Future Generation Computer Systems 29 (2013) 1216–1234

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Modeling and performance analysis of large scale IaaS Clouds☆

Rahul Ghosh [a,*], Francesco Longo [b], Vijay K. Naik [c], Kishor S. Trivedi [a]

[a] Duke University, Durham, NC 27708, USA

[b] Università degli Studi di Messina, 98166 Messina, Italy

[c] IBM T. J. Watson Research Center, Hawthorne, NY 10532, USA

ARTICLE INFO

Article history:
Received 15 January 2012
Received in revised form
10 June 2012
Accepted 11 June 2012
Available online 27 June 2012

Keywords:
Analytic model
Cloud
CTMC
Fixed-point iteration
IaaS
Interacting sub-models
Monolithic model
Performance
Provisioning

ABSTRACT

For Cloud based services to support enterprise class production workloads, Mainframe like predictable performance is essential. However, the scale, complexity, and inherent resource sharing across workloads make the Cloud management for predictable performance difficult. As a first step towards designing Cloud based systems that achieve such performance and realize the service level objectives, we develop a scalable stochastic analytic model for performance quantification of Infrastructure-as-a-Service (IaaS) Cloud. Specifically, we model a class of IaaS Clouds that offer tiered services by configuring physical machines into three pools with different provisioning delay and power consumption characteristics. Performance behaviors in such IaaS Clouds are affected by a large set of parameters, e.g., workload, system characteristics and management policies. Thus, traditional analytic models for such systems tend to be intractable. To overcome this difficulty, we propose a multi-level interacting stochastic sub-models approach where the overall model solution is obtained iteratively over individual sub-model solutions. By comparing with a single-level monolithic model, we show that our approach is scalable, tractable, and yet retains high fidelity. Since the dependencies among the sub-models are resolved via fixed-point iteration, we prove the existence of a solution. Results from our analysis show the impact of workload and system characteristics on two performance measures: mean response delay and job rejection probability.

1. Introduction

An Infrastructure-as-a-Service (IaaS) Cloud, such as Amazon EC2 [1], IBM SmartCloud Enterprise, IBM SmartCloud Enterprise+ [2,3], and IBM Smart Business Desktop Cloud [4], delivers on-demand operating system (OS) instances provisioning computational resources in the form of virtual machines deployed in the Cloud provider's data center. Such Cloud based services are gaining popularity leading to increasing business competitions and hence, performance and dependability guarantees are becoming critical. Providers of IaaS Clouds (e.g., IBM and Amazon) offer service level agreements (SLA) to Cloud users. Violations of such SLAs can cause loss of revenue and business reputation. We observe that, for most of the IaaS Cloud providers, offered SLAs are in terms of guaranteed availability. As technology and business models for Cloud services are getting mature, in future, users will also expect SLAs on Cloud performance besides availability. However, performance of an IaaS Cloud depends on a large number of factors including: (i) nature of physical infrastructure (e.g., CPU, RAM, disk characteristics), (ii) nature of virtual infrastructure (e.g., hypervisor characteristics), (iii) nature of management and automation tools (e.g., request provisioning steps), (iv) workload (e.g., arrival rate, request types) and (v) available capacity (e.g., number of physical machines). Hence, systematic performance assessment of Cloud infrastructure is difficult and non-trivial.

This paper presents a scalable analytic approach for model driven performance analysis of an IaaS Cloud. Traditional measurement based performance evaluation requires extensive experimentation with each workload and system configuration and may not be feasible in terms of cost due to the sheer scale of Cloud. A simulation model can be developed and solved but in contrast with an analytic model, it might be time consuming as the generation of statistically significant results may require many simulation runs [5]. A stochastic model is a more attractive alternative because of lower relative cost of solving the model while covering large parameter space. However, such stochastic analytic models are presumed not to scale well when dealing with the rising complexity associated with Cloud service architectures. Simplifying the model to make it more tractable can result in lower fidelity and, in the process, the effects of important parameters affecting the service performance metrics may not be captured [6]. To overcome this

☆ This research was supported in part by IBM Research and the US National Science Foundation under grant NSF-CNS-08-31325.

* Corresponding author.
E-mail addresses: rg51@ee.duke.edu, rahul.ghosh@duke.edu (R. Ghosh), flongo@unime.it (F. Longo), vkn@us.ibm.com (V.K. Naik), kst@ee.duke.edu (K.S. Trivedi).

0167-739X/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2012.06.005

difficulty, in our recent research [7], we described a scalable approach using interacting stochastic sub-models where an overall solution is composed by iteration over individual sub-model solutions. We quantified effects of changes in workload (e.g., job arrival rate, job service rate) and system capacity (e.g., number of physical machines, number of virtual machines per physical machine) on service quality as measured by mean response delay and job rejection probability. In this paper, we extend our previous research in several directions:

(1) A monolithic Cloud performance model is constructed using our variant of stochastic Petri net (SPN) called stochastic reward net (SRN) [8]. We compare the scalability and accuracy of the interacting stochastic sub-models approach proposed in [7], w.r.t. the monolithic model. Our analysis shows that the monolithic model becomes intractable and fails to produce results as the scale of Cloud increases, while interacting sub-models approach quickly provides model solutions without significantly compromising the accuracy. Thus, we provide an analytic verification and validation of the proposed approach.

(2) Closed-form solutions of sub-models are shown whenever feasible. Such closed-form expressions can complement the use of analytic modeling software packages such as SHARPE [9] and SPNP [10], when dealing with large number of model states.

