Contents lists available at SciVerse ScienceDirect

Int. J. Production Economics

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

Resource management in software as a service using the knapsack problem model

Fotis Aisopos*, Konstantinos Tserpes, Theodora Varvarigou

Department of Electrical and Computer Engineering, National Technical University of Athens, 9, Heroon Polytechniou Str, 15773 Athens, Greece

ARTICLE INFO

Article history:
Received 14 January 2011
Accepted 6 December 2011
Available online 16 December 2011

Keywords:
Software as a Service
Resource allocation
Service Level Agreement
Quality of Service
Fractional Knapsack problem
Service oriented architecture

ABSTRACT

This paper proposes a resource allocation model for "Software as a Service" systems that maximizes the service provider's revenues and the resource utilization under a heavy load. Employing the elasticity of virtualized infrastructures, the proposed model dictates that system resources must be fully exploited by incoming jobs, even if they do not satisfy their requirements completely. This yields a higher Service Level Agreement violation probability, which is mitigated by the assignment of more resources when these become available. The problem is deduced to the Fractional Knapsack problem and the heuristic solution is implemented in the frame of a SOA environment.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

A distributed environment is a system comprised of a set of interconnected resources, with a virtualization layer applied on top of them for interoperability and scalability, delivering a single point of interaction to the customer. A feverish discussion is taking place in the research community about the various distributed computing environments utilities, especially after the emergence of Cloud Computing as a business solution on the web, when Grid computing had only started addressing the same space adequately. Cloud Computing enables the distinction between the application development framework and the underlying resources. It is therefore possible to provide to the end user a platform for application development and/or deployment, an infrastructure or even the application itself, all through web services and usable front ends. The fact that computing or storage resources are virtualized as web resources broadens the customer target group making Cloud services available to various stakeholders. Grid computing on the other hand seems to be addressing the needs of larger organizations. The Grid services' consumer agrees to use Grids without having much control over the application, which is solely provided and known to a service provider and/or the application developer.

Although these prominent distributed system paradigms present differences in the way they propagate the control of their resources to the service customers and the level of usability that each one exposes towards the customers, the principle remains the same: they are both service provisioning systems which, when it comes to resource management, they both attempt to abstract/virtualize the underlying resources providing a seamless way to access and manage them. Thus, solutions such as Software as a Service (SaaS) are basically common approaches in distributed computing.

Generally, existing distributed SaaS environments support the provision of dynamically scalable and virtualized resources as a service over the Internet. In such systems, the infrastructure resources can be provided exteriorly as a "pool" of resources, like a seamless, single infrastructure with aggregated capabilities. Service customers run various jobs by invoking the respective services with a set of specific requirements for a certain tariff. The resources assigned to each job are elastic, i.e. the provider can dynamically assign the amount of memory, CPU and disk space to a specific job and therefore their performance and capabilities can vary based on the set up. Taking advantage of this property (elasticity), providers are able to maintain high resource utilization while trying to deliver services at the requested quality. An important issue in this supply chain is the establishment of appropriate performance measures, ensuring the supply chain effectiveness and efficiency. These can be categorized as either qualitative (e.g. customer satisfaction with the service received) or quantitative (e.g. objectives depended directly on cost or profit defined in the service tariff) (Beamon, 1998).

Service quality and performance is evaluated in a form of electronic contract, called Service Level Agreement (SLA). SLA

* Correspondence to: Distributed Knowledge and Media Systems Laboratory, Department of Electrical and Computer Engineering, 3rd Floor, Room B.3.19, National Technical University of Athens Campus, 9 Heroon Polytechniou Str, 15773 Athens, Attica, Greece. Tel.: +30 210 7722568; fax: +30 210 7722132.
E-mail addresses: fotais@mail.ntua.gr (F. Aisopos), tserpes@mail.ntua.gr (K. Tserpes), dora@mail.ntua.gr (T. Varvarigou).

0925-5273/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2011.12.011

is a contracting tool keyed to a client's service performance expectations, for identifying the responsibilities of both the customer and provider (Manthou et al., 2004), by defining the Quality of Service (QoS) level promised to be delivered, using quantifying metrics. These metrics specify – explicitly or implicitly – the resources that are to be allocated by the provider to the consumer. Among others, SLAs define the period that this agreement will be active, the cost for the resources as well as clauses that safeguard both the sides from potential SLA term violations (e.g. execution deadline). The latter is usually expressed into monetary compensations, thus, their avoidance is a priority for providers that wish to preserve a good reputation.

