Available online at www.sciencedirect.com

ScienceDirect

Future Generation Computer Systems 00 (2011) 1–10

Future Generation Computer Systems

Feedback-based Optimization of a Private Cloud

Hamoun Ghanbari[a,b,1,*], Bradley Simmons[a], Marin Litoiu[a], Gabriel Iszlai[b]

[a]Department of Computer Science and Engineering, York University, 4700 Keele St, Toronto, Ontario, Canada, M3J 1P3
[b]Centre for Advanced Studies (CAS), IBM Toronto Lab, 8200 Warden Avenue, Markham, Ontario, Canada, L6G 1C7

Abstract

The optimization problem addressed by this paper involves the allocation of resources in a private cloud such that cost to the provider is minimized (through the maximization of resource sharing) while attempting to meet all client application requirements (as specified in the SLAs). At the heart of any optimization based resource allocation algorithm, there are two models: one that relates the application level quality of service to the given set of resources and one that maps a given service level and resource consumption to profit metrics. In this paper we investigate the optimization loop in which each application's performance model is dynamically updated at runtime to adapt to the changes in the system. These changes could be perturbations in the environment that had not been included in the model. Through experimentation we show that using these tracking models in the optimization loop will result in a more accurate optimization and thus result in the generation of greater profit.

Keywords: Optimization, Modeling, State Estimation, Private Cloud, PaaS, IaaS

1. Introduction

Advances in virtualization [1] techniques and the construction of numerous large commodity data centers around the world [2] have resulted in a new approach to computing referred to as *cloud computing* [3, 2, 4, 5] becoming an important topic of research and development. The *cloud*, though still in its infancy, typically refers to some set of computing resources (infrastructure (IaaS), platform (PaaS) or software (SaaS)) being provided on demand over the Internet to users as a service.

A *private cloud* represents a set of virtualized data centers under the ownership of a single administrative domain (i.e., the cloud service provider). Unlike in a *public cloud* where the various layers may be offered by multiple providers the entire stack (IaaS,PaaS and SaaS) is controlled by the single provider and so it has access and control over the various applications, middlewares and infrastructure simultaneously. The main objective of a private cloud provider is to maximize *profit* (i.e., revenue-cost). Optimization techniques allow the provider to determine resource allocations to various clients in order to best maximize its revenue while minimizing its costs[2]. Due to these economic benefits, optimization has been the subject of much investigation [11, 12, 13, 14, 15, 16, 17, 18].

The decisions made by a provider with regards to deployment of application tiers in the cloud and resource allocation to application environments can be enforced

*Corresponding author
Email addresses: hamoun@cse.yorku.ca (Hamoun Ghanbari), bsimmons@yorku.ca (Bradley Simmons), mlitoiu@yorku.ca (Marin Litoiu), giszlai@ca.ibm.com (Gabriel Iszlai)
URL: http://www.ceraslabs.com/people/hamoun-ghanbari (Hamoun Ghanbari), http://www.ceraslabs.com/people/dr-bradley-simmons (Bradley Simmons), http://www.ceraslabs.com/people/dr-marin-litoiu (Marin Litoiu)
[1]Tel:+1-416-265-7585

[2]Notice that in layered cloud, optimization is decomposed into dynamic infrastructure pricing mechanism offered by provider [6, 7] and elastic resource allocation policies employed by individual consumers [8, 9, 10] to satisfy their QoS requirements.

through *scale-up/down* (i.e., adding/removing resource to individual virtual machines (VM)), *scale-out/in* (i.e., adding/removing VMs to an application environment, and *migration* (i.e., moving VMs over the physical infrastructure) and will directly impact both the performance of an application and the provider's cost of operations. Here we focus only on scale-up/down.

The optimization problem addressed by this paper involves the allocation of resources in a private cloud such that cost to the provider is minimized (through a maximization of resource sharing) while attempting to meet all client application requirements as specified in their respective Service Level Agreements (SLA)s[3] [19, 20, 21, 22] (see Figure 1). Many of the current optimization approaches, while efficient, assume static models [11, 23, 24, 12, 13, 14, 15, 16, 17, 18]. In this paper we attempt to solve the optimization problem through a feedback-based loop with dynamic models. The resulting optimizer will be compared to one using static models to demonstrate the benefits of this proposed approach.

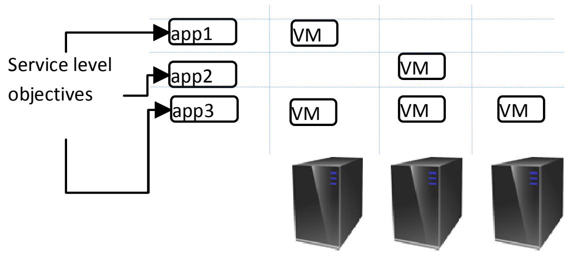


Figure 1: The interaction of layers in our optimization mechanism.

The remainder of this paper is structured as follows: Section 2 describes our feedback based approach for optimization of resource shares in a private cloud. This involves the introduction of general formal definitions used during problem formulation and the description of the estimator component of the feedback loop. Section 3 explains the optimizer component of the loop. The applicability of the proposed approach is demonstrated by the case studies in Section 4. Related work, conclusions and future works are discussed in Sections 5 and 6.

2. Proposed Approach

We propose a feedback based Cloud Optimization Manager (COM) as shown in Figure 2. In this approach, each application maintains both a dynamically updated performance model (see Section 2.3) and a utility model (which defines a specific service level objective e.g., response time). The COM has access to the utility model of each application. Periodically, each application submits its performance model to the COM which performs a system-wide global optimization (see Section 3) using this information and determines new resource allocations for each application for the following period.

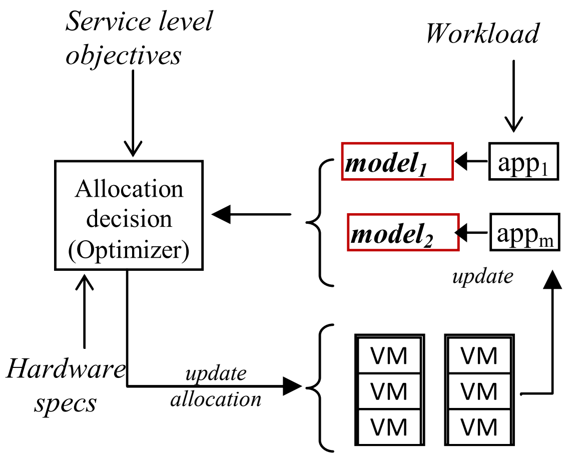


Figure 2: Feedback based optimization of resource shares.

Consider applications app_1,..., app_m. Each application runs within one or more VMs and experiences a particular workload. Assume that there are n physical machines (PM)s in the data center, represented by the set $P = p_0, ..., p_n$. VMs are hosted on PMs on behalf of applications. Let us assume that the physical server environment is homogeneous and each physical machine, say p_i has one resource, and, thus has a fixed capacity c_i in one dimensional space [4]. The allocation of VMs on PMs can be represented by an $n \times m$ matrix A. Each

[3]An SLA is a contract which defines the relationship between a service provider and its clients that fully specifies all obligations for both parties, the price to be paid for the service(s) offered and associated penalties should obligations be unmet. It can be quite complex and comprehensive (e.g., considering aspects of both functional and non-functional requirements); however, in this work, only performance objectives that can be extracted from an SLA are considered. No attempt is made to fully model or develop an SLA or an SLA management framework.

[4]In the case of multi-resource modeling c_i can be substituted with c_i^r , where c_i^r is the capacity of resource r of p_i.

element a_{ij} of A denotes a resource allocation defining the percentage of the total resource (i.e. CPU) capacity of the PM i allocated to a running VM of application j.

2.1. Problem formulation

A global utility function U_0 is expressed as the difference of the sum of application-provided resource-level utility functions and an operating cost function as follows:

$$U_0 = \left(\sum_{j \in App} u_j(s_j(A_j)) \right) - \omega.cost(A) \qquad (1)$$

where ω denotes an adjustable weight (working as tunable parameter for the administrator), s_j denotes the service level function which maps the application j's resource allocations (i.e. A_j) to the service level measure (e.g. response time) of the application, u_j is the local utility function for application j, A represents the allocation matrix of VMs on PMs (defined earlier), and A_j represents j'the column of A. U_0 can be associated with the profit of the cloud and the two terms of equation (1) represent the revenue and the cost respectively.

Our objective is to maximize U_0 subject to a set of capacity constraints which come form the physical layer of the private cloud. The optimization problem addressed here can be expressed as follows:

$$\begin{aligned} &\text{maximize: } U_0 \\ &\text{subject to: } \forall i(\sum_{j \in App} a_{ij} < c_i) \\ &\qquad\qquad \forall i \forall j (a_{ij} \in [0, c_i]) \end{aligned} \qquad (2)$$

It is assumed that each allocation signal a_{ij} is constrained to lie in the interval $[0, c_i]$ meaning that an application can get the whole capacity of a PM.

Notice that the problem has a best effort nature and we treat a service level objective (i.e. target on a specific QoS metric) as a soft constraint by incorporating it into the objective function.

Figure 3 represents a sample service level utility function where the vertical line indicates the SLA target of an application and utility decreases as the value of s_j approaches the SLA limit.

It is worth noting that any other decreasing differentiable function can also be used and the only feature of this function is simplicity of presentation in mathematical form. However, one should note that the shape of a function, especially after passing the SLA threshold, will impact the behavior of the optimization algorithm.

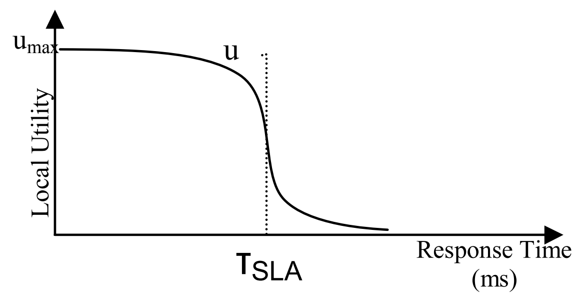


Figure 3: Smooth service level utility function; vertical line indicates the servicel level objective of an application (as defined in SLA).

2.2. Modeling QoS Attributes Based on Resource Allocation

A performance model can be used to quantitatively relate the physical layer specification of a customer's application (e.g. its given resource shares) and its associated workload to a service level measure (e.g. response time). We assume that the service level function of each application j, referred to as s_j, can be modeled by a function g with the column j of A $(a_{1j}, a_{2j}, \ldots, a_{kj})$ and τ_j, d_j as parameters:

$$s_j = g(a_{1j}, a_{2j}, \ldots, a_{nj}, \tau_j, d_j) \qquad (3)$$

where a_{ij} is the number of allocated CPU cycles (i.e., measured in MIPS), τ_j denotes the request inter-arrival time in seconds associated with the workload of the application [5] and d_j is application j's hardware demand per request (i.e. the number of CPU cycles for each request to be processed).

The model is further simplified by using the aggregate resource capacity measure $\gamma_j = \sum_{i=1...n} a_{ij}$ instead of the vector of individual values $(a_{1j}, a_{2j}, \ldots, a_{nj})$.We will assume that all applications deployed on the infrastructure adhere to the same performance model differing only in the setting of various configurable parameters. While there are several types of models (the model we use is described fully in Section 4) the online-estimation techniques and optimization method are general and will be considered next in Subsection 2.3 and Section 3.

2.3. Online Estimation

In general the goal of any optimal control approach is to choose a sequence of feasible *control actions* that

[5] s_j is usually the average response time of the application.

maximizes a defined *performance criterion* (or objective function)[6]. Since calculating the cost and performance criterion involves projection of system's behavior and state in the future, there is a need to have a *model* of the system over time. This model can be either deterministic or stochastic, and continuous-time, discrete time, or steady-state.

Adaptive models are used to dynamically take into account the system differences and re-tune the model. In adaptive models, parameter estimates of a model are bootstrapped and improved iteratively over time as more data is revealed. This is different from off line construction of the model using training data in supervised learning.

In adaptive models the process of estimation (i.e. finding the most likely hidden (or non-measurable) state sequence given a model and an observation sequence) and model identification (i.e. finding the probabilistic relation state variables over time) are mixed together to form an unsupervised learner. The Kalman filter [25] is a well-known parameter estimator of this type which can compensate for system differences from the base model at runtime for linear dynamical systems. In the algorithm presented in the next section we performed dynamic tuning of the models s_j using an extended Kalman filter, a variant of the Kalman filter for a non-linear measurement equation. The filter is able to take into account the measurements during runtime and update the model. Focusing only on one application called j, let $B_{j,n}$ be a vector representing the model coefficients and $\phi_{j,n} = \left[\gamma_{j,n}, \tau_{j,n}, d_{j,n}\right]$ be the measured inputs of the system (i.e. capacities, inter-arrival time and demand) at step n. Let $z_{j,n}$ be the measured system output (e.g. response time) at step n. The model maps $\phi_{j,n}$ and $B_{j,n}$ to the modeled output $s_{j,n}$:

$$s_{j,n} = f(\phi_{j,n}, B_{j,n}) \tag{4}$$

where f is a runtime approximation of the service level function(or performance model) g defined earlier.

The filter maintains the estimate of $B_{j,n}$ and updates it using the linear feedback equation based on new measurements $\phi_{j,n}$ and $z_{j,n}$:

$$B_{j,n} = B_{j,n-1} + K_{j,n}(z_{j,n} - s_{j,n}) \tag{5}$$

where n denotes a discrete time index, $K_{j,n}$, the dynamically computed "Kalman Gain" matrix and $z_{j,n} - s_{j,n}$ denotes the prediction error[7]. With certain assumptions, the calculated Kalman gain $K_{j,n}$, is guaranteed to minimize the estimation errors for $B_{j,n}$. The updated state will result in a dynamic model $f(\phi_{j,n}, B_{j,n})$ that will be accessed multiple times during optimization cycles. Function s_j and its non-measurable and measurable parameters can have many representations (as shown in Section 4).

[6]One can instead say optimal control minimizes an expected total cost function

[7]Notice that $s_{j,n}$ was calculated using equation 4.

3. Optimization

In this section, we solve the introduced optimization problem, as defined in equation 2, through primal decomposition. Algorithm 1 represents a centralized solution for this problem using the primal method. The equations referenced by the algorithm are listed in Table 1.

```
input  : initial capacities $A_0$, $k_{max}$, performance model
         $s_j$ and utility model $u_j$ for each app $j$
output : optimal allocation $A_{opt}$, Maximum utility $fp_{best}$
1  initialize: $fp = -\infty$, $fp_{best} = -\infty$, $A = A_0$
2  while $k < k_{max}$ do
3      compute constraint value $U_i$ for each PM using
       equation 1.
4      if no constraint is violated, move towards optimum:
       $(max(U_i(A)) > 0)$ then
5          compute global utility function using equation 2.
6          record the global maxima using equation 3.
7          calculate objective function's subgradient using
           equation 4.
8          move towards subgradient and the optimum
           using equation 5.
9      else if there is some violated constraint then
10         find most violated constraint using equation 7.
11         compute the subgradient value of most violated
           constraint using equation 8.
12         select step size based on equation 9.
13         move away from subgradient and the optimum
           using equation 6.
14     project each individual variable based on its local
       constraint using equation 10.
15     $k = k + 1$;
16 $A_{opt} = A$;
```

Algorithm 1: A centralized solution for the problem using primal method. The equations referred to are presented in Table 1.

The input to the algorithm includes the initial capacities (i.e., A_0), the number of iterations during optimization ($k_{\max}$), the performance model (s_j) and utility model u_j for each application. The output of the algorithm is the optimal allocation matrix A_{opt}, and maximum utility gained from that allocation, fp_{best} which is obtained iteratively using $k_{\max}$ iterations.

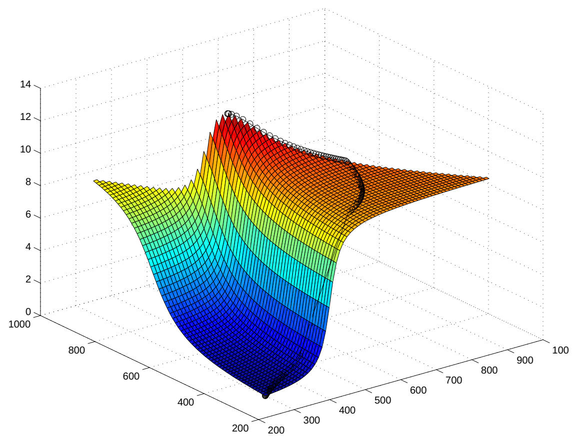


Figure 4: One sample run of subgradient optimization algorithm, finding the maximum utility using variations in capacity.

