RC25050 (W1010-012) October 6, 2010
Computer Science

IBM Research Report

Interacting Stochastic Models Approach for Performability Analysis of IaaS Cloud

Rahul Ghosh[1], Kishor S. Trivedi[1], Vijay K. Naik[2], DongSeong Kim[1]

[1]Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708 USA

[2]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598 USA

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

Interacting Stochastic Models Approach for Performability Analysis of IaaS Cloud

Rahul Ghosh*, Kishor S. Trivedi*, Vijay K. Naik†, and DongSeong Kim*
*Dept. of Electrical and Computer Engineering, Duke University, Durham, NC 27708, USA
Email: {rg51, kst, dk76}@ee.duke.edu
†IBM T. J. Watson Research Center, Hawthorne, NY 10532, USA
Email: vkn@us.ibm.com

***Abstract*—In this paper, we describe an analytical modeling approach for performability analysis of cloud provided services. We use infrastructure-as-a-service as an example of a cloud based service, where service availability and provisioning response delays are key service level agreements (SLAs) to users while cost reductions are important to service providers. A novelty of our approach is in reducing the complexity of the analysis by dividing the overall model into multiple interacting sub-models and then obtaining the overall solution by iteration over sub-model solutions. This results in a high fidelity model that is scalable. We summarize our modeling approach, showcase our ongoing work through initial results and outline future avenues of research using such an approach.**

I. Introduction

This paper presents an analytic modeling approach for performability analysis of a cloud service using interacting stochastic process models. To analyze large scale cloud systems, the overall system model is composed of interacting sub-models and the solution is obtained by iterating over the sub-models. Such an approach provides a model that is tractable, scalable and yet of high fidelity. Using service unavailability (defined as the job rejection probability) and mean provisioning response delay as the two quality-of-service measures, we evaluate the effects of changes in job-arrival, service time, and the effects of system capacity on the cloud service quality.

System Model and Assumptions. We consider infrastructure-as-a-service (IaaS) cloud. Amazon EC2 [1] and IBM SBDT Cloud [2], [3] are examples of such services. In such systems, in response to a user request, pre-built images are used to create Virtual Machine (VM) instances. When the VM instances are deployed, they are provisioned with request specific CPU, RAM, and disk capacity. VMs are deployed on Physical Machines (PMs) each of which may be shared by multiple VMs. To reduce overall VM provisioning delays and operational costs, we assume that PMs are grouped into three server pools; hot (i.e., running), warm (turned on, but not ready) and cold (turned off). A pre-instantiated VM can be provisioned and brought to ready state on hot PMs with minimum provisioning delay. Instantiating a VM from an image and provisioning it on a warm PM needs additional provisioning time. PMs in the cold pool need additional startup time to be turned on before a VM deployment.

In the subsequent discussions we use the term job to mean a user request for provisioning a VM and making it available for use to a cloud user. We assume that all requests are homogeneous where each request is for one VM with fixed size CPU cores and RAM.

User requests (i.e., jobs) are submitted to a global resource provisioning decision engine (RPDE) that processes requests on a first-come, first-served (FCFS) basis as follows. The request at the head of the queue is provisioned on a hot PM if there is capacity to run a VM on one of the hot PMs. If all hot PMs are serving jobs, a PM from warm pool is used for provisioning the requested VM. If all warm PMs are busy, a PM from cold pool is used. If none of these PMs are available, the request is rejected (service unavailable). When a running job exits, the capacity used by that VM is released and becomes available for provisioning the next job. Failures in PMs can lead to reduction in available capacity and repairs lead to refurbishing the available capacity.

Problem Statement. For the above described scenario, we investigate how changes in job arrivals, job service time and available system capacity can affect the provisioning response delays and the service availability to user requests.

II. Our Approach

Shown in Figure 1 is the life-cycle of a job as it moves through system. In the following, we describe

three pure performance models (resource provisioning decision, VM provisioning, run-time) and an availability model capturing the PM failures and repairs.