With a few exceptions such as Gallizo et al. (2009), SLAs are long-term and resource-oriented (Vs job-oriented) which means that the provider guarantees the allocation of a certain amount of resources to the consumer who in turn can use them at anytime and any extend during the contract's lifetime. This implies, however, that it is not necessary for all resources to be reserved on behalf of the consumer during the whole SLA lifetime. This observation is critical because providers tend to employ inflexible allocation schemes such as resource reservation in fear of conducing SLA violations. However, at any given instance, if the jobs submitted by customers demand in total more resources than those available based on their SLAs, the provider comes across a problem that needs to be solved. If the resource pool does not suffice to cover the demand, providers resort to outsourcing jobs or grant extra resources, given that they are keeping valuable resources as a backup. Although this is not a significant problem for large organizations like Amazon (Amazon Web Services LLC, 2010) with practically unlimited resources available, for Small Medium Enterprises (SMEs) this means leasing extra resources or simply denying jobs (violating the SLA). All these alternatives cost money and indicate a lack of flexibility when it comes to assess the risk of undertaking a job using uncommitted resources, even if they do not suffice to cover demanded quality, at the particular time.

The implementation of a mechanism to effectively allocate resources on run-time comes under a set of NP-hard optimization problems with numerous parameters being required to be counted in. This paper presents a solution for maintaining the maximum resource utilization at the cost of risking potential SLA violations, on the pending jobs that will yield the smaller profit for the provider. It does however make two basic assumptions that may affect the feasibility of the proposal: (a) all submitted jobs are preemptive, i.e. they can be interrupted and resumed at a later stage; (b) the pool of resources is homogeneous (or treated as such).

The proposed solution considers the resource infrastructure as a pool of elastic resources using any technology for resource virtualization. Therefore the resources are divided in small chunks and allocated for incoming jobs. The amount of given resources can vary based on the workload generated by the job requirements. The term "workload" refers to the amount of resources that need to be allocated in order to fully satisfy the QoS requirements of a customer when he submits a job. A more specific term could be the "demanded workload", which is calculated by the provider when mapping high level requirements to low level parameters. It is thus depended on the customer requirements but also on time, because the demanded workload is reduced when part of the job has been executed and can be increased when a job remains in the queue for execution for a long time.The problem now is to maximize the resource utilization while at the same time maximizing the profit. For solving this we resort to a well-known problem with which our own presents analogies: a variant of the knapsack problem: "Given a set of items (jobs to be submitted), each with a weight (workload) and a value (profit), determine the number of each item to include in a collection (set of jobs to be executed/resources to be committed) so that the total weight (total workload) is less than or equal to a given limit (infrastructure limitations) and the total value (total profit) is as large as possible".

Given the above analogy, we investigate the possible solutions and formulate our allocation as a knapsack problem. In order to conclude to the most well-fitted solution, we also model the profit from job execution. Profit in the examined case, is a multi-parametric function, depended on the probability of an SLA violation given the amount of committed resources, the compensation cost and the infrastructure operating cost yielding from a job execution.

To evaluate the proposed solution, we implement a resource allocation scheme at the level of a distributed system's meta-scheduler. The implementation approach takes into consideration the cost of a job with specific requirements, the cost of an SLA violation and the probability of an SLA violation, given the job's workload in order to calculate the profit at any time. In the evaluation phase, various pricing strategies are employed so as to test the model against the potential profit it yields.

The document is organized as follows: Section 2 presents all the related work in the specific area. Section 3 describes our resource allocation model deducting it to the Fractional Knapsack problem (the variant of knapsack) and presenting a heuristic greedy solution for the current multi-dimensional problem. Section 4 describes the architecture of the system that the evaluation of the provided solution took place, while Section 5 presents the experiment performed and analyses the results. Finally, Section 6 presents the conclusions of the current work.

2. Related work

So far, there has been a great deal of research on resource allocation and notable progress in maximizing resource utilization in systems as the ones described above.

An implementation of a distributed, market-based resource allocation system was provided by Lai et al. (2004), where distributed clusters like the Grid and PlanetLab were studied. The purpose of the platform developed (Tycoon) was to allocate computing resources like CPU cycles, memory, network bandwidth, etc. to users in an economically efficient way. Tycoon system introduced three new allocation components, while using Linux VServer for virtualization, focusing on the maximization of the revenues of the specific system. Another business-focused resource managing model for service oriented architectures (SOA) was introduced by Pueschel et al. (2009). This work involves Cloud Computing, proposing decision support mechanisms for the Cloud provider, in order to get a revenue optimization in the respective business model. Resource management for Clouds was also studied by Nguyen Van et al. (2009), who presented an autonomic virtual resource management mechanism for service hosting platforms. Zhao and Sakellariou (2007) studied advance resource reservation in Grids, while Chase et al. (2003) presented new mechanisms for dynamic resource management in a cluster manager, allocating servers from a common pool to multiple virtual clusters.

The papers presented above tackle the resource allocation problem focusing on the available resources and the demands of specific applications. Our approach involves both the low-level parameters of the reservation process, as well as the high-level Quality of Service provisioning details, which are a result of the agreement between the service provider and the customer. As mentioned in Section 1, these details are usually encompassed in the Service Level Agreement in the form of contractual terms and define the level of the provided quality. Job Scheduling based on