To demonstrate how the optimization algorithm performs, we optimize a pre-built model of one physical machine hosting two applications, each running on a single VM. Both applications have CPU demands of 5100 and 4100 MIPS per request respectively and response time thresholds (i.e. r_{thr_j}) of 16 and 7 respectively. The request interarrival time of both applications is 22 seconds, and their $u_{\max}$ is 10. A single PM's CPU, which is used to host these applications, has total capacity of 1200 MIPS. Assuming capacities a_{11} and a_{12} are the CPU allocation for VM1 and VM2 Figure 4 shows the whole configuration space over these allocations, constrained by inequalities: $a_{11} + a_{12} < 1200$, $a_{11} > 280$, and $a_{12} > 280$. Notice how the optimizer climbs up the global utility function, hits the constraint (i.e., $a_{11}+a_{12} < 1200$) then continues to climb and reach the maximum.

4. Case Studies

For the case studies in this paper we simulated a private cloud with CloudSim [26] simulator. It was assumed that the administrator responsible for configuring the feedback optimizer had access to enough information regarding the deployed applications so as to construct simple models for them[8].

A data-set of a sample application's performance, was generated synthetically by invoking the CloudSim simulator using random values for capacity, demand and inter-arrival time (Figure 5 presents a partial visualization of the data-set).

Three steady-state models were considered:

1. A simple linear model only containing predictor variables[9] ($s_j = \beta_1 + \beta_2 d_j - \beta_3\tau_j - \beta_4\gamma_j$).
2. A more complex linear model with interaction terms [10] $\gamma_j\tau_j$ and $\gamma_j d_j$ to capture the effect of γ_j on response time at different arrival rates and demands ($s_j = \beta_1 + \beta_2 d_j - \beta_3\tau_j - \beta_4\gamma_j - \beta_5\gamma_j d_j + \beta_6\gamma_j\tau_j$).
3. A non-linear model derived from the mean value based formula for open queuing networks ($R = D/(1 - \lambda D)$) [27] extended to account for capacity ($s_j = \frac{\beta_1 d_j}{\gamma_j - \beta_2 d_j/\tau_j}$) [11]

A regression analysis was then performed on the resultant data for each model allowing confidence intervals to be computed for all model coefficients. In all cases, the approximate 95% confidence intervals for coefficients were quite narrow and the margin of error was orders of magnitude smaller than the coefficient. This implies that the amount of data for modeling was adequate. However, in linear models the intervals were very near to zero and coefficients were not significant. In the end as a result of this analysis, the non-linear model (i.e., 3) was used.

We further considered a very simple local utility function for applications:

$$u_j = u_{\max_j} \cdot \frac{\left(\arctan(r_{\text{thr}_j} - r_j) + \pi/2\right)}{\pi} \tag{6}$$

where $u_{\max_j}$ denotes the maximum possible utility as defined by the curve, r_{thr_j} denotes the SLA limit (in terms of response time) and r_j denotes the actual response time.

[8]Only a single application was used for which a model was designed. Multiple instances of this application were deployed.

[9]The model is linear in the parameters β_i not predictor variables.

[10]The interaction term is the product of the subset of our original predictors.

[11]This equation can be derived by substituting D_j with d_j/γ_j and λ_j with $1/\tau_j$ from the base formula, and adding model coefficients (i.e. β_1 and β_2) to compensate for the differences between the system and pure queuing model. The substitution requires the arrival rate (λ) to be a homogeneous Poisson process, and inter-arrival times (τ) to be exponentially distributed with parameter λ (mean $1/\lambda$).

Num	Equation	Description
1	$U_i(A) = \left(\sum_{j \in app} a_{ij}\right) - c_i$	A constraint for each PM_i ($i > 0$) is the sum of all resources given to VMs deployed on that PM minus the PM's total capacity.
2	$U_0 = \left(\sum_{j \in App} u_j(s_j(\gamma_j, \tau_j, d_j))\right) - k.cost(A)$	The global utility function U_0.
3	$fp_{\text{best}} = max(U_0(A), fp_{\text{best}})$	Incremental recording of objective value.
4	$g = \partial U_0(A)/\partial A$	Calculating the objective subgradient as if the problem were unconstrained.
5	$A = A + \alpha g_0$	Moving towards the objective function subgradient with a fixed step size (i.e. optimality update).
6	$A = A - \alpha g_i$	Projecting the current point onto the set of points that satisfy the inequality constraint i.
7	$i = argmax_i(\|U_i(A)\|)$	Choosing the most violated constraint.
8	$\forall k \forall j\{[(k = i \wedge placement(k, j) = 1) \rightarrow (g_i)_{kj} = 1] \wedge [k \neq i \rightarrow (g_i)_{kj} = 0]\}$	Calculating the subgradiet g_i of a constraint i (i.e. $\partial U_i(A)/\partial A$) while $i > 0$.
9	$\alpha = (U_i(A) + \epsilon)/(\|g\|_2^2)$	Calculating the step size in feasibility update, while $\epsilon > 0$ is a tolerance and set to 10^{-3}.
10	$\forall i \forall j\{a_{ij} = min(max(a_{ij}, 0), c_i)\}$	Check that each allocated capacity is feasible and is between the local bounds.

Table 1: Equations referenced by Algorithm 1.

4.1. Case Study One

Consider a small cloud configuration with three applications (app1, app2, app3) deployed on two PMs with the following placement matrix:

$$placement = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

That is, app1 is deployed on PM1, app3 is deployed on PM2, and app2 uses both PMs by having a VM on each.

Both of the PMs have single core CPUs with each core having processing speed of 1200 MIPS and 2 GB of ram. The VMs for applications are identical with image size of 1 GB, ram of 512 MB, and bandwidth of 1 GB. The CPU resource share for each VM is initialized to 280 MIPS but will be adjusted by the optimizer. At the PaaS layer applications are assumed to have workloads with mean CPU demand per request of 4100, 5700 and 500 MIPS respectively. For app2, the load gets distributed evenly between the two VMs. We chose the arrival-rates of applications, from FIFA '98 workload [28]. The interarrival time of requests for all dynamic pages were extracted, on a per-minute basis, from the first six hours of day 21 of FIFA 98 workload. To have the time-of-a-day effect we divided this into three 2-hour periods and applied each as an application workload. Moreover we multiplied interarrival-times of applications by 14, 11 and 8 respectively. The third application is more transactional in nature (i.e. lower demand and higher arrival rate) while the other ones are characterized as batch-like.

The utility functions are defined for each application based on equation 6 with the following parameters for each application:

$$\mathbf{r}_{\text{thr}} = [15, 16, 2]$$
$$\mathbf{u}_{\max} = [20, 10, 5]$$

For each 120 samples of interarrival times, we submit 20 transactions on each application in the simulator. We ran the experiment twice. One run used the static model with the coefficients computed off line as mentioned above. The second run used the same model but tuned at runtime. In the case of the tuned model, the new measurements obtained from the simulator (e.g. response times) are passed to the filter. The Kalman filter introduced in Subsection 2.3 re-tunes the dynamic model's coefficients and passes the model to the optimizer. In the case of a static model, the optimizer uses the model obtained through regression analysis as-is and no tuning is happening. The optimizer then recalculates the new resource allocations, passes the new allocation vector to actuators to apply it to the system.

Figure 6 presents the results. The small mismatch of

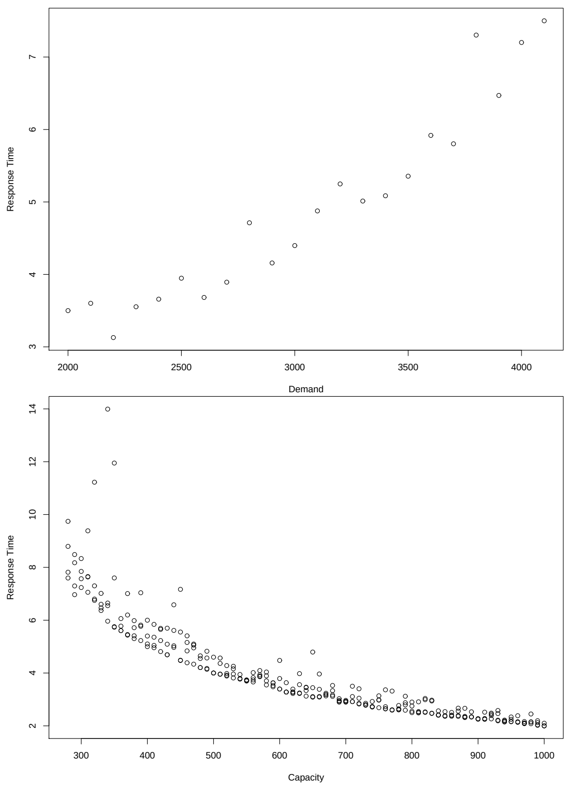


Figure 5: Visualization of pairwise relation between input parameters (capacity,demand) and the response attribute (i.e. response time) for a simulated application running on a single VM.

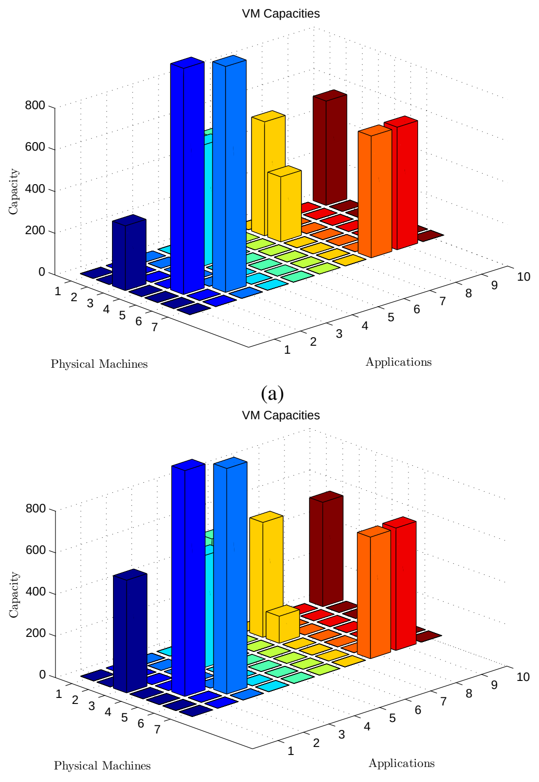


Figure 7: A snapshot of resource allocation both (a) before and (b) after an increase in workload is detected.

model and measurements due to the use of static models by the applications (see Figure 6(a)) results in a bad allocation decision for app2, this leads to failure in meeting its SLA (see Figure 6(e)). In contrast, using dynamic models results in more efficient resource allocations being made and better commitment to SLAs (i.e, response time for app2 (see Figure 6(f)).