A. Model descriptions.

(1) Resource provisioning decision model. This is a homogenous continuous time Markov chain (CTMC) where all sojourn times are exponentially distributed [4]. We assume a finite length provisioning decision FCFS queue, where a job inserted in the queue waits for its provisioning decision until decisions are taken for all jobs ahead of it in the queue. In our model, we increase buffer length with system capacity (maximum number of VMs that can simultaneously run across all the machines in cloud). Outputs of this model are job rejection probability due to buffer full (P_{block}), rejection probability due to insufficient capacity (P_{drop}), mean queuing delay ($E[T_{q\_dec}]$) and mean decision delay ($E[T_{decision}]$) conditional upon the job being accepted (assuming no PM failure). These output measures are computed by assigning proper reward rates to each state of the CTMC [4].

(2) VM provisioning model. A PM can accept a job for provisioning only if it has sufficient capacity to execute the job. VM provisioning models capture the actual provisioning and deployment of requested VMs to accepted jobs. For each hot, warm and cold server, we design one CTMC which keeps track of the number of VMs running. VM provisioning model of a pool is the union of individual provisioning models of each PM in that pool. These models provide the probability that a PM in a pool can accept a job for resource provisioning, mean queuing delay within each PM ($E[T_{q\_vm}]$), and the weighted mean VM provisioning delay ($E[T_{prov}]$).

(3) Run-time model. Once a job is successfully provisioned, it utilizes the resources until its execution is complete. We design a Discrete Time Markov Chain (DTMC) to capture the details of job execution and to compute mean resource holding time [4].

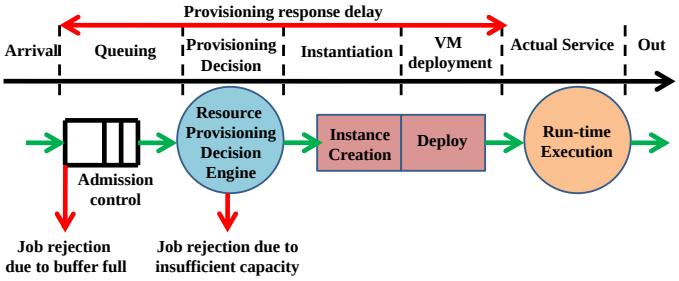

Fig. 1. Request provisioning and servicing steps in IaaS cloud.

(4) Availability model. Availability model determines the steady state probability of a certain number of PMs being available. We assume that mean time to failure (MTTF) of warm PMs is at least $2-4$ times longer than that of hot PMs, and cold PMs do not fail. Failure of a PM in one pool also triggers migration of a new PM from other pools.

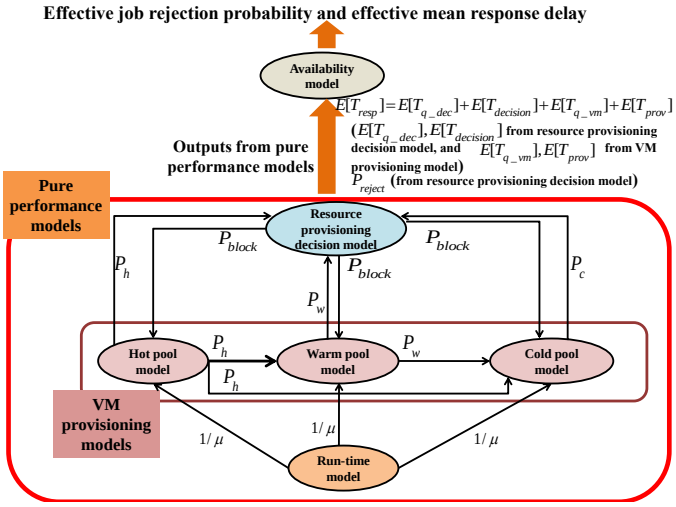

Fig. 2. Interactions among the sub-models.

B. Model Interactions, Performability

Interactions among sub-models are shown as an import graph [5] in Figure 2. For a particular type of a job, run-time model provides mean resource holding time ($1/\mu$). This output is utilized as an input parameter to each type (hot, warm or cold) of the VM provisioning model. VM provisioning models compute the probabilities (P_h, P_w and P_c) that at least one PM in a particular pool (hot, warm and cold, respectively) can accept a job for provisioning. These probabilities are used as input parameters to the resource provisioning decision model. Resource provisioning decision model computes blocking probability (P_{block}) due to buffer full, rejection probability due to insufficient capacity (P_{drop}) and net job rejection probability (P_{reject} as sum of P_{block} and P_{drop}). The net mean response delay ($E[T_{resp}]$) is also computed from pure performance models. Two components ($E[T_{q\_dec}], E[T_{decision}]$) of net mean response delay are obtained from the resource provisioning decision model and two other components ($E[T_{q\_vm}], E[T_{prov}]$) are computed from VM provisioning models. In the topmost availability model, ($E[T_{resp}]$, P_{reject}) are used as reward rates to compute effective mean response delay and effective job rejection probability. We note that there

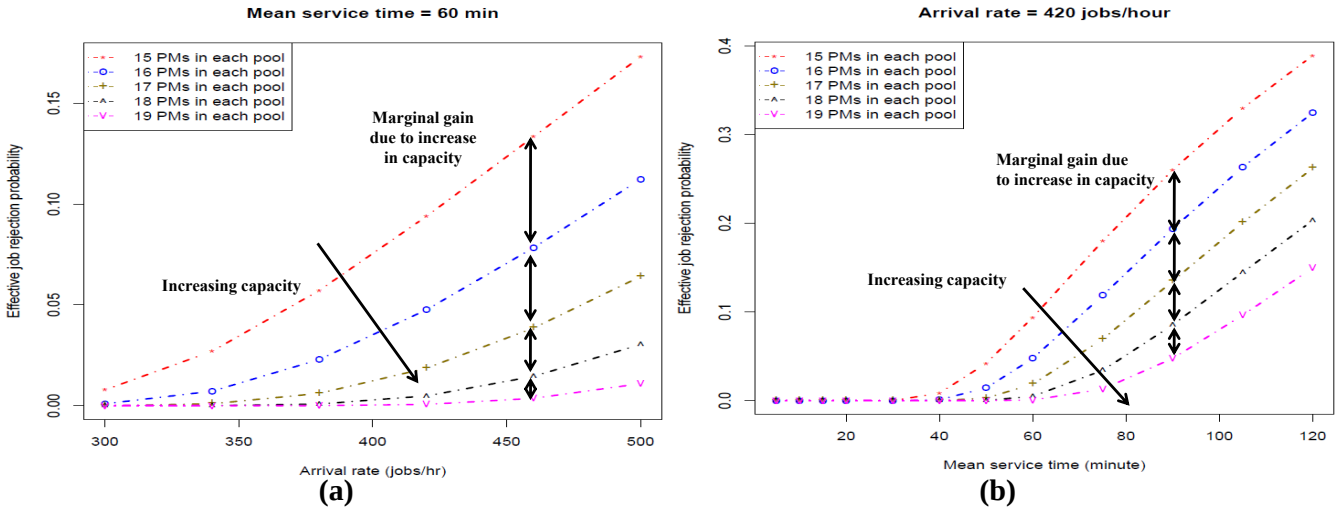

Fig. 3. Effect of (a) arrival rate and (b) job service time on effective job rejection probability.

