This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Analysis of response time percentile service level agreements in SOA-based applications

Keerthana Boloor*, Rada Chirkova†‡, Tiia Salo‡, Yannis Viniotis*‡
*Department of Electrical and Computer Engineering
†Department of Computer Science
North Carolina State University, Raleigh, North Carolina, USA
‡IBM Software Group, Research Triangle Park, North Carolina, USA

***Abstract*—A large number of enterprise, web-based, distributed software applications are designed on Service Oriented Architecture (SOA) principles and hosted in large scale datacenters managed by cloud providers. Typically, Service Level Agreements (SLAs) are negotiated between the consumers of the cloud platform services and the cloud provider. In this work, we consider SLAs that involve percentiles of response times as part of the performance metrics; the SLAs stipulate that a penalty be charged to the cloud provider if the SLA targets are not met. The main motivation for considering such SLAs is their potential for price differentiation. We focus our analysis on the effects the penalty function has on the achieved response time percentiles. In particular, we analyze the effect of three commonly deployed choices (linear, exponential or step-wise functions) to relate the penalty charged and the achieved percentile. This analysis is NP-hard, so we employ a heuristic algorithm that is based on simulated annealing. Our results indicate that the linear penalty charging function is "best" in the sense that it maximizes the achieved response time percentiles.**

I. Introduction and Motivation

A Service Level Agreement (SLA) formally specifies terms and conditions negotiated between a service provider and a service consumer. The SLA may involve a variety of metrics, called Key Performance Indicators (KPI), that describe the service; typical examples of KPIs include cost, security, reliability, and delays. SLAs have been employed in the IT industry (e.g., Internet Service Providers, telecommunication providers, technical support providers). In the telecommunication industry, in particular, they have been used extensively for more than 30 years. More recently, they are being used in the cloud computing industry, for offering and pricing services such as Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [1].

Recently, enterprise, web-based applications are largely built on Service Oriented Architecture (SOA) principles. SOA-based applications are loosely composed of functional software components called service end-points. The functionality of the application can now be offered as a service collectively provided by composition of the individual service end-points. Application providers can thus be viewed as providing SaaS to their end-customers.

The most common platform for deployment of SOA-based enterprise applications is the cloud computing system [2]. A typical, distributed cloud computing system is composed of a large number of networked resources that can be provisioned on-demand; it offers the potential of seamlessly managing hosted applications, making the platform flexible, robust and energy-efficient. Apart from such technical advantages, the main business motivation SaaS providers have for deploying their applications in a cloud computing system (also known as a datacenter) is, ultimately, cost efficiency; this derives from the ability to obtain fine-grained control on provisioning resources and thus from the option to negotiate accurate pricing models with the cloud/datacenter provider. In the SLAs analyzed in this work, the **provider** is the cloud provider who offers resources provisioned on-demand and manages hosted applications as a service; the **consumer** is the SaaS vendor whose applications are hosted in the datacenters. Typical KPIs of interest to the SaaS vendors include availability of the software to their end-customers, response times, data capacity (if storage is required) and so on.

In this work, we focus on percentile response times as the KPI metric of interest in the SLA negotiated between the provider and the consumer. From the consumer's point of view, when used as a KPI, percentiles of response times, as opposed to averages, provide a better control of variability in delays experienced by user requests. Controlling the percentiles enforces a "shape" on the entire distribution of response times, and thus goes beyond simply bounding the average delay; this capability is deemed more suitable for (near real-time) SOA applications.

SLAs using percentile response time KPIs can be stated in many different ways; moreover, there is no standardization in place. This presents numerous technical challenges and is a potential business barrier. We define such an SLA more precisely in the next section. Suffice it here to say that, such SLAs are usually and broadly expressed by specifying "the fraction of service requests to be executed within a defined response time deadline", along with the penalty charged on the cloud provider on the non-conformance of the percentile requirement. Note that the provider may host multiple classes of applications (from one or more consumers) in the same datacenter, each negotiated with its own percentile SLA.

From the cloud provider's point of view, controlling the penalty charged is, of course, important. The mechanism for that is the "request management policy" the provider can put in place. In general, service requests are queued in the datacenter they arrive. Then, the policy determines how they

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

are allocated to a server and how they are scheduled at the allocated server in order to, for example, minimize the penalty charged according to the SLAs negotiated.

The exact relation between the metric of percentile conformance and the monetary penalties charged on the provider due to non-conformance (hereon referred to as the "penalty function") is an important parameter of the SLA [3]. As already mentioned, there are multiple ways in relating the two and thus a variety of penalty functions can be chosen. In this work, we investigate three variants of the penalty function: linear, exponential and step-wise. We utilize a heuristic request management policy (based on simulated annealing) which is shown in [4] to maximize response time percentiles in comparison to variants. This scheme is independent of the penalty function utilized and is thus suitable for evaluating the variants. In our simulations, we show that utilizing a linear response time percentile SLA enables the highest percentile conformance. To the best of our knowledge this is the first work to analyze the effect of the choice of a penalty function to the total response time percentile conformance for SOA applications. The results of the analysis are of use to SaaS vendors and cloud providers for negotiation of appropriate pricing models.

The paper is organized as follows. In Section II, we define precisely the percentile SLA we will consider and define the three penalty function variants. In Section III, we explain our evaluation methodology. In Section IV, we present the results of our analysis. Finally, in Section V, we list related work.

II. Percentile Service Level Agreements

In this section, we define the response time percentile SLAs more precisely and detail the three penalty function variants one can use in such SLAs.

A. SLA Parameters

We identify the following parameters of interest in the percentile SLA:

1) Response time, R: The response time we consider in this work is defined as the duration of the time it takes from when a request first arrives at the datacenter and to the time the response for the request leaves the datacenter. The request is said to "complete" once the response leaves the datacenter. The request could involve invoking a composition of multiple service end-points across various applications and/or request functionality across multiple tiers of the same application. The response time includes the queuing and execution delays at each component of the application. R can be monitored by both the consumer and the cloud provider.

2) Response time threshold, R_T: The response time threshold, R_T, is the "deadline" by which a request must leave the datacenter. The value of R_T is negotiated between the consumer and the cloud provider. We say that a request is "conforming" if $R \leq R_T$, i.e., if the request completes within the negotiated response time.

3) Desired conforming percentile, X_T: The desired conforming percentile, X_T, is the fraction of requests for which the consumer wishes execution to be completed by the "deadline", i.e., for which the cloud provider must ensure that $R \leq R_T$. The value of X_T is negotiated between the consumer and the cloud provider.

4) Actual conforming percentile, X: The actual (or simply the) conforming percentile, X, is the fraction of requests for which the cloud provider has ensured that $R \leq R_T$. The value of X is controlled by the request management policy chosen by the cloud provider.

5) Observation interval, T: The percentile SLA includes a period of time called the observation interval over which the fraction of requests conforming to the negotiated response times is computed and penalties (if any) are charged for the observation interval. For example, a typical percentile SLA could require 70% of requests of a particular application to have a response time of less than 10 milliseconds, computed over an observation interval of 10 days.

6) Penalty, P: The percentile SLAs considered in this paper call for economic penalties if the fraction of conforming requests is less than a certain value. The SLAs also define the method of calculating the penalties. In this work we consider two ways of penalty calculation:

- Per-request penalties: If the fraction of conforming requests is less than a given value, the penalty is calculated in proportion to the number of requests contributing to the reduction in conformance. For example, suppose that the percentile SLA requires $X_T = 70\%$ of the requests of a particular class to have a response time of less than $R_T = 10$ milliseconds. Out of the 1000 requests that arrive at the datacenter only 600 requests conform i.e. have a response time of less than 10 milliseconds resulting in a conformance percentile of $X = 60\%$. Penalty of \$P is charged on each of the 100 requests resulting in reduction in the conformance percentile. In this case, penalties depend on the number of requests sent to the datacenter.
- Per-fraction penalties: If the fraction of conforming requests is less than a given value, the penalty is calculated in proportion to the fraction of requests contributing to the reduction in conformance. For example, consider the percentile SLA requires $X_T = 70\%$ of the requests of a particular class to have a response time of less than $R_T = 10$ milliseconds. Out of the 1000 requests that arrive at the datacenter only 600 requests conform i.e. have a response time of less than 10 milliseconds resulting in a conformance percentile of $X = 60\%$. Penalty of \$P is charged on the fraction of 10% resulting in reduction in the conformance percentile. In this case, penalties do not depend on the number of requests sent to the datacenter.

