

# Towards Modelling-Based Self-adaptive Resource Allocation in Multi-tiers Cloud Systems

Mehdi Sliem<sup>1(✉)</sup>, Nabila Salmi<sup>1,2</sup>, and Malika Ioualalen<sup>1</sup>

<sup>1</sup> MOVEP Laboratory, USