; “innovators” such as Amazon and Salesforce; and “enablers” such as CapGemini and RightScale.

Despite its benefits, the adoption rate of cloud computing in Australian regional municipal government sectors has been lower compared to that in urban areas (IT Industry Innovation Council 2011). This lag in adoption could be due to factors to challenges and issues (IT Industry Innovation Council 2011; Department of Finance and Deregulation 2011). That is, various challenges and issues relating to security, privacy, and trust (Ali & Soar 2014; Buyya et al. 2011; Ghanam et al. 2012; Kim 2009; Takabi et al. 2010).

The paucity of empirical studies about the policies that need to be considered for the adoption of cloud computing in regional municipal governments has hindered understanding and strategy development to improve its adoption (IT Industry

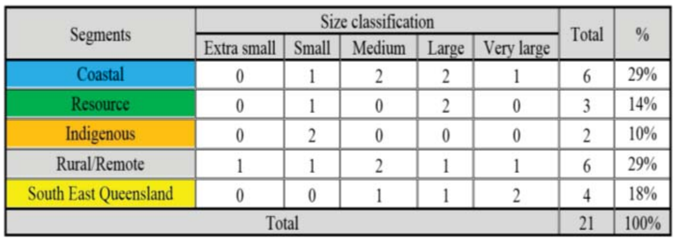
Innovation Council 2011). There are many policies need to be considered for the adoption of cloud computing which needs an attention and for that reason the Australian regional municipal governments have encouraged further research into this phenomenon (Department of Innovation Industry Science and Research 2011) to enhance the adoption rate. The research questions addressed in this paper relate to "what" policies need to be considered for the adoption of cloud computing in regional municipal governments and "what" the current level of policy is.

This research identifies and provides an overview about polices that need to be considered in the adoption of cloud computing in Australian regional municipal governments. The paper is structured: first, we provide an overview of cloud computing based on the extant literature. Next, we explain the methodology used to collect data for this research. This section involved in-depth interviews with IT managers in Australian local governments. Then, we conclude this paper with conclusion section.

II. Data Collection And Analysis

This research is exploratory in nature, seeking to provide a qualitative overview of the concepts with the highest salience relating to the factors and polices that need to be considered for the cloud computing adoption, and based on the encouragement by the Australian regional municipal governments. This research was intended to explore the significant factors that need to be considered for the adoption of cloud computing and identify the specific polices needed. A series of in-depth interviews were conducted between May 13, 2014 and August 12, 2014. These obtained inputs from 21 at top management levels: IT Manager (7); IT Coordinator (4); Technical Director (2); Information Service Manager (2); IT Officer (1); IT Consultant (1); IT Network Manager (1); Chief Information Officer (1); Enterprise Architecture Manager (1); and Team Leader ICT Operation (1). These occupational groups were selected based on the assumption that they represent key stakeholder groups likely to be responsible for planning and adoption of cloud computing for regional municipal governments.

The sample reflects the geographical spread and size classifications of local councils throughout Queensland (Coastal – 29%; Resource – 14%; Indigenous – 10%; Rural/Remote – 29%; South East Queensland – 18%).



Segments	Size classification					Total	%
	Extra small	Small	Medium	Large	Very large		
Coastal	0	1	2	2	1	6	29%
Resource	0	1	0	2	0	3	14%
Indigenous	0	2	0	0	0	2	10%
Rural/Remote	1	1	2	1	1	6	29%
South East Queensland	0	0	1	1	2	4	18%
Total						21	100%

TABLE 1: Size classification

To improve the reliability of this research, a model by Kirsch (2004) was followed; this model defines a set of procedures: Firstly, identify and select the research issues. Secondly, determine who to interview. Finally, determine how the interviews will be conducted.

An interview protocol was developed and used to guide the interview process. The interviewer followed a sequence of steps: Planning the interview, introductions at the commencement of the interview and establishing rapport with the respondent through small chat (Gaskell 2000). Ethical clearance was obtained through University Southern Queensland (USQ). Each interview was structured around four questions, with the interviewers asking probing questions based on responses. The questions required the participants to describe:

(1) Please describe your role in the field of IT/IS?
(2) What is your background, experience and knowledge in relation to cloud computing?
(3) How long have you been involved with cloud computing projects and in what capacity?
(4) What are the policies that need to be considered when focusing in the adoption of cloud computing?
(5) What is the current level of policy for adoption of cloud computing in the Australian regional and municipal government?

The interviews lasted between 30 and 50 minutes. The interview questions were designed as largely open questions to encourage the interviewees to provide answers that revealed their attitudes and perceptions relating to the research topic (Carson et al. 2001). Altogether, 24 interviews were carried out with IT managers of the chosen councils. The research reached the saturation level within the interview number 18, when the researchers noticed that, there is no more new information or patterns in the data emerging from the interview. Another six interviews were conducted to ensure inclusion of all segments and size classification of the councils to obtain a comprehensive overview of issues (refer to table 1). About 21 interviews were take a part in the analysis process, and the other 3 interviews excluded from the analysis process because it was discovered during the interview that these 3 IT managers did not coming from an IT background and did have not any experience or knowledge in relation to cloud computing.

The interviews data was analysed using manual content analysis method (Miles & Huberman 1984), and using Leximancer. Manual content analysis was undertaken as a first step in the analysis which included three concurrent flows of activities: data reduction, data display and conclusion drawing/verification (Faust 1982; Hsieh & Shannon 2005; Miles & Huberman 1984). After the completion of each interview session, the recorded interviews were immediately transcribed. Interview transcripts were reviewed to create summary sheets for every interview (Rao & Perry 2007). This summary sheet included main themes, issues, problems and brief answers to each question, resulting in an overall summary of the main points in the contact (Patton 2002; Schilling 2006). Then the summary sheets were reviewed to develop a pattern code for the research data. The next step of the analysis was to develop data display, which organised assembly of information to permit the researcher to draw conclusions and taken actions (Miles & Huberman 1984). Once manual coding was completed, the data was then reanalysed using Leximancer to improve the reliability of the findings (Middleton et al. 2011; Smith & Humphreys 2006).

Leximancer is a data mining tool that can be used to analyse the content of collections of textual documents and to visually display the extracted information (Smith 2003). It uses ontological relativity and dynamics to assemble bits of information to structure and evaluate concepts (Cummings & Daellenbach 2009). Words are combined to form concepts (thematic analysis) and identify relationships (semantic analysis) between concepts. A 'concept map' displays the main concepts in the text data, depicting the relationships through visual summaries of concepts and their co-occurrences – similar to a mind map (Cummings & Daellenbach 2009). Combined use of both manual and software analytical approaches provided a robust basis for clearly delineating concepts, themes and aggregate dimensions (Middleton et al. 2011; Smith & Humphreys 2006).

III. FINDINGS AND DISCUSSION

With the ability to energetically organize, grant, redesign servers to attend to an assortment of requirements, cloud computing is defined as a computing podium. Meanwhile, policy issues based on cloud computing are not being extensively conversed or recognized. On the other hand, it is escalating swiftly as a utility used worldwide by numerous organizations and many individuals. According to Jaeger et al. (2008), as cloud computing is being utilized far and wide, there has been a decline of policy creating in regard to the adoption of cloud computing as there is an extensive range of problems concerning cloud computing that require substantial concentration. Some researchers such as Delaney and Vara (2007); and Ma (2007) shed light on the range of policy issues that comprises of government inspection, confidentiality, communication aptitude and defence problems. On the subject of the stipulation and expansion of cloud computing, in relation to the aforementioned problems, there are noteworthy ambiguities about and apprehensions between industrial capability and public policy (Jaeger et al. 2008).

The main level of policy identified by the participants based on their knowledge and experience as IT managers is presented in Figure (1).

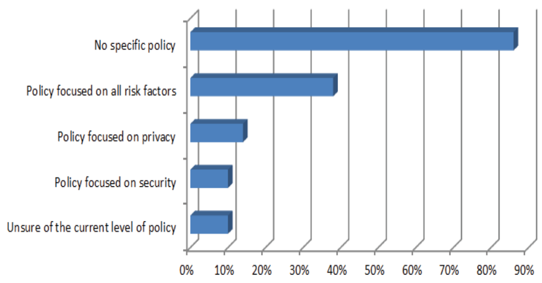


FIGURE 1: Level of policy for cloud computing adoption

This research investigated the current level of policy for cloud computing adoption in Australian regional municipal governments. This research found that there was no specific policy on adoption of cloud computing is new important finding that emerged upon the current level of policy for cloud computing adoption. Approximately 86 per cent (18/21) of the sample population reported that there was no specific policy for cloud computing adoption. Some commented that there was no national policy on cloud computing commented. The others mentioned that an information technology committee has been established to explore cloud computing "*we do not have any formal policy in place here. We do have an information technology steering committee that is involved and they oversight these types of changes*" (C15-RAL). Although there are no specific and formal policies currently being implemented, participants still explained that it can be a case to case basis for some organizations and their employment of the cloud model "*there is no specific policy that say we can or cannot doing. It is done based on case by case at this stage and also that depend on the actual solution that been offered*" (C61-URM).

Another finding was that the regulations are present and focused on all risk factors. Approximately 38 per cent (8/21) of the sample population commented that regulations are present and focused on all risk factors. Risk factors identified included the cost of the adoption, guarantee of quality of the services to the customers, and internet speed "*will at our regional council anyway there is no adopted policy by council to move IT infrastructure and services into the cloud it is more of a cost benefit analysis actually so it is rather than a dedicated policy*" (C42-URL).

The other finding that emerged was a regulation targeted on the privacy of the data and the users. Approximately 14 per cent (3/21) of the sample population observed that the government has placed and implemented a policy securing the privacy of the stakeholders "*as long as you can satisfy yourself that you are meeting your legislative obligations around data sovereignty, information privacy and so forth which you should reveal under any contract even if it is on shore*" (C68-URL). Another finding was that the regulations are present and focused on security. Approximately 10 per cent (2/21) of the sample population in this research reported that, there were rules for security of data and the stakeholders. Similar to the previous point mentioned, this particular rule is aimed to protect the data uploaded or stored through cloud "*there is been some clarification about the security level of data that the feds will store off-shore and state governments made some decisions there, but from a council perspective we are neither discouraged by senior, by policy or the other two levels of government or encouraged by the policy*" (C40-UDV). Finally, 10 per cent (2/21) of the sample population commented that they were not aware of any policy being implemented by their organization in regard to cloud computing technology. These participants reported that they were not aware any legislation pertaining to cloud computing in local government "*we do not know of any, or ever heard of policies or legislation here to say. Yes, it seems probably there are not any*" (C21-RTX).

Comparative Analysis

As stated in the methodology, the interview data was reanalysed using Leximancer to enhance the reliability of the findings from the manual content analysis (Middleton et al. 2011; Smith & Humphreys 2006). The first step it focused on the wide range of business-related words used by the respondents and identified from the exploratory Leximancer analysis. The second step for analysing the data was to examine the thematic groupings. Leximancer uses a natural language processing algorithm, so the theme is titled by the concept with highest prominence in the thematic aggregation. In this analysis, Leximancer clustered the concepts into seven themes (government, cloud, services, information, aware, down, sure), each theme aggregating two or more concepts and represented by labelled circles as they have been illustrated in Figure (2). Figure (2) illustrates the IT managers' views related to the level of policy for cloud computing adoption in regional municipal governments. This figure depicts the central theme within the map was 'government', and being strongly linked to the themes cloud, services, aware, and down. The dominate theme government has strong associations with most other concepts on the map. Government is multifaceted in its use: relating to councils, service, cost effective, stuff, strategy, data, and local. The concepts strategy, local, data, cost effective, and service are shown to be frequently occurring and strongly connected to the theme Government. Other themes illustrated but not connected to the theme 'government' include 'information' and 'sure'. The centrality of this theme provides a starting point for the research analysis.

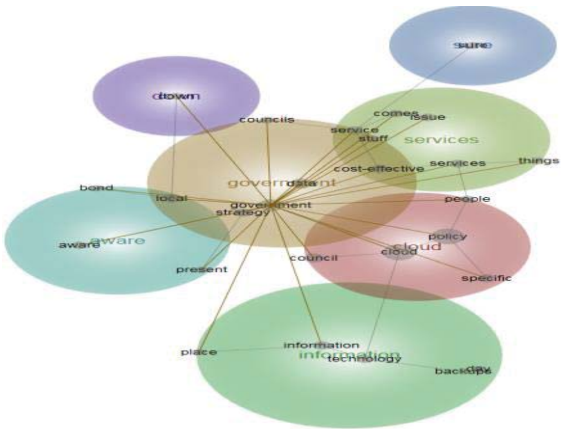


FIGURE 2: policy key concepts map

Each concept from the previously discussed concept map has been depicted in a bar chart as shown in Figure (3). It was demonstrated by the IT managers that the top six ranking concepts were strategy, local, data, cost effective, service, and councils.

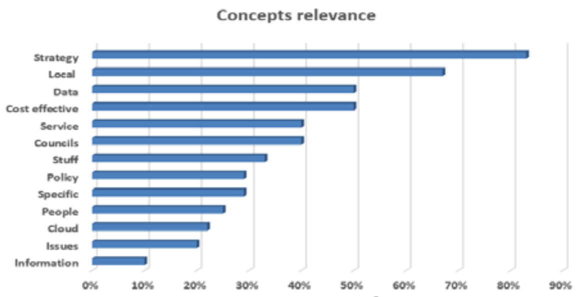


FIGURE 3: Concepts relevance

Because this research concentrated to find out the level of policy for the adoption of cloud computing in the regional and municipal governments, the theme 'cloud' which contains the concept 'policy' links strongly to the findings within the manual content analysis that suggested that IT managers saw the current level of policy on cloud computing adoption as having an impact on their organizational unit (See Figure 4).

The concept 'policy' and it's linkages on the concept map, through the analysis, have been illustrated through Figure (4). This concept is linked to all other concepts on the map. These linkages are to be expected with 'policy' being the top ranking concept. The strongest linkages shown in Figure (4) are: (a) between *policy* and *specific*, (b) between *policy* and *stuff*, (c) between *policy* and *issues*, (d) between *policy* and *government*, (e) between *policy* and *data*, (f) between *policy* and *service*, (g) between *policy* and *council*, (h) between *policy* and *information*, (i) between *policy* and *cost effective*, (j) between *policy* and *people*. These strengths are expected due to the focus of the research study and the qualitative questions asked, which were related to the current policy for the cloud computing adoption.

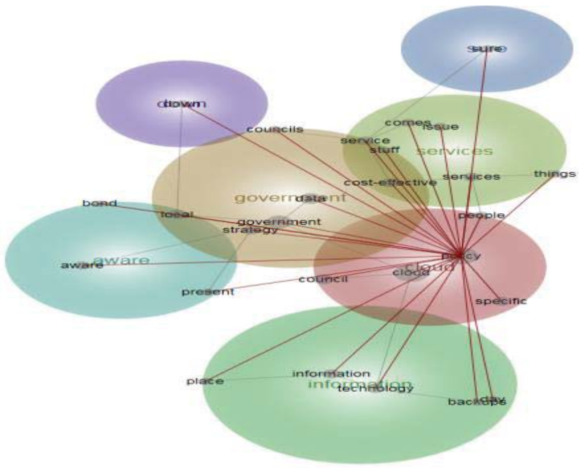
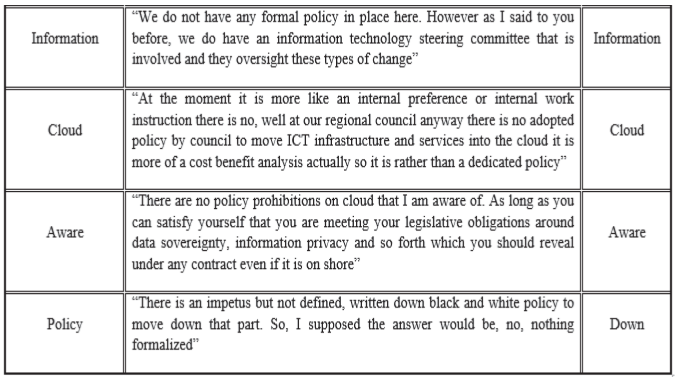


FIGURE 4: Policy and related linkages

When discussing the concept 'policy' the IT managers were referring to the impact of councils, government, strategy, specific, issues, information, cloud, aware, and policy. In relation to the mentioned concepts Table (2) illustrate the representative quotes of each concepts.

Leximancer-derived concepts	Representative quotes	Themes
Councils	"There is not one that covers local government in general. Each council would have their own approach for policy and generally it won?"	Government
Government	"The other issue from the government perspective is also how you guarantee a service level when you are relying on the internet. Because the internet, nobody really controls the internet"	Government
Strategy	"There is no policy, it might be more of a strategy which would be along the lines of what ways which is as opportunities arise, look at cloud-based solutions and if they are cost-effective to implement"	Government
Specific	"There is no specific policy that says we can or cannot do. It is done based on case by case at this stage"	Cloud
Issue	"Well the policy is, I mean, probably the major issue around the policy is around the storing of data off-shore"	Services



Information	"We do not have any formal policy in place here. However as I said to you before, we do have an information technology steering committee that is involved and they oversight these types of change"	Information
Cloud	"At the moment it is more like an internal preference or internal work instruction there is no, well at our regional council anyway there is no adopted policy by council to move ICT infrastructure and services into the cloud it is more of a cost benefit analysis actually so it is rather than a dedicated policy"	Cloud
Aware	"There are no policy prohibitions on cloud that I am aware of. As long as you can satisfy yourself that you are meeting your legislative obligations around data sovereignty, information privacy and so forth which you should reveal under any contract even if it is on shore"	Aware
Policy	"There is an impetus but not defined, written down black and white policy to move down that part. So, I supposed the answer would be, no, nothing formalized"	Down

TABLE 2: The representative quotes of each concept

After having a comparison between the results from Leximancer and the manual analysis, it was found by the researchers that both the methods gave the same result in a relation to policy for the adoption of cloud computing.

As cloud computing becomes extra widespread platform, it is likely to increase more issues concerning policy. According to Singh et al. (2006), it will be essential to locate technology and policy elucidations for adopting guidance that will guarantee the confidentiality and information precautions, to certify the adoption and escalation of cloud computing. There is an obligation to expand policies, rules or directives to envelop cloud computing adoption that will prove cooperative in cataloguing the ambiguities and apprehensions of contributor and associations regarding cloud computing and on the basis of the outcome of this segment, the above mentioned is suggested by the study. Nonetheless, the lack of supervision acts as a hindrance for the development of potential and user execution of cloud computing, with the existing deficiency of policies regarding the execution of cloud computing.

Instruction by an elected neighbourhood, state and national regime division or constitution authorizing better accuracy in service accord amid the contributors and associations are the two methods that can be taken under deliberation both in individual or concomitantly, based on the prior discussion. As stated by Best et al. (2005); Jaegar (2007); and Carrico and Smalldon (2004) that for the subsequent purposes, either of these methods would have the objective of ensuring the acknowledged principles of cloud computing.

- Fundamental thresholds for authenticity
- Obligation of accountability for failure or other infringement of the data.
- Opportunities for information sanctuary
- Confidentiality protection
- Any possible prospects for secrecy
- Admittance and consumption privileges
- International equivalence to encourage trans-border information streaming in clouds.

Through directives or formation of constraints for upcoming service accords, the particular rudiments of the abovementioned principles can be recognized. These issues will be chief elements to forge user confidence in cloud computing and deal with the policy issues related to it regardless of which method is assumed.

IV. CONCLUSION

Government organizations have started looking for new options for interacting with other organizations as well as individual citizens (Wyld 2010). Cloud computing has potential to significantly change the roles of IT departments in business and government sectors because of its potential benefits.

The main concerns about the policies that significantly need to be considered for the adoption of cloud computing services are concentrating on security, privacy and risk factors such as quality of services and the cost of services as well. Also, it appears that most of the local government councils lack policy for the adoption of cloud computing services. Local government councils need to develop transparent guidelines on the adoption of cloud computing with respect to these factors.

Compliance with Ethical Standards

The authors have no known potential conflicts of interest (financial or non-financial) with the research. Informed consent was obtained from participants and a Participant Information Sheet was provided. Participation was voluntary, there were no consequences for non-participation, no personally-identifying data was captured and data was anonymised. Authors whose names appear on this paper have contributed sufficiently to the and therefore share collective responsibility and accountability for the results. This research was funded through an Australian Post-graduate awarded at the University of Southern Queensland. The research was approved by the USQ Ethics Committee and was performed in accordance with the ethical standards as laid down in the 1964 Declaration of Helsinki and its later amendments or comparable ethical standards.

REFERENCE

[1] O. S. Kuyoro, A. A. Omotunde, C. Ajaegbu, and F. Ibikunle, "Towards building a secure cloud computing environment", *International Journal of Advanced Research in Computer Science*, vol. 3, no. 4, 2012, pp. 166-171.

[2] M. Nicho, and M. Hendy, "Dimensions of security threats in cloud computing: a case study", *Review of Business Information Systems*, vol. 17, no. 4, 2013, pp. 160-170.

[3] N. Subashini, and V. Kavitha, "A survey on security issues in service delivery models of cloud computing", *Journal of Network and Computer Applications*, vol. 34, 2011, pp. 1-11.

[4] P. Gupta, A. Seetharaman, and J. R. Raj, "The usage and adoption of cloud computing by small and medium businesses", *International Journal of Information Management*, vol. 33, no. 4, 2013, pp. 861-874.

[5] S. Paquette, P. T. Jaeger, and S. C. Wilson, "Identifying the security risks associated with governmental use of cloud computing", *Government Information Quarterly*, vol. 27, no. 3, 2010, pp. 245-253.

[6] I. Son, and D. Lee, "Assessing a New IT Service Model", *Cloud Computing*, 2011.

[7] T. Oliveira, and M. F. Martins, "Understanding e-business adoption across industries in European countries", *Industrial Management and Data Systems*, vol. 110, no. 9, 2010, pp. 1337-1354.

[8] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing: The business perspective", *Decision Support Systems*, vol. 51, no. 1, 2011, pp. 176-189.

[9] IT Industry Innovation Council, Cloud Computing-Opportunities and Challenges, 2011, pp. 1-31.

[10] Department of Finance and Deregulation, Cloud Computing Strategic Direction Paper, 2011, pp. 1-45.

[11] O. Ali, and J. Soar, “Challenges and issues within cloud computing technology”, *The 5th International Conference on Cloud Computing, GRIDs, and Virtualization*, 2014, pp. 55-63.

[12] R. Buyya, J. Broberg, and A. Gościński, Cloud Computing: Principles and Paradigms. John Wiley and Sons, 2011.

[13] Y. Ghanam, J. Ferreira, and F. Maurer, “Emerging issues and challenges in cloud computing - a hybrid approach”, *Journal of Software Engineering and Applications*, 2012, pp. 923-937.

[14] W. Kim, “Cloud computing: today and tomorrow”, *Journal of Object Technology*, vol. 8, no. 1, 2009, pp. 65-72.

[15] H. Takabi, J. Joshi, and G. Ahn, “Security and privacy challenges in cloud computing environments”, *Proceeding in the IEEE Security and Privacy*, vol. 8, no. 6, 2010, pp. 24-31.

[16] Department on Innovation Industry Science and Research, Cloud Computing - Opportunities and Challenges, IT Industry Innovation Council, 2011, pp. 1-31.

[17] L. J. Kirsch, “Deploying common systems globally: The dynamics of control”, *Information Systems Research*, vol. 15, no. 4, 2004, pp. 374-395.

[18] G. Gaskell, “Individual and group interviewing”, in M. Bauer, and G. Gaskell, (eds), Qualitative researching with text, image and sound, Sage, London, 2000.

[19] D. Carson, A. Gilmore, C. Perry, and K. Gronhaug, Qualitative Marketing Research, Sage Publications, London, 2001.

[20] M. B. Miles, and M. Huberman, Qualitative Data Analysis, Sage Publication, Newbury Park, 1948.

[21] D. Faust, “A needed component in prescription of science: Empirical knowledge of human cognitive limitations”, *Knowledge*, vol. 3, 1982, pp. 555-570.

[22] H. F. Hsieh, and S. E. Shannon, “Three approaches to qualitative content analyses”, *Qualitative Health Research*, vol.15, no. 9, 2005, pp. 1277-1288.

[23] S. Rao, and C. Perry, Convergent interviewing: A starting methodology for an enterprise research program, in D. Hine, and D. Carson, (eds), Innovative Methodologies in Enterprise Research, Edward Elgar, Northampton, Massachusetts, 2007.

[24] M. Q. Patton, Qualitative Research and Evaluation Methods, Thousand Oaks, CA: Sage Publications, 2002.

[25] J. Schilling, “On the pragmatics of qualitative assessment: Designing the process for content analysis”, *European Journal of Psychological Assessment*, vol. 22, no. 1, 2006, pp. 28-37.

[26] S. Middleton, P. W. Liesch, and J. Steen, “Organizing time: Internationalization narratives of executive managers”, *International Business Review*, vol. 20, no. 2, 2011, pp. 136-150.

[27] A. E. Smith, and M. S. Humphreys, “Evaluation of unsupervised semantic mapping of natural language with Leximancer concept mapping”, *Behaviour Research Methods*, vol. 38, no. 2, 2006, pp. 262-279.

[28] A. E. Smith, “Automatic extraction of semantic networks from text using Leximancer”, Paper presented at the North American Chapter of the Association for Computational Linguistics - Human Language Technologies, Edmonton, Canada, 2003.

[29] S. Cummings, and U. Daellenbach, “A guide to the future of strategy? The history of long range planning”, *Long Range Planning*, vol. 42, no. 2, 2009, pp. 234-263.

[30] P. T. Jaeger, J. Lin, and J. M. Grimes, “Cloud computing and information policy: Computing in a policy cloud?”, *Journal of Information Technology and Politics*, vol. 5, no. 3, 2008, pp. 269-283.

[31] K. J. Delaney, and V. Vara, “Google plans services to store users’ data’, Wall Street Journal, accessed on March 16, 2014, available at: http://online.wsj.com/article/SB119612660573504716.html?mod=hps_us_whats_news, 2007.

[32] W. Ma, “Google’s G drives raise privacy concerns’, Popular Mechanics, accessed on March 16, 2014, available at: http://www.popularmechanics.com/technology/industry/4234444.html, 2007.

[33] A. Singh, B. Gedik, and L. Liu, “Agyaat: Mutual anonymity over structured P2P networks”, Internet Research, vol. 16, no. 2, 2006, pp. 189-212.

[34] S. J. Best, B. S. Kreuger, and J. Ladewig, “The effect of risk perceptions on online political participatory decisions”, *Journal of Information Technology and Politics*, vol. 4, no. 1, 2005, pp. 5-17.

[35] P. T. Jaeger, “Information policy, information access, and democratic participation: The national and international implications of the Bush administration’s information politics”, *Government Information Quarterly*, vol. 24, 2007, pp. 840-859.

[36] J,. C. Carrico, and K. L. Smalldon, “Licensed to ILL: A beginning guide to negotiating e-resources licenses to permit resource sharing”, *Journal of Library Administration*, vol. 40, no. 1-2, 2004, pp. 41-54.

[37] D. C. Wyld, “The cloudy future of government IT: Cloud computing and the public sector around the world”, *International Journal of Web Semantic Technology*, vol. 1, no. 1, 2010, pp. 1-20.

New Protection of Kernel-level Digital Rights Management in Cloud-based Consumer Electronics Environments

Woei-Jiunn Tsaur[1] **and Lo-Yao Yeh**[2]
[1]Department of Information Management, Da-Yeh University, Changhua, Taiwan
[2]Network and Information Security Division, National Center for High-Performance Computing, Taichung, Taiwan

Abstract – *Controlling and managing rights of digital contents has been becoming very critical in cloud-based consumer entertainment devices. The kernel-level digital rights management (DRM) software can offer stronger protection of digital contents. For effectively preventing unauthorized copying, the rootkit stealth technologies may be employed in consumer electronics (CE) environments to conceal kernel-level DRM driver. Therefore, to stop unauthorized users from deleting the DRM software by employing anti-rootkit tools to remove the rootkit, this paper presents new rootkit-based stealth technologies to reinforce DRM driver in cloud-based CE environments. The proposed new driver-hidden rootkit stealth technologies can successfully evade detection and removal of well-known rootkit detectors in cloud-based CE environments. In summary, the contributions of this paper are that in cloud-based CE environments the proposed novel stealth technologies can be used to effectively reinforce the kernel-level DRM software for ensuring the rights of legitimate consumers and providing forceful protections for copyright owners.*

Keywords: Cloud service, consumer electronics, digital rights management, consumer security system, Linux.

1 Introduction

With the advent of new technologies such as cloud computing, it seems to cause problems of piracy even more than before. A huge amount of transactions of digital contents are expected under cloud computing environments. And the cloud computing environment is a place where many computers are available everywhere throughout the physical world connected seamlessly to the information systems. Therefore, it will be a more critical issue to control and manage rights of digital contents. Piracy and illegal distribution of digital contents are severe issues. Digital Rights Management (DRM) [1]-[6] aims at protecting digital contents from being abused through regulating the usage of digital contents. The DRM scheme is a digital protection technique that protects and manages the access rights of digital contents. It can prevent the confidential information of a digital content from unauthorized usages by illegal users. Although each DRM system may have its different DRM implementation and infrastructure, the basic DRM process is the same, which usually involves three parties: content provider, DRM technology provider (i.e. cloud server in cloud-based consumer electronics (CE) environments), and consumer device. The implementation of a DRM controller in the kernel of an operating system, instead of implementing it at the application layer potentially provides some great benefits [7]. The main advantage is that it should be possible for any application to access protected works, as the main underlying protection is provided by the operating system and not the individual applications. Furthermore, it also means that DRM protection can be offered to any data types, not just multimedia, and remain transparent to any user-space application trying to access the DRM protected contents. This means that DRM can be used as a privacy enhancing mechanism for consumers, who can determine the exact access control rules for their own private data.

As stated in the literature [8], [9], rootkit is a stealth technology, and the intent with which this technology is used determines their malicious or otherwise legitimate purpose. The same technology used by rootkits is also used in security software such as firewalls and host-based intrusion prevention systems to extend the protection of the operating system. Therefore, the rootkit technologies may be employed in CE devices to conceal the DRM software for preventing unauthorized copying [10]. However, Tsaur's scheme [10] cannot be adopted in cloud-based CE environments because his proposed approach did not completely consider the properties of virtual machines and operating systems in cloud-based CE environments. In addition, cloud environments offer particularly attractive malware targets as they incorporate vast numbers of computing resources and high network bandwidths, and are increasingly becoming the operational home to many high-valued software systems and services. Attacks targeting clouds can provide significant chances to obtain control over resources and extract proprietary information. Thus, how to control and manage rights of digital contents has been becoming very critical in cloud-based consumer electronics environments.

In order to prevent unauthorized users from removing the rootkit of concealing kernel-level DRM driver by employing anti-rootkit tools in cloud-based CE environments, this paper is to propose new driver-hidden rootkit stealth technologies for strengthening the DRM driver in protecting against the illegal distribution and consumption of copyrighted digital contents. In cloud-based environments, though many companies or organizations have been developing rootkit detectors to the public and undoubtedly they can detect known rootkits effectively, they cannot foresee what the result is when meeting unknown rootkits. In this paper, the proposed unknown rootkit stealth technologies are constructed in Linux-based cloud operating systems, and have been verified that it can successfully evade detection and removal of a variety of well-known rootkit detectors. The contributions of this paper are mainly twofold. Firstly, when consumers use their CE devices to connect to cloud servers, the proposed new rootkit stealth technologies can be employed to extend the protection of the kernel-level DRM driver in cloud-based consumer entertainment environments, which can be a great inspiration to DRM software makers to effectively ensure the rights of legitimate consumers and provide strong protections for copyright owners. Secondly, the stealth tricks of the proposed subtle rootkits can be a great inspiration to defenders who need to effectively strengthen the legitimate uses in cloud-based service environments.

The remainder of this paper is organized as follows. Section 2 surveys current rootkit stealth and detection technologies. Section 3 presents the system design for developing new rootkit stealth technologies in Linux-based cloud operating systems. Section 4 depicts the experimental results of testing the proposed rootkit's stealth ability in Linux-based cloud operating systems. Finally, some concluding remarks are included in the last section.

2 Related work

There are essentially two different technologies that a rootkit can use to hide computer resources. One is hooking that intercepts the requests of accessing resources. The other is Direct Kernel Object Manipulation (DKOM) that manipulates the data used by operating systems to keep track of resources. The oldest kernel mode rootkit [8] uses table hooking to alter System Service Descriptor Table (SSDT) to hide processes, drivers, files, etc. Although it is a simple, stable and efficient method, it is easily detected by current rootkit detectors [11]. Hunt and Brubacher [12] introduced Detours, a library for intercepting arbitrary binary function. Later, this method is also applied by a rootkit, which replaces the first few instructions of a specific function with "jump" to point to the rootkit's code instead of targeting system tables. The aforementioned is named inline hooking. But VICE [8], a heuristic detector, is created to detect hookers no matter table hooking or inline hooking. In order to enhance inline hooking, brilliant rootkit makers combine a polymorphic technique [13] whose purpose is to generate different appearances of a piece of code. These appearances may look different but have the same functionality. On the other hand, Butler *et al.* [14] used DKOM to target EPROCESS, a kind of kernel data structures to record information related to a process, to alter an affiliated doubly linked list, and let the desired processes be hidden. When using DKOM, rootkit makers need to clearly understand the data structures in kernel, but it is more furtive than using hooking [8], [15]. The DKOM technique was first used in the FU rootkit and then used in FUTo to hide their drivers [8], [9]. In 2007, the DKOM-based Unreal rootkit was created and shown off that all of the famous detectors cannot detect it. However, at present several well-known detectors are capable of effectively detecting the above-mentioned three driver-hidden rootkits.

As for identifying rootkits, there are two main approaches to develop rootkit detectors. The first detection approach targets hiding mechanisms by detecting the presence of API hooking [15], [16]. It is similar to the signature-based detection [17], and thus it cannot catch unknown rootkits whose signatures of hiding mechanisms do not exist in its signatures repository in advance. The second approach targets hiding behaviors by detecting any discrepancies between the original and the fake. It collects resources information from two different storage places, and then compares each other to find rootkits. This approach is to belong to the cross-view rootkit detection [18]. It is noticed that in this approach both targeted information cannot be modified simultaneously by rootkits, otherwise a detector using this approach cannot distinguish the differences between the two places storing targets and then it must be useless. Although this method has the drawback, it does not need to maintain a signature database as used in the signature-based detection method.

3 System design

In this section, new rootkit stealth techniques which have abilities to escape well-known anti-rootkit tools are proposed to hide the driver-format DRM software in effectively protecting against the illegal distribution and consumption of copyrighted digital contents. This paper discloses six places, some of which may not be known by anti-rootkit developers, to hide driver information in Linux-based cloud operating systems. The proposed new rootkit stealth technologies are composed of six tricks which will be presented in the following Items A-F, respectively.

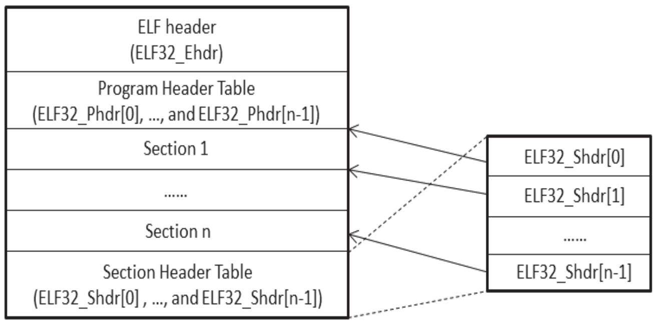


Fig. 1. The ELF structure [18].

A. Removing the signature of ELF (Executable and Linkable Format) image

The ELF file format is executable in operating systems environments and can be executed in multiple platforms. Almost all executable files (also including kernel mode drivers) are to use the ELF file format which inherits the characteristics of the COFF (Common Object File Format) file format in UNIX operating systems. The ELF file format contains four main parts [19]: ELF Header, Program Header

Table, a variety of Sections, and Section Header Table, as shown in Fig. 1. Driver programs are loaded to memory via initializing RAM disk system function, and therefore their complete ELF images exists in memory after loading. In order to avoid detectors to scan the ELF files, the characteristics of the ELF images must be removed. The method of removing the ELF images is to first load the ELF file to memory, and then find the list of the shared library link, including ELF32_Shdr[0] , …, and ELF32_Shdr[n-1] in Section Header Table as shown in Fig. 1. Afterwards, the corresponding symbol link and library in the file system can be found from ELF32_Phdr[0], …, and ELF32_Phdr[n-1] in Program Header Table, and then the value of ELF Header is set to zero.

B. Removing Object Drivers and Object Devices from Object Directory

In the internal of operating systems kernel, the most fascinating part is objects. They contain all kinds of resources that are queried by kernel functions. The program of object management is responsible to manage objects. All of them are kept in a tree of Object Directory whose definition is shown in Fig. 2. The Object Directory is established with several HashBuckets. Each one points to an Object Directory Entry structure whose definition is shown in Fig. 3. In Fig. 3, it can be found that the Object member refers to an object, and ChainLink member points to another Object Directory Entry. Most of Object Drivers have at least one Object Device pointing to themselves, so both of them are needed to consider when a driver-hidden rootkit is made. Thus, the purpose of hiding can be achieved by exploring the whole Object Directory to find the desired Object Drivers and Object Devices to hide.

C. Removing Object Drivers from driver's Object_Type

An Object_Type defines the common properties of the same kind of objects as shown in Fig. 4. Each kind of objects has a dedicated Object_Type. An Object_Type has a List_Entry data structure which keeps all of the same kind of Object_Creator_Info. The definition of Object_Creator_Info is described in Fig.5. Here the type of objects is referred to Object Driver whose definition is shown in Fig.6. And every object body is immediately preceded by an Object_Header structure whose Type member points to the Object_Type. Therefore, the loaded rootkit driver can be exploited to get its Object Driver, then move to its Object_Header to get the pointer to the Object_Type, and finally check its TypeList member to find the desired Object Drivers to hide.

D. Removing Object Devices from device's Object_Type

It is the same as the method described in Item C, except for different kind of objects. Here the type of objects is referred to Object Device whose definition is shown in Fig. 7. First, through the loaded rootkit Driver the Object Device can be found. After getting the Object Device, an Object_Type can also be found. Finally, its List_Entry is traversed to locate the desired Object Devices to hide.

```
typedef struct Object_Directory
{
/0x000/  [37] Ptr32 _Object_Directory_Entry     HashBuckets
/0x094/  _EX_PUSH_LOCK                           Lock
/0x098/  Ptr32_DEVICE_MAP                        DeviceMap
/0x09c/  Uin4B                                   SessionId
/0x0a0/  Uin2B                                   Reserved
/0x0a2/  Uin2B                     SymbolicLinkUsageCount
}
Object_Directory  * pObject_Directory
```

Fig. 2. Definition of an Object Directory.

```
typedef struct Object_Directory_Entry
{
/0x000/  Ptr32 _Object_Directory_Entry        ChainLink
/0x004/  Ptr32 Void                           Object
}
Object_Directory_Entry   * pObject_Directory_Entry
```

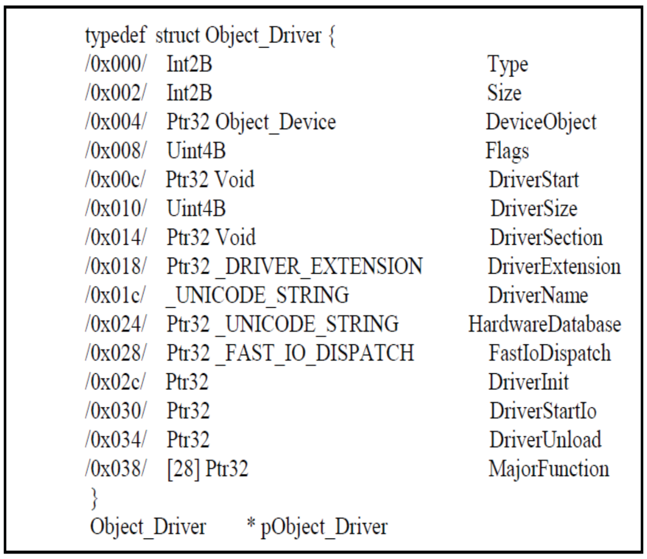
Fig. 3. Definition of an Object Directory Entry.

```
typedef struct Object_Type
{
/0x000/  _ERSOURCE                  Mutex
/0x038/  _List_Entry                TypeList
/0x040/  _UNICODE_STRING            Name
/0x048/  Ptr32 Void                 DefaultObject
/0x04c/  Uint4B                     Index
/0x050/  Uint4B                   TotalNumberOfObjects
/0x054/  Uint4B                   TotalNumberOfHandles
/0x058/  Uint4B                 HighWaterNumberOfObjects
/0x05c/  Uint4B                 HighWaterNumberOfHandles
/0x060/  _TYPE_INITIALIZER          TypeInfo
/0x0ac/  Uint4B                     Key
/0x0b0/  [4] _ERESOURCE             ObjectLocks
}
Object_Type     *pObject_Type
```

Fig. 4. Definition of an Object_Type.

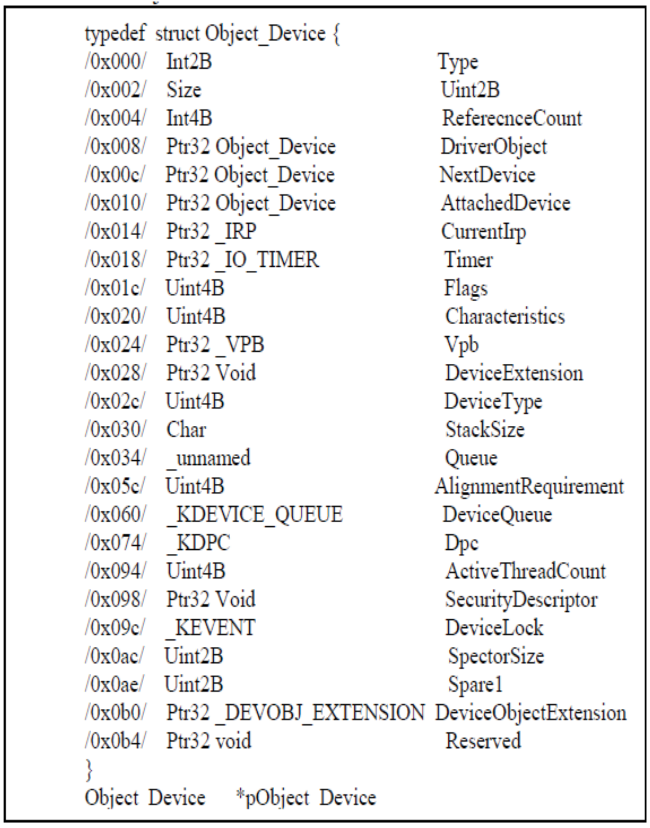
```
typedef struct Object_Creator_Info
{
/0x000/  _List_Entry                TypeList
/0x008/  Int4B                    CreateUniqueProcess
/0x00c/  Int4B                    CreateBackTraceIndex
/0x00e/  Ptr32 Void                 Reserved
}
Object_Creator_Info   * pObject_Creator_Info
```

Fig. 5. Definition of an Object_Creator_Info.



```
typedef  struct Object_Driver {
/0x000/   Int2B                          Type
/0x002/   Int2B                          Size
/0x004/   Ptr32 Object_Device            DeviceObject
/0x008/   Uint4B                         Flags
/0x00c/   Ptr32 Void                     DriverStart
/0x010/   Uint4B                         DriverSize
/0x014/   Ptr32 Void                     DriverSection
/0x018/   Ptr32 _DRIVER_EXTENSION        DriverExtension
/0x01c/   _UNICODE_STRING                DriverName
/0x024/   Ptr32 _UNICODE_STRING        HardwareDatabase
/0x028/   Ptr32 _FAST_IO_DISPATCH        FastIoDispatch
/0x02c/   Ptr32                          DriverInit
/0x030/   Ptr32                          DriverStartIo
/0x034/   Ptr32                          DriverUnload
/0x038/   [28] Ptr32                     MajorFunction
}
Object_Driver      * pObject_Driver
```

Fig. 6. Definition of an Object Driver.

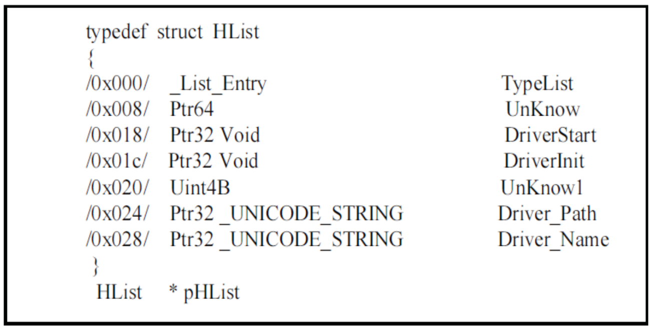


```
typedef  struct Object_Device {
/0x000/  Int2B                        Type
/0x002/  Size                         Uint2B
/0x004/  Int4B                        ReferecneCount
/0x008/  Ptr32 Object_Device          DriverObject
/0x00c/  Ptr32 Object_Device          NextDevice
/0x010/  Ptr32 Object_Device          AttachedDevice
/0x014/  Ptr32 _IRP                   CurrentIrp
/0x018/  Ptr32 _IO_TIMER              Timer
/0x01c/  Uint4B                       Flags
/0x020/  Uint4B                       Characteristics
/0x024/  Ptr32 _VPB                   Vpb
/0x028/  Ptr32 Void                   DeviceExtension
/0x02c/  Uint4B                       DeviceType
/0x030/  Char                         StackSize
/0x034/  _unnamed                     Queue
/0x05c/  Uint4B                      AlignmentRequirement
/0x060/  _KDEVICE_QUEUE               DeviceQueue
/0x074/  _KDPC                        Dpc
/0x094/  Uint4B                       ActiveThreadCount
/0x098/  Ptr32 Void                   SecurityDescriptor
/0x09c/  _KEVENT                      DeviceLock
/0x0ac/  Uint2B                       SpectorSize
/0x0ae/  Uint2B                       Spare1
/0x0b0/  Ptr32 _DEVOBJ_EXTENSION  DeviceObjectExtension
/0x0b4/  Ptr32 void                   Reserved
}
Object Device    *pObject Device
```

Fig. 7. Definition of an Object Device.

E. Removing drivers from Linux loadable kernel modules

A Linux loadable kernel modules (LKMs) structure whose definition is shown in Fig. 8 can be effectively employed to hide drivers. In this trick, the loaded rootkit driver can be checked to get its Object Driver's DriverSection, and further find and traverse LKMs to get the desired driver addresses to hide.



```
typedef  struct  HList
{
/0x000/   _List_Entry                  TypeList
/0x008/   Ptr64                        UnKnow
/0x018/   Ptr32 Void                   DriverStart
/0x01c/   Ptr32 Void                   DriverInit
/0x020/   Uint4B                       UnKnow1
/0x024/   Ptr32 _UNICODE_STRING        Driver_Path
/0x028/   Ptr32 _UNICODE_STRING        Driver_Name
}
HList    * pHList
```

Fig. 8. Definition of a Linux loadable kernel modules list.

F. Altering Object Driver's appearance

The targeted Object Driver appearance is modified to let it look different as compared to a normal one. This method tries to escape signature-based detectors. For example, if the value stored in the offset 0x000h of an Object Driver should be 0x04, then it can be altered with a random value to accomplish the purpose of stealth.

4 Experimental result and analysis

In Section 3, the six stealth technologies of the proposed new rootkit in Linux-based cloud operating systems have been depicted. In the following, the experimental results of testing the proposed rootkit's stealth ability in Linux-based cloud operating systems (Linux Mint 15 Cinnamon (Olivia)) are demonstrated by the following two phases: (1) rootkit loading operations, (2) test and analysis of rootkit stealth ability.

A. Rootkit loading operations

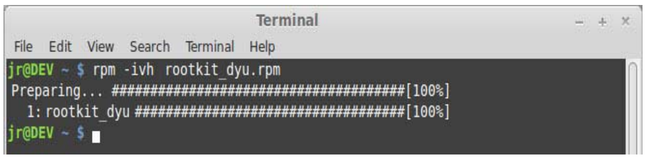
The proposed Linux-based rootkit named rootkit_dyu is a driver format and executed in a cloud computing service where multiple virtual machines are co-located on the same physical server. In such systems, physical resources are transparently shared by the virtual machines belonging to multiple users. For the rootkit loading operations, the proposed rootkit is installed but its stealth functionality is not invoked, as shown in Figs. 9 and 10. In Fig. 10, the rootkit driver name "rootkit_dyu" can be clearly found after it is loaded, and therefore it attests that the proposed rootkit is successfully loaded into the Linux-based cloud operating system.

B. Test and analysis of rootkit stealth ability

In order to provide greater flexibility in the testing process, the GUI (Graphical User Interface) interface is designed to test the stealth ability of the proposed rootkit, as shown in Fig. 11. In Fig. 11, since the six hiding techniques have been implemented in the proposed "rootkit_dyu" drivers, testers can check off their needs of the concealment mechanism to deploy a variety of different type of rootkit.

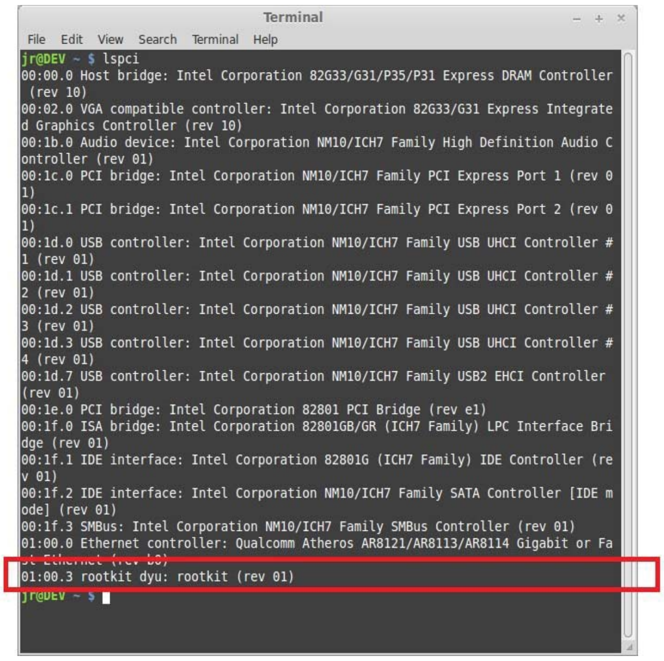
When the six stealth tricks of the proposed rootkit "rootkit_dyu" are checked off in Fig. 11, "rootkit_dyu" disappears in the Linux-based cloud operating system, as shown in Fig. 12, and thus proves that it have launched the hiding feature. As stated in the literature [8], [20]-[27], it can be found that a variety of detectors are highly effective for identifying rootkits. Therefore, the prestigious rootkit detectors introduced by the literature [8], [20]-[27] have been chosen to effectively test the stealth ability of the proposed rootkit driver. When the proposed rootkit is installed but its stealth functionality is not invoked, all of the detectors have listed it. When its stealth functionality has been invoked, all of the well-known detectors cannot detect the presence of the proposed rootkit in cloud computing environments.

It can be concluded that why all of the tested detectors cannot detect the proposed rootkit driver should be the following reasons. One is that some detectors cannot detect the rootkit with the abilities of removing the signature of ELF image. Another is that some detectors employ memory scan with predefined signatures, but they cannot recognize hidden Object Drivers with an abnormal object appearance. The other is that some detectors do not completely check whether the data structures of Object Directory, Object Driver, Object Device and LKMs may be modified, and thus the rootkit with the tricks of modifying the aforementioned data structures can avoid the heuristic-based detectors.



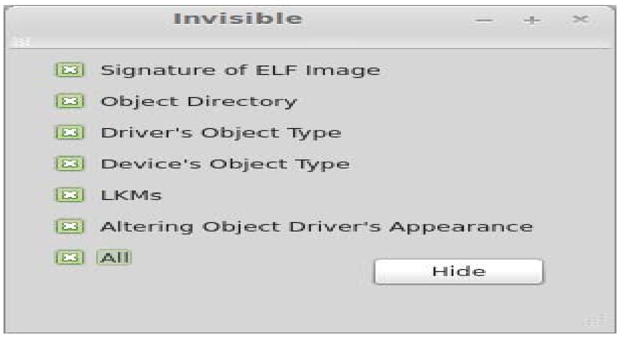
```
jr@DEV ~ $ rpm -ivh rootkit_dyu.rpm
Preparing... ##################################[100%]
   1:rootkit_dyu ##################################[100%]
jr@DEV ~ $
```

Fig. 9. The proposed rootkit "rootkit_dyu" is installed.



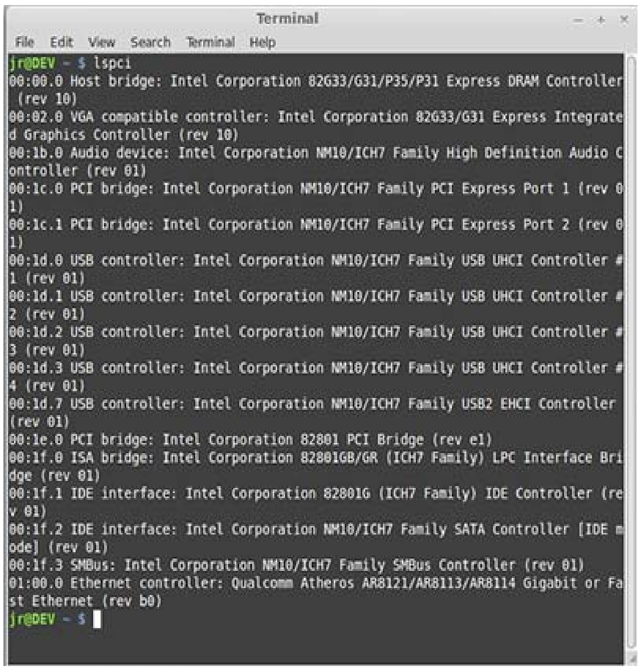
```
jr@DEV ~ $ lspci
00:00.0 Host bridge: Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller
 (rev 10)
00:02.0 VGA compatible controller: Intel Corporation 82G33/G31 Express Integrate
d Graphics Controller (rev 10)
00:1b.0 Audio device: Intel Corporation NM10/ICH7 Family High Definition Audio C
ontroller (rev 01)
00:1c.0 PCI bridge: Intel Corporation NM10/ICH7 Family PCI Express Port 1 (rev 0
1)
00:1c.1 PCI bridge: Intel Corporation NM10/ICH7 Family PCI Express Port 2 (rev 0
1)
00:1d.0 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
1 (rev 01)
00:1d.1 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
2 (rev 01)
00:1d.2 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
3 (rev 01)
00:1d.3 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
4 (rev 01)
00:1d.7 USB controller: Intel Corporation NM10/ICH7 Family USB2 EHCI Controller
(rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GB/GR (ICH7 Family) LPC Interface Bri
dge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801G (ICH7 Family) IDE Controller (re
v 01)
00:1f.2 IDE interface: Intel Corporation NM10/ICH7 Family SATA Controller [IDE m
ode] (rev 01)
00:1f.3 SMBus: Intel Corporation NM10/ICH7 Family SMBus Controller (rev 01)
01:00.0 Ethernet controller: Qualcomm Atheros AR8121/AR8113/AR8114 Gigabit or Fa
st Ethernet (rev b0)
01:00.3 rootkit dyu: rootkit (rev 01)
jr@DEV ~ $
```

Fig. 10. The stealth functionality of the proposed rootkit "rootkit_dyu" is not invoked, and therefore it is shown in the Linux-based cloud operating system.



- [✓] Signature of ELF Image
- [✓] Object Directory
- [✓] Driver's Object Type
- [✓] Device's Object Type
- [✓] LKMs
- [✓] Altering Object Driver's Appearance
- [✓] All

Fig. 11. The GUI for testing the stealth functionality.



```
jr@DEV ~ $ lspci
00:00.0 Host bridge: Intel Corporation 82G33/G31/P35/P31 Express DRAM Controller
 (rev 10)
00:02.0 VGA compatible controller: Intel Corporation 82G33/G31 Express Integrate
d Graphics Controller (rev 10)
00:1b.0 Audio device: Intel Corporation NM10/ICH7 Family High Definition Audio C
ontroller (rev 01)
00:1c.0 PCI bridge: Intel Corporation NM10/ICH7 Family PCI Express Port 1 (rev 0
1)
00:1c.1 PCI bridge: Intel Corporation NM10/ICH7 Family PCI Express Port 2 (rev 0
1)
00:1d.0 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
1 (rev 01)
00:1d.1 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
2 (rev 01)
00:1d.2 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
3 (rev 01)
00:1d.3 USB controller: Intel Corporation NM10/ICH7 Family USB UHCI Controller #
4 (rev 01)
00:1d.7 USB controller: Intel Corporation NM10/ICH7 Family USB2 EHCI Controller
(rev 01)
00:1e.0 PCI bridge: Intel Corporation 82801 PCI Bridge (rev e1)
00:1f.0 ISA bridge: Intel Corporation 82801GB/GR (ICH7 Family) LPC Interface Bri
dge (rev 01)
00:1f.1 IDE interface: Intel Corporation 82801G (ICH7 Family) IDE Controller (re
v 01)
00:1f.2 IDE interface: Intel Corporation NM10/ICH7 Family SATA Controller [IDE m
ode] (rev 01)
00:1f.3 SMBus: Intel Corporation NM10/ICH7 Family SMBus Controller (rev 01)
01:00.0 Ethernet controller: Qualcomm Atheros AR8121/AR8113/AR8114 Gigabit or Fa
st Ethernet (rev b0)
jr@DEV ~ $
```

Fig. 12. After the stealth functionality of the proposed rootkit "rootkit_dyu" is invoked, it disappears in the Linux-based cloud operating system.

5 Conclusions

In this paper, novel Linux-based rootkit stealth technologies are presented for enhancing kernel-level DRM driver in preventing the confidential information of digital contents from unauthorized usages by illegal users in cloud-based CE environments. The proposed new driver-hidden rootkit technologies executed on Linux-based cloud operating systems has successfully evaded the well-known anti-rootkit detectors, and thus can be effectively used to prevent unauthorized users from removing the rootkit of concealing the DRM driver by employing anti-rootkit tools in cloud-based CE environments.

To the best of the author's knowledge, there is no literature exploring the rootkit-based technologies for enhancing the kernel-level DRM driver in CE environments at present, so this paper is the first attempt to develop kernel-level DRM protection technologies against unauthorized usages of digital contents in cloud-based CE environments. The proposed technologies are valuable for extending the protection of DRM software, and can be a great inspiration to cloud-based DRM software makers to effectively improve the current techniques of defending against the illegal distribution and consumption of copyrighted digital contents. Furthermore, this study also inspires defenders to effectively strengthen the legitimate uses in cloud service environments by the proposed subtle hiding tricks.

Acknowledgement

This work was supported by the Ministry of Science and Technology of Republic of China under contract numbers NSC 101-2221-E-212-006-MY3 and MOST 103-2622-E-212-009-CC2.

References

[1] S. Lian, “Digital rights management for the home TV based on scalable video coding,” *IEEE Trans. Consumer Electron.*, vol. 54, no. 3, pp. 1287-1293, Aug. 2008.

[2] Y. Zou, T. Huang, W. Gao, and L. Huo, “H.264 video encryption scheme adaptive to DRM,” *IEEE Trans. Consumer Electron.*, vol. 52, no. 4, pp. 1289-1297, Nov. 2006.

[3] S. Park, J. Jeong and T. Kwon, “Contents distribution system based on MPEG-4 ISMACryp in IP set-top box environments,” *IEEE Trans. Consumer Electron.*, vol. 52, no. 2, pp. 660-668, May 2006.

[4] P. Zou, C. Wang, Z. Liu, and D. Bao, “Phosphor: A cloud based DRM scheme with sim card,” *in Proc. 12th International Asia-Pacific Web Conference*, Busan, Korea, pp. 459-463, Apr. 2010.

[5] Q. Huang, Z. Ma, Y. Yang, X. Niu, and J. Fu**,** “Attribute based DRM scheme with dynamic usage control in cloud computing**,**” *China Communications*, vol. 11, no. 4, pp. 50-63, Apr. 2014.

[6] E. Tsilichristou and D. Tsolis, “A P2P cultural multimedia network - maximizing cultural dissemination and supporting copyright protection and management,” *in Proc. 5th International Conference on Information, Intelligence, Systems and Applications*, Chania, Greece, pp. 406-411, Jul. 2014.

[7] A. Arnab, M. Paulse, D. Bennett, and A. Hutchison, “Experiences in implementing a kernel-level DRM controller,” *in Proc. Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, Barcelona, Spain, pp. 39-46, Nov. 2007.

[8] L. Stevenson and N. Altholz, *Rootkits for Dummies*, Wiley Publishing, 2007.

[9] E. U. Kumar, “Battle with the unseen — understanding rootkits,” *in Proc. 9th Association of anti-Virus Asia Researcher's Conference*, Auckland, New Zealand, pp. 82-97, Dec. 2006.

[10] W. Tsaur, “Strengthening digital rights management using a new driver-hidden rootkit,” *IEEE Trans. Consumer Electron.*, vol. 58, no. 2, pp. 479-483, May 2012.

[11] C. Keong, “Defeating kernel native API hookers by direct service dispatch table restoration,” *Technical Report*, SIG2 G-TEC Lab, Oct. 2004.

[12] G. Hunt and D. Brubacher, “Detours: binary interception of functions,” *in Proc. Third USENIX Symposium*, Seattle, USA, pp. 135-143, Jul. 1999.

[13] P. Beaucamps, “Advanced polymorphic techniques,” *International Journal of Computer Science*, vol. 2, no. 3, pp. 194-205, Sept. 2007.

[14] J. Bulter, J. L. Undercoffer and J. Pinkston, “Hidden process: the implication for intrusion detection,” *in Proc. 2003 IEEE International Workshop on Information Assurance*, West Point, USA, pp. 116-121, Jun. 2003.

[15] A. Baliga, L. Iftode, X. Chen, “Automated containment of rootkits attacks,” *Computers & Security*, vol. 27, no. 7-8, pp. 323-334, Dec. 2008.

[16] A. Baliga, P. Kamat and L. Iftode, “Lurking in the shadows: identifying systemic threats to kernel data,” *in Proc. 2007 IEEE Symposium on Security and Privacy*, Oakland, USA, pp. 246-251, May 2007.

[17] C. Kreibich and J. Crowcroft, “Honeycomb: creating intrusion detection signatures using honypots,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 51-56, Jan. 2004.

[18] N. L. Petroni Jr., T. Fraser, A. Walters and W. Arbaugh, “An architecture for specification-based detection of semantic integrity violations in kernel dynamic data,” *in Proc. 15th USENIX Security Symposium*, Vancouver, Canada, pp. 289-304, Aug. 2006.

[19] N. Kajtazovic, C. Preschern and C. Kreiner, “A component-based dynamic link support for safety-critical embedded systems,” *in Proc. 20th IEEE International Conference and Workshops on the Engineering of Computer Based Systems*, Scottsdale, USA, pp. 92-99, Apr. 2013.

[20] J. Rhee, R. Riley, D. Xu and X. Jiang, “Defeating dynamic data kernel rootkit attacks via VMM-based guest-transparent monitoring,” *in Proc. 4th International Conference on Availability, Reliability and Security*, Fukuoka, Japan, pp. 74-81, Mar. 2009.

[21] Y. Wen, J. Zhao, H. Wang, and J. Cao, “Implicit detection of hidden processes with a feather-weight hardware-assisted virtual machine monitor,” *in Proc. 13th Australasian Conference on Information Security and Privacy*, Wollongong, Australia, pp. 361-375, Jul. 2008.

[22] Y. Wen, J. Zhao, and H. Wang, “Implicit detection of hidden processes with a local-booted virtual machine,” *International Journal of Security and Its Applications*, vol. 2, no. 4, pp. 39-48, Dec. 2008.

[23] C. Xuan, J. Copeland and R. Beyah, “Shepherding loadable kernel modules through on-demand emulation,” *in Proc. 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Como, Italy, pp. 48–67, Jul. 2009.

[24] D. Lobo, P. Watters and X. Wu, "Identifying rootkit infections using data mining," *in Proc. 2010 International Conference on Information Science and Applications*, Seoul, Korea, pp. 1-7, Apr. 2010.

[25] D. Lobo, P. Watters and X. Wu, "RBACS: rootkit behavioral analysis and classification system," *in Proc. Third International Conference on Knowledge Discovery and Data Mining*, Phuket, Thailand, pp. 75-80, Jan. 2010.

[26] A. Baliga, V. Ganapathy and L. Iftode, "Detecting kernel-level rootkits using data structure invariants," *IEEE Trans. Dependable and Secure Computing*, vol. 8, no. 5, pp. 670-684, Sept. 2011.

[27] J. Li, Z. Wang, T. Bletsch, D. Srinivasan, M. Grace and X. Jiang, "Comprehensive and efficient protection of kernel control data," *IEEE Trans. Information Forensics and Security*, vol. 6, no. 4, pp. 1404 - 1417, Dec. 2011.

Distributed XML with Tag Shuffling in Cloud Computing

Behrooz Seyed-Abbassi and Jamie Gordon
School of Computing, University on North Florida, Jacksonville, Florida, USA

Abstract - *This research describes a process in which an XML document is analyzed for distribution across multiple cloud systems and partitioned based on the number of available clouds using a method for balanced distribution. The distribution of the document among the clouds could also utilize a tag shuffling technique to provide a more secure mechanism for XML data in cloud computing. To ensure that stored data will be accessible if a cloud is unavailable or hacked, a backup method has been devised to store a second copy of the XML partition in a different cloud. A mapping of elements to elements are created and stored as a kernel document to keep track of where documents and clouds in the system are located, where backup documents are located, and how tags have been shuffled. The proposed algorithm provides a method for sensitive documents to be stored and protected in cloud computing.*

Keywords: Cloud Computing, XML, Database, Distributed, Document

1 Introduction

As a versatile storage technology, XML has frequently been used for a standard data format to transfer information over the Internet [1]. The World Wide Web Consortium [2] group wrote the specifications for XML and published the standard definition in 1998. While originally designed as a simple markup language derived from the International Organization of Standardization (ISO) standard, the flexibility of the Standard Generalized Markup Language (SGML) format for large-scale online publishing has become popular in a variety of areas.

XML is said to be well formed, since it has a block structured format and support the following specifications [3].
1. Each document begins with a prolog containing an XML declaration, which specifies the XML standard being used.
2. There is one “root” element under which all other elements are nested.
3. For every starting tag, there is an ending tag with a matching case. XML is case sensitive.
4. All element tags are properly nested with none missing and none overlapping.

Another reason for its popularity is that all tags are user-defined, unlike HTML. The files can also have a well-defined structure through either a Document Type Definition (DTD) [4] or an XML Schema Definition (XSD) [5], [3]. This allows a user to define the types of tags that can be used. If a document conforms to a specified DTD or XSD, it is said to be valid. If two organizations use the same DTD or XSD, they know that each other’s XML documents are valid and will work with one another.

Various XML documents can take advantage of the XML query language, XQuery. The syntax for XQuery is often described as FLWOR or For, Let, Where, Order by, and Return [6]. XQuery uses the XPath syntax to find elements. XPath syntax is similar to a file path, where the items in the path describe the nesting patterns of elements.

XML documents have typically been seen as single document structures. Distributed XML documents, which may be called distributed trees, are documents which have been partitioned and sent to various nodes and are linked together to form a complete XML document [7]. The process can be accomplished through embedded function calls to the separate documents over a network from within a centralized node in the distributed system [7]. This document is a document that lies at the center of the distributed XML document and that brings together all the partitioned, distributed parts into a single document.

The kernel document is the central hub of the XML document. The other XML documents in the distributed system are connected through the leaf-nodes of the kernel document. The leaf nodes are called function-nodes, which connect to external XML documents through XML web services over a network. The activation of these function nodes causes the resource to be retrieved from an external service [7]. Figure 1 shows a centralized node (Main Computer) which houses a kernel document. This kernel document could contain links (through XML web services) to cloud servers to house the individual components of the distributed document.

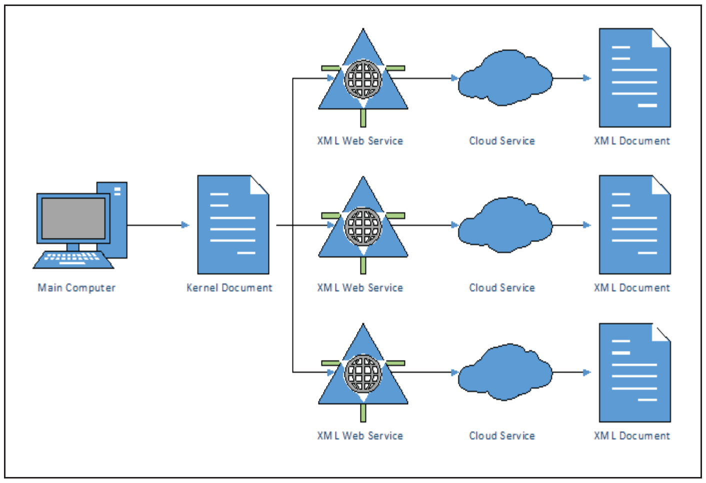


Figure 1. Distributed XML Document

Recently, work has been done on the topic of distributed XML[7], [8]. There are implementations that allow a distributed XML document to come together as a single document through querying the distributed system. The distributed system uses web service calls to create a document from a peer-to-peer architecture [9]. More of the research in this area has been on the topic of efficient query processing in the distributed environment. These methods seek to use the distributed nodes to perform tasks in parallel, rather than combining the document from the nodes and then running queries. Some authors involve not only efficiency in query processing, but also efficiency in data partitioning and distribution, as well as load balancing throughout the nodes of the distributed environment [8]. Others use partial evaluation to perform the queries on individual nodes [10].

Another topic of study has been the verification and validation of distributed XML documents. The authors of [7] were concerned with the creation of document specifications (in the form of DTD or XSD). They described two different design paradigms: bottom-up and top-down. In the bottom-up design, a specification is designed for each node in the distributed environment. Each of these specifications are then genericized into a global specification for the entire system. The top-down design creates a system-wide specification that is then specialized for each node in the system.

Security is another focus of study in the area of XML. For Guo [11], authentication is examined for individual elements of an XML document. XML authentication has been attempted through XML schemas, but the authors suggest creating authorization only on leaf nodes (where information is stored) and a privacy protection model resulting from separating the XML structure from its content.

2 Considerations for cloud XML

When designing the distributed system for XML, a number of factors must be taken into consideration.

1. Data partitioning for XML information should preserve the tree structure of the XML document and provide an even or as equally partitioned segments as possible. This means that the system must be made aware of both the data size of the XML document as well as the structure of the document itself. Knowing the size and shape of the XML document allows the system to load balance across clouds and to preserve the tree structure.
2. Once the data is partitioned into equal parts, the data must be split up to different locations for data distribution and there must be a way to recombine those parts into a centralized document [8]. That process is called materialization [7]. The data should be distributed in a way to partition the amount of data equally and to ensure load balance processing.
3. If the data becomes unbalanced in terms of either space or processing time, a third consideration of dynamic data relocation needs to be taken into consideration. When any single node becomes too large, is accessed frequently, changes frequently, or any other threshold is reached that affects its data size or processing cost, the data will need to be redistributed throughout the nodes so that the system becomes balanced again [8].
4. In distributed query processing, the considerations include where materialization occurs in the system, how the decomposed data is gathered, and how it is compiled into results [8]. This process is often handled by a third party tool, such as Active XML. Active XML is an extension to the XML framework that allows distribution of an XML document throughout a network and the recreation of the parts into one. This is done through embedded function calls that connect to external services to build an entire XML document from various sources [9]. For this paper's methodology, the kernel document will contain all of the embedded function calls.
5. The final consideration is the document specification that is done in either DTD or XSD and concerns the design of the entire distributed XML document in both the kernel and distributed clouds. It describes the rules of how the documents must be structured. The design can be done in one of two ways (bottom-up or top-down). The bottom-up method begins with designing a specification for each distributed cloud, and then combining and generalizing the specifications for the kernel node. The top-down method creates a global specification that is then specified for each distributed cloud. This research is concerned with the top-down design because each document is partitioned from an already known XML document thus global specification is available.

3 Storing XML documents in clouds

The proposed methodology consists of two main methods of distribution. The first method of distribution consists of taking a basic XML document and distributing it into equal parts based on the number of distribution cloud nodes available. The second method is more complex. In this method, all minimal subtrees are found. The load of each subtree is defined as the number of nodes in the subtree. An algorithm of least load is then used to group the subtrees into groups of near-equal total load.

An example of an XML document in a tree structure is shown in Figure 2. In the example, there are the two address elements, delivery notes, and the two item elements, excluding the deepest leaf nodes which would most likely not be distributed independently of one another.

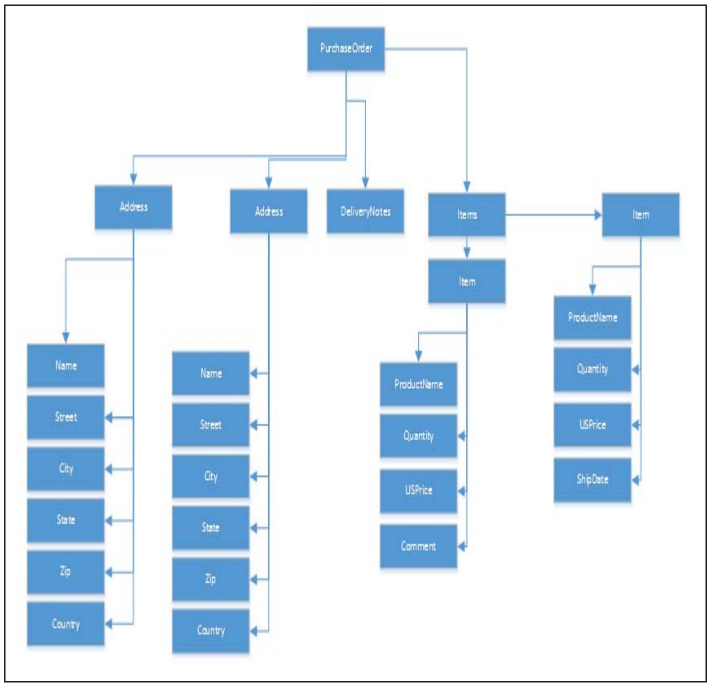


Figure 2. Tree Layout of Original Document

Because the tree structure needs to be preserved, sibling tree nodes must be kept together no matter where they are distributed. This means that both <Item> subtrees (when distributed) still must be contained underneath the <Items> element on order to preserve the original tree structure. Possible subtrees for the example are (Address, Address, Delivery Notes, Item, Item and Items (Item, Item)) excluding the lowest level leaf nodes and the tree as a whole. For subtree distribution, the tolerance factor needs to be calculated.

The tolerance of each group is decided as follows. The process is based on the space balancing component describe by [8].

1. A margin (ε) is chosen ahead of time.
2. The average group size (avg_{group}) is calculated by taking the total number of tree nodes (n_{nodes}) in the tree and dividing it by number of cloud nodes (n_{dist}) available.
3. The tolerance range (permissible subtree size range) is then calculated as
   $avg_{group} - \varepsilon \le range \le avg_{group} + \varepsilon$

The tolerance range gives the system a little leeway for deciding how to partition data for subtrees sizes that do not partition entirely evenly.

In the XML tree diagram given in Figure 2, there are 26 nodes available for distribution (the root node is not counted). Next, the tolerance margin needs to be chosen. The size of the smallest available subtree, "Delivery Notes", is used as the margin and there are four cloud services for document distribution. As a result, the average group size is 26 / 5 = 5.2, round up to 6, since 0.2 elements cannot be saved. The reason five is chosen instead of four is so that all available nodes are used, including the kernel document. This ensures load balancing system-wide. The tolerance range is between 6 - 1 and 6 + 1, or between 5 and 7. In summary: nnodes = 26, ndist = 5, margin = 1, avggroup = 26 / 5 = 5.2 |6| and tolerance range = 5 =< range <= 7.

4 Distribution algorithm

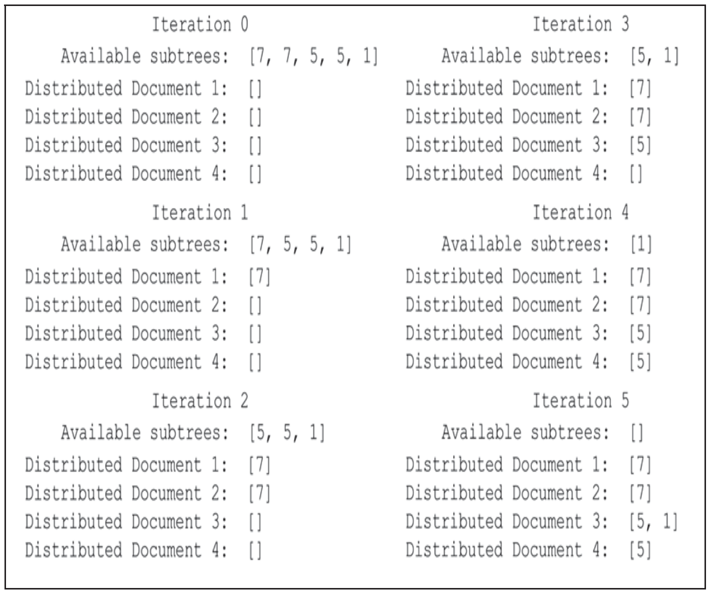
Once the tolerance range is decided, each of the subtrees are partitioned into groups. If a subtree is too large for the permissible size range, it is removed from consideration for distribution as long as there are subsets of that subtree available. This is done through the following algorithm. The algorithm splits each of the subtrees into parts based on which distribution node has the most space still available on each step (the "least load").

```
// array is an array of the subtrees' load sizes
split(array, num_groups)
    sort array descending and load_array = []
    // instantiate load_array with num_groups blank arrays
    for i = 0; i < num_groups; i++
        load_array[i] = []
    for each item in array
// least_load is a method that returns the array in
// load_array with the smallest load sum
        min = least_load(load_array) and min.add(item)
    return load_array
```

Once the load is determined, each subtree can be stored in a cloud. The kernel document stores which cloud service was given which partitioned document.

Figure 3 shows the distribution of the documents. Based on the algorithm, the list of their loads is sorted in descending order of 11, 7, 7, 5, 5, 1. The subtree corresponding to the size 11 (the "Items" subtree) is already outside the permissible range and can be discarded. Its contents are already covered by the two "Item" subtrees. That leaves the stack [7, 7, 5, 5, 1]. There are also four distributed sets (clouds) to work out.

Whatever elements are left out of these sets will be kept in the kernel document, such as “Items”.



```
          Iteration 0                               Iteration 3
    Available subtrees:  [7, 7, 5, 5, 1]      Available subtrees:  [5, 1]
Distributed Document 1:  []               Distributed Document 1:  [7]
Distributed Document 2:  []               Distributed Document 2:  [7]
Distributed Document 3:  []               Distributed Document 3:  [5]
Distributed Document 4:  []               Distributed Document 4:  []
          Iteration 1                               Iteration 4
    Available subtrees:  [7, 5, 5, 1]         Available subtrees:  [1]
Distributed Document 1:  [7]              Distributed Document 1:  [7]
Distributed Document 2:  []               Distributed Document 2:  [7]
Distributed Document 3:  []               Distributed Document 3:  [5]
Distributed Document 4:  []               Distributed Document 4:  [5]
          Iteration 2                               Iteration 5
    Available subtrees:  [5, 5, 1]            Available subtrees:  []
Distributed Document 1:  [7]              Distributed Document 1:  [7]
Distributed Document 2:  [7]              Distributed Document 2:  [7]
Distributed Document 3:  []               Distributed Document 3:  [5, 1]
Distributed Document 4:  []               Distributed Document 4:  [5]
```

Figure 3. Example Run of Algorithm

From the start, all distributed documents have their maximum tolerance set to 7 (which is shown in parentheses).

1. On the first iteration, the document of least load is chosen to receive the first subtree in the stack. Because all documents have the same load, the first document is given the subtree with the load of 7. This means that Distributed Document 1(Cloud 1) cannot receive any more subtrees.
2. On the second iteration, the same thing happens with Distributed Document 2 (Cloud 2). It is the next document with least load. It can no longer receive any more subtrees.
3. On the third iteration, Distributed Document 3 (Cloud 3) is given a subtree of load 5, giving it 2 remaining load and putting it within the tolerance range. The same thing happens with the fourth iteration and Document 4 (Cloud 4).
4. In the final iteration, the document with least load, Distributed Document 3 (Cloud 3) is given the subtree of load 1 and all subtrees.

The kernel document will be given links to all these distributed items and any remaining elements, in this case the <Item> element. Subtrees that are within the same external document are identified using an “element” attribute so they can be found and inserted back into the kernel document independently, such as for <Delivery Notes> and one of the <Item> elements. The example is continued below with the original document followed by the kernel document. The kernel document holds a links to each of the sources located where the documents contained in each source should go in order to form the original document.

Original Document

<PurchaseOrder number="99503" date="1999-10-20"> <AddressType="Shipping"> <Name>Ellen Adams </Name> <Street>123 Maple Street</Street> <City>Mill Valley</City> <State>CA</State> <Zip>10999</Zip> <Country>USA</Country> </Address> <Address Type="Billing"> <Name>Tai Yee</Name> <Street>8 Oak Avenue</Street> <City>Old Town</City> <State>PA</State> <Zip>95819</Zip> <Country>USA</Country> </Address> <DeliveryNotes> Please leave packages in shed by driveway. </DeliveryNotes> <Items> <ItemPartNumber="872-AA"> <ProductName>Lawnmower</ProductName> <Quantity>1</Quantity><USPrice>148.95</USPrice> <Comment>Confirm this is electric</Comment> </Item> <ItemPartNumber="926-AA"> <ProductName>Baby Monitor</ProductName> <Quantity>2</Quantity><USPrice>39.98</USPrice> <ShipDate>1999-05-21</ShipDate></Item> </Items> </PurchaseOrder>

Kernel Document

<kernel> <PurchaseOrder number="99503" date="1999-10-20"> <external> <!-- connection to cloud 1 --> </external> <external> <!-- connection to cloud 2 --> </external> <external element=”DeliveryNotes”> <!-- connection to cloud 3 --></external> <Items> <external element=”Item”><!-- connection to cloud 3 --> </external> <external> <!-- connection to cloud 4 --> </external> </Items> </PurchaseOrder> </kernel>

External Document 1

<document1> <Address Type="Shipping"> <Name>Ellen Adams</Name> <Street>123 Maple Street </Street> <City>Mill Valley</City> <State>CA</State> <Zip>10999</Zip> <Country>USA</Country> </Address> </document1>

External Document 2

<document2> <Address Type="Billing"><Name>Tai Yee </Name><Street>8 Oak Avenue</Street> <City>Old Town </City><State>PA</State><Zip>95819</Zip><Country>USA </Country> </Address> </document2>

External Document 3

<document3> <Item PartNumber="872-AA"><ProductName> Lawnmower</ProductName><Quantity>1</Quantity> <USPrice>148.95</USPrice> <Comment>Confirm this is electric </Comment> </Item> <DeliveryNotes> Please leave packages in shed by driveway.</DeliveryNotes> </document3>

External Document 4

<document4> <Item PartNumber="926-AA"><ProductName> Baby Monitor</ProductName><Quantity>2</Quantity> <USPrice> 39.98</USPrice><ShipDate>1999-05-21 </ShipDate> </Item> </document4>

5 Redundant distribution for security

Once the document distribution is complete, the distribution repeats again. This time, each of the clouds is given the data of one of the other cloud servers. This redundancy accomplishes the task of allowing data to be recoverable if one of the clouds is unavailable due to electrical issues, being compromised, hacking, or any other problem. Figure 4 shows this feature in action.

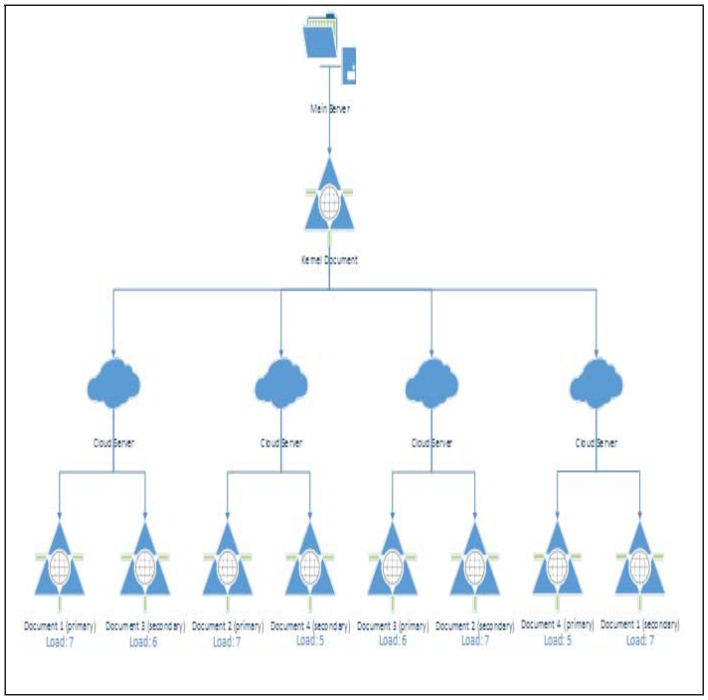

Main Server
Kernel Document
Cloud Server
Cloud Server
Cloud Server
Cloud Server
Document 1 (primary) Load:7
Document 3 (secondary) Load:6
Document 2 (primary) Load:7
Document 4 (secondary) Load:5
Document 3 (primary) Load:6
Document 2 (secondary) Load:7
Document 4 (primary) Load: 5
Document 1 (secondary) Load: 7

Figure 4. Distribution with Redundancy

The kernel document now stores two sources for each partition, a primary and secondary (backup) cloud source. If the primary source is ever compromised, the secondary source will be used. To get to the above distribution, the same algorithm as for the initial distribution is performed. However, for this distribution, the loads of subtrees are no longer used, but the loads of the clouds themselves. To decide how to distribute these documents again, the algorithm of least load again is used but with the array: [7, 7, 6, 5] to describe the loads of each of the clouds. Unlike the previous example, each of the cloud arrays need to be initialized to their previous load. The design that now stores that information can come from many sources. The kernel document is the only place where this is stored in the system. The kernel document now has nested <source> tags underneath the <external> tags. Each of the cloud services is given a rank. The sources are accessed in rank order. The lower rank will be called before the higher ranks unless something has gone wrong with the primary rank. Ideally, the distributed parts stored with each other on the cloud servers should be as unrelated as possible.

Kernel Document

```
<kernel>
      <PurchaseOrder number="99503" date="1999-10-20">
      <external name="document1"> <source name="cloud1" rank=1>
       <!-- connection to cloud 1 --> </source>
      <source name="cloud3" rank=2> <!-- connection to cloud 3 -->
      </source> </external>
      <external name="document2"> <source name="cloud2" rank=1>
      <!-- connection to cloud 2 --> </source>
      <source name="cloud4" rank=2> <!-- connection to cloud 4 -->
      </source> </external>
      <external name="document3" element="DeliveryNotes">
      <source name="cloud3" rank=1> <!-- connection to cloud 3 -->
      </source>
      <source name="cloud2" rank=2> <!-- connection to cloud 2 -->
      </source> </external>
      <Items> <external name="document3" element="Item">
      <source name="cloud3" rank=1> <!-- connection to cloud 3 -->
      </source>
      <source name="cloud2" rank=2> <!-- connection to cloud 2 -->
      </source> </external>
      <external name="document4"> <source name="cloud4" rank=1>
      <!-- connection to cloud 4 --> </source>
      <source name="cloud1" rank=2> <!-- connection to cloud 1 -->
      </source> </external> </Items>  </PurchaseOrder>
</kernel>
```

Each external tag now specifies which document on the cloud it references, because there are now multiple documents on each cloud services. The difference is that the source is the physical location in the cloud, i.e. each cloud service and the external document is the distributed portion of the XML document. Each source is given multiple documents, and each document is sent to multiple sources.

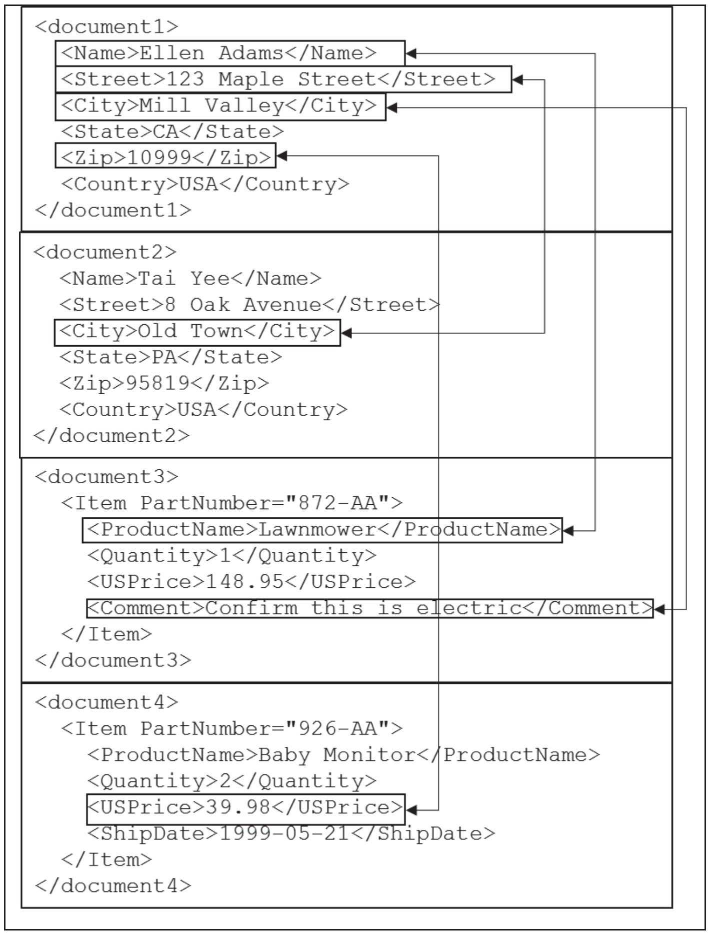
6 Tag shuffling

The basic idea behind tag shuffling is to further obfuscate the meaning of any single distributed XML document by switching the contents of element tags with one another. The only way to decode this shuffling would be to have access to the kernel document, which would store the mapping from tag to tag. For a basic example of this technique, External Source Document 1is examined.

External Document 1

```
<document1>
        <Name>Ellen Adams</Name>
       <Street>123 Maple Street</Street>
       <City>Mill Valley</City> <State>CA</State>
       <Zip>10999</Zip>
       <Country>USA</Country> </document1>
```

This source has six tags of Name, Street, City, State, Zip and Country. The shuffle technique would replace the contents of each of these tags with those of another element in a different distributed node. A sample result is shown in Figure 5. It is important to point out that not all tags need to be shuffled necessarily, just enough so that the meaning as a whole is undecipherable. In Figure 5, State and Country are left because there is very little someone could do with that kind of information.



```
<document1>
  <Name>Ellen Adams</Name>
  <Street>123 Maple Street</Street>
  <City>Mill Valley</City>
  <State>CA</State>
  <Zip>10999</Zip>
  <Country>USA</Country>
</document1>

<document2>
  <Name>Tai Yee</Name>
  <Street>8 Oak Avenue</Street>
  <City>Old Town</City>
  <State>PA</State>
  <Zip>95819</Zip>
  <Country>USA</Country>
</document2>

<document3>
  <Item PartNumber="872-AA">
    <ProductName>Lawnmower</ProductName>
    <Quantity>1</Quantity>
    <USPrice>148.95</USPrice>
    <Comment>Confirm this is electric</Comment>
  </Item>
</document3>

<document4>
  <Item PartNumber="926-AA">
    <ProductName>Baby Monitor</ProductName>
    <Quantity>2</Quantity>
    <USPrice>39.98</USPrice>
    <ShipDate>1999-05-21</ShipDate>
  </Item>
</document4>
```

Figure 5. Tag Shuffling Plan for Document 1

The shuffling process involves some level of human interaction and the steps are listed below.

1. Get User Input - user specifies which tags contain identifying information. These are the tags which will be shuffled by the system. Other options include what percentage of tags should be shuffled and whether tags with similar names should be shuffled.
2. The system starts with one of the documents. It then swaps tags based on the user input provided.
3. This shuffle pattern is recorded in the kernel document (shown below). This secures the mapping from the cloud services.
4. The system marks that these tags have already been shuffled and that they should not be shuffled again.
5. This same process is done for the rest of the documents in the system. Note: this is only done for the first distribution. The mapping will carry on to the secondary distributions.
6. When a user tries to get data, the XML document looks first at the mapping pattern and swaps the tags back to their original format on the local machine, or wherever the kernel document is stored.

In Figure 5, four of the tags will be swapped with four elements from other distributed sources for document 1. In this case, Name, Street, City, and Zip were swapped as they were considered to have identifying information for the document. The tags for swapping were randomly selected from those available in the other partitions. The kernel document contains the mapping of tag to tag. This allows the system to re-materialize the information into its pre-shuffled state. Each tag that has been swapped is listed and includes which document was swapped as well as which tag should be replaced. For <Name> in the below example, the system would go to document3, find the name tag, and exchange it for <ProductName> in document1. Tags that have not been shuffled are not included.

```
<kernel>
  ...
  <document1>
    <ProductName> <swap>document3</swap> <tag>Name</tag>
    </ProductName>
    <City> <swap>document2</swap> <tag>Street</tag> </City>
    <Comment> <swap>document3</swap> <tag>City</tag>
    </Comment>
    <USPrice> <swap>document4</swap> <tag>Zip</tag>
     </USPrice>
  </document1>
  ...
</kernel>
```

This would be done for the other external documents as well. Tags that have already been swapped will be marked as such. The marked elements will not be considered for swapping throughout the rest of the shuffling process. Shuffling tags in the cloud services add a further level of security outside of the distribution itself. If any of the cloud services happen to have sensitive information even after the distribution, the tag shuffling further secures the distribution by ensuring that sensitive information cannot be easily recombined without access to the kernel document. The kernel document is not stored on external services, and should be more easily secured locally than securing each cloud service independently.

7 Conclusion

XML is a widely used storage format for systems on the Internet. As single documents, the format is supported well, but there has been little research done for a distribution of the format. There are tools for accessing XML data in a distributed environment, but there are few tools for the design and actual distribution of an XML document among many nodes.

The paper lays out a plan for distributing an XML document throughout a system of cloud services using a kernel document and several distributed cloud nodes. The first portion of the system distributes the XML document through an algorithm of least load based on the number of cloud servers available, the size of the original XML document, and a predetermined margin.

The system includes a method for backing up document partitions through the redundancy of having a secondary distribution. The partitioned documents are distributed in a manner similar to the first part, based on the already distributed load on the cloud server.

This redundancy is done in case of server malfunctions or the server becoming compromised. In addition, the kernel, or central, document stores a mapping of elements in the portioned documents that have been shuffled from document to document so as to obfuscate the meaning of any single partition in the case of a cloud service being compromised. This is done so that if a cloud service is compromised and an intruder has access to possibly confidential data, they cannot discern the meaning of what data they have.

8 References

[1] H. F. El-Sofany, F.F.M. Ghaleb, and S.A. El-Seoud. “The Impact of XML Databases Normalization on Design and Usability of Internet Applications”, International Journal of Advanced Corporate Learning, 3.2 (2010), pp. 4-13.

[2] World Wide Web Consortium, “Extensible Markup Language (XML)”, http://www.w3.org/XML, last revision October 29, 2013, last accessed April 22, 2014.

[3] World Wide Web Consortium, “Extensible Markup Language (XML) 1.0 (Fifth Edition)”, http://www.w3.org/TR/2008/REC-xml-20081126/, last revision February 7, 2013, last accessed April 22, 2014.

[4] W3Schools.com, “DTD Tutorial”, http://www.w3schools.com/dtd/default.asp, last accessed April 23, 2014.

[5] W3Schools.com, “XML Schema Tutorial”, http://www.w3schools.com/Schema/default.asp, last accessed April 23, 2014.

[6] W3Schools.com, “XQuery Tutorial”, http://www.w3schools.com/xQuery/default.asp, last accessed April 27, 2014.

[7] S. Abiteboul, G. Gottlob, and M. Manna. “Distributed XML Design”, Journal of Computer and System Sciences, 77.6 (2011), pp. 93-964.

[8] H. Kurita, K. Hatano, J. Miyazaki, and S. Uemura. "Efficient Query Processing for Large XML Data in Distributed Environments”, 21st International Conference on Advanced Information Networking and Applications, (2007). pp.317,322, 21-23.

[9] S. Abiteboul, O. Benjelloun, and T. Milo. “The Active XML Project: An Overview”, The VLDB Journal, 17 (2008), pp. 1019-1040.

[10] G. Cong, W. Fan, A. Kementsietsidis, and et al. “Partial Evaluation for Distributed XPath Query Processing and Beyond”, ACM Trans. Database Syst., 37, 4 (2012), pp. 1-43.

[11] L. Guo, J. Wang, and H. Du. “XML Privacy Protection Model Based on Cloud Storage”, Computer Standards & Interfaces, 36 (2014), pp. 454-464.

An Approach For Securing Software as a Service Model of Cloud Computing

Fayza Rekaby[1]
Dept. of Computer and *Information Sciences*
Institute of Statistical Studies and Research, ISSR
Cairo University
[1]fayzarekaby@gmail.com

A. A. Abd El-Aziz[2]
Dept. of Computer Science
and *Information Sciences*
ISSR
Cairo, Egypt
[2]a.ahmed@cu.edu.eg

Mahmood A. Mahmood[3]
Dept. of Computer Science
and *Information Sciences*
ISSR
Cairo, Egypt
[3]mahmoodissr@cu.edu.eg

Hesham A. Hefny[4]
Dept. of Computer Science
and *Information Sciences*
ISSR
Cairo, Egypt
[4]hehefny@ieee.org

LATE BREAKING PAPER

Abstract— **Today, undoubtedly cloud-computing has turned into the popular expression in the IT business. There are many features that make cloud computing attractive for users; however, it has revealed new security issues. This paper provides an insightful analysis of the current security issues of cloud-computing. Moreover, it proposes a Security Model for SaaS (SSM) as a guide for assessing and enhancing security in each layer of SaaS model.**

Keywords- component; Cloud computing, SaaS, Security, SSM model

1 *Introduction*

Today, cloud computing has become the buzzword in the IT industry. Looking at the potential impact, it has numerous business applications used in our everyday life. It provides services over the internet; cloud computing user can utilize the online services of different softwares instead of purchasing or installing them at their own computers. The National Institute of Standards and Technology (NIST) has defined cloud computing as a model for enabling convenient and on-demand network access to a shared pool of configurable computing resources, such as networks, applications, and services, that can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

According to Gartner, cloud computing defines as a style of computing in which elastic IT-enabled capabilities are delivered as a service by using internet technologies [2]. According to NIST [1] and Seccombe [3], it has four different deployment models and three serves models as follows:

1) Private cloud; it may be owned, managed, and operated by the organization, a third party, or some combination of them.

2) Community cloud; it may be owned, managed, and operated by one or more of the organizations in the community.

3) Public cloud; the cloud infrastructure is provisioned for open use by the general public. It may be owned, managed and operated by business, academic, or government organization.

4) Hybrid cloud; the cloud infrastructure is composition of two or more distinct cloud infrastructures (Ease of Use).

1) SaaS (Software as a Service); it enables the user to access online applications and software that are hosted by the service providers.

2) PaaS (Platform as a Service); it provides platform for developing applications by using different programming languages.

3) IaaS (Infrastructure as a Service); it provides the use of virtual computer infrastructure environment, online storage, hardware, servers, and networking components.

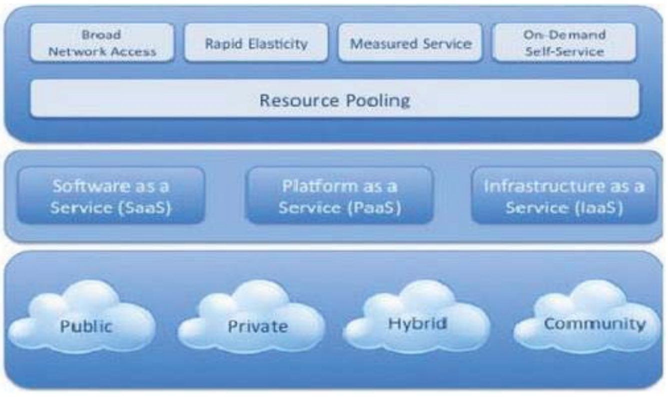


Fig1: NIST cloud definition Frame work [1]

In SaaS, there is the Divided Cloud and Convergence coherence mechanism whereby every data item has either the "Read Lock" or "Write Lock". Two types of servers are used by SaaS: the Main Consistence Server (MCS) and Domain Consistence Server (DCS). Cache coherence is achieved by the cooperation between MCS and DCS. In SaaS, if the MCS is damaged, or compromised, the control over the cloud environment is lost. Hence securing the MCS is of great importance [4].
In SaaS, the client has to depend on the service provider for proper security measures. The provider must do the work to ensure that the multiple users' from seeing each other's data. So it becomes difficult to the user to ensure that right security measures are in place and also difficult to get assurance that the application will be available when needed [5].
The cloud customer during using SaaS model who will, by definition, be substituting new software applications for old ones. Therefore, the focus is not upon portability of applications, but on preserving or enhancing the security functionality provided by the legacy application and achieving a successful data migration. [6] As interest

in software-as-a-service grows, so too do concerns about SaaS security. Total cost of ownership used to be the most frequently cited roadblock among potential SaaS customers. But now, as cloud networks become more frequently used for strategic and mission-critical business applications, security tops the list [7].
The trend Worldwide toward marches onward in SaaS by a Gartner Group estimate, SaaS sales in 2010 reached $10 billion, and was projected to increase to $12.1bn in 2011, up 20.7% from 2010. Customer relationship management (CRM) continues to be the largest market for SaaS [8]. There are indicators the trend toward marches onward in SaaS.

In fact, Enterprise Strategy Group (ESG) research shows that ranks as the 3rd most from now, the majority of companies will deliver at least 20% of their applications via SaaS. Of the companies surveyed, 25% reported that they already use or plan to use SaaS [9]. Now, we're really beginning to see some interesting trends, where more companies have adopted software-as-a-service (SaaS) over the past few years. The fourth annual "Future of Cloud Computing" survey revealed SaaS adoption rose from 13 percent in 2011 to 74 percent 2014 [10].
SaaS provides application services on demand, such as email, conferencing software, and business applications. It can be consumer-centric (e.g. Flickr photo storage, management and sharing offering), enterprise-centric (e.g. Salesforce.com's and Microsoft's customer relationship management (CRM) offering), or both (e.g. Google's Gmail email offering). A smaller, but growing, number of providers also use SaaS for more complex applications, such as enterprise resource planning (ERP) and supply chain management. Users While using SaaS mode; have less control over security among the three fundamental delivery models in the cloud. The adoption of SaaS applications may raise some security concerns [11].

Our main area of concern in this paper is to highlight security issues and their existing solutions in SaaS model and give recommended existing solution for the concern security issues at the SaaS level and also proposed layered security architecture SaaS Security Model (SSM) as a guide for assessing and improved the scale of security. The organization of the paper is as follows: Section 2 describes the security issues that are posed by the Software as a Service delivery model. Section 3 lists some of the current solutions. Section 4 Roles Customer and the Provider of security-relevant .Section 5 discuss the proposed model that will help assessment to security issues. Finally, Section 6 is conclusions.

2 SECURITY ISSUES IN SOFTWARE AS SERVICE

Today, enterprises view data and business transactions as strategic and guard them with access control and compliance policies. However, in the SaaS model, enterprise data is stored at the SaaS provider's data center, along with the data of other enterprises. Moreover, if the SaaS provider is leveraging a public cloud computing service, the enterprise data might be stored along with the data of other unrelated SaaS applications. The cloud provider might, additionally, replicate the data at multiple locations across countries for the purposes of maintaining high availability. Most enterprises are familiar with the traditional on- promise model, where the data continues to reside within the enterprise boundary, subject to their policies. Cloud computing providers need to solve the common security challenges being faced by traditional communication systems. At the same time, they also have to deal with other issues inherently introduced by the cloud computing paradigm itself. In the following section, the SaaS security issues have been categorized as traditional and new cloud specific security challenges, for sake of convenience.
Over the past decade, the fact is that enterprises have been outsourcing their services and technology. Hence, most enterprises are aware with the traditional on- promise model, where the data continues to reside within the enterprise boundary, subject to their policies. Enterprises view data and business transactions as strategic and guard them with access control and compliance policies may give up some control to these service providers when they upgrade to a cloud-based environment. However, in the SaaS model, Service providers already deliver hosted data and stored at the SaaS provider's data center, along with the data of other enterprises. Moreover, if the SaaS provider is leveraging a public cloud computing service, the enterprise data might be stored along with the data of other unrelated SaaS applications. The cloud provider might, additionally, replicate the data at multiple locations across countries for the purposes of maintaining high availability. As a result, cloud computing providers need to solve the common security challenges that traditional communication systems face.
The key security rudiments should be carefully considered as a fundamental part of the SaaS application development and deployment process include; Data security, Network security, Data locality, Data integrity, Data segregation, Data access, Authentication and authorization, Data confidentiality, Web application security, Data breaches, Virtualization, Availability, Backup and Identity management [12]. The SaaS security issues have been categorized as traditional and new cloud specific security challenges [23]. Due to the research, we study the protection subject of every each component and discuss the proposed solutions and recommendations.

2.1 Data Security

Data security risks constitute the biggest barrier for any technology, but it becomes a major challenge when SaaS users have to rely on their SaaS vendor for proper security then data security is the major issue [13 and 12]. In other words, in the SaaS model, the enterprise data is stored outside the enterprise boundary, at the SaaS vendor end. In SaaS, organizational data is often processed in plaintext and stored in the cloud. The SaaS vendor is the one responsible for the security of the data while is being processed and stored [14]. Consequently, these SaaS vendor must ensure data security and prevent breaches due to security vulnerabilities in the application or through malicious employees by adopt additional security. This involves the use of strong encryption techniques for data security and fine-grained authorization to control access to data. In cloud vendors such as Amazon, Security within Amazon EC2 is provided on multiple levels: The operating system (OS) of the host system, the virtual instance operating system or guest OS, a stateful firewall and signed API calls. Administrators do not have access to customer instances and cannot log into the Guest OS. EC2 Administrators with a business need are required to use their individual cryptographically Strong Secure Shell (SSH) keys to gain access to a host [15]. All such accesses are logged and routinely audited. Data stored within Amazon S3 is not encrypted at rest in. However, users can encrypt their data before it is uploaded to Amazon S3 so that the data cannot be accessed or tampered with by unauthorized parties.

2.2 Network security

In a SaaS deployment model, sensitive data is obtained from the enterprises, processed by the SaaS application and stored at the SaaS vendor end. Network security measures are needed to protect data during their transmission. The network needs to be secured in order to prevent leakage of sensitive information, using the strong network traffic encryption techniques, such as Secure Socket Layer (SSL) and the Transport Layer Security (TLS) for security. The Amazon Web Services (AWS) network layer provides significant protection against traditional network security issues, such as MITM (Man-In-The-Middle) attacks, IP spoofing, port scanning, packet sniffing, etc. and the customer can implement further protection. Amazon S3 is accessible via SSL encrypted endpoints. The encrypted endpoints are accessible from both the Internet and from within Amazon EC2, ensuring that data is transferred securely both within Amazon Web Services AWS and to and from sources outside of AWS.

2.3 Data locality

In a SaaS model of a cloud environment, end-users use the applications provided by the SaaS and process their business data without knowing exactly where the data is getting stored. Moreover, most compliance standards do not envision compliance with regulations in a world of cloud computing, consumers do so on its own initiative and is responsible for compliance with applicable laws. For example, in many EU and South America countries, certain types of data cannot leave the country because of potentially sensitive information [16].

The European Union has issued a Directive 95/46/EC to protect the user privacy at all costs [17]. The directive prohibits transfers of personal data to countries which do not ensure an adequate level of protection. For example, the recent Dropbox users have to agree to the ‘‘Terms of Service’’ which grant the providers the right to disclose user information in compliance with laws and law enforcement requests [18]. This can be an issue of local laws in the world of SaaS, the process of compliance is complex because data is located in the provider’s datacenters, which may introduce regulatory compliance issues such as data privacy, segregation, and security, that must be enforced by the provider.

2.4 Data Segregation

Multi-tenancy is one of the major characteristics of cloud computing. However, Multi-tenancy in cloud means sharing of resources and services to run software instances serving multiple consumers and client organizations (tenants) that users can store their data using the applications provided by SaaS. Intrusion of data of one user by another becomes possible in this environment. This intrusion can be done either by hacking through the loop holes in the application or by injecting client code into the SaaS system. A client can write a masked code and inject into the application. If the application executes this code without verification, then there is a high potential of intrusion into other’s data. A SaaS model should therefore ensure a clear boundary for each user’s data. Multiple clients (tenants) may be sharing the same application stack (database, app/web servers, and networking). That means the data from multiple tenants may get stored in the same database, may get backed up and archived together, may be moving on common networking devices (unencrypted), and managed by common application processes. This puts a heavy emphasis on logical security built within the application to separate one tenant’s users from others. The boundary must be ensured not only at the physical level but also at the application level. The service should be intelligent enough to segregate the data from different users.

2.5 Data access

Data access issue is mainly related to security policies provided to the users while accessing the data. Once data is stored in the cloud, the provider has access to that data by other entities. In a typical scenario, a small business organization can use a cloud provided by some other provider for carrying out its business processes. This organization will have its own security policies based on which each employee can have access to a particular set of data. The security policies may entitle some considerations where in some of the employees are not given access to certain amount of data. These security policies must be adhered by the cloud to avoid intrusion of data by unauthorized users [19, 20, and 21].

The SaaS model must be flexible enough to incorporate the specific policies put forward by the organization. The model must also be able to provide organizational boundary within the cloud because multiple organization will be deploying their business processes within a single cloud environment [22 and 23].

2.6 Cloud Computing Standards

Cloud is still an emerging technology, and standards are still developing , with certain areas maturing more quickly than others. Standards should promote trusted and reliable cloud offerings that encourage security, interoperability, data transferability and reversibility. Cloud standards are needed across different standard developing organizations. The abundance of standards and awareness of gaps which are present in standards may led to add to the confusion for cloud users [24]. Many concerns could be addressed with the necessary standards in place. There are currently a large number of standards bodies with different interests, such as IEEE Cloud Computing Standard Study Group [25], ITU Cloud Computing Focus Group [26], Cloud Security Alliance (CSA), Distributed Management Task Force [27], Storage Networking Industry Association [28], OGF’s Open Cloud Computing Interface [29], Open Cloud Consortium [30], and Organization for the Advancement of Structured Information Standards [31]. The standards are immature and insufficient for handling the rapidly changing and evolving technologies of cloud computing. Standards should promote the wide use of cloud computing to move services between clouds in the following areas are needed to increase cloud interoperability execution, Portability, management services, data services, resource management services, security services, self-management services and information service:

-Interface standards for email and productivity in the office

-Standards for user account and credential management

-standards for identity management across domains, single sign-on to multiple cloud systems, standards for policies, processes and controls for audits, regulation and compliance

-Standards for SLA description, standards for cloud service resource description, standards for metering and billing of cloud service usage

Potential customers will have increased confidence in the cloud when many concerns could be addressed with the necessary standards in place.

2.7 Authentication and Authorization

The authorization and authentication applications used in enterprise environments need to be changed, so that they can work with a safe cloud environment. Strong authentication is a mandatory requirement for any cloud deployment. Strong user authentication and authorization have not yet been extending into the cloud. K. Yassin et al. [32] proposed an authentication process based on a one-time password (OTP) with mutual authentication of the user and the cloud server. There are anonymous password for 2FA in cloud computing environment, encryption scheme with ASPE, and RSA digital signature for a cloud computing model Pratap Murukutla [33]. They have proposed a solution with de-facto standards of open authorization by a trust party auditor which maintains all the credentials and cloud provider can uniquely distinguish one user from other. Moreover, they allows user to use a single set of credentials.

2.8 Availability

Availability these concerns center on critical applications and data being available. The SaaS application providers are required to make sure that the systems are running as it should be when needed and enterprises are provided with services around the clock. Well-publicized incidents of cloud outages such as Amazon S3's over seven-hour downtime on July 20, 2008 (Amazon S3 Availability Event, 2008), and FlexiScale's 18-17 hour outage on October 31, 2008 (Flexiscale Outage) [34]. Maintaining the uptime, preventing denial of service attacks and ensuring robustness of computational integrity are some of the major issues in this category of threats.

2.9 Web Application Security

Software as a Service application development may use various types of software components and frameworks which run is deployed behind a firewall in local area network or personal computer. SaaS identifying characteristics include Network-based access to, and management of, commercially available software and managing activities from central locations rather than at each customer's site, enabling customers to access application remotely via the Web. Web applications introduce new security risks that cannot effectively be defended against at the network level, and do require application level defenses which create a weakness to the SaaS application. The Open Web Application Security Project (OWASP) identifying the ten most critical web applications security threats [35]. Web application security

2.10 Backup Data

The traditional backup methods are not optimally designed for the applications running in the cloud so must develop. It used with earlier applications and data centers that were primarily designed for web and consumer applications. The SaaS vendor needs to ensure that all sensitive enterprise data is backed up on regular basis to smooth the progress of quick recovery in case of disasters. Backup data is recommended use of strong encryption schemes to protection and also to prevent accidental leakage of sensitive information. In the case of cloud vendors such as Amazon, the data at rest in S3 is not encrypted by default. The users need to separately encrypt their data and backups so that it cannot be accessed or tampered with by unauthorized parties.

3 SOLUTIONS SECURITY ISSUES

There are many research documents in the area of cloud security. Nowdays, several organizations and groups are interested in developing security solutions and standards for the cloud. National Institute for Standards and Technology (NIST) is a key organization in defining various standards for cloud computing [36]. The Cloud Security Alliance (CSA) is non-profit organization provides security guidance for several areas of focus in cloud computing, gathering solution, covers key issues and provides advice for both cloud computing customers and providers within various strategic domains, best practices for information assurance in the cloud [37 and 38]. The Open Web Application Security Project (OWASP) raise awareness about application security by identifying some of the most critical risks facing organizations, maintains list of top vulnerabilities to cloud-based or SaaS models which is updated as the threat landscape changes [39]. The Cloud Standards website collects and coordinates information about cloud related standards under development by the groups. Organization for the Advancement of Structured Information Standards (OASIS) is a not-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards (OASIS Homepage). OASIS has technical committees In cloud computing domain .OASIS promotes industry consensus and produces worldwide standards for security, cloud computing and other [40].

[41] Proposed a data protection framework that is consists of three key components: policy ranking, policy integration and policy enforcement. The framework addresses challenges during the life cycle of a cloud service. For each component there are various models. Policy ranking aims satisfying users' privacy policy requirements, to find the service provider through three models: (i) User-oriented ranking model; (ii) Service-provider-oriented ranking model; and (iii) Broker based ranking model. Policy Integration Models, created policies to be agreed by involving parties. Finally, the policy enforcement to guarantee correct policy enforcement in the cloud (uses either tight coupling or loose coupling) examines whether the confidentiality of data and policies are guaranteed at any time and at any location or not.
[42] Proposed a different approach for securing data using offensive decoy technology to mitigate insider data theft. In this approach, monitoring data access patterns by profiling user behavior. Decoy documents that stored in the Cloud alongside the user's real data also act as sensors to detect illegitimate access. Once unauthorized data access or exposure, it is verified using challenge questions.
[43] Proposed an authentication approach using federated identity management together with hierarchical identity-based cryptography (HIBC). It provides keys (public key and private key) the key distribution and the mutual authentication can be simplified in the cloud. It allows users to access services from other Clouds with a single digital identity. For web services, this approach simplifying public key distribution and reducing SOAP header size. It is used to create a session between two parties without message exchange. However, it creates trust issues since third party key distribution is involved. The key escrow problem can be restricted.

4 ROLES CUSTOMER AND THE PROVIDER

With respect to security incidents, the definition and understanding between a customer and a provider of security-relevant roles and responsibilities are important to be a clear for both. Each one service model has a distinctive division will vary greatly between SaaS offerings and IaaS offerings in the case of cloud service [44].

4.1 Responsibilities of Customer

Customer is responsible for implementing and managing themselves, Cloud customers should also carry out their own, Compliance with data protection law in respect of customer data collected and processed, Maintenance of identity management system,Management of identity management system,Management of authentication platform.

4.2 Responsibilities of Provider

Provider able to support Physical infrastructure (facilities, rack space, power, cooling, cabling, etc) and management fast dynamic allocation, Management Physical infrastructure security and availability (servers, storage, network bandwidth, etc). OS patch management and hardening procedures (check also any conflict between customer hardening procedure and provider security policy). Control Security platform configuration (Firewall rules, IDS/IPS tuning, etc), management Systems monitoring, Security platform maintenance (Firewall, Host IDS/IPS, antivirus, packet filtering), Log collection and security monitoring. Cloud providers may have not to prevent (directly or indirectly) the portability of their customers services and data.

5 PROPOSED SECURITY ARCHITECTURE FOR SAAS

As a result of those research, we propose a SaaS Security Model (SSM) as a guide for assessing and enhancing security in each layer of SaaS service model as shown in Fig.2 . We assume to perform a security assessment by specialist farm. The most practical way to perform a security assessment associated with using a service in the cloud is to get a third party to do it. [45] SSM model consists of three layers,1) Network and Application Layer2) Controller Access Control and 3) Data Security layer. Now, I am going to discuss each layer in detail.

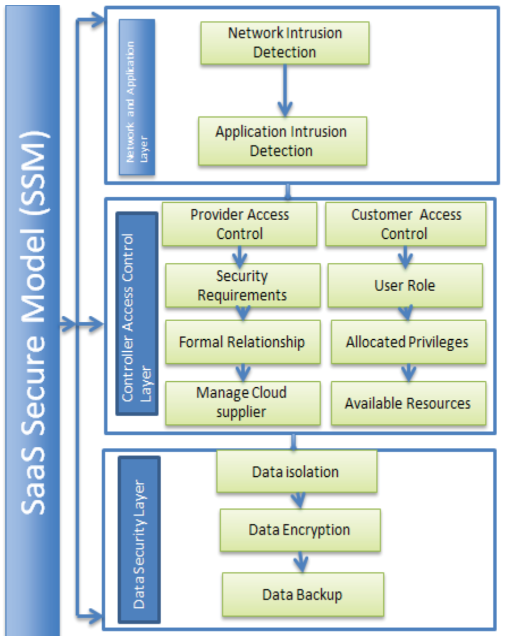


Figure 2: Proposed Security Layered Architecture

5.1 Network and Application Layer

This is the first layer which manage the controls to the cloud services by detecting any feasible intrusions and deal with them. It can be divided into two sub-layers. One is network and another at application level. Network based Intrusion Detection/Prevention (NIDP) attempt to expose illegal entry to network by configuring the network traffic packets like UDP, TCP and IPX/SPX and analyze the contents against a collection of rules. The most common tactics for network intrusion detection or prevention are digital signatures, Network penetration and packet analysis, Session management weaknesses, network behavior analysis or traffic analysis, Insecure SSL trust configuration, and protocol analysis or heuristics. NIDS is prepared up of a mix of uni-utility sensors and configured according to the rules and policies of the provider that are settled at dissimilar points within the network. Such that it can accept the packet and discard the packets according to the rules define by the cloud provider. Using the good authentication tool, the one is application level intrusion detection system. It can handle like unauthorized access by malicious bots or user.

5.2 Responsible Access Control

Controller Access Control Layer is combination of two sub-layers; Customer Access Control (CAC) and Provider Access Control (PAC), each sublayer is combination of thee Levels. Customer Access Control sub-layer is combination of three levels. Cloud customers carry out their own. The first check the user or customer roll in the cloud system is that it ensures customer cannot swap their roles .the second is responsible for verifies the privileges, privileges for the entire cloud system and, if so, for what operations (read/write/delete).
The last manages the available resources according to their roles and privileges. This layer channels the clients on the premise of their Predefined roles. The 3 levels further include security as they further verification that the client can't swap parts or any Gate crasher can't get access to the assets. The principal of these levels verities the parts relegated to the client and can even relegate default part to any new client. This level is likewise in charge of keeping note of any part Acceleration that has happened and can even rollback the Acceleration. The next level checks the benefits relegated to the client and checks whether they match the role that the client has and any conflict could prompt repudiating of any extra benefits. The third level deals with conceding the resources to the client that focused around his roles and privileges.
Provider Access Control sublayer is combination of three levels. The first level customers security requirements, manages cloud service providers meet to the customers' information security requirements that derived from the organization's own policy, legal and regulatory obligations, and may carry through from other contracts or SLAs that the company has with its customers. The second level customers formal relationship is responsible for verifies from formal relationship between cloud customers and cloud provider. Prospective cloud customers should undertake proper due-diligence on providers. Detailed due-diligence investigations can provide an unbiased and valuable insight include a providers' past track record, its financial status, legal action taken against the organization and its commercial reputation. Certification schemes such as ISO27001 also provide customers with some assurances that a cloud provider has taken certain steps in its management of information security risks.
The third level the outsourcing of services supplier ,the outsourcing of services supplier – This layer Manage cloud supplier risks. The outsourcing of key services to the cloud may require customer organizations to seek new and more mature approaches to risk management and accountability. Effective risk management also requires maturity both in vendor relationship management processes and operational security processes.

5.3 Layer :Data Security

Lowest layer in this model is data security layer (DSL). It is combination of 3 sublayer there sub-layer. DSL is responsible for prevention of illegal use of data and data protection from deletion, accidental problem. All sublayer achieve together to maintain the security of data. DSL is governed by the rules set done by the Cloud computing system providers.
The cloud data layer security is the least level of security that the cloud supplier can apply. This incorporates security of information from coincidental access or deletion and counteractive action of unapproved utilization of data the three sub layers can cooperate to guarantee security of the information. The layers are completely legislated by the strategies set by the cloud supplier. The first layer

guarantees isolation, This means that isolation between resources used by different customers must be strong of information regardless i.e. regardless of the fact that the encryption tails at any stage this layer guarantees the isolation of information. Furthermore encryption can be used for protecting data. Data encryption is a key some piece of putting away the information on cloud as the information is extremely defenseless against outside access. The information encryption that utilized ought to be solid enough however in the meantime it ought to have the capacity to adapt with colossal volumes of information. This issue however has been countered, all things considered, by the Hard circles which encode any information that is put away on them and likewise. By the accessibility of secure fittings which don't let the supplier get to the memory when information is, no doubt transformed. An alternate perspective is that of recovery to information, the suppliers must give a backup to all the information that the client has so if there should arise an occurrence of any accidental deletion the information could be effectively recovered.

6. CONCLUSION

Cloud computing is a disruptive technology with profound implications not only for Internet services but also for the IT sector as a whole. As described in this paper, currently security has lot of issues which scares away several potential users. Until a proper security module is not in place, potential users will not be able to leverage the true benefits of this technology. This security module should cater to all the issues arising from all directions of the cloud. In a cloud, where there are heterogeneous systems having a variation in their asset value, a single security system would be too costly for certain applications and if there is less security then the vulnerability factor of some applications like financial and military applications will shoot up. On the other side, if the cloud has a common security methodology in place, it will be a high value asset target for hackers because of the fact that hacking the security system will make the entire cloud vulnerable to attack. In this paper, propose a SaaS Security Model (SSM) as a guide for assessing and enhancing security in each layer of SaaS service model. We assume to perform a security assessment by specialist farm. In this model A decently characterized layered methodology is important to guarantee the security at every level. The methodology might be effortlessly connected and changed according to the client requirements with available software also promote a common level of understanding between the consumers and providers of cloud computing regarding the necessary security requirements. Cloud computing has the possibility to turn into a leader in, Secure virtual and financially reasonable IT service. In the future, we will give and implement some security strategies with technology and management ways.

7 References

[1] NIST SP 800-145, "A NIST definition of cloud computing", http://csrc.nist.gov/publications/drafts/800- 145/Draft-SP-800-145_cloud-definition.pdf

[2] Gartner, "What you need to know about cloud computing security and compliance"(HeiserJ),[online]2009,https://www.gartner.com/doc/1071415/need-know-cloud-computing Security.

[3] Seccombe A.., Hutton A, Meisel A, Windel A, Mohammed A, Licciardi A, (2009). Security guidance for critical areas of focus in cloud computing, v2.1. Cloud Security Alliance, 25 p.

[4] G. Rakesh Reddy , Dr. M. Bal Raju , "Augmentation Data Security Aspects of Cloud Computing", International Journal of Advanced Trends in Computer Science and Engineering, Vol.2 , No.1, Pages : 139-142 (2013) Special Issue of ICACSE 2013 - Held on 7-8 January, 2013 in Lords Institute of Engineering and Technology, Hyderabad icacsesp28

[5] Choudhary V. Software as a service: implications for investment in software development. In: International conference on system sciences, 2007, p. 209.

[6] Seccombe A.., Hutton A, Meisel A, Windel A, Mohammed A, Licciardi A, (2009). Security guidance for critical areas of focus in cloud computing, v2.1. Cloud Security Alliance.

[7] 5 problems with SaaS security, http://www.networkworld.com/news/2010/092710-software-as-service-security.html

[8] McHall, Tom (7 July 2011). "Gartner Says Worldwide Software as a Service Revenue Is Forecast to Grow 21 Percent in 2011". Gartner.com. Gartner. Retrieved 28 July 2011.

[9] McClure, Terri and Kristine Kao, "Market Landscape Report: Online File Sharing and Collaboration in the Enterprise," Enterprise Strategy Group, December 2011.

[10] Dan Kobialka, "Future of Cloud Computing Survey: PaaS, SaaS Adoption on the Rise"Jun 19, 2014 .

[11] ESG-Market-Landscape-Online-File-Sharing-and-Collaboration-Dec-1

[12] Subashini S, Kavitha V. A survey on security issues in service delivery models of cloud computing. Journal of Network and Computer Applications; 2011;34(1):1–11.

[13] JW Rittinghouse, JF Ransome Security in the Cloud. In: Cloud Computing. Implementation, Management, and Security, CRC Press 2009 - books.google.com.sci-hub.org

[14] Ju J, Wang Y, Fu J, Wu J, Lin Z (2010) Research on Key Technology in SaaS. In: International Conference on Intelligent Computing and Cognitive Informatics (ICICCI), Hangzhou, China. IEEE Computer Society, Washington, DC, USA, pp 384–387.

[15] Amazon. Amazon Elastic Compute Cloud (EC2). [accessed on: 14 May 2015] http://aws.amazon.com/ec2/

[16] Vijayapriya, "SECURITY ALGORITHM IN CLOUD COMPUTING: OVERVIEW" / International Journal of Computer Science & Engineering Technology (IJCSET) 2013.

[17] European Union. Directive 95/46/EC of the European parliament and of the council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data ;1995.

[18] Dropbox. Blog. Privacy, security & your dropbox; 2011. <http://blog.dropbox.com/?p=735> [Accesed :May 2015

[19] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure ealization," in Proceedings of the 4th International Conference on Practice and Theory in Public Key Cryptography (PKC'11). pringer, 2011, pp. 53–70.

[20] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP'08). Springer, 2008, pp. 579-591.

[21] J. Bethencourt,A.Sahai, and B.Waters, "Ciphertext-policy attributebased encryption," in Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P'07). IEEE Computer Society, 2007, pp. 321–334.

[22] Tejaswini R M1, Roopa C K2 , Ayesha Taranum3, "Securing Cloud Server & Data Access with Multi-Authorities" International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online) Vol. 2, Issue 2, pp: (297-302), Month: April-June 2014, Available at: www.researchpublish.com.

[23] Rashmi 1 , Dr.G.Sahoo2 , Dr.S.Mehfuz, "Securing Software as a Service Model of Cloud Computing: Issues and Solutions", International Journal on Cloud Computing: Services and Architecture (IJCCSA) ,Vol.3, No.4, August 2013

[24] Fogarty Kevin. Cloud computing standards: too many, doing too little; 2011. http://www.cio.com/article/679067/

[25] IEEE CCSSG. IEEE Cloud Computing Standard Study Group. http://www.computer.org/portal/web/sab/cloud.

[26] ITU,2015. Cloud Computing Focus Group. http://www.itu.int/en/ITUT/focusgroups/cloud/Pages/default.asp x

[27] DTMF,Distributed Management Task Force. <http://www.dmtf.org/>.

[28] SNIA.(2013).Storage Networking Industry Association. http://www.snia.org/

[29] OGF,.Open Grid Forum, http://www.ogf.org/ [Accessed: May 2015]

[30] OCC,OpenCloudConsortium. http://www.opencloudconsortium.org/

[31] Organization for the Advancement of Structured Information Standards. Homepage URL: http://www.oasis-open.org

[32] Yassin AA, Jin H, Ibrahim A, Qiang W, Zou D. Cloud authentication based on anonymous one-time password. In: Han Y-H, Park D-S, Jia W, Yeo S-S, editors. Ubiquitous information technologies and applications. Lecture notes in electrical engineering, vol. 214. New York: Springer Dordrecht Heidelberg; 2013. p.

[33] Pratap Murukutla, K.C. Shet (2012).Single Sign On for Cloud .In: International Conference on Computing Sciences,2012 IEEE DOI 10.1109/ICCS.2012.66

[34] Amazon S3 Availability Event: (2008). URL: http://status.aws.amazon.com/s3-20080720.html (Accessed on May 2015).

[35] Cloud-10 Multi Tenancy and Physical Security 2010.[Online]. Available: https://www.owasp.org/index.php/Cloud-10_Multi_Tenancy_and_Physical_Security

[36] Badger, L., Grance, T., Patt-Corner, R., & Voas, J. (2011). Draft Cloud Computing Synopsis and Recommendations. National Institute of Standards and Technology (NIST) Special Publication 800-146. US Department of Commerce. May 2011. Available online at: http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf (Accessed on: May 19, 2015).

[37] M. Vijayapriya, “Security algorithm In Cloud Computing: Overview”/ International Journal of Computer Science & Engineering Technology(IJCSET)

[38] Cloud Security Alliance. Guidance for identity & accessmanagement V2.1,2010a

https://cloudsecurityalliance.org/guidance/csaguide dom12-v2.10.pdf

[39] OWASP.(2013) The Ten most critical Web application Security risks. Available: https://www.owasp.org/index.php/Category: OWASP_Top_Ten_Project [Accessed:May 2015].

[40] Organization for the Advancement of Structured Information Standards. Homepage URL: http://www.oasis-open.org

[41] Lin D, Squicciarini A (2010) Data protection models for service provisioning in the cloud. In: Proceeding of the ACM symposium on access control models and technologies, SACMAT’10

[42] Stolfo SJ, Salem MB, Keromytis AD (2012) Fog computing: mitigating insider data theft attacks in the cloud. In: 2012 IEEE symposium on security and privacy workshops. IEEE Press, New York, pp 125–128

[43] Yan L, Rong C, Zhao G (2009) Strengthen cloud computing security with federal identity management using hierarchical identity-based cryptography. In: Proceedings of the 1st international conference on cloud computing, CloudCom’09, pp 167–177

[44] Benefits, risks and recommendations for information securityEuropean Network and Information Security Agency (ENISA), 2009

[45] J. Heiser and M. Nicolett. Assessing the Security Risks of Cloud Computing, June 2008. assessing-the-security-risks

SESSION

SCALABLE COMPUTING + DATA CENTERS + WEB-BASED SYSTEMS AND APPLICATIONS

Chair(s)

TBA

High Throughput Data Replication Model with Asynchronous Event Tracking Method

Ki Sung Jin, Young Kyun Kim
Electronics and Telecommunications Research Institute, Daejeon, Korea

Abstract - *In data intensive scalable computing, the replication mechanism plays an important role in supporting a high availability, fault tolerance, and scaling up of the performance. However, although many related researches have already proven that a replication can scale up the read performance proportionally to the number of replicas, a traditional replication model still suffers from a poor write performance. To resolve this problem, we propose the event tracking chain replication (ETCR) model. Our model supports asynchronous update message propagation for scaling up the write throughput, as well as an additional event handling mechanism for keeping replicas in a consistent state. We show that our model greatly reduces the write latency and improves the write performance.*

Keywords: Replication, Distributed Storage, Filesystem

1 Introduction

In data-intensive scalable computing, the replication mechanism plays an important role in supporting a high availability, fault tolerance, and scaling up of the performance [1]-[3]. The key idea is that the replication be able to guarantee the synchronization of all replicas by propagating the original updates to the replica servers. Since the data are stored in one or more replica servers, the user can read the required data from the nearest server at any time as long as one of the replicas is alive. Based on the advantage of such a distributed access to replicas, many related researches have already proven that a replication can scale up the read performance in proportion to the number of replicas [4]-[5]. However, the previous replication model still suffers from a poor write performance owing to the expensive message overhead for assuring the data consistency among all replicas. Lots of replication algorithms struggle to keep all replicas rigorously in the same state as part of one atomic transaction. Such a mechanism needs expensive additional messages for maintaining synchronized data among all the servers [6]-[8]. This messaging cost is the main reason for increasing the response time and decreasing the write performance. As a Microsoft research report has stated, lots of algorithms have been theoretically well proven for quite a long while, but it is very hard to find concrete practical products of their research developed in large-scale distributed systems [9].

To solve such a limitation of the write performance, we propose the event tracking chain replication (ETCR) model. Our model supports asynchronous update message propagation for scaling up the write throughput, as well as an additional event handling mechanism for keeping replicas consistent. Based on a cost estimation and experimental evaluation, we show that our model greatly improves the write performance while preserving data consistency among the replicas.

2 Related Works

The Quorum-based protocol can support strict consistency [10]-[13]. The quorum-based protocol gives access permission to one of the replica nodes, which is selected through a consensus approach. Generally, a quorum is a subset of replica nodes in a system. Read and write quorums are defined for each read and write operation, respectively. At any case, each operation can begin only if permission is acquired from all participant nodes within the quorum. Such a consensus mechanism can support constant latency for all operations, and can support resiliency from various failures. However, this advantage also has drawbacks in practical systems because it incurs too much consensus messaging cost and increases the system complexity. In addition, a quorum-based protocol needs 2f+1 nodes to tolerate f failures.

The chain replication is a fast replication protocol whose goals are both high throughput and availability in a large-scale distributed system [14][15]. The basic approach is to construct all replicas in a serialized chain. The first link of a chain is called a head, and the last link of a chain is called a tail. For the write operation, the head only has a responsibility for updating the data. After the update message arrives, the head instantly propagates the update message to the next node along with the chain. If the update message arrives at the tail, a chain replication believes that all nodes in the chain have the same data, and then sends an ack message to the client. The chain replication does not need complex consensus mechanisms because the state machine is in itself implicit. Such a simple ordering is an important factor to be applicable to large-scale distributed systems. However, a chain replication still suffers from write throughput degradation owing to write message latency. Because every write operation should be blocked until a response message arrives from the tail, completion of the write operation is delayed proportionally to the number of replicas.

3 Proposed Replication Model

3.1 Data Transfer

We consider a large-scale storage environment where n remote servers are connected through a trusted network. When the update is stored on one of servers, it is simultaneously applied to the other servers. We do not constrain the number of servers or number of replicas. According to the notion described in the previous chain replication mechanism [14], we also assume the following: i) each failed server instantly goes to a fail-stop state instead of waiting for a failed server to return to an erroneous state; ii) the failure status of each server can be detected by the system without omission; iii) a stateful TCP connection is maintained from a client to all participant replicas in a single ordered channel; and iv) consistency is maintained using a general concurrency control mechanism such as a lease or version vector.

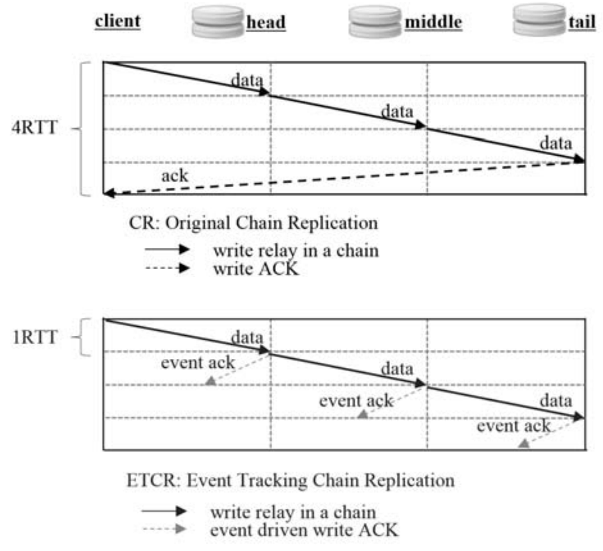


Figure 1. Event tracking replication protocol. While the CR needs *4RTT* until a replication is completed, the ETCR needs just *1RTT* for the same request. In the ETCR, each server sends an event notification to the client only when an error occurs in updating the data.

We set each request window as W = { r0, ri, ..., rN } of N ≥ 1, where r0, ri, and rn are called a head, a middle, and a tail respectively. The constant N is an arbitrary number that can be increased to the number of servers. Each W is formed when data are first created from the user, and all of r{0..n} is composed in an ordered fashion within W. Such ordering information remains continuously in an unchanged state if the data are not removed or a server failure is not detected. Consistent ordering of W is a core concept of the replication process. When the client issues an update request, every update moves to replicas followed by W. In addition, an ordered fashion can give us a very intuitive and simple method to avoid update conflicts from multiple requests.

Figure 1 shows the concept of our proposed model compared to the traditional chain replication (CR) mechanism. For simplicity, we suppose an environment in which a single client continuously issues an update request to the same object. In CR, if the client sends an update to the head, the update is delivered to the next server followed by W. When the update arrives at the tail, the tail sends an ack reply to the client, and the client completes the request. This means that the client inevitably stays in blocked state until the response message arrives from the tail. Because the client can send the next update after confirming a response for the current update, the write performance may be declined in proportion to the response latency. Paying careful attention to this point, a few questions may arise: Why should we always receive reply messages from the tail? Is there any solution to guarantee data consistency without a response?

To solve such problems, we define two types of response messages: i) a regulated response, and ii) an event response. While a regulated response means that the update is successfully applied to all replicas, an event response means that a failure in W has occurred. The most important fact is that most messages are regulated responses, and event responses are rare in a real storage environment. Nevertheless, CR always sends regulated messages without considering whether the update has succeeded or not, which is why the write performance is limited in CR. To solve this problem, we propose event tracking chain replication (ETCR), which refines the messaging protocol of CR.

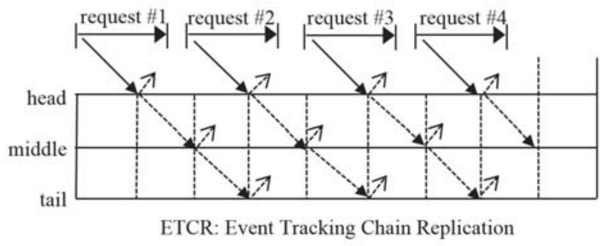


Figure 2. Update message flow in ETCR

In ETCR, the tail no longer sends a regulated response to the client. Instead, we only send an event response to the client at the moment an error event occurs in W. Because the client does not need to wait for a regulated response from the tail, the next update can be transmitted without blocking. Figure 2 shows the difference in update message sequencing between ETCR and CR.

However, this messaging protocol can undergo an abnormal state when a failure occurs. That is, the client can believe that an update has successfully been completed even if the update has failed owing to various types of errors. To resolve such an inconsistency problem, the client maintains recent sending updates for a certain period. If ETCR detects an event response notified from any replicas within W, the client then re-sends the recent update to the live replicas to make the data consistent during the recovery process.

3.2 Write Event Tracking

The key idea of ETCR is not handling a regulated response in the tail. As mentioned before, a regulated response is the main reason for increasing the update latency. Instead, ETCR manages an event response when an error occurs. For this, we support an extra event tracker mechanism in the client, as shown in Figure 3. In ETCR, each client controls the event tracker consisting of a sustained buffer and an event handler. The sustained buffer is used for keeping previously transmitted data, and the event tracker is used for monitoring update errors during the whole replication process.

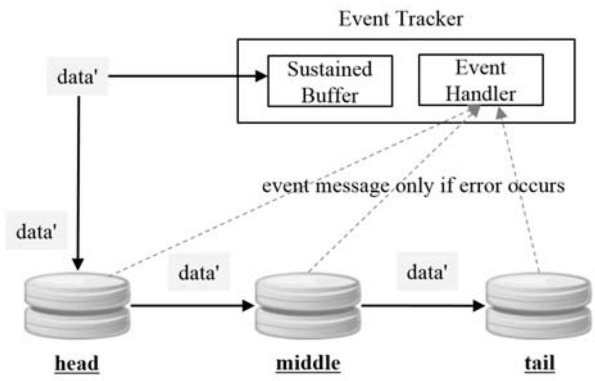


Figure 3. Architecture of event tracker for handling errors

In ETCR, the client must put each update into the sustained buffer before sending the update to the head following the write ahead log (WAL) protocol. However, this buffer does not remain permanently but for only a certain period of time. To avoid continuously growing the size of the sustained buffer, the tail sends complete notifications of each update periodically. The completed update is then removed from the sustained buffer. Because the tail sends several completed groups of messages, it does not affect the system performance. In ETCR, the sustained buffer is used to keep all of replicas in a consistent state even if any of them fails. If the event tracker detects an error, the event tracker retransmits all entries of the sustained buffer to the live replicas. To describe this procedure in detail, we define three error cases: i) a head failure, ii) a middle failure, and iii) a tail failure.

In the case of a head failure, the client can detect an error event upon sending the update to a TCP socket. The client then instantly stops the sending task and prepares a recovery state. Prior to the recovery process, the client reorganizes replication window W' except the failure replica. The client then asks the sustained buffer if previous updates exist. If no previous update exists in the sustained buffer, it is because the client sends an update for the first time. In this case, the client continues to conduct a normal update with W'. If any previous update exists in the sustained buffer, the client sends an entry of the sustained buffer to W' prior to sending a normal update. After successfully re-sending a sustained buffer, the client resumes the normal update task.

Recovery of a middle or tail failure follows a similar procedure. For simplicity, suppose a failed replica as r_i. The server r_{i-1} always detects the failure of ri. After recognizing the failure of r_i, the server r_{i-1} sends an event response to the event handler of the client. The event response includes the failure status of r_i. The rest of the recover procedure is similar to the method used for dealing with a head failure. The event tracker instantly stops the normal updates and reorganizes replication window W' except for r_i. The client then re-sends the sustained buffer to W', and resumes a normal update task.

4 Evaluation and Discussion

In this section, we present an experimental evaluation of ETCR. Because our read protocol is very similar with CR, we only focus on measuring the performance for a write case. In addition, we describe our evaluation in two ways: i) normal write throughput and latency, and ii) the failure effect. The former is for verifying the performance of our model in an errorless environment, and the latter is for observing the ability to handle a server failure.

4.1 Environmental Setup

We implemented our algorithm in C. To compare our model with CR, we implemented a basic prototype for both ETCR and CR. The implementation consists of a client and a server. The client issues a write request to the server, and each server stores data in internal storage. We split each file into several replicas with a size of 64 MB. Every replica is distributed to each server. To reflect a real-world workload, the client is implemented on FUSE, which is an open-source software that provides a fast way to develop a user-level filesystem. The implementation is mounted on a local filesystem, and thus we can evaluate our model using the iozone, which is widely used for benchmarking a filesystem.

We performed the evaluation using up to 11 homogeneous linux servers: one for the client operation and the others for storing the replicas. All machines consist of the same hardware specifications: 2.4 GHz Intel Xeon E5620, 32 GB of memory, and a 256 GB SSD disk. All machines run on Linux kernel 2.6.32 and are connected through a gigabit Ethernet network. To ensure the fairness test, we used the default TCP settings. The load was generated by one dedicated client server using the iozone benchmark tool. The detailed input option is iozone -Ra -i 0 -r 16k -q 64k -s 4g -f /etcr/iozone. Every measurement was performed at least three times, and the average of all measurements was recorded.

4.2 Update Performance in Errorless Environment

To measure the ideal update performance in our model, we first suppose an errorless environment. All updates are successfully propagated to the next servers along with the chain topology. Each chain is made up of a number of replicas from 2 to 10. In addition, to reflect the various workloads in the real world, we let the I/O size be from 16 KB to 64 KB. Because we use the iozone benchmark tool, the client repeatedly sends update requests to the replica server.

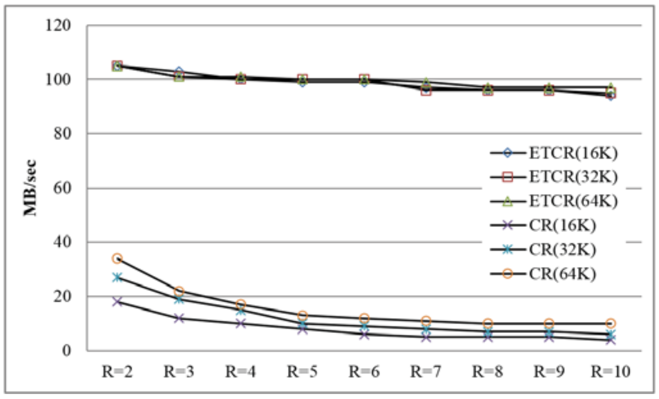


Figure 4. Write throughput in both ETCR and CR

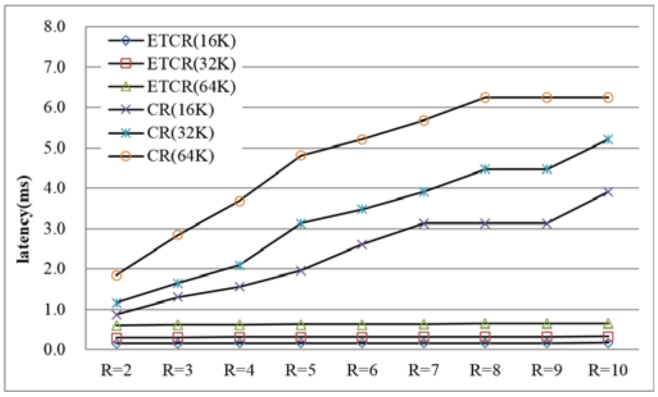


Figure 5. Update latency in both ETCR and CR

As shown in Figure 4, our model behaves as predicted by our analysis. First, our model provides an update performance of about 105 MB/s when the number of replicas is two, while CR only shows 19 to 38 MB/s. Such a difference is caused by the messaging protocol. In CR, sending of the next update is blocked until a response message for the previous one arrives from the tail. Because the client should always wait for response messages even if the update succeeds, a serious degradation in the update performance occurs. On the other hand, the results of our model outperform those of the CR. Because our model only handles an event response instead of a regulated response, the client can always send the next update to the head if the update does not fail. Second, another important point can be found in the X-axis. Our model shows a sustained performance regardless of the number of replicas. For ten replicas, our model shows 94 MB/s, which is a decrease of 10% compared with the case of two replicas. However, the decreasing ratio in the CR is almost 70% for the same conditions.

Figure 5 shows the results from the latency perspective. To achieve accurate results, we added the time-measuring macro into the client code. We measured the latency of every write request in microseconds, and the average of all latency was calculated after completing a file write. The chart shows the remarkable differences between our model and the CR. Our model keeps up the sustained latency in all cases, while the CR shows a relatively high latency as well as an increasing tendency in proportion to the number replicas. These differences result from the massaging protocol, as in the performance results in Figure 4.

4.3 Failure Effect

In a real storage environment, there are various failures that may occur. Although some systems can provide a great performance, they may be unsuitable for a real service if they are unable to handle failures. We therefore evaluated our model in terms of server failures. For this, we kill the process in a server during the iozone runs, and then observe both the message latency and I/O throughput until the recovery process is complete. The whole measurement is performed for both single server and multiple server failures.

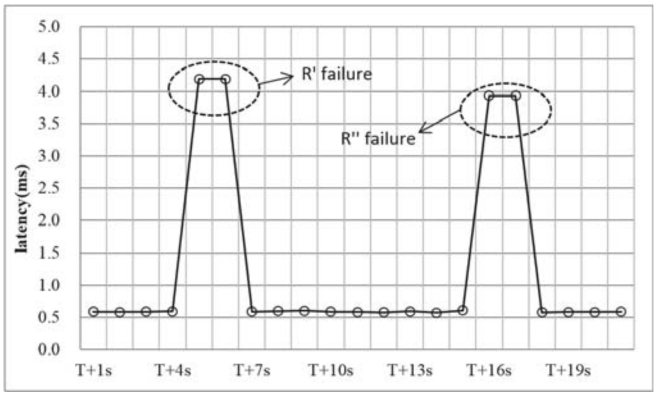


Figure 6. Latency effect on server failure

Figure 6 shows the effect of latency on such failures. The X-axis indicates a time history whose interval is 1 s. After killing the process in one of the servers in T+4, the latency is suddenly increased from 0.5 ms to 3.7 ms. This is because the event tracker instantly stops a normal update and goes into the recovery process. As mentioned earlier, the event tracker collects previous updates from the sustained buffer and sends them to the live replicas. Such a recovery process is performed from T+4 to T+6. After the whole recovery process is complete, the event tracker resumes the normal update in T+7. In addition, to evaluate the latency effect on multiple failures, we kill processes running on two servers in intervals of a few seconds. As shown in the chart on the right, our model can handle multiple failures as long as any of the replicas stay alive.

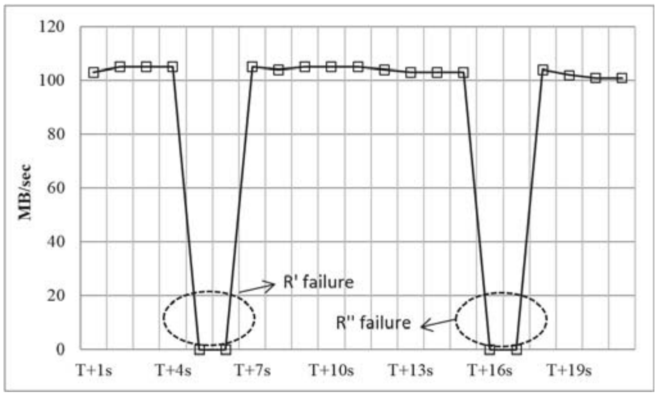


Figure 7. Throughput effect on server failure.

The effect of the throughput is shown in Figure 7. The results are similar to the latency effect. When a server failure occurs in any position in time, the whole recovery process is completed in 2 to 3 s. In such a period, the throughput drops to 0 because the event tracker stops sending normal updates. After completing the recovery process, the throughput reaches 100 MB/s again.

5 Conclusions

In this paper, we introduced the event tracking chain replication (ETCR) model designed for a large-scale distributed environment. The ETCR supports asynchronous update message propagation for scaling up the write throughput, as well as an additional event handling mechanism for keeping data in a consistent state. According to our experimental results, the ETCR achieves low latency and high throughput for write requests. We also observed a failure effect of the ETCR and showed that the recovery process is successfully completed in a few seconds for both a single server failure and multiple server failures. Further studies will focus on applying the ETCR to real distributed storage and justifying the effectiveness in the real world.

6 Acknowledgement

This work was supported by the ETRI R&D Program grant funded by the Korea government(MSIP) (15ZS1400, Development of Big Data Platform for Dual Mode Batch-Query Analytics).

7 References

[1] L. Gao et al., “Improving Availability and Performance with Application-Specific Data Replication,” IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 1, pp. 106-200, 2005.

[2] M. M. Deris et al., “Binary Vote Assignment on Grid For Efficient Access of Replicated Data,” Intl. Journal of Computer Mathematics, vol. 80, no. 12, pp. 1489-1498, 2003.

[3] M. Tang et al., “The impact on data replication on Job Scheduling Performance in the Data Grid”, Journal of Future Generation of Computer Systems, vol. 22, pp. 254-268, 2006.

[4] S. Ghemawat, H. Gobioff, H., and S.T. Leung, “The Google file system,” 19th Symposium on Operating Systems Principles, Lake George, NY, pp. 29-43, 2003.

[5] Hadoop: Open source implementation of MapReduce. http://lucene. apache.org/hadoop.

[6] Herlihy et al., “Linearizability: A correctness condition for concurrent objects,” TOPLAS, vol. 12, no. 3, pp. 463-492, 1990.

[7] Bernstein et al., “The failure and recovery problem for replicated databases,” 2nd Symposium on Principles of Distributed Computing(PODC), pp. 114-122. 1983.

[8] Bernstein et al., Concurrency Control and Recovery in Database Systems, Addison Wesley, Boston, MA, 1987, Available at http://research.microsoft.com.

[9] I. Keidar and L. Zhou, “Building reliable large-scale distributed systems: When theory meets practice,” ACM Special Interest Group on Algorithms and Computation Theory(SIGACT), vol. 40, no. 3, Sept. 2009.

[10] L. Lamport and K. Marzullo, “The part-time parliament,” ACM Transactions on Computer Systems, vol. 16, no. 133-169, 1998.

[11] F. B. Schneider, “Implementing fault-tolerant services using the state machine approach: A tutorial,” ACM Computing Surveys, vol. 22, no. 4, pp. 299-319, 1990.

[12] J. Yang et al., “Modist: Transparent model checking of unmodified distributed systems,” NSDI, 2009.

[13] R. D. Prisco and B. Lampson, Revisiting the paxos algorithm,” Proceedings of the 11th International Workshop on Distributed Algorithms (WDAG 97), vol. 1320 Lecture Notes in Computer Science, pages 111-125. Springer-Verlag, 1997.

[14] R. van Renesse and F. B. Schneider, “Chain replication for supporting high throughput and availability,” USENIX Operation Systems Design and Implementation(OSDI), 2004.

[15] Jeff Terrace, Michael J. Freedman, "High Throughput Chain Replication Model for read-mostly workloads", In Proceeding of International Conference on USENIX, 2009

Analysis of Network Segmentation Techniques in Cloud Data Centers

Ramaswamy Chandramouli
Computer Security Division, Information Technology Laboratory
National Institute of Standards & Technology
100 Bureau Drive, Gaithersburg, MD, USA

Abstract – *Cloud Data centers are predominantly made up of Virtualized hosts. The networking infrastructure in a cloud (virtualized) data center, therefore, consists of the combination of physical IP network (data center fabric) and the virtual network residing in virtualized hosts. Network Segmentation (Isolation), Traffic flow control using firewalls and IDS/IPS form the primary network-based security techniques with the first one as the foundation for the other two. In this paper, we describe and analyze three generations of network segmentation techniques – Virtual Switches & Physical NIC-based, VLAN-based & Overlay-based. We take a detailed look at the overlay-based virtual network segmentation and its characteristics such as scalability and ease of configuration*.

Keywords: Virtual Machine; Virtual Network; Hypervisor; VLAN; Overlay-based Network

1 Introduction

Cloud data centers are computing and networking infrastructures configured for offering cloud-based services such as Infrastructure as a Service (IaaS). A great majority of servers or hosts in these centers will be found to be virtualized hosts (having the server virtualization product – the hypervisor running inside them) for reasons of scalability, agility, cost-efficiency of operations and perhaps even security. We will call these hosts as hypervisor hosts or virtualized hosts throughout this paper. Cloud data centers, because of the predominant presence of hypervisor/virtualized hosts are also called Virtualized data centers as well. A Hypervisor host has multiple virtual machines (VMs) running in each of them and in each VM one or more applications may be hosted. These applications therefore are referred to as Virtual workloads.

From the point of view of user accessibility, connectivity and security, the VMs play the same role as physical hosts in non-virtualized data centers, since as computing nodes (or endpoints), VMs house the resources that provide the functionality for one or more aspects of any application – user interface, application logic or data access. Hence it is no surprise that applications exclusively providing only one of these functional aspects are categorized as belonging to a Tier. Hence, a common architecture for applications is a 3-tier architecture consisting of Web, Application and Database tiers providing respectively the functions – user interface, application logic and data access. In other words, a VM could be hosting a Web tier, Application tier or Database tier or a combination of tiers in a data center.

From the description of the role of VMs, it should be obvious that they are the counterparts of Web Server, Application Server and Database Server in virtualized data centers, their only difference being that they run on virtual machines instead of on physical machines. Hence VMs are entities or nodes that need to be connected to each other (e.g., enable one application tier to communicate with another – A web server to communicate with Application server) and also be accessible to entities external to the data center (e.g., users accessing the application running on the VMs). The twin needs of accessibility and network connectivity, in turn, requires that each of these VMs (just like their physical counterparts) must have a distinct set of network identifiers or addresses (i.e., MAC address, IP address etc). Therefore, it goes without saying, that VMs are targets to be protected as well.

In spite of the the above common requirements between VMs and physical servers, the connectivity paradigm in which VMs are involved in brings in a different networking picture for the data center as a whole. Since multiple VMs reside in a single physical host, they may need to be connected to each other and also to the network on which the physical (virtualized) host itself is a node (enterprise or data center network). Hence there exists a capability in every server virtualization product, to define a network inside a virtualized host. This network unlike the overall data center network, is entirely software-defined and is a feature needed to provide connectivity among the VMs residing in a virtualized host as well as to provide connectivity to VMs residing in a virtualized host to the data center network. This network inside a physical (virtualized) host is therefore a virtual network with its nodes (i.e., VMs) being the virtual nodes. Connectivity among the VMs (i.e., virtual nodes) and between a VM to the data center network through the physical network interface cards (Physical NICs) of the virtualized host are all enabled through another software-defined element or a set of elements called the virtual switches. The VMs themselves communicate to the virtual switches using software-defined virtual network interface cards (virtual NICs). Thus we see that a virtual network inside a virtualized host is made up of Virtual NICs and virtual switches with some communication links from virtual switches (called uplinks of the virtual switches) terminating in one or more physical NICs of the virtualized host.

The above description, together with our knowledge of the network topologies found in conventional data centers, now provides us with the picture of networking infrastructure in a cloud (or virtualized) data center - as one that is made up of the combination of the physical network (called the datacenter fabric) and the virtual network residing in each of the virtualized hosts. We all know the *three network-based security techniques* used for protection of resources in a conventional data center without any virtualized hosts, such as: (a)Network segmentation or isolation, (b) control of network traffic flows based on various parameters using devices called Firewalls and (c) use of special-purpose network snooping devices (Intrusion Detection Systems (IDS)/Intrusion Prevention Systems (IPS)) to detect malicious traffic entering into or going out of the network. We consider these to be *primary network-based security protections*, although we do recognize that network services such as Dynamic Host Configuration Protocol (DHCP), Network Address Translation (NAT), Load Balancers etc do contribute to the availability aspect of the network security. Out of these three primary network-based security techniques, the network segmentation forms a foundational technique for supporting other two forms of network-based security protection. The goal of network segmentation in cloud data center environment is to enable logical separation (or isolation) among customers or tenants of (say) an IaaS cloud service.

The objectives of this paper are twofold. One is to describe in sufficient level of detail, the network segmentation techniques available in cloud data centers whose network infrastructure are made up of physical network and virtual network components. The second objective is to analyze the security strengths and weaknesses of each of these network segmentation techniques. Based on the nature of evolution and the features they possess, we categorize these network segmentation techniques into three generations.

The organization of this paper is as follows. Since the focus of this paper is on network segmentation, we would like to provide some clarity on the concept of network segmentation in section 2. In Section 3, we describe and outline the strengths and limitations of the first generation network segmentation technique for cloud data centers. This solution is based on the coarse segmentation of a data center network into external, demilitarized zone (DMZ) and internal network. The segmentation of the network using the concept of virtual LANs (segmentation at the layer 2 (L2) or data link layer) and its advantages and weaknesses are the topic of Section 4. In section 5, we present in somewhat great detail (compared to the other two network segmentation techniques), the network segmentation technique using the concept of network overlays and analyze in detail the advantages of this technique in terms of the security assurances it can provide. The conclusions from our analysis are presented in Section 6.

2 Clarification on the Concept of Network Segmentation

The term network segmentation in most contexts only implies logical segmentation and not physical segmentation. Physical segmentation of a data center network is enabled using physical devices such as the Top of the Rack (ToR) switches, aggregate switches, core switches and routers as well as the physical Network Interface Cards (NICs) in each of the hosts. Space availability (for housing all physical networking devices), costs (costs of equipment procurement, installation and power requirements), and finally the complexity of configuration and management limit the extent of physical segmentation that is possible in a data center. Logical network segmentation, on the other hand, requires the deployment of (is only relevant in the context of) a logical or virtual network on top of the physical network in the data center. Hence, in this paper, the network segmentation approach is always in the context of the underlying virtual networking technology. For example, when we talk about VLAN-based network segmentation, it is in the context of a *VLAN-based virtual network*, which is the type of the underlying virtual network.

3 Description and Analysis of First Generation Network Segmentation (Virtual Switches & Physical NICs)

The first generation network-based solution for protection of VMs merely consisted of creating a single DMZ [1] (to act as a buffer between an enterprise's internal and external network) or a combination of DMZ and targeted network segments. This solution uses the virtual switches (VS) inside the hypervisor and the physical network interface cards (physical NICs or pNICs) of the hypervisor. A configuration for creating a single DMZ within a hypervisor host is given in Figure 1.

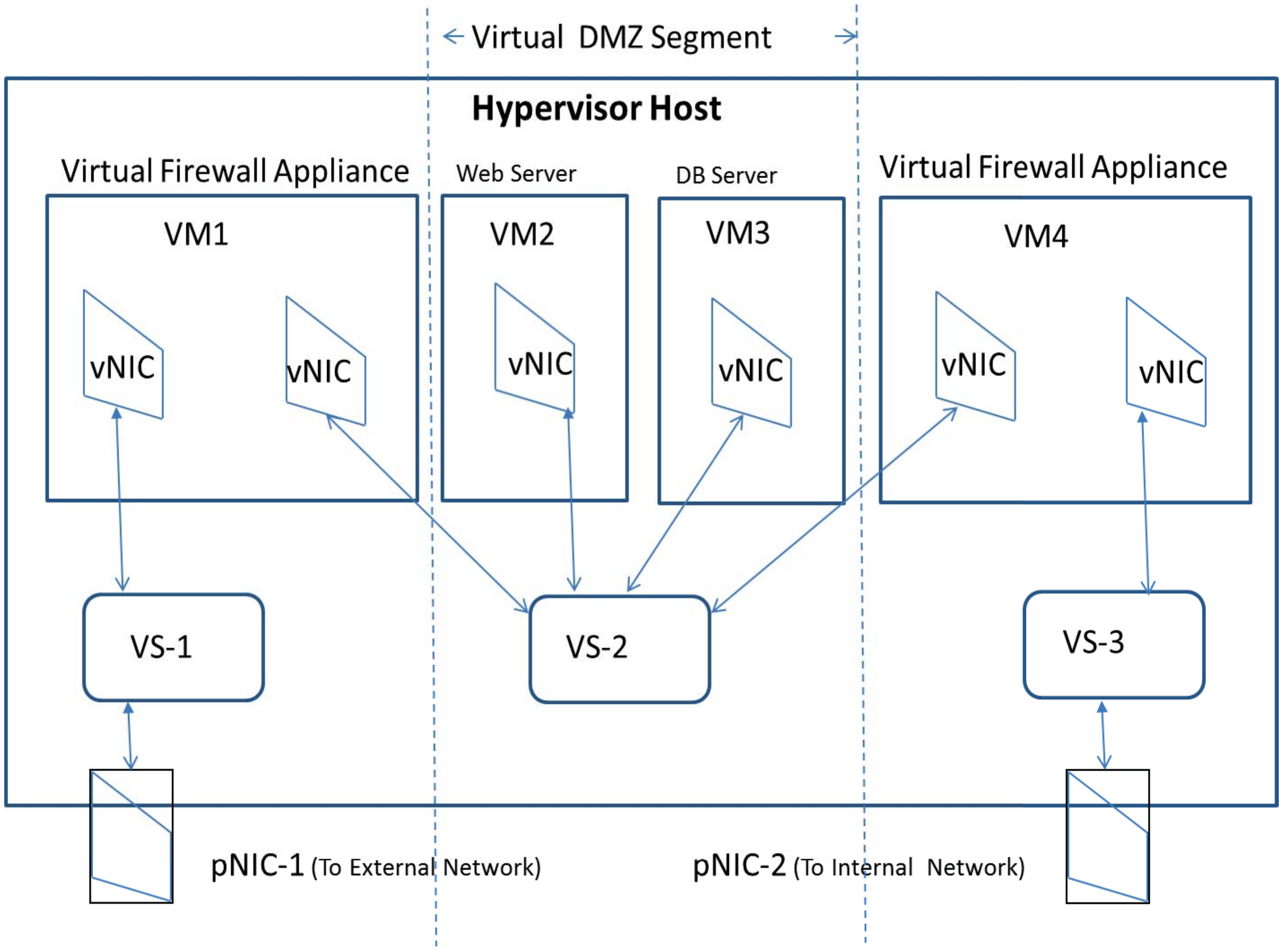

Virtual DMZ Segment
Hypervisor Host
Virtual Firewall Appliance
Web Server
DB Server
Virtual Firewall Appliance
VM1
VM2
VM3
VM4
vNIC
vNIC
vNIC
vNIC
vNIC
vNIC
VS-1
VS-2
VS-3
pNIC-1 (To External Network)
pNIC-2 (To Internal Network)

Figure 1 –Network Segmentation (Using Virtual Switches and Physical NICs)

As one can see from Figure 1, a virtual network-based DMZ has been constructed using the same architectural principles that are used to construct a DMZ in a physical network. The external firewall (between the external network and DMZ) is constructed using a virtual firewall (or a firewall appliance) running in VM1. VM1 is a multi-homed virtual server that has two virtual network interface cards (virtual NICs). One virtual NIC is connected to the external network through the physical NIC labeled as pNIC-1 via the virtual switch VS-1. The other virtual NIC on VM1 is connected to the virtual switch VS-2 to which VMs hosting a webserver (VM2) and database server (VM3) are also connected. The internal firewall in the virtual network-based DMZ (between the DMZ and internal network) is constructed using a virtual firewall hosted on VM4. Like VM1,VM4 is a multi-homed virtual server that has two virtual NICs. One virtual NIC is connected to the internal network through the physical NIC pNIC-2 of the hypervisor host. The other virtual NIC on VM4 is connected to the virtual switch VS-2 which we have said already is connected to VMs hosting the web server and database server. Any external packet landing in VM1 through vNIC-1 can only reach the webserver in VM2 (or database server in VM3) if allowed by firewall rules of the virtual firewall hosted in VM1. Similarly any traffic from VM2 or VM3 can only reach the internal network through VM4, if it is allowed by firewall rules of the virtual firewall hosted in VM4.

In figure 1, there is only one internal logical network segment since there is only one internal-only virtual switch (not connected to any physical NIC). It is also possible to have multiple internal segments as well by using more internal-only virtual switches. The main characteristic features and limitations of this approach to achieve network segmentation are the following:

(a)Lack of Scalability: Increase in the number of logical network segments has to be achieved by increasing the number of VMs with multiple vNICs, by increasing the number of virtual switches inside a hypervisor and possibly by increasing the number of pNICs for the physical host where the hypervisor resides. Since there is limit to these values, this configuration is not scalable.

(b)Another limitation of this approach for network segmentation is configuration complexity and proneness to errors. Identical configurations have to be configured in many VMs, making the configuration error-prone.

(c)A network segment cannot span more than virtualized (hypervisor) host (since segmentation is obtained through virtual switch connectivity inside the virtualized host) – again making scalability an issue.

4 Description and Analysis of Second Generation Network Segmentation (VLAN)

The second generation of network segmentation for protecting virtualized infrastructure uses the concept of VLANs. A VLAN (Virtual Local Area Network) is a logical group of devices or users, grouped by function, department or application irrespective of their physical location on the LAN [2,3]. The grouping is logical and obtained by assignment of an identifier called VLAN ID to one or more ports of a switch and connecting the computing units (physical servers or VMs) to those ports. The basic objective of VLAN is logical network segmentation in order to provide broadcast containment [4]. Devices on one VLAN can only communicate directly with devices on the same VLAN and a router is needed for communication between devices on different VLANs. The VLAN implementation in virtualized hosts is enabled using virtual switches that are VLAN-aware so that VMs (just like physical hosts/servers) can become VLAN end nodes. In other words, VLAN IDs are assigned to ports of a virtual switch inside a hypervisor kernel and VMs are assigned to appropriate ports based on their VLAN membership. Since in a cloud data center, VMs may belong to different consumers or cloud users, the cloud provider is thus able to provide one or more logical or virtual network segments for each tenant (for isolation of their computing/storage resources) by making VMs belonging to each of them being assigned/connected to a different VLAN segment. These VLAN-capable virtual switches can perform tagging of all packets going out of a VM with a VLAN tag (depending upon which port it has received the packet from) and can route an incoming packet with a specific VLAN tag to the appropriate VM by sending it through a port whose VLAN ID assignment equals the VLAN tag of the packet. An example of a VLAN-based virtual network segmentation inside a hypervisor host is given in Figure 2.

The characteristics, advantages and limitations of a VLAN-based network segmentation approach are:

(a) Unlike the first generation network segmentation approach achieved using just a combination of vNICs, virtual switches and pNICs, VLAN configuration is based on the individual ports or groups of ports within each virtual switch, enabling a large number of virtual network segments (a virtual switch typically can support 64 ports). Further, the virtual network segments (each identified by a VLAN ID) can span more than one virtualized host.

(b) However, since the size of a VLAN ID is 12 bits, the maximum number of virtual segments possible throughout the data center is limited to approximately 4000.

(c) A VLAN implementation expects all switches (ToR and Core) to know the MAC addresses of all VMs in all VLANs. Hence, there exists the possibility of a situation where the VLAN ID to MAC address table of a ToR switch overflows, and packets intended for a particular VLAN ID, may flood all links emanating from that switch to all servers instead of on just those links going into servers that hosts VMs belonging to that VLAN.

(d) A top of the rack switch (ToR) switch has many ports, at least one port for each physical server in the rack. In the simplest implementation of VLAN, every server port on the ToR switch is enabled for all VLANs. Hence the connection linking those ports to the hypervisor host becomes a trunking link (carry traffic corresponding to multiple VLANs). Hence in the case of multicast (broadcast) messages, it becomes the responsibility of hypervisor kernel to process each of these messages for every VLAN on the network, even when the hypervisor is not hosting any active VM belonging to that VLAN [5].

(e) In a network that is designed to be aware of the presence of VMs, there will still be flooding on some ToR switch to server links. The flooding will be proportional to the number of VLANs active in that hypervisor host and the number of VMs assigned to those VLANs.

(f) There are several configuration issues that must be carefully addressed in the VLAN-based network segmentation solution. First of all, every hypervisor host is connected to an external physical switch for linking the former to the enterprise network. The VLAN configuration in this physical switch must exactly match with the VLANs configured in the virtual switches of the hypervisor host to which the physical switch is connected. Further, the links connecting the hypervisor host to these physical switches must all be configured as trunk ports (capable of supporting traffic belonging to multiple VLANs). Thirdly VMs can be migrated from one hypervisor host to another only if the source and target port group number (or VLAN identifier) is the same. In the interest of load balancing and availability, there must be a large population of hypervisor hosts available for this task and this in turn necessitates building a large VLAN (one that spans many hypervisor hosts).

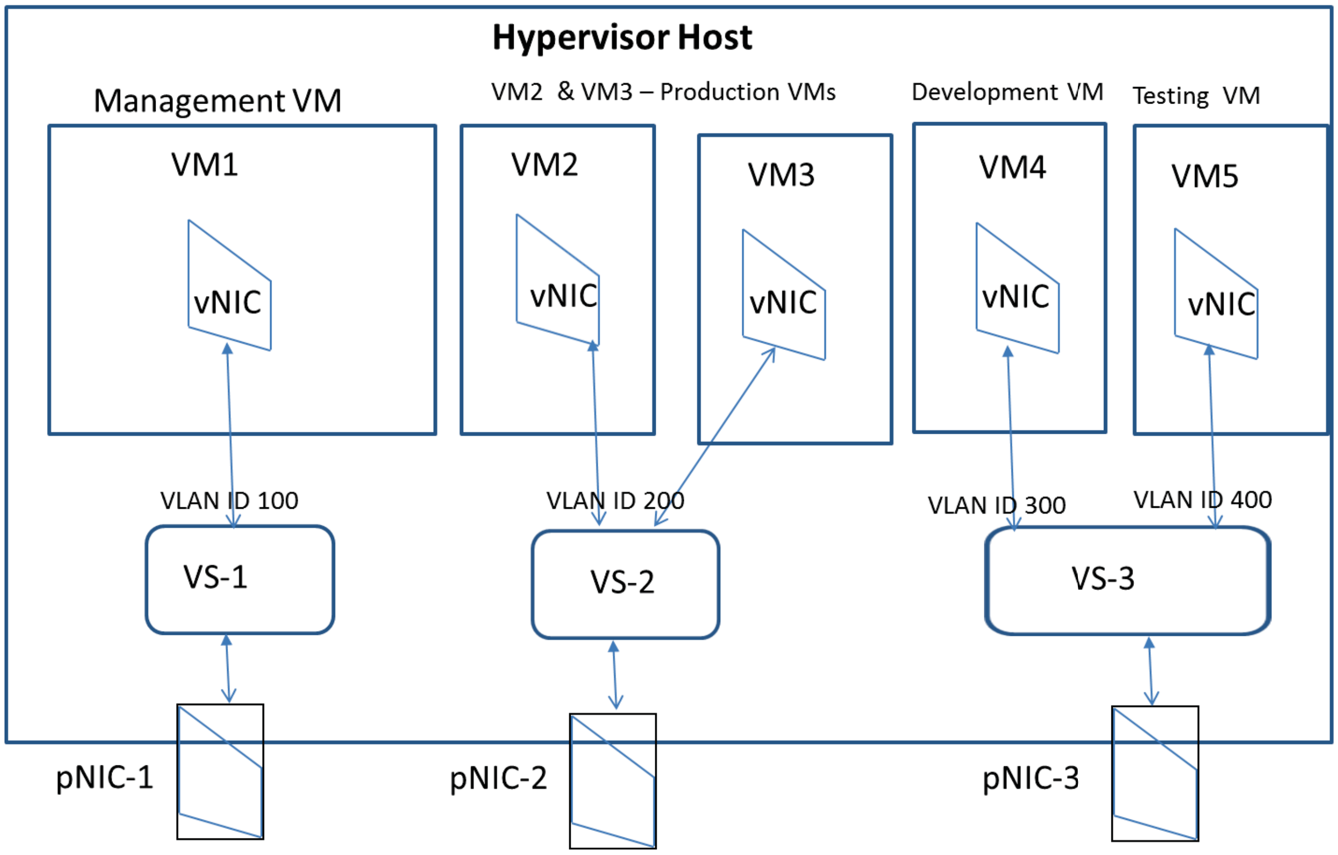


Figure 2. VLAN-based Network Segmentation

5 Description and Analysis of Third Generation Network Segmentation

We would like to characterize the overlay-based virtual networking technology as third generation network segmentation solution. We will look at a brief description of Overlay-based virtual networking before delving into the analysis of its features and strengths.

5.1 Description of Overlay-based Virtual Network Segmentation in Cloud Data Center

Let us briefly look at the implementation of Overlay Network and its associated segmentation in a cloud data center. The physical network topology of the data center is made up of 2 or 3 layers with a reliable IP backbone. A two layer topology consists of just Top of the Rack switches and the core switches. The top of the rack (ToR) switches sit on top of a rack that contains multiple servers or hosts. Most of the hosts in our context are virtualized hosts that run hypervisor software inside each of them which in turn hosts multiple VMs inside each. Several ToR switches are connected to some core switches. The core switches provide connectivity to the outside world for the data center as a whole by connecting to the Internet or a VPN. In a data center with three layer network topology, there is an intermediate layer of switches between ToR switches and core switches called Aggregation switches. Aggregation switches provide connectivity between ToR switches and core switches as they are connected in a mesh topology to ToR switches (every aggregation switch is connected to all ToR switches in the data center).

In the Overlay-based virtual networking, isolation is realized by encapsulating an Ethernet frame received from a VM as follows. Out of the three encapsulation schemes (or overlay schemes) – VXLAN, GRE and STT [5], let us now look at the encapsulation process in VXLAN [6] through components shown in Figure 3. First, the Ethernet frame received from a VM, that contains the MAC address of destination VM is encapsulated in two stages: (a) First with the 24 bit VXLAN ID (virtual Layer 2 (L2) segment) to which the sending/receiving VM belongs and (b) two, with the source/destination IP address of VXLAN tunnel endpoints (VTEP) [7], that are kernel modules residing in the hypervisors of sending/receiving VMs respectively. The source IP address is the IP address of VTEP that is generating the encapsulated packet and the destination IP address is the IP address of VTEP in a remote hypervisor host sitting anywhere in the data center network that houses the destination VM. Thus, we see that VXLAN encapsulation enables creation of a virtual Layer 2 segment that can span not only different hypervisor hosts but also IP subnets within the data center.

The VXLAN based network segmentation can be configured to provide isolation among resources of multiple tenants of a cloud data center as follows. A particular tenant can be assigned two or more VXLAN segments (or IDs). The tenant can make use of multiple VXLAN segments by assigning VMs hosting each tier (Web, Application or Database) to the same or different VXLAN segments. If VMs belonging to a client are in different VXLAN segments, selective connectivity can be established among those VXLAN segments belonging to the same tenant through suitable firewall configurations, while communication between VXLAN segments belonging to different tenants can be prohibited.

5.2 Analysis of Overlay-based Network Segmentation

The features and advantages of Overlay-based network segmentation are as follows:

(a) With network devices supporting programmability through standardized interfaces (being part of the SDN framework) [8], many of the network security configuration can be automated – such as the firewall rules etc

(b) A VXLAN network identifier (VNID) is a 24 bit field compared to the 12 bit VLAN ID. Hence the namespace for VXLANs is about 16 million as opposed to 4096 for VLANs. Further VXLAN is Layer 2 overlay scheme over a Layer 3 work. Hence unlike a VLAN scheme which has to use the Spanning Tree Routing Protocol to forward packets, VXLANs can use the ECMP protocol of Layer 3 [9], thus efficiently utilizing all the available network links in the network fabric of the data center.

(c) A highly scalable network security configuration is possible not only due to the unlimited availability of VXLAN IDs, but also due to the fact that the encapsulating frame is IP/UDP packet, and hence the number of virtual networks is limited only by the size of IP subnets that can be defined within the data center and not by the number of ports in virtual switches as in the case of VLAN-based network configuration. Further, by using internal, non-routable IP addresses for VMs (using DHCP and NAT capabilities) running within virtualized hosts, the number of virtual networks that can be realized is even higher.

(d) In a data center that is offered for IaaS cloud service, isolation between the tenants (cloud service subscribers) can be achieved by assigning each of them at least one VXLAN segment (denoted by a unique VXLAN ID). Since VXLAN is a logical L2 layer network (called overlay network) running on top of a physical L3 layer (IP) network inside the data center, the latter is independent of the former. The consequence of this feature is that it gives the freedom to locate the computing and/or storage nodes belonging to a particular client in any physical segment of the data center network. This freedom and flexibility in turn, helps to locate those computing/storage resources based on performance (high performance VMs for data/compute intensive workloads) and load balancing considerations.

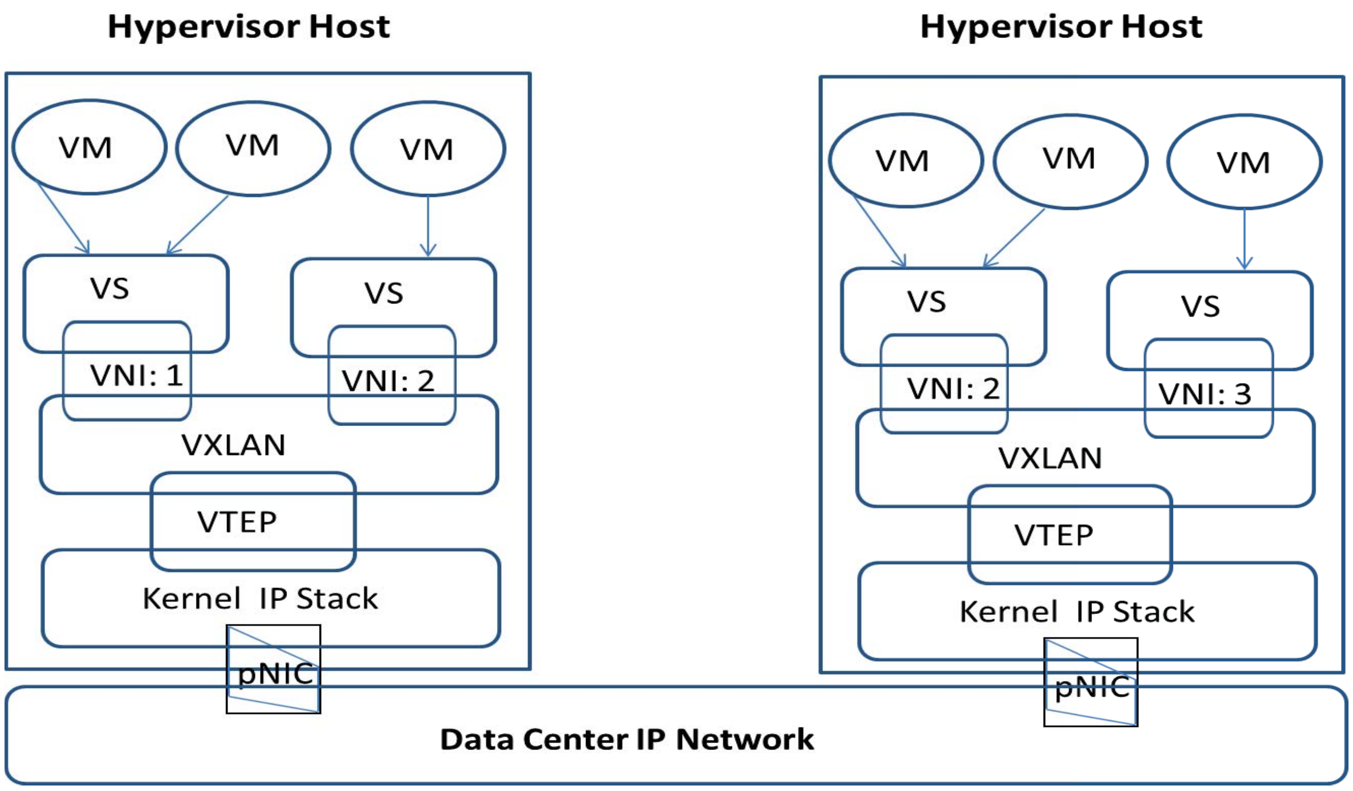


Figure 3 – Overlay-based Virtual Network Segmentation

6 Conclusions of our Analysis

With the increasing adoption of cloud services by large enterprises that have to host multi-tier applications, the data center network administrators need a flexible virtual networking topology with capability to provide the necessary isolation through network segmentation. At the same time, it is necessary that these virtual network segments span multiple, arbitrary IP subnets of the data center and also several hypervisor clusters. As of now, the only virtual networking technology that can provide these capabilities without a great deal of physical network reconfiguration or addition of networking resources is the overlay-based virtual networking. This degree of independence between the virtual networks and the physical networks provided by overlay-based techniques also provides the *scalability and configuration ease* that are needed for maintaining the logical network segmentation within large data centers. Thus, the overlay-based network segmentation, has become an economically and operationally viable approach for securely supporting multi-tenant workloads for IaaS cloud providers.

7 References

[1] “DMZ Virtualization with VMware Infrastructure”. http://www.vmware.com/files/pdf/dmz_virtualization_vmware_infra_wp.pdf

[2] “MAC Bridges and Virtual Bridged LANs”. https://www.ietf.org/meeting/86/tutorials/86-IEEE-8021-Thaler.pdf

[3] “IEEE 802.1Q Virtual LANs (VLANs)”. http://www.ieee802.org/1/pages/802.1Q.html une 2014]

[4] Abdul Hameed, Adnan Noor Mian.“Finding Efficient VLAN Topology for better broadcast containment”. Third International Conference on the Network of the Future (NOF), Gammarth, Nov 21-23, 2012.

[5] “Overlay Virtual Networking and SDDC”. **http://my.ipspace.net/bin/list?id=xSDNOverlay**

[6] **“**Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks”. https://tools.ietf.org/html/rfc7348

[7] “VXLAN Overview: Cisco Nexus 9000 Series Switches”. http://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/white-paper-c11-729383.pdf

[8] Open Networking Foundation, http://www.opennetworking.org

[9] “Scaling Overlay Virtual Networks”. http://content.ipspace.net/get/Scaling%20Overlay%20Virtual%20Networks.pdf

A Dynamic and Efficient Coalition Formation Game in Cloud Federation for Multimedia Applications

Mohammad Mehedi Hassan, Abdulhameed Alelaiwi and Atif Alamri
College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
E-mail: mmhassan, aalelaiwi, atif@ksu.edu.sa

Abstract— *Recently, the growing demand for various multimedia services and applications has imposed great challenges in terms of real-time processing, storing, analyzing and sharing of large media data. Federation among media cloud providers can be a promising solution for effectively hosting multimedia applications and provisioning of Quality of Service (QoS)-assured services. In this paper, we address the problem of efficient federation formation mechanism among media cloud providers (CPs) that motivates them to cooperate to fulfill the dynamic resource demands of users for running multimedia workloads. We design a novel media cloud federation formation mechanism based on cooperative game theory. By collaborating to trustworthy providers, our proposed method looks for building federations to minimize the penalties due to possible violation of service quality by untrustworthy providers, and thereby, generate more profit. Simulation results demonstrated that the media cloud federation obtained by the proposed mechanism is stable and provide higher profit for the participating CPs.*

Keywords: media cloud federation, coalition formation, game theory, multimedia services

1. Introduction

Recently, multimedia is emerging as a popular service that involves in live broadcast, on demand videos, games, video conference and others. Using wireless or wired networks, various users can ubiquitously access and share diverse media services. As the media service demand is increasing day by day, the media traffic volume for all types of services will also increase in near future. This speedy growth of the demand for multimedia services has led to great challenges in terms of large volume of data processing, storing, analyzing and sharing among millions of users with high interactivity [1].

In order to address the above issues, we propose to utilize the federation paradigm [2][3][4] among various media cloud providers to serve dynamic and fast growing multimedia services. Cloud federation presents a promising approach for effectively hosting multimedia applications and provision of QoS-assured services. Leveraging infrastructure provisioned by multiple partnership media cloud service providers, we solve problems like dynamic resource allocation and improvement of user experience. The federation creates a pool of virtualized resources which are offered to users as different types of VM instances. The CPs in a federation cooperate in order to provide the resources requested by users with guaranteed QoS. As a result forming cloud federations helps to achieve greater scalability and performance. It may also provide the resources at lower costs[5].

However, one of the major challenging issues in this environment is how to define an effective federation formation mechanism among CPs that motivates them to cooperate to fulfill the resource demands of users [5][6][7] for big data workloads. This issue is very important since it determines how much revenue each CP gets when participating in a federation by proving a certain number of VM resources. In addition, the fairness is also an important issue to be considered which ensures that each CP should gain a revenue based on the amount of VM resources contributed to the federation (Gain-as-you-contribute in short). The stability of a federation is also important and needs to be analyzed for finding whether it is profitable for a CP to work with other CPs in the federation or not.

Currently, several strategies [5][6][8][9][1][10][7][11] are present in the literature for forming cloud federations. Among these strategies, game-theory-based federation formation mechanisms are becoming popular[5][6][8]. However, all of these mechanisms mostly focus on forming federation based on the maximum individual profit gained by the CPs in a federation. The profit of any CP varies based on the revenue it can gain by running various types and number of VM resources and the cost of providing those resources. None of these approaches take into account the risk of selecting unreliable collaborating CPs in a federation that may result in additional penalty cost due to SLA violation and finally affect the other CPs reputation. Therefore, we argue that a trust model is also necessary to find the most promising collaborators for forming cloud federation.

In this paper, we model the federation formation among CPs as a coalition game, where CPs decide to form a federation to allocate various types of VM instances dynamically, based on usersâĂŹ requests with QoS guarantee. The coalition game is modeled as a hedonic game [5][6] whereby each CP can order and compare all the possible coalitions it belongs to based on its preference model satisfying fairness and stability properties. We derive a trust model specifically designed to assist a CP in making its own preference model about federation formation. We analyze the properties of our proposed cloud federation formation game and perform extensive experiments to investigate its properties.

The paper is organized as follows: In section 2, we present the overall system model and mathematical problem formulation. In section 3, we describe the proposed cloud federation formation game in detail. In section 4, we evaluate the effectiveness of the proposed game in a cloud federation environment, and finally, in section 5, we present our conclusions and future work.

2. System Model and Mathematical Formulation

We first describe the system model consisting of a set of media CPs, several trust evaluation agent (TEA), some cloud users and a broker as a mediator. Each media CP has a TEA which provides trust-related information services to other CP's TEAs and users [12]. One TEA can represent multiple CPs, and one CP can delegate multiple TEAs. A TEA can derive the objective trust of CPs for each QoS factors from the trust information received from monitoring information process agents and users. It also collects the trust feedback ratings sent by other TEAs of other CPs for services that they have used, in order to build up the subjective trust of each service. Furthermore, TEAs can also interact with each other in order to exchange and propagate trust information. The broker in a federation is a trusted third party responsible for receiving cloud users VM requests, motivating various CPs to cooperatively supply required amount of VM resources in the federation, receiving the payment from users, and dividing the profit among CPs in the federation.

We assume that a set of IaaS CPs $P = \{P_i^t \,|i = 1...m\}$ is available at time t where each CP i consists of a set of physical servers defined as $S = \{S_{ik}|k = 1...ns\}$. In order to describe a physical server in general, we use ϕ_{ik}, μ_{ik}, and θ_{ik} to represent any CP i's total amount of CPU cores (expressed in GHz), total amount of memory space (expressed in GB), total amount of storage (expressed in TB), respectively. Note that each provider reserves a specific capacity for its own users, and specifies the available capacity to be shared in the federation based on its load. Each physical server of a CP i provides its resources (CPU, memory and storage) in the form of VM instances to cloud users. The IaaS CPs offer various types of VM instances (small, medium or large) [], where each VM instance provides a specific number of CPU capacity, amount of memory, and amount of storage. Let $v_{ij}^t(i = 1...m, j = 1...n)$ denotes the available VM resources of type n provided by any CP i at time t. Each VM resource of type $v_{ij}^t(i = 1...m, j = 1...n)$ is characterized by the number of cpu cores, z_{ij}^{CPU}, the amount of memory, z_{ij}^{Mem} and the amount of storage provided, z_{ij}^{Str}.

A cloud user sends a request to a broker, consisting of the number of VM instances of each type needed. A request is denoted by $G = \{G_j^t|j = 1...n\}$, where G_j^t is the total number of requested VM instances of type j , $j = 1...n$.. To fulfill the request of the user, the broker first sets a price $\Pr_{CP}^t(v_j^t)$ on each type of VM instance j. Based on the price, cloud providers are encouraged to form a federation to provide the required VM resources. Therefore, the sum of the VM resources supplied to any broker should be $\sum\limits_{i=1,j=1}^{m,n} v_{ij}^t = G_j^t$. Note that the final price given to broker by users is independent of the price given to CPs for providing the VM instances.

As all CPs are rational, they form a coalition in such a way that maximizes their individual profit. The total profit of any CP i is the difference between its global revenue rate (obtained for hosting a set of v_{ij}^t units of VM resources) and its global cost rate (that it incurs to run such VMs). It can be defined as follows:

$$Profit(v_{ij}^t) = v_{ij}^t \cdot \underset{CP}{\overset{t}{\Pr}}(v_j^t) - \sigma_{ij}^{\cos t} \tag{1}$$

The first term $v_{ij}^t \cdot \Pr_{CP}^t(v_j^t)$ in eq. (1) is the total revenue a CP can earn by providing a number of VM resources. The second term $\sigma_{ij}^{\cos t}$ is the total cost of a CP P_i when providing VM resources. This cost consists of the following:

- production cost, denoted by Y_i^t, which is the cost of producing first unit of VM resource for any provider i during a certain period t and
- the risk cost, denoted by $\underset{i,l}{\overset{RiskCost}{\Pr}}$, which is the additional cost a CP P_i need to pay it it selects an unreliable CP P_ls to form a federation that violates the SLA with users. This cost can be reduced if CP i joins a federation with trustworthy collaborators. The risk cost varies based on the trust levels of collaborator CP P_l in a federation.

Now, the total cost of a CP P_i in order to supply v_{ij}^t units of the VM resource, is defined as follows:

$$\sigma_{ij}^{\cos t} = \frac{Y_i^t \cdot v_{ij}^{t^{1+\log_2\alpha}}}{1+\log_2\alpha} + \underset{i,l}{\overset{RiskCost}{\Pr}}) \tag{2}$$

The production cost Y in eq. (2) is based on the learning curve model [1] which assumes that as the number of production units are doubled, the marginal cost of production decreases by a fixed factor α. It has been reported [1] that the learning factors are typically within the range (0.75, 0.9).

The risk cost in eq. (2) can be calculated as follows:

$$\underset{i,l}{\overset{RiskCost}{\Pr}} = v_{ij}^t \cdot \underset{CP}{\overset{t}{\Pr}}(G_j^t) \times (1 - e^{-\left(1-TL_{(i,l)}(t,c)\right)}) \tag{3}$$

The first term in eq. (3) is the total revenue the CP i can obtain by providing the V_{ij} resources and the second term is the trust value of CP i towards CP l based on the trust level $TL_{(i,l)}(t,c)$, where t is the time and c is the context. The detail description of calculating the trust level for any CP can be found in Section 4.3.

Now, the goal is to form a suitable federation of trustworthy media CPs based on the unit price and total amount of VM resource, announced by the broker. Such a federation will allow the CPs to cooperatively provide required number of VM resources in such a way to maximize their own utility or profit.

3. The Proposed Cooperative Cloud Federation Formation Game

In this section, we introduce our proposed media cloud federation formation game.In this game, a broker announces a price or revenue rate along with the total amount of VM resources required by the media cloud users. Accordingly, a set of available CPs forms a coalition to cooperatively fulfill the requested user demands. However, there are many different coalitions can be formed, each one differing from the other ones in terms of the profit they bring to their members. Therefore, the proposed game model will help to investigate the stability of different federation structures and find the best federation model that maximizes each CPs profit by considering trustworthy collaborator CPs. We consider a coalition formation cooperative game with transferable utility [5][6]. The proposed game also satisfies the two main properties, fairness and stability, which are very important for any CP to decide which coalition to join. The profit obtained by a federation should be fairly divided among the participating CPs. In addition, a federation should be stable, that is, the participating CPs should not have incentives to leave the federation. In the following sections, we describe the proposed game in detail.

3.1 Characteristic Function

The proposed coalition formation algorithm is based on a hedonic game [5][6], a class of coalition formation cooperative games, where each player behaves as a selfish agent and where her/his preferences over coalitions depend only on the composition of his/her coalition.

Given the set $P = \{P_i^t \,|i = 1, 2, 3...m\}$ of CPs, a coalition $S^t \subseteq P$ represents an agreement among the CPs at any given time t in S to act as a single entity. Now, the set of CPs is partitioned into a coalition partition π, that we define as the set $\Pi = \{S_1^t, S_2^t 1, S_3^t ... S_b^t\}$. That is, for $x = 1, ..., b$, each $S_x^t \subseteq P$ is a disjoint coalition such that $\cup_{x=1}^b S_x^t = P$ and $S_h^t \cap S_x^t = 0$ for $h \neq x$.

Given a coalition partition Π, for any CP $i \in P$, we denote by $S_\Pi^t(i)$ the coalition $S_x^t \in \Pi$ such that $i \in S_x^t$.

Each coalition S^t is associated with its coalition value $v(S^t)$, that we define as the total profit of coalition S^t, that is:

$$v(S^t) = \sum_{i=1}^{m} \sum_{j=1}^{n} \left(v_{ij}^t \cdot \Pr_{CP}^{t}(G_j^t) - \sigma_{ij}^{\cos t} \right) \tag{4}$$

The strategies for CPs is to maximize the value of the cloud federation $v(S^t)$ so that each CP can get its profit $Profit(v_{ij}^t)$ as much as high without violating the fairness requirement, and also maintain the stability of the federation. Thus, a profit division rule is needed in order to computes the profit of each federation member and to ensure fairness. To this end, we use the Shapley value [17], a profit allocation rule that ensures that the larger is the contribution provided by a player to a coalition, the higher is the payoff allocated to it.

Now, we formulate the cloud federation formation problem as follows:

$$Max \quad v(S^t) = Max \sum_{i=1}^{m} \sum_{j=1}^{n} \left(v_{ij}^t \cdot \Pr_{CP}^{t}(v_j^t) - \sigma_{ij}^{\cos t} \right) \tag{5}$$

or

$$Max \sum_{i=1}^{m} \sum_{j=1}^{n} Profit(v_{ij}^t) = Max \sum_{i=1}^{m} \sum_{j=1}^{n} \left(v_{ij}^t \cdot \Pr_{CP}^{t}(G_j^t) - \sigma_{ij}^{\cos t} \right) \tag{6}$$

s.t.

$$0 \leq v_{ij}^t \leq \sum_{i=1}^{m} \Psi_i^t \tag{7}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{ns} a_{ik}^j = 1, i \in P, k \in S \tag{8}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ik}^j \leq o_{ik}^j, i \in P, k \in S \tag{9}$$

$$v_{ij}^t \geq 0, \forall j = 1...n \tag{10}$$

In the above optimization model, the decision variables v_{ij}^t represent the number of available VM instances of type vm J provided by a CP i. Constraint (7) ensures that the total number of VM resources provided by any CP in the federation do not exceed its total capacity denoted by Ψ_i^t, in all dimensions of VM resources v_{ij}^t. Constraint in (8) imposes that each VM J is hosted by exactly one physical server. Constraint in (9) ensures that only servers that are powered on can have VMs allocated to them.

Since the problem in eq. (6) has a boundary constraint for the variable v_{ij}^t, it can be formulated as a constrained optimization, which can be solved by the method of Lagrangian Multiplier.

$$L = Profit(v_{ij}^t) - \gamma v_{ij}^t + \varphi(v_{ij}^t - \Psi_i^t) \tag{11}$$

where γ and φ are the Lagrangian constant. The Karush Kuhn Tucker (KKT) condition is as follows:

$$\frac{\partial L}{\partial v_{ij}^t} = Profit'(v_{ij}^t) - \gamma + \varphi = 0, \;\; j = 1, ..., n \tag{12}$$

$$\gamma \geq 0,\ \varphi \geq 0,\ \gamma v_{ij}^t = 0,\ \varphi(v_{ij}^t - \Psi_i^t) = 0, \quad 0 \leq v_{ij}^t \leq \Psi_i^t,\ j = 1, ...,n \tag{13}$$

$$Profit'(v_{ij}^t) = Y_i^t \cdot v_{ij}^{t^{\log_2 \alpha_i}} = \underset{CP}{\overset{t}{\Pr}}(G_j^t) - \Pr_{i,l}^{RiskCost} \tag{14}$$

$$v_{ij}^{t*} = \left[\frac{Y_i^t}{\underset{CP}{\overset{t}{\Pr}}(G_j^t) - \Pr_{i,l}^{RiskCost}} \right]^{\frac{1}{\log_2 \alpha_i}} \tag{15}$$

By solving v_{ij}^t of eq. (14), the optimal v_{ij}^{t*} of each CP i is either v_{ij}^{t*} if the maximum located in the range of $(0, \Psi_i^t)$, or the boundary value 0 or Ψ_i^t. Now, it is observed that for some CPs, the v_{ij}^{t*} value can be less than or equal to 0. So those CPs will be removed from the game.

3.2 Federation Formation Process and Algorithm

In the proposed media cloud federation formation game process, a preference relation needs to be defined so that each CP can order and compare all the possible coalitions it belongs to, and hence it can build preferences over them. Unlike existing works, we consider that the preference of a CP to join/leave any federation will depend on the maximum profit it can gain from that federation with minimum risk of penalty costs by selecting trustworthy collaborators.

Specifically, for any CP $i \in P$ and given federation set $S_1^t, S_2^t \subseteq P$ containing CP i, the preference of CP i being a member of S_1^t over S_2^t or at least i prefers both coalitions equally, can be defined as follows:

$$S_1^t \succ_i S_2^t \Leftrightarrow f_i(S_1^t) > f_i(S_2^t) \tag{16}$$

$$S_1^t \succeq_i S_2^t \Leftrightarrow f_i(S_1^t) \geq f_i(S_2^t) \tag{17}$$

Here, the eq. (16) defines that CP i strictly prefers being a member of S_1^t over S_2^t and the eq. (17) shows that CP i prefers both federation equally. The function $f_i(S^t)$ is a preference function, defined for any CP $i \in P$ and any federation S^t containing i. The value of $f_i(S^t)$ is the $Profit(v_{ij}^t)$, the payoff received by CP i in S^t. Based on the payoff value $Profit(v_{ij}^t)$, each CP prefers to join to a federation that provides the larger payoff. To keep track of which federation CP i visited and left in the past, let $\overset{visit}{H_i}$ be a history set of CP i's past federations. Now the federation formation method is described step by step as follows:

Step 1 : Initially there are no federations, $\Pi_0 = \{\{1\}, \{2\}, ...\{P\}\}$ and the history set $\overset{visit}{H_i}$. Now CP i evaluates all the possible federations it can form, starting by leaving its current one $S_{\Pi_c}^t(i)$ to join another already existing federation S_x^t by applying the hedonic shift rule [], that is, its payoff in the new federation is higher than the one it is getting in its current federation.

Step 2 : If such federation S_x^t is found, CP i updates its history $\overset{visit}{H_i}$ by adding its current federation $S_{\Pi_c}^t(i)$ as visited, leaves the current federation and joins the new federation S_x^t. Finally the current partition Π_c is updated as follows:

$$\Pi_{c+1} = \big(\Pi_c \backslash \{S_{\Pi_c}^t(i), S_x^t\}\big) \cup \big(S_{\Pi_c}^t(i) \backslash \{i\}, S_x^t \cup \{i\}\big) \tag{18}$$

Otherwise, CP i remains in the same federation so that $\Pi_{c+1} = \Pi_c$.

The proposed federatoin formation algorithm has two important properties- convergence guarantee and Nash-stability, which are proved in [green cloud]. In the next section, we show how to calcualte the trust value of any CP when forming a federation.

4. Performance Analysis

In this section, we present our evaluation methodology and simulation results to show the effectiveness of the proposed cloud federation formation game. In our game model, we consider that each CP will calculate its total cost of providing VM resources based on the production cost of VMs, running energy cost of servers and risk or penalty cost due to various trust levels to other CPs. We assume that the trust level of different CPs exists based on our trust model executed by various TEAs of CPs.

4.1 Simulation Setup

Our simulation environment consists of the participation of 8 independent CPs. Each CP consists of a number of hosts (servers), VMs and a trust value picked randomly from the range presented in Table 1. The three types of available VMs in these hosts and their costs are set similar to those offered by the Amazon EC2 [Amazon EC2 Pricing. http://aws.amazon.com/ec2/pricing/]. The three types of VM instances are presented in Table 2.

Table 1: Different parameters used in the simulations

Parameters	Values
Number of CPs	8
Hosts in each CP	$[20, 40]$
VMs in each host	$[20, 40]$
α	[0.78, 0.84]
Trust Value	$[0.5, 1]$
Electricity Price	[0.4, 0.5]

We set the cost of VMs randomly from the ranges presented in Table 2. To set up the ranges, we follow the half of VM prices of the Amazon EC2 regions as a guideline. We also assume that The electricity price range is set to 0.4 -0.5 \$/kWh. The cost of switch on and off for each server are computed according to [6]. We generate total 90 requests such that requests with less than 25%, 35%, and 40%, of the total available capacity belong to the VM 1 (small), VM 2 (medium), and VM 3 (large) classes, respectively. For every parameter setup, we randomly generate 30 requests for each class and run our simulation on them. The averages of the different output parameter indicating the performances found in these simulations are presented in the plots and tables unless specified otherwise.

Table 2: The available VM instances in the simulations

Parameters	VM Class 1	VM Class 2
Number of cores (1.6 GHz CPU)	1	2
Memory (GB)	1.7	3.75
Storage (TB)	0.22	0.48
Cost, Y, of producing the first VM ($)	[0.02, 0.07]	[0.08, 0.14]
Price ($ per hour)	0.12	0.24

To compare our proposed method with other mechanisms, we have adopted two other state of the art approaches. One [6] of which considers cost in terms of production and running energy cost of servers but does not take into account the penalty costs due to trust; while the other one [5] considers neither the penalty nor the energy cost when deriving its profit from the federation. For the sake of simplicity in referring to them, we refer to the first approach here as Federation game with energy cost (as because it considers the production costs and the running costs) method and the second approach as Federation game without energy and trust penalty cost, in the rest of this section.

5. Experimental Results

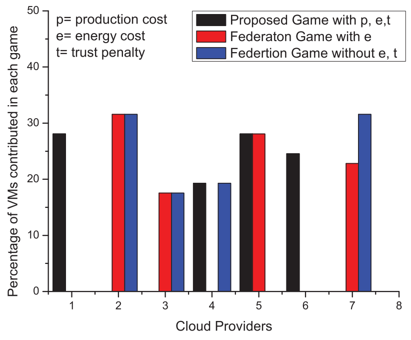


Fig. 1: Percentage of VM resources supplied by each CP in all the games for VM Class 1 (small size requests)

Figs. 1 and 2 demonstrate the numbers of VMs offered by the CPs to serve requests of different classes of VMs. When the requests are in VM class 1 (small VM requests), the different methods can have the chance to pick the âĂŸgoodâĂŹ CPs to form a federation. At this point, the differences in the performances of different approaches to build federations appear to be much clear (performance based on different metrics are presented later in this section) as demonstrated in Fig. 1. Here, different approaches have picked different CPs to make the federations. However, as the size (number of VMs required) of requests get larger the freedom making choices becomes narrower, which is demonstrated in Figs. 2, where the contributions of different CPs have become more similar.

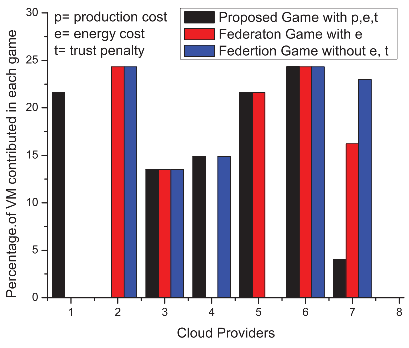


Fig. 2: Percentage of VM resources supplied by each CP in all the games for VM Class 2 (medium size requests)

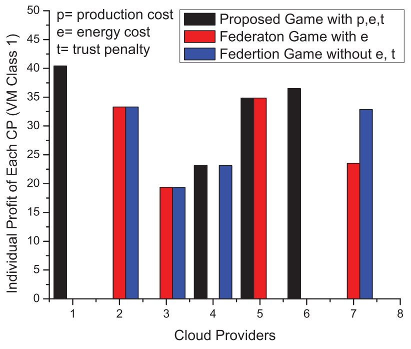


Fig. 3: Individual profit of each CP in all the games for VM Class 1 (small requests)

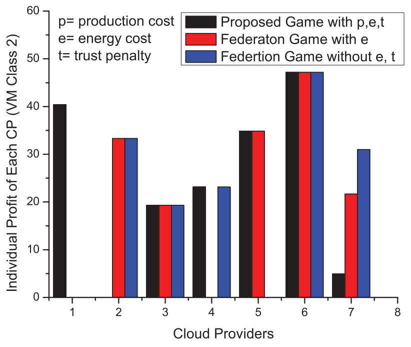


Fig. 4: Individual profit of each CP in all the games for VM Class 2

Figs. 3 and 4 present the individual profits made by the different CPs. Here the federations formed by the proposed approach are found to make higher total profits than the other two approaches. This is because the other two approaches do not consider the costs related to running the servers and trust of the CPs which minimizes the expected profit. Here we can also notice that some CPs have made profits in the PR and P approaches while the proposed approach have not picked them in the federation. Actually, the proposed method has avoided them in the federation because such CPs may not lead to the expected profit due to their trust values. Considering the penalty costs which may incur due to the low trust value, the proposed method may not always pick a CP in a federation even though it promises to make high profit.

6. Conclusions

This paper presents a novel cloud federation formation approach that helps media cloud providers to form a federation dynamically to provide various types of VM instances to users' with QoS guarantee for media data workloads. Based on coalition game theory, the proposed mechanism help CPs to form different federations- each one consisting of a subset of the CPs. Unlike existing works, we consider that the preference of a CP to join/leave any federation will depend on the maximum profit it can gain from that federation with minimum risk of penalty costs by selecting trustworthy collaborators. The proposed model uses the overall trust level of a collaborating CPs. Simulation results demonstrated that the media cloud federation obtained by the proposed mechanism can satisfy the fairness and stability property, while at the same time provides maximum profit to the media CPs as compared to the state-of-the-art approaches.

Acknowledgement

This project was funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, Award Number (12-INF2613-02).

References

[1] Hassan MM, Song B, Huh EN. Distributed resource allocation games in horizontal dynamic cloud federation platform. *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, IEEE, 2011; 822–827.

[2] El Zant B, Amigo I, Gagnaire M. Federation and revenue sharing in cloud computing environment. *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, IEEE, 2014; 446–451.

[3] Goiri Í, Guitart J, Torres J. Economic model of a cloud provider operating in a federated cloud. *Information Systems Frontiers* 2012; **14**(4):827–843.

[4] Toosi AN, Calheiros RN, Thulasiram RK, Buyya R. Resource provisioning policies to increase iaas provider's profit in a federated cloud environment. *High Performance Computing and Communications (HPCC), 2011 IEEE 13th International Conference on*, IEEE, 2011; 279–287.

[5] Mashayekhy L, Nejad M, Grosu D. Cloud federations in the sky: Formation game and mechanism. *IEEE Transaction on Cloud Computing* 2014; .

[6] Guazzone M, Anglano C, Sereno M. A game-theoretic approach to coalition formation in green cloud federations. *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, IEEE, 2014; 618–625.

[7] Niyato D, Vasilakos AV, Kun Z. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. *Proceedings of the 2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE Computer Society, 2011; 215–224.

[8] Samaan N. A novel economic sharing model in a federation of selfish cloud providers. *IEEE Transactions on Parallel and Distributed Systems* 2013; :1.

[9] Mashayekhy L, Grosu D. A coalitional game-based mechanism for forming cloud federations. *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing*, IEEE Computer Society, 2012; 223–227.

[10] Xu X, Yu H, Cong X. A qos-constrained resource allocation game in federated cloud. *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013 Seventh International Conference on*, IEEE, 2013; 268–275.

[11] Truong-Huu T, Tham CK. A game-theoretic model for dynamic pricing and competition among cloud providers. *Utility and Cloud Computing (UCC), 2013 IEEE/ACM 6th International Conference on*, IEEE, 2013; 235–238.

[12] Fan W, Yang S, Pei J. A novel two-stage model for cloud service trustworthiness evaluation. *Expert Systems* 2014; **31**(2):136–153.

Web-based Data Management System Design and Development for the Multisite Renewable Energy Research and Education Partnership

B. Asiabanpour[1], S. Aslan[1], Z. H. Salamy[2], Almusaid[1] , J. Warren[1]
[1]Ingram School of Engineering, Texas State University, San Marcos, Texas, USA
[2]School of Engineering, University of Saint Thomas, Saint Paul, Minnesota, USA

Abstract – *There is no single factor that can be universally identified as the most significant factor in a solar panel losing its efficiency. Variety of variables such as solar panel condition, location, season, time, and environmental conditions can play a significant role in solar panel efficiency loss. In this study, we propose a Web-based Data Management System Design and Development for the Multisite Renewable Energy to appropriately provide infrastructure needed for optimization efforts in different geographical places and considering variety of factors simultaneously.*

Keywords: Electricity generation, solar energy, Web-based Data Management, Energy grid, Research and Education Partnership

1 Introduction

The rapid global demand for energy derived by: the growth in population, improved quality of life and high level of industrialization. With the limitation of fossil fuels and their production plus the high possibilities of pollution when extracted and used. The need for clean and sustainable energy has become inevitable. Solar energy has the potential to be one of the key alternative clean and renewable sources to supply the increased demand. In 2009, the world's consumption of energy was 11164.3 million tons of oil equivalent (mtoe) [1]. Comparing this figure with the amount of received solar radiation during the same year, we find that the input solar radiation is 11,300 times greater than the world's total primary energy consumption [2]. Solar energy is clean and friendly to the environment, and it will not be affected by fluctuations in the market, such as oil. The Solar Energy Industry Association (SEIA) report for 2014 shows a rapid growth in solar industry. 6201 MW of PV panels have been installed plus 767 MW of concentrating solar power. This makes 20 Giga watts of total installed capacity, which is enough to power four million houses in the United States. The report also shows that the industry has 175,000 workers, more than Google, Apple, Facebook, and Twitter combined*. Solar energy is now estimated for one-third of the United States new generating capacity in 2014, surpassing both wind energy and coal for the second year in a row [3 and 4].

Generating electricity from solar energy has a long history, but due to the inefficiency of the process, solar energy has not fulfilled its potential. Some of the major causes of energy loss in solar panels include DC-to-AC power conversion, panel orientation, temperature, dust and dirt, aging and degradation, and even humidity and wind speed. Common practices to remedy these problems include sun tracking, protection shields, material improvements, regular cleaning, cooling, and AC solar generating without inverters. None of the solutions addressed by the investigators seems to be a universal remedy for all users. Additionally, all the published articles studied by the authors show that only one or very limited important factors have been investigated in each study. It appears that the efficiency loss should be studied as a regional issue, and remedies would be tailored to directly address these local factors. Solar panel performance can also vary because of different manufacturing processes and materials used.

To study and optimize the energy generation by the solar panels, variety of factors including sun tracking, protection shields, material improvements, regular cleaning, cooling, and AC solar generating without inverters as well as other key issues, such as location, season, time, and environmental conditions needs to be considered simultaneously. An extensive multifactor design of experiments in different regions and sites involving major affecting factors will facilitates better understanding of each factor. Such study requires a central integrated data collection and processing system capable of collecting, storing, demonstrating, and analyzing the results seamlessly.

In this study, we propose a Web-based Data Management System Design and Development for the Multisite Renewable Energy to appropriately provide infrastructure needed for optimization efforts in different geographical places.

2 System development

The growing need for renewable energy and improvements for efficiency shows the importance of data collection and analysis. Most of the efficiency and operational improvements focus mainly on local and regional problems relating to solar panels such as dirt/mold type, heat, and optimum orientation. To obtain the proper data and multifactor design of experiments, different sites are used to generate test data.

2.1 Nature of Input Data:

There are two main sensor components, the Egauge and the SMA SunnySensor Box. Together these sensors collect voltage, current, frequency, irradiance, solar module temperature and ambient temperature. Every minute these sensors collect data with a timestamp and sent data to the webserver and store them in the database. Basic connection scheme of the sensor boxes are shown in Figure 1 below.

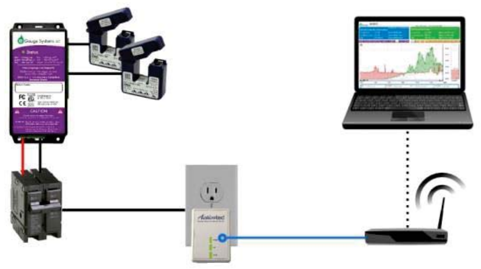


Figure 1: Web-based electric energy and power meter system [5]

2.2 Database format, structure, and details:

Instead of data being pulled directly and served through MySQL, a restful API was implemented in Go where it offers built-in support for JSON encoding/decoding and built-in custom data types support. This format is standard and is used for many different applications to transport data in a compact, but easy to read format. Our API has several different routes. Each route returns a different set of data depending on the route and the parameters passed into the route. For example, the route /registersInfo/location/TxState/serial/0001 returns all the registers for serial 0001 at location TxState. It may return some JSON that look like one shown in Figure 2.

```
[
    {
        name: “CT2I”,
        type: “I”
    },
    {
        name: “L1V”,
        type: “V”
    }
]
```

Figure 2: JSON data code sample

2.3 Server info:

A cloud based server was necessary to build this project. It provides a location to store and host our webserver and MySQL data server. The entire project runs on a Windows Server 2012 operating system. This system runs 24/7 as data is sent to it and stored one minute intervals. The entire web server was built in Go, also commonly referred to as Golang. Go is chosen due to wide range of web implementation and built in custom data support. The web applications that is used in this project can be summarized in two parts. There is the API and the Website. The API provides a raw source of data for computers or programs. The website provides an experience where data can be analyzed by a human. The block diagram of the proposed system is shown in Figure 3 below.

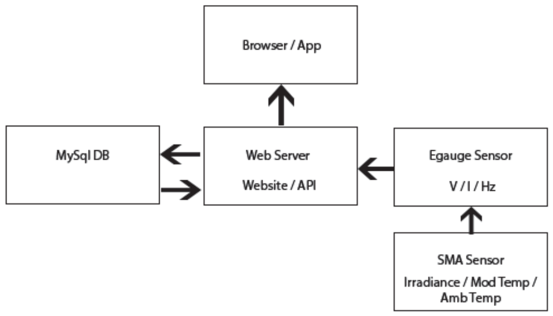


Figure 3: Web server data collection and API block diagram

2.4 Graphical User Interface (GUI) (Website input selection):

The website basic interface is shown in Figure 4 below provides a location on the Internet for any user to download, search or interact with the data that is collected and stored. The website also offers a way for others to communicate and share information. The data that is presented here using a JavaScript graphing library called HighCharts [6].

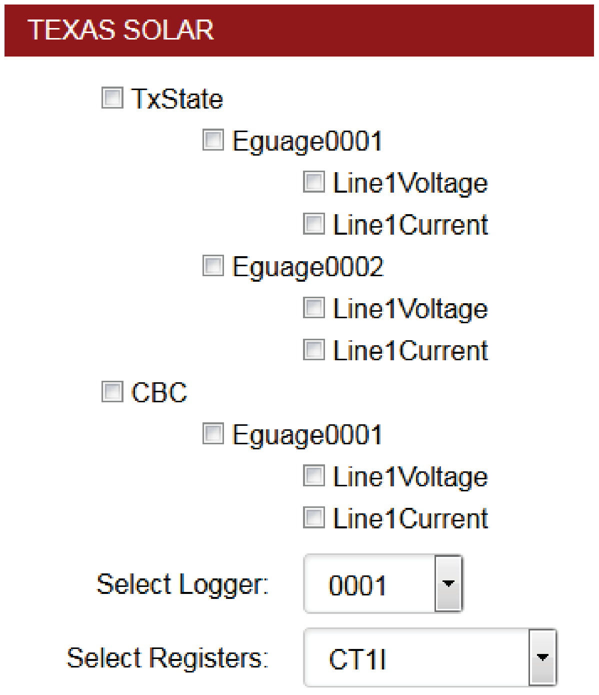


Figure 4: Texas Solar webpage interface

A graph can be generated for any measurement taken and compared to the other locations over any interval of time. A graph data display and time interval change are shown in Figures 5 and 6 respectively.

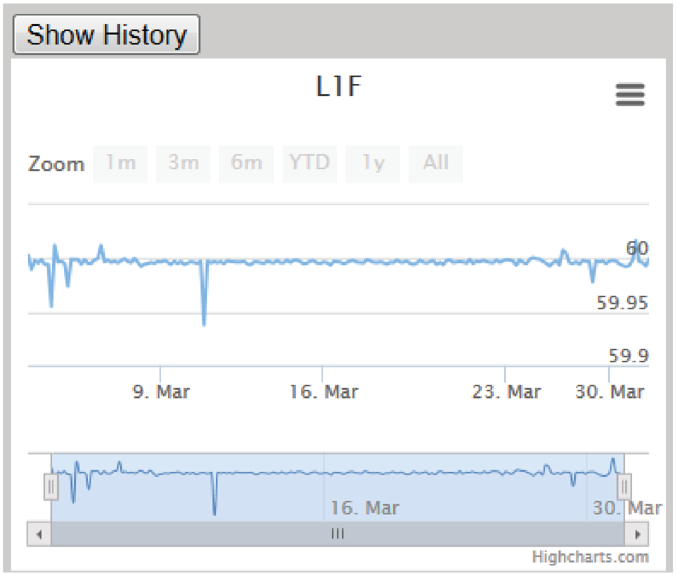


Figure 5: Data display using HighCharts

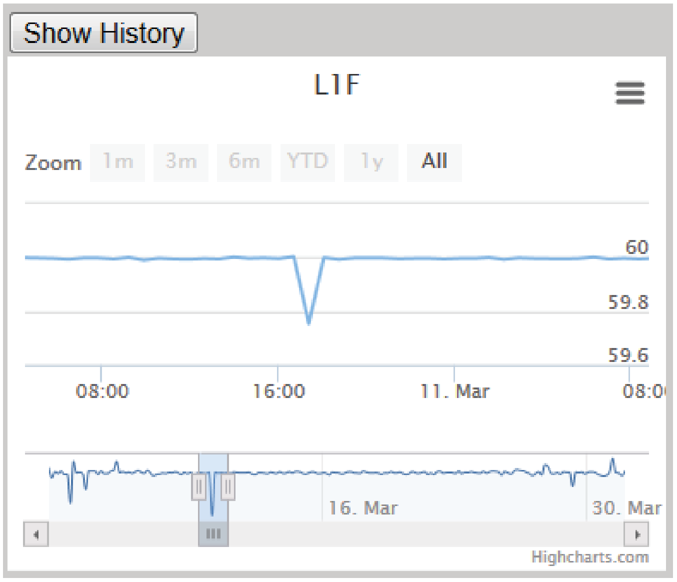


Figure 6: Data time interval change

2.5 App development:

This project currently has an android app in development. The app is being developed in Native android code with Android studio. The app allows access to live environmental data collected from different sites, and does so in a way that is tailored specifically to mobile devices. This is ideal for someone doing field analysis, and will be usable by anyone

3 Results, case study, and expandability

The proposed web-based data management system is deployed to manage data readings from different environmental sensors including temperature, humidity, light intensity, and wind speed as well as system performance specifications of the solar panels including current and voltage connected to multiple solar panels in the System Modelling and Renewable Technology (SMART) lab at Texas State University. The same system (solar panels and sensors) is also deployed at satellite labs at community colleges in neighboring cities (Coastal bend college-Beeville, San Antonio College, and Southwest Texas junior college at Uvalde) and real-time data are uploaded in the central database to manage data for comparisons and further analysis.

As the main hub for the proposed system, the SMART Laboratory is a multidisciplinary research and teaching lab focusing on green and renewable energy solutions for manufacturing and residential applications. The lab was established mainly through grant funding from the Texas State Energy Conservation Office (SECO) and Department of Education. The lab is located on a rooftop patio between two buildings shown in Figure 7. The SMART lab is a research, education, and outreach lab that develops outreach programs and workshops to promote renewable energy and SMART technology. The SMART Lab is 100% green and self-sustaining using energy produced from solar panels and wind turbines. The schematic of the SMART lab is shown in Figure 8.

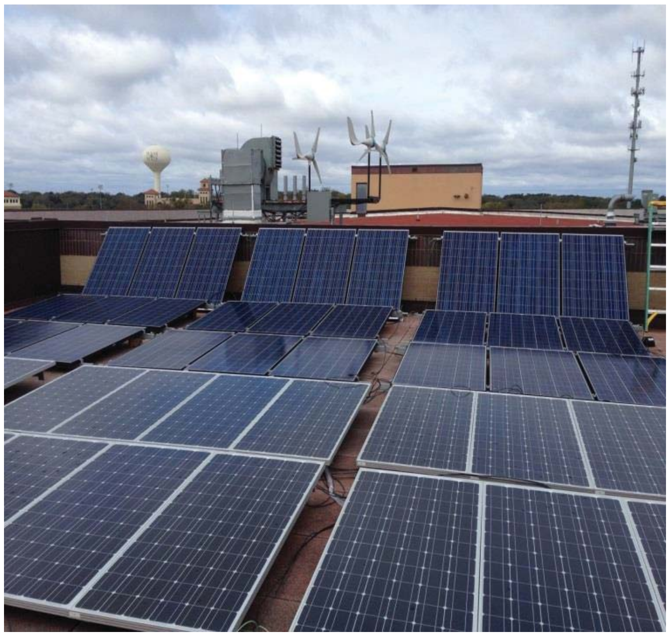


Figure 7: Solar Panels and Wind Turbines in the SMART Lab.

The solar and wind systems depicted in Figure 8 are arranged electrically into a number of complimentary sub-systems. Each individual sub-system may be separately configured in order to gauge the effectiveness of configuration changes on the efficiency of the system. The several sub-systems are all wired to provide power back to the SMART Lab for both the operations of the facility and for testing with various laboratory loads. The current data gathering systems in the SMART Lab allow for an aggregate sub-system level of granularity. For each sub-system, the voltage, current, and power stored into and used from each sub-system are read and managed by our proposed system and the data is recorded on the data server at regular intervals. The system also manages other useful set of weather condition data such as temperature, light illumination and humidity.

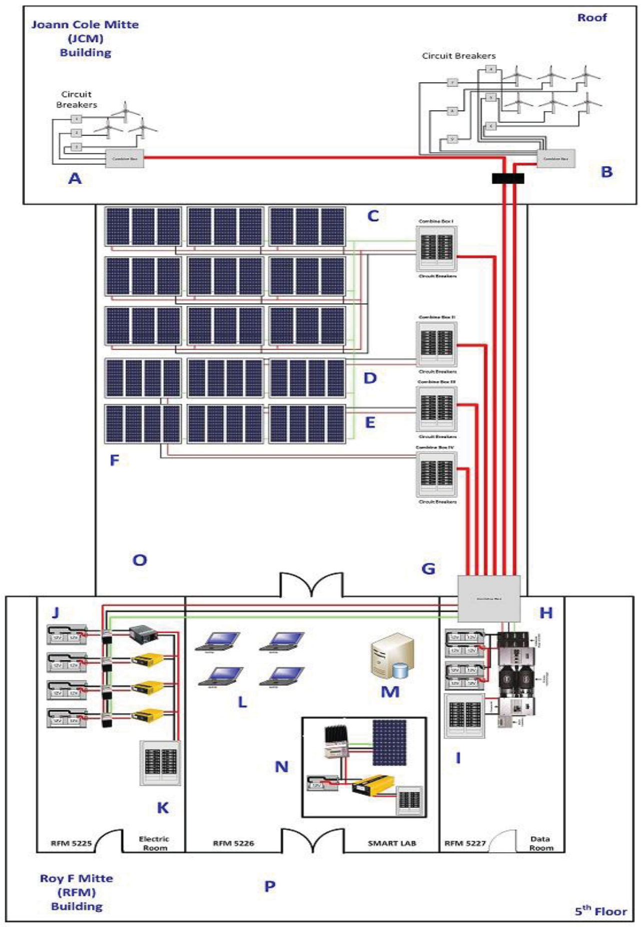


Figure 8: Schematic Representation of the SMART Lab

The proposed integrated web-based data management system is designed with minimum constraints to facilitate scalability, expandability and applicability to other scenarios. The next point in our agenda is to use the data management system to manage a medium size Internet of Things (IoT) sensor network spanning multiple campuses through a cloud cluster. The "Internet of Things" (IoT) concept is a natural outgrowth of the advancement of microprocessor and sensor technologies that has brought significant compute and network capability to the point of ubiquity. The technical and financial barriers to adding advanced computing and communication to essentially anything have been erased. Adding a microprocessor and radio link to a consumer product would not have been practical a decade ago, yet it costs pennies today. This creates the possibility of a dense network of interacting elements, enabling applications that were previously impossible. The resulting network density offers significant challenges, especially as the network grows in complexity and the ubiquitous devices generate large, real-time datasets.

The proposed IoT network segment as part of our ongoing and future work to support this research would include a significant number of connected sensors gathering electrical usage data at a much finer granularity. In the collective, the SMART Lab systems alone contain 53 solar panels, 9 wind turbines, 18 storage batteries, and 7 dc-to-ac inverters. Ideally, a finely-grained sensor network would monitor voltage, current, and power for each of these elements individually. Systemic variables such as atmospheric conditions, wind speed, and temperature would supplement this data pool. This would form a dense network of sensor modules.

Collecting the amount of data from the proposed ZigBee based IoT sensor network from the SMART lab and the four other sites will represent a large case study that will test the scalability and expandability of the proposed Web-based Data management system. Aside from research pertaining to ZigBee IoT scalability, performance and reliability, the amount of data and its versatility will open the door to more multidisciplinary research in renewable technology, data mining, big data, and web development.

4 Discussion and conclusion

The developed system is providing the infrastructure for comprehensive research and wide variety of statistical optimization activities, which requires long-term momentarily data collections of the local environmental conditions (e.g., temperature, wind speed, light intensity, humidity) and solar panels status (e.g., cleanness, orientation, temperature) as well as system performance (e.g., current and voltage). The system is expandable to more sites and online data provides the capability to many researchers that have no access to the physical renewable energy facility to be able to selectively use the data in their research.

5 Acknowledgement

This work has been completed with funding from the Department of Education-MSEIP program (grant number P120A140065). The authors would like to thank the Department of Education and Texas State University for providing funding and access to infrastructure and laboratories. Sponsors are not responsible to the content and accuracy of this article.

6 References

1- BP Statistical Review of World Energy, (June 2010 & 2011), British Petroleum Plc, London.

2- Shepherd, W., & Shepherd, D. W. (2014). Energy studies (3rd Ed.). London, England: Imperial College Press.

3- Solar Energy Industrial Application (SEIA), “U.S. Solar Market Insight”, Retrieved on March 1, 2015. http://www.seia.org/research-resources/us-solar-market-insight

4- EcoWatch Transforming Green, “U.S. Solar Energy Industry Achieves Record-Shattering Year”, Retrieved on March 1, 2015. http://ecowatch.com/2015/03/10/rhone-resch-solar-shattering-year/

5- EGauge Systems LLC, www.egauge.net

6- Highsoft AS, http://www.highcharts.com/