B. Percentile SLA definition

Based on the above parameters, an SLA using percentile of response time KPIs can be precisely stated as follows:

"Over the observation interval of T time units, at least X_T% of requests of the application should have a response time less than R_T time units, else a penalty \$P is charged to the datacenter provider."

C. Enforcement of the SLA

The cloud provider has multiple control mechanisms at his/her disposal to monitor and control such an SLA. For example, among the most common ones would be allocation of requests to an available server (e.g., request allocation) and allocation of CPU time to requests inside a server (e.g., scheduling). These controls are collectively called the request management policy. The policy is chosen to satisfy management goals. A typical goal in datacenters would be to determine a request allocation and scheduling algorithm that provides the minimum in Equation 1 below:

$$\min \sum_{1 \le k \le K} penalty_k \tag{1}$$

where $penalty_k$ is the penalty charged for non-conformance of the requests from users of application k; K is the total number of SOA-based applications the cloud is hosting.

D. Variants of penalty functions

In typical response time percentile SLAs, the penalties charged (\$P) are obtained as a function $P(X)$ of the fraction of conformance (X%). In this section we present three forms of penalty functions that have been used as parts of SLAs.

1) Linear penalty function: The Linear penalty function shown in Figure 1 is formally defined as:

$$P(X) = \begin{cases} P_{\max} - X \cdot m, & \text{if } X \le X_{\text{zero}}, \\ 0, & \text{if } X_{\text{zero}} < X \le 100. \end{cases}$$

where $P_{\max}$ is the penalty charged when the conformance is 0%, $m = P_{\max}/X_{\text{zero}}$, X_{zero} is the percentile (%) conformance at which the penalty charged is zero.

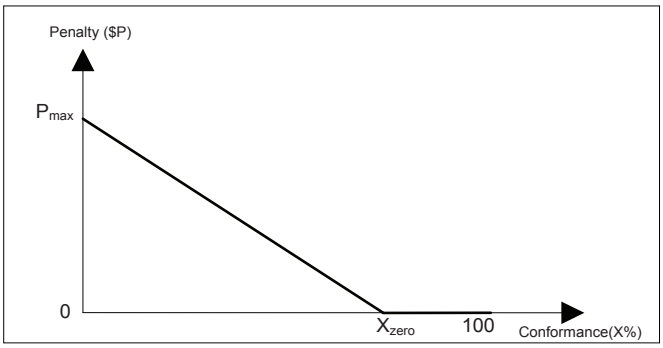

Fig. 1: Linear penalty function.

2) Exponential penalty function: The Exponential penalty function shown in Figure 2 is formally defined as:

$$P(X) = P_{\max} \cdot e^{-\lambda \cdot X},$$

where $P_{\max}$ is the penalty charged when the conformance is 0%, $\lambda = \ln(10)/X_\alpha$, and X_α is the percentile at which the penalty should be $(1/10) \cdot P_{\max}$ (X_α is similar to the half-life value in exponential decaying of unstable atoms, except we set the value to (1/10)th the maximum instead of (1/2)).

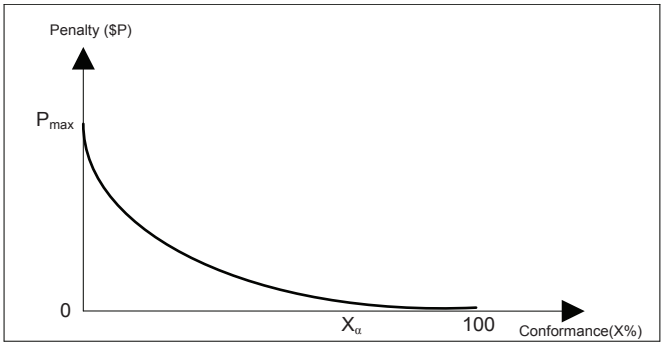

Fig. 2: Exponential penalty function.

3) Step-wise penalty function: The Stepwise penalty function is shown in Figure 3 (for the special case of $n = 3$ steps). The function is defined via n pairs (P_i, X_i), where P_i is the penalty charged when the conforming percentile is X_i.

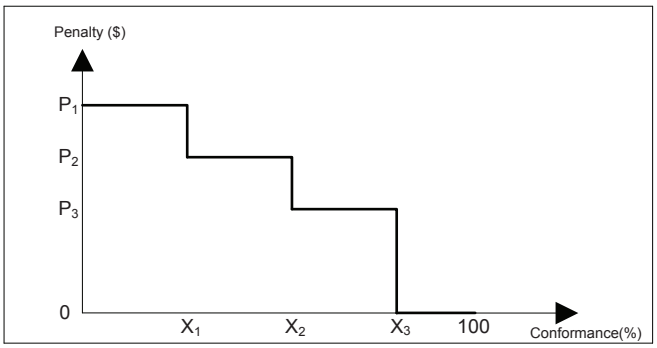

Fig. 3: Stepwise penalty function.

E. Selection of penalty functions

The linear function is the most intuitive and commonly used in pricing models in SLAs across industries [5]. Stepwise functions are heavily used in pricing models as they enable categorization of a range of service parameter values into groups or classes having the same utility value, making it essential to analyze the effect of a utilizing a piecewise linear function in percentile SLAs [6]. We have chosen the exponential function as a model "in-between".

The main focus of this work is to analyze the effect of the penalty function on the achieved percentile conformance, under a given request management policy. In the next section, we present the evaluation methodology used to compare the three variants of the percentile SLA described.

III. Evaluation methodology

The main criterion for comparison is the total number of requests that have a response time less than that specified in the SLA negotiated, when the "best known" allocation and scheduling policy is used for minimizing penalty in each of the three percentile SLAs. The problem we consider belongs to the class of problems encompassing utility-maximizing schemes for allocation and scheduling of multi-class jobs to

multiple resources. This class of problems has been proven to be NP-hard [7]. The evaluation methodology is thus based on heuristics. Also, due to the NP-hardness nature of the problem, it is not possible to analytically prove optimality of the proposed heuristics-based solution.

In order to evaluate the three variants, we adopted a methodology based on simulated annealing. Simulated annealing is a probabilistic metaheuristic method for finding the global minimum of a cost function with many local minima. It works by emulating the physical process whereby a solid is cooled slowly and eventually the object "freezes" at the lowest energy configuration [8]. Following this analogy, at each step of simulated annealing, the current solution is replaced by a "nearby" solution which is chosen based on a probability that depends on the difference between the current function value and the new function value and a global parameter called the temperature that is gradually decreased in each step. The dependency on the temperature is such that the "nearby" solution is random when temperature is large, but randomness decreasingly "descends" as the temperature tends to zero. The randomness allows for uphill movements which prevents searching only at a particular local minimum, thus resulting in a better solution than greedy methods. This characteristic of simulated annealing based solution search technique makes it a good fit for evaluation of the three percentile SLAs as the solution will not be dependent on the shape of the cost function.

Algorithm 1 describes the heuristic simulated annealing based allocation and scheduling scheme used for evaluation of the three percentile SLA variants. We assume a work conserving system. If all end-servers are busy, requests are queued at the datacenter as they arrive. A request from the queue is allocated to an end-server as soon as it is free. The request to be serviced is chosen by a simulated annealing based heuristic described in Algorithm 2. In [4], we have shown that simulated annealing based scheduling schemes result in maximizing utility of percentile response times. In the iterative scheme adopted, the first seed schedule we begin with is First Come First Serve (FCFS), a global start temperature, end temperature and step values are chosen (t, ϵ, α respectively in Algorithm 2 that determine the number of iterations). In each iteration, two requests from the queue are chosen at random and their positions are interchanged in the schedule and the penalty ensuing from the resulting schedule is calculated. If this schedule is the lowest obtained, the schedule is saved as the final schedule. This schedule now becomes the seed schedule in the next iteration. If the penalty obtained is not the lowest obtained among all the iterations, then the probability that the schedule is used as the seed schedule in the next iteration, depends on the difference of the penalty obtained from that obtained in the previous iteration and the current value of the global temperature variable. At the end of the iterations, the final schedule is the one which has resulted in the least penalty. The request at the head of the final schedule is allocated to the end-server.

Algorithm 1 Allocation and scheduling of requests.

Input: Number of servers in the datacenter: N
Incoming requests are inserted at the end of the request queue at the datacenter
while Queue of requests is not empty **do**
  if any server i in N is free **then**
    $best\_schedule$ = Schedule found in Algorithm 2
    Allocate request at the head of $best\_schedule$ to server i
  end if
end while

Algorithm 2 Simulated annealing based heuristic algorithm for best schedule resulting in least penalty.

Input:
- Number of requests queued: R
- Observation interval: T
- Starting temperature: t
- Ending temperature: ϵ
- Step-value: α
- $final\_penalty$: Infinity
- Seed schedule of requests: $current\_schedule$

Output: Schedule resulting in lowest penalty: $final\_schedule$

while $(t > \epsilon)$ **do**
  $old\_penalty$ = penalty resulting from $current\_schedule$
  $new\_schedule$ = Interchange positions of 2 randomly chosen requests in $current\_schedule$
  $new\_penalty$ = penalty resulting from the $new\_schedule$
  if $(new\_penalty < final\_penalty)$ **then**
    $final\_penalty = new\_penalty$
    $final\_schedule = new\_schedule$
    $current\_schedule = new\_schedule$
  else
    if $(random\_number < e^{(new\_penalty - old\_penalty)/t})$ **then**
      $current\_schedule = new\_schedule$
    end if
  end if
  $t = t * \alpha$
end while

This scheduling scheme is compute-intensive and is not practical to implement in real-world deployments. However, since this work focuses on analysis of the three variants of the penalty functions, we chose the scheduling policy which results in the "best" schedules [4] for percentiles.

In the next section, we evaluate the effect of the three penalty functions based on the methodology described above.

IV. Evaluation results

We have built a discrete event simulator for the analysis of the choice of the penalty function when the best-known request allocation and scheduling scheme based on simulated annealing technique is employed for minimizing penalties charged. This section details simulation results to answer the following top-level questions:

1) Do we have a clear winner among the 3 variants of the response time percentile SLA considered?

2) Is there a difference between conformance in the per-request penalties and per-fraction penalties in each of the 3 variants considered?
3) What is the relation between step-size and observation interval in step-wise SLA?

A representative scenario involves a datacenter with (a) 10 end-servers in each datacenter, (b) 5 distinct applications with distinct values of the different SLA parameters for each of the 3 variants, (c) the input arrival process is Poisson, (d) the service processes are exponential.

A. Comparison of conformance in the 3 variants

Typical results in Figure 4 (with 95% confidence intervals) show that linear percentile Service Level Agreement consistently results in higher number of conforming requests even as input request rates are increased. Step-wise SLAs with lower number of steps results in the lowest number of conforming requests. However, if the number of steps are increased (essentially more accurately approximating linear SLA), the number of conforming requests increase.