4.2. Case Study Two

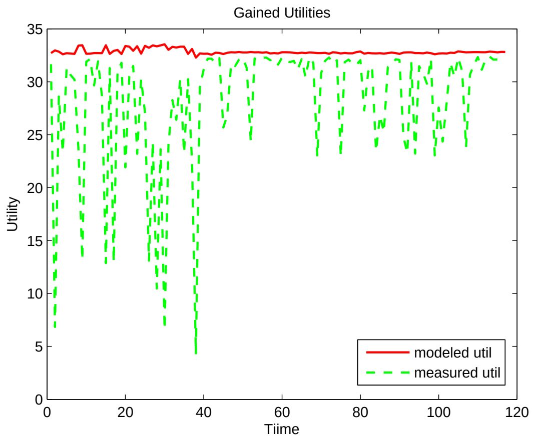
The second case study was performed to demonstrate the ability of our approach to successfully update resource shares in a way that compensates for additional workload. Ten applications were deployed on seven PMs. VMs were assigned randomly to PMs. All application workloads were set to have an interarrival time of 40 seconds except app1 whose workload was monotonically increasing (i.e., interarrival time was decreasing from 40 to seven seconds).

Figure 7 presents both before (a) and after (b) snapshots of resource allocations. Notice that both app1 (dark blue) and app7 (yellow) are resident on the same PM. Initially, app1 is allocated approximately 300 MIPS while app7 is allocated approximately 310 MIPS. After the increase in workload app1 is allocated approximately 510 MIPS while app7 is allocated approximately 100 MIPS.

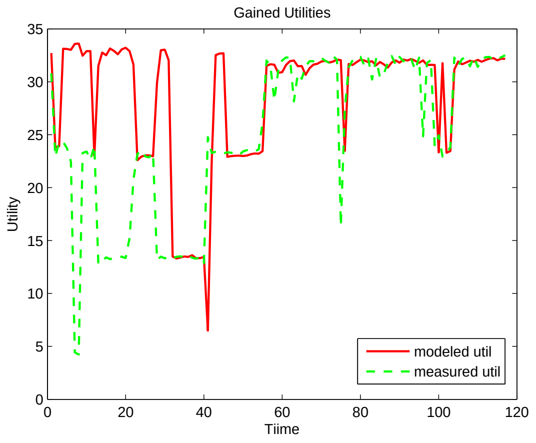
5. Related Work

Maintaining application level QoS guarantees has been the subject of much investigation. Recently satisfying the dual objectives of QoS guarantees and server consolidation in cloud computing environments has received considerable attention. Current approaches formulate the optimization problem in different forms.

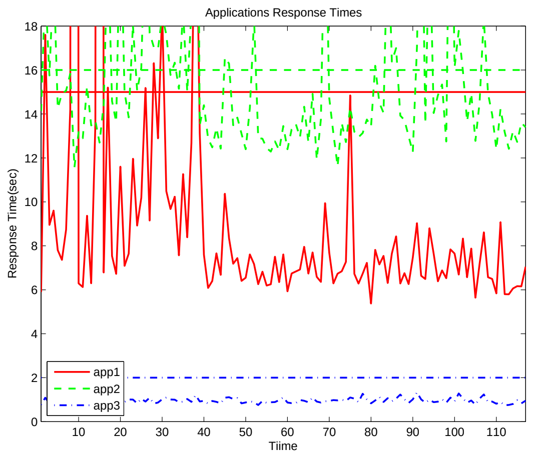
One approach attempts to minimize cost subject to performance constraints [12, 13]. In this approach the SLA represents constraints rather than flexible goals. Constraints could be based on response time or throughput. For example [12] finds the minimum cost deployment subject to processing capacity and user throughput constraints. It seeks deployments which minimizes the overall cost of the hosts used, subject to meeting average



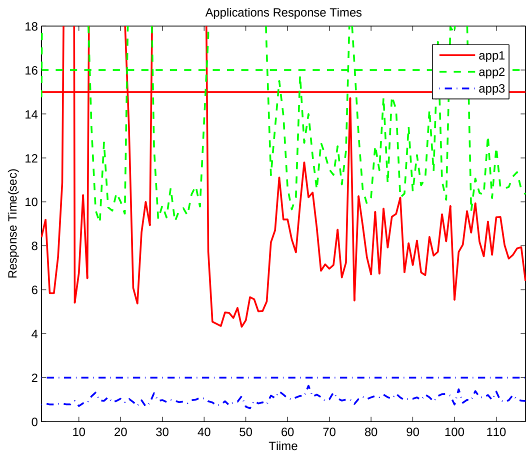
(a)



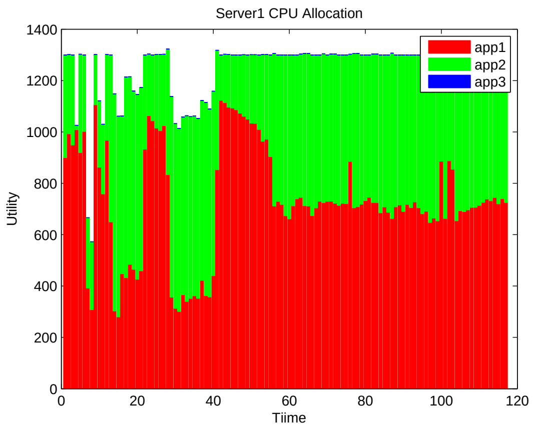
(b)