(3) Since dependencies among the sub-models are resolved via fixed-point iteration, in this paper, we prove the existence of a solution for the associated fixed-point equation.

(4) Numerical results are expanded and discussions are presented on how the proposed model can be extended to include different Cloud management aspects. Our developed model can be applied in capacity planning, forecasting, sensitivity analysis to find bottlenecks, *what-if* analysis or in an overall design optimization problem during design, development, testing and operational phases of an IaaS Cloud.

The rest of the paper is organized as follows. In Section 2, we present a system description, assumptions, and formulate the problems of interest. Section 3 describes interacting sub-models approach for performance analysis of IaaS Cloud. An equivalent monolithic model is described in Section 4. Numerical results are presented in Section 5. Generalizations of the interacting sub-models approach and future avenues of research are outlined in Section 6. Related research is highlighted in Section 7. Finally, we conclude this work in Section 8. Appendix A summarizes the symbols used in the paper and detailed steps of interacting sub-model closed form solutions are shown in Appendix B.

2. System description, assumptions, and problem formulation

In an IaaS Cloud, when a request is processed, a pre-built image is used to deploy one or more Virtual Machine (VM) instances or a pre-deployed VM may be customized and made available to the requester. VMs are deployed on Physical Machines (PMs) each of which may be shared by multiple VMs. The deployed VMs are provisioned with request specific CPU, RAM, and disk capacity. This process of provisioning and deploying VMs involves delays which may be reduced by various optimization techniques. One such approach is to group the PMs into multiple pools characterized by different degrees of provisioning delays. Maintaining the PMs in multiple pools also helps to minimize power and cooling costs without incurring high startup delays for all VMs. In this paper, we show the analysis for a Cloud, where the PMs are grouped into three pools: hot (running), warm (turned on, but not ready) and cold (turned off). A pre-instantiated VM can be readily provisioned and brought to ready state on a running PM (hot PM) with minimum provisioning delay. Instantiating a VM from an image and deploying it on a warm PM needs additional provisioning time. PMs in the cold pool are turned-off when not in use and deploying a VM on such a PM suffers from additional startup delays. In this paper, we assume that the size of the hot, warm, and cold pools are predetermined. We do not focus on the optimal size of the three pools. Those issues are left as future research.

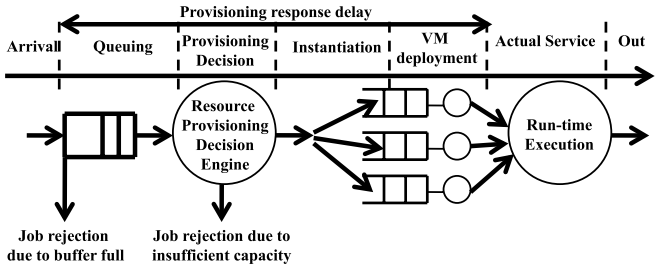

Fig. 1. Request provisioning and servicing steps.

We assume that all PMs in a pool are identical and all requests are homogeneous, where each request is for one VM with fixed size CPU cores, RAM and disk capacity. Under this assumption, the maximum number of VMs (denoted by m) that can be deployed on a PM, is given by:

$$m = \min\{\lfloor (C_t/c_v) \rfloor, \lfloor (R_t/r_v) \rfloor, \lfloor (D_t/d_v) \rfloor\} \quad (1)$$

where, C_t is the total number of cores per PM, R_t is the total amount of RAM per PM, D_t is the total disk capacity per PM, c_v is the number of cores required per VM, r_v is the amount of RAM required per VM and d_v is the disk capacity required per VM. Having a homogeneous environment can bring certain benefits to a Cloud service provider [11], e.g., information assurance, security response activities, fault management, load balancing, and system maintenance. Through standardization, on-demand and rapid delivery of commodity computing resources is also facilitated in a homogeneous Cloud [12]. Examples of such homogeneous environments are Cloud federations [13], where, each collaborating Cloud provider leverages homogeneous Cloud services from other providers. Examples of Clouds with homogeneous requests include MapReduce workloads, which often have repeated queries on similar or identical datasets [14]. For heterogeneous PMs and/or requests, the resource provisioning analysis is more complex (see [15], for example). We plan to address that in the future.

Shown in Fig. 1 is the life-cycle of a request as it moves through the system. Submitted user requests (i.e., jobs) enter a first-come, first-served (FCFS) job queue. The request at the head of the queue is processed by a Resource Provisioning Decision Engine (RPDE) as follows. The request is provisioned on a hot PM if a pre-instantiated but unassigned VM exists. If no hot PM is available, a PM from the warm pool is used for provisioning the requested VM. If all warm PMs are busy, a PM from the 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. For the above described scenario, we investigate the effects of varying job arrival rates, job service rates and system capacity on two QoS metrics: (i) mean response delay and (ii) job rejection probability.

Problem formulation. We are interested in analyzing the end-to-end performance of the IaaS Cloud as described above. End-to-end Cloud service delivery is composed of three main steps: (i) resource provisioning decision, (ii) VM provisioning and deployment, and (iii) run-time execution. We first translate these individual steps into analytic sub-models which are tractable and yet of high fidelity. We then connect these sub-models into an overall model to compute the Cloud QoS metrics. To compare the scalability and accuracy of the proposed interacting sub-models approach, we construct an SRN based monolithic Cloud performance model. Finally, we solve these models and show how QoS metrics are affected by variations in workload (job arrival rate, job mean service time) and system capacity. We explain our approach systematically in the following sections.