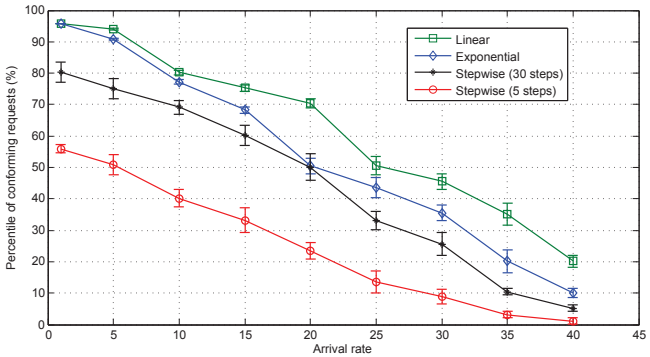

Fig. 4: Comparison of percentile SLA variations (linear, exponential and stepwise)

B. Difference in utilizing per-request and per-fraction penalties

We now compare the effect of utilizing per-request and per-fraction penalties. As shown in Figure 5 (with 95% confidence intervals), for linear and exponential response time percentile SLAs, per-request penalties result in higher conformance. This is as expected as the penalties are charged at a lower granularity resulting in improved adaptability of the management scheme to the dynamic nature of the system. However, for step-wise percentile SLAs the improvement in utilizing per-request percentile SLA is lower than that in the former penalty functions.

C. Relation between step-size and observation interval

Figure 4 shows that as the number of steps in the step-wise interval increase, thereby approximating the linear SLA, the number of conforming requests increase. In Figure 6 (with 95% confidence intervals) we show that, as the length of the observation interval increases, the effect of a smaller number of steps in the step-wise SLA reduces resulting in larger percentile of conforming requests. The observation interval

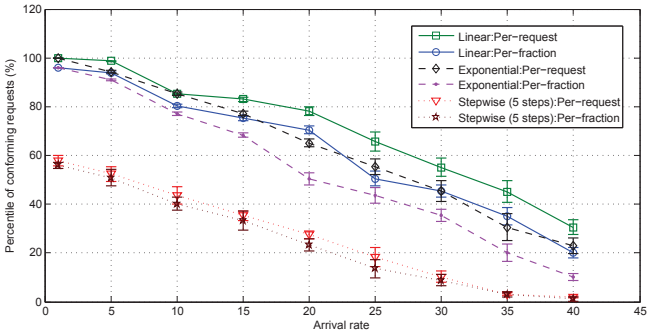

Fig. 5: Comparison of per-request and per-fraction penalties

is expressed as the number of times the average service time of an end-server in the datacenter. This representation of the observation interval is chosen as the average service time the total number of requests served over a time period and the observation interval of the SLA should be large enough to incorporate a high number of completed requests to accommodate variations in the input rate.

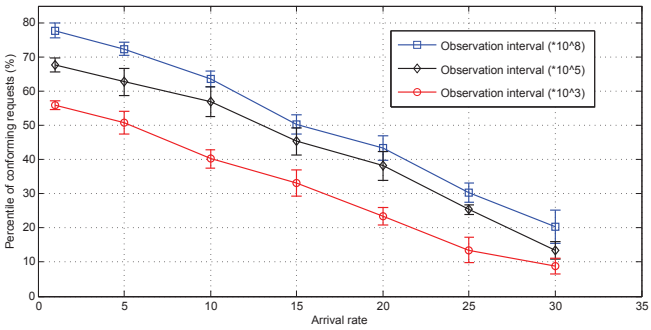

Fig. 6: Relation between step-size and observation interval

V. Related work

As mentioned in previous sections, to the best of our knowledge, the present work is the first effort towards analyzing the effect of the choice of the penalty function on the total number of requests that have conformed to their negotiated response time. However, there have been several past research efforts that have focussed on management schemes to meet percentile Service Level Agreements. Agarwal et al. [9] propose a scheduling scheme called gi-FIFO and analytically prove that it is the best schedule for multi-class jobs in a single server to maximize response time percentiles; Cardellini et al. [10] show that enforcing percentile performance criterion as a management objective, results in better conformance and improved response times in comparison to average performance criterion guarantees and present a brokering service for management of composite services under percentile-based SLAs; Xiong et al. [11] provide an analytical solution of resource optimization subject to percentile response time and price by modelling the system as an overtake free open tandem queuing network with feedback and provide closed form expressions of the probability distribution function of the response time; Gmach et al. [12] consider step-wise percentile SLAs and propose scheduling algorithms for a single database server. Schroeder

et al. [13] propose a framework for providing class-based QoS targets in transactional workloads, where the QoS targets are response time percentiles.

VI. Conclusion and Future Work

In this paper, we have considered response time percentile Service Level Agreements that call for economic penalties if percentile response time targets are not met. We analyzed three variants of the penalty functions: linear, exponential and stepwise. We identified that the choice of the penalty function results in variations in the total number of conforming requests when the "best known" allocation and scheduling request management policy is employed. We proposed a penalty minimizing, simulated annealing based request allocation and scheduling heuristic scheme to evaluate the three variants of the percentile response time SLA. Our simulation based evaluation shows that the linear variant results in the highest number of requests that have conformed to the negotiated response time threshold.

Our future work involves analyzing the three variants of the response time percentile SLA when a clairvoyant scheduling policy is employed. Recently, research efforts have been directed towards pro-active request management schemes that maximize utility [14], essentially developing clairvoyant systems. The analysis of the improvement (or lack of) in conformance when clairvoyant solution is used in each of the three variants of the percentile SLA aims to determine the variant of the penalty function that benefits the most from adoption of clairvoyant solutions.

References

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

[2] W. Cellary and S. Strykowski, "e-government based on cloud computing and service-oriented architecture," in *Proceedings of the 3rd international conference on Theory and practice of electronic governance*, ser. ICEGOV '09. New York, NY, USA: ACM, 2009, pp. 5–10. [Online]. Available: http://doi.acm.org/10.1145/1693042.1693045

[3] Y. C. Lee, C. Wang, A. Y. Zomaya, and B. B. Zhou, "Profit-driven service request scheduling in clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, May 2010, pp. 15 –24.

[4] K. Boloor, R. Chirkova, T. J. Salo, and Y. Viniotis, "Heuristic-based request scheduling subject to a percentile response time sla in a distributed cloud," in *GLOBECOM*. IEEE Computer Society, 2010, pp. 1–6.

[5] D. Ardagna, M. Trubian, and L. Zhang, "Sla based resource allocation policies in autonomic environments," *J. Parallel Distrib. Comput.*, vol. 67, pp. 259–270, March 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1224555.1224640

[6] C. A. Yfoulis and A. Gounaris, "Honoring slas on cloud computing services : a control perspective," *Control*, 2009.

[7] K. R. Baker, *Principles of sequencing and scheduling*. Hoboken, N.J, John Wiley, 2009.

[8] B. Dimitris and T. John, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

[9] N. Agrawal and I. Viniotis, "Performance space of a gi/g/1 queueing system under a percentile goal criterion," in *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 1995. MASCOTS '95., Proceedings of the Third International Workshop on*, Jan. 1995, pp. 53 –57.

[10] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti, "Adaptive management of composite services under percentile-based service level agreements," in *ICSOC*, 2010.

[11] K. Xiong and H. Perros, "Sla-based service composition in enterprise computing," in *16th International Workshop on Quality of Service, IWQoS*. Washington,DC,USA: IEEE Computer Society, 2008, pp. 30 –39.

[12] D. Gmach, S. Krompass, A. Scholz, M. Wimmer, and A. Kemper, "Adaptive quality of service management for enterprise services," *ACM Trans. Web*, vol. 2, no. 1, pp. 8:1–8:46, 2008.

[13] B. Schroeder, M. Harchol-Balter, A. Iyengar, and E. Nahum, "Achieving class-based qos for transactional workloads," in *Proceedings of the 22nd International Conference on Data Engineering*, ser. ICDE '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 153–. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2006.11

[14] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, "A hybrid reinforcement learning approach to autonomic resource allocation," in *In Proc. of ICAC-06*, 2006, pp. 65–73.