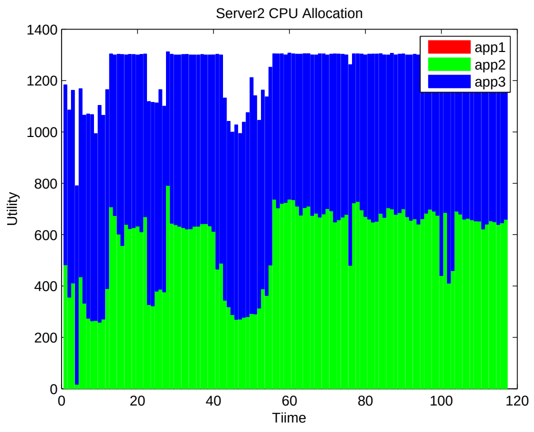
(c)



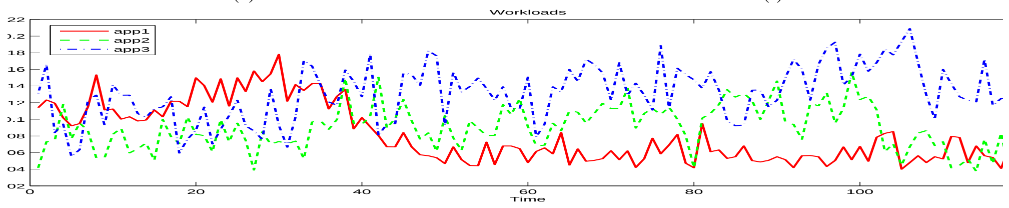
(d)



(e)



(f)



(g)

Figure 6: The (a) measured and (b) modeled gained global utility using static and dynamic models over time. The response time of applications together with SLA response times for (c) static and (d) dynamic models. The allocated capacity to applications over time on (e) server one and (f) server two respectively using dynamic models. The (g) workload of applications.

delay and throughput constraints for each application as posed by its SLA.

A second approach attempts to simultaneously minimize cost while maximizing QoS attributes, through multi-objective optimization or MOO [14]. For example, Pareto-optimal solutions can find a good trade-off between conflicting performance and cost-saving goals rather than finding a single global optimum [29]. Geometrically, these well-balanced solutions concentrate around the "knee" of a multi-objective curve.

A third approach is the one that was used in this paper, that is, to optimize a utility function combining application-level SLAs and resource costs with tunable parameters for the administrator to specify trade-offs between the two [14]. In this approach a system-level global utility is defined in terms of local utilities which are in turn based on the achieved service level of the applications. These local utilities are combined with a set of coefficients that allows for the high level control of performance goal fulfillment and the resource cost savings.

Finally, some techniques try to satisfy SLAs using a pure control-based feedback loop [11, 23, 24]. These approaches are categorized as control theoretic constraint satisfaction, rather than optimization; however, they are used in our estimation technique. Also the idea of dynamically adjusting the resource shares of multiple applications in order to meet a specified level of service differentiation, was used in [11] and [30], although using an adaptive multivariate controller and for a more limited scenario than ours (i.e. maximum one PM per application layer).

6. Conclusions and Future Work

The purpose of this work was to demonstrate the advantage of "adaptive" models relative to "static" models in optimization. We investigated model based optimization of a private cloud where applications are clustered across a known, homogeneous set of PMs. In this optimization, we modified the resource shares of applications, in order to minimize the SLA violations. While we focused only on response time, considering multiple service level objectives will not change the approach (just the complexity of solving the optimization problem).

The main contribution of this work was dynamically using tracking models (for each application) within the global optimization loop. These models update themselves at runtime in order to adapt to perturbations in the environment not captured in initial model specification. This results in more accurate models being passed to the COM allowing for better resource utilization on a global scale.

The relationship between the number of simulated VMs and optimization time per step was also considered and a non-linear relation was observed. While we acknowledge that this could pose a problem for our approach (i.e., when working with large numbers of VMs) a possible solution would be to split the problem into subsets and solve them individually.

Future work will involve implementing the optimization algorithm in a distributed manner in which applications interact in a peer-to-peer fashion to determine how much resource should be allocated to each. We are also investigating optimizing the dual problem, which is more suitable for mapping to an agent-oriented optimization approach (see [31, 32]).

Another direction will involve extending the performance model to include more details about an application's structure. For example the model can be aware of different layers of an application such as web server, application server, and database server. Further, consideration of heterogeneous resources is also interesting and should be considered as well.

Acknowledgments

This research was supported by OCE, the Ontario Centres of Excellence, and by the IBM Toronto Centre for Advanced Studies, as part of the program of the Centre for Research in Adaptive Systems (CERAS).

References

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM, New York, NY, USA, 2003, pp. 164–177.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, Above the clouds: A berkeley view of cloud computing, Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (February 2009).
URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`

[3] B. Hayes, Cloud computing, Communications of the ACM 51 (7) (2008) 9–11.

[4] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. M. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, F. Galan, The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 4–11.

[5] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems. 25 (6) (2009) 599–616.

[6] H. H. Huang, A control-theoretic approach to automated local policy enforcement in computational grids, Future Generation Computer Systems 26 (6) (2010) 787 – 796.

[7] C. S. Yeo, S. Venugopal, X. Chu, R. Buyya, Autonomic metered pricing for a utility computing service, Future Generation Computer Systems 26 (8) (2010) 1368 – 1380.