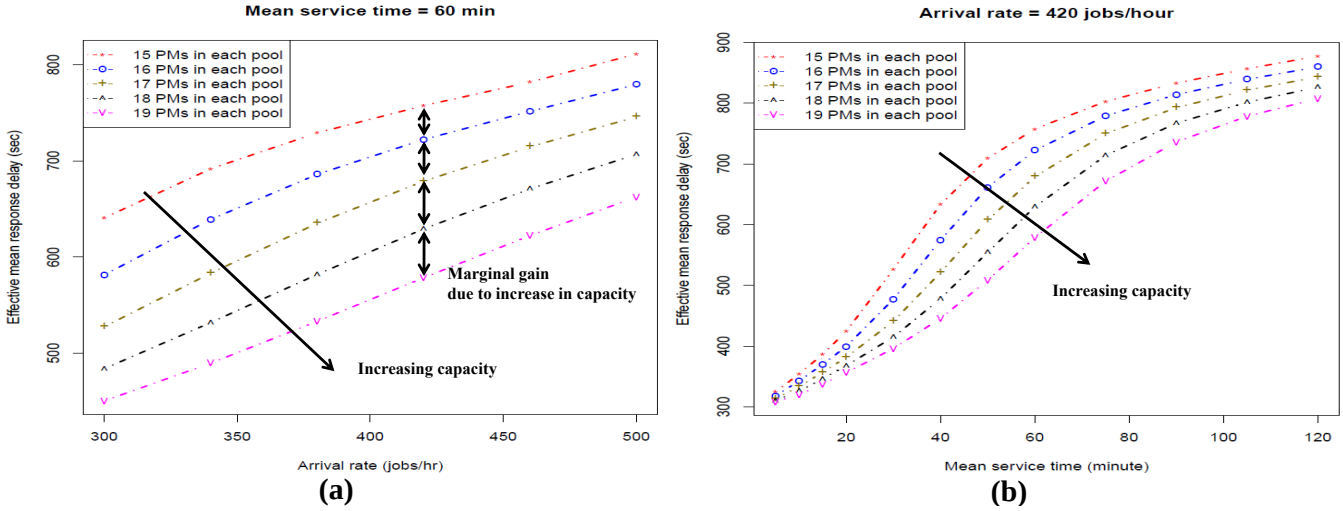

Fig. 4. Effect of (a) arrival rate and (b) job service time on effective mean response delay.

are cyclic dependencies among the sub-models. P_{block} computed in the provisioning decision model is used as an input parameter in VM provisioning models. However, to solve the resource provisioning decision model, outputs from VM provisioning models (P_h, P_w, P_c) are needed as input parameters. This cyclic dependency is resolved via fixed-point iteration.

III. Numerical Results

We evaluated cloud services for two metrics; (1) effective job rejection probability and (2) effective mean response delay (conditional upon being accepted). Here we highlight the salient results to show the effects of changes in job arrival rates, mean service time of jobs and system capacity (number of PMs in each pool). In all cases, we assume exponential distribution for inter-arrival times, service times and PM failures/repairs; though these restrictions can be and will be relaxed in the future. Models were solved using the SHARPE [6] software package. Figure 3(a) shows, at a fixed mean service time, increasing arrival rate increases effective job rejection probability. This effect is more pronounced if the system capacity is low. Increasing system capacity reduces effective job rejection probability. Although, marginal gain (reduction in job rejection probability, as shown by vertical arrows) diminishes with increasing capacity. Similar effect can be observed in Figure 3(b) where mean service time is increased at a given arrival rate. Figure 4(a) shows that with increasing arrival rate, effective mean response delay increases for a fixed

number of machines in each pool. Keeping the arrival rate same, increasing PMs in each pool reduces mean provisioning delay. Similar effect of increasing mean service time and system capacity on mean provisioning delay is shown in Figure 4(b).

IV. Future Research

(1) Capacity planning. Results show that system capacity is tightly coupled with a variety of system/job parameters. We want to determine the optimal capacity of a cloud system that can uphold the given SLA requirements. **(2) Energy profiling.** Based on the granularity of energy consumptions (joules per core or joules per server), our models can be exploited to compute energy consumptions through reward rate assignments. **(3) Heterogenous requests.** We want to evaluate the impact of request heterogeneity [7] (different job types having different arrival rates and service time parameters) on cloud services. **(4) Model verification and validation.** We have developed a stochastic Petri-net based model as a monolithic model to cross-validate the interacting Markov chains model approach. Models will also be validated against simulative solution and real measurements.

V. Conclusions

We evaluated QoS measures of cloud services through interacting analytical models. Using this approach, model complexity can be reduced significantly without removing realistic features. We believe that the general approach and results presented in this paper will be very useful to solve specific problems for public, private and hybrid cloud systems.

ACKNOWLEDGMENTS

Research by Duke authors was supported in part under a 2009 IBM Faculty Award and a NSF grant NSF-CNS-08-31325. Authors would like to thank Dan Dias and Murthy Devarakonda from IBM Research for their support. This work was completed during Rahul's internship at IBM T. J. Watson Research Center.

References

[1] "Amazon EC2," http://aws.amazon.com/ec2.
[2] "IBM SBDTC," http://www-180.ibm.com/cloud/enterprise/beta/dashboard.
[3] "IBM Cloud," http://www.ibm.com/ibm/cloud/.
[4] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. Wiley, 2001.
[5] G. Ciardo and K. S. Trivedi, "A decomposition approach for stochastic reward net models," *Performance Evaluation*, vol. 18, no. 1, pp. 37–59, July 1993.
[6] K. S. Trivedi and R. Sahner, "SHARPE at the age of twenty two," *ACM Perf. Eval. Rev.*, March 2009.
[7] P. Garbacki and V. Naik, "Efficient resource virtualization and sharing strategies for heterogeneous grid environments," in *Proceedings IM*, 2007.