[8] W. Iqbal, M. N. Dailey, D. Carrera, P. Janecek, Adaptive resource provisioning for read intensive multi-tier applications in the cloud, Future Generation Computer Systems 27 (6) (2011) 871 – 879.

[9] L. Rodero-Merino, L. M. Vaquero, V. Gil, F. Galn, J. Fontn, R. S. Montero, I. M. Llorente, From infrastructure delivery to service management in clouds, Future Generation Computer Systems 26 (8) (2010) 1226 – 1240.

[10] M. A. Murphy, S. Goasguen, Virtual organization clusters: Self-provisioned clouds on the grid, Future Generation Computer Systems 26 (8) (2010) 1271 – 1281.

[11] X. Liu, X. Zhu, P. Padala, Z. Wang, S. Singhal, Optimal multivariate control for differentiated services on a shared hosting platform, in: Decision and Control, 2007 46th IEEE Conference on, 2007, pp. 3792–3799.

[12] J. Z. Li, J. Chinneck, M. Woodside, M. Litoiu, Fast scalable optimization to configure service systems having cost and quality of service constraints, in: Proceedings of the 6th international conference on Autonomic computing, ACM, 2009, pp. 159–168.

[13] J. Li, J. Chinneck, M. Woodside, M. Litoiu, G. Iszlai, Performance model driven QoS guarantees and optimization in clouds, in: Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, IEEE Computer Society, 2009, pp. 15–22.

[14] H. Li, G. Casale, T. Ellahi, SLA-driven planning and optimization of enterprise applications, in: Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering, ACM, 2010, pp. 117–128.

[15] H. N. Van, F. D. Tran, J. M. Menaud, SLA-Aware virtual resource management for cloud infrastructures, in: IEEE Ninth International Conference on Computer and Information Technology, IEEE, 2009, pp. 357–362.

[16] G. Tesauro, W. E. Walsh, J. O. Kephart, Utility-Function-Driven resource allocation in autonomic systems, in: Proceedings of the Second International Conference on Automatic Computing, IEEE Computer Society, 2005, pp. 342–343.

[17] X. Wang, Z. Du, Y. Chen, S. Li, D. Lan, G. Wang, Y. Chen, An autonomic provisioning framework for outsourcing data center based on virtual appliances, Springer Netherlands 11 (3) (2008) 229–245.

[18] X. Wang, D. Lan, G. Wang, X. Fang, M. Ye, Y. Chen, Q. Wang, Appliance-Based autonomic provisioning framework for virtualized outsourcing data center, in: Proceedings of the Fourth International Conference on Autonomic Computing, IEEE Computer Society, 2007, p. 29.

[19] J. Sauvé, F. Marques, A. Moura, M. C. Samaio, J. Jornada, E. Radziuk, Sla design from a business perspective, DSOM 2005: Proceedings of the 16th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (2005) 72–83.

[20] I. n. Goiri, F. Julia, J. Ejarque, M. d. Palol, R. M. Badia, J. Guitart, J. Torres, Introducing virtual execution environments for application lifecycle management and sla-driven resource distribution within service providers, in: Proceedings of the 2009 Eighth IEEE International Symposium on Network Computing and Applications, NCA '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 211–218.

[21] M. Comuzzi, C. Kotsokalis, G. Spanoudakis, R. Yahyapour, Establishing and monitoring slas in complex service based systems, in: Proceedings of the 2009 IEEE International Conference on Web Services, ICWS '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 783–790.

[22] A. Ferrer, F. Hernández, J. Tordsson, E. Elmroth, C. Zsigri, R. Sirvent, J. Guitart, R. Badia, K. Djemame, W. Ziegler, et al., OPTIMIS: a Holistic Approach to Cloud Service Provisioning, Future Generation Computer Systems, accepted.

[23] E. Kalyvianaki, T. Charalambous, S. Hand, Self-adaptive and self-configured CPU resource provisioning for virtualized servers using kalman filters, in: Proceedings of the 6th international conference on Autonomic computing, ACM, Barcelona, Spain, 2009, pp. 117–126.

[24] T. F. Abdelzaher, K. G. Shin, N. Bhatti, Performance guarantees for web server end-systems: A control-theoretical approach, IEEE Transactions on Parallel and Distributed Systems (2002) 80–96.

[25] G. Welch, G. Bishop, An introduction to the kalman filter, University of North Carolina at Chapel Hill, Chapel Hill, NC. URL `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`

[26] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience 41 (1) (2011) 23–50.

[27] E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik, Quantitative system performance: computer system analysis using queueing network models, Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1984.

[28] M. Arlitt, T. Jin, A workload characterization study of the 1998 world cup web site, Network, IEEE 14 (3) (2000) 30–37.

[29] A. Soror, U. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, S. Kamath, Automatic virtual machine configuration for database workloads, ACM TRANSACTIONS ON DATABASE SYSTEMS 35 (1) (2010) Article 7.

[30] H. C. Lim, S. Babu, J. S. Chase, S. S. Parekh, Automated control in cloud computing: challenges and opportunities, in: Proceedings of the 1st workshop on Automated control for datacenters and clouds, ACM New York, NY, USA, 2009, pp. 13–18.

[31] P. Huang, H. Peng, P. Lin, X. Li, Macroeconomics based grid resource allocation, Future Generation Computer Systems 24 (7) (2008) 694 – 700.

[32] H. Izakian, A. Abraham, B. T. Ladani, An auction method for resource allocation in computational grids, Future Generation Computer Systems 26 (2) (2010) 228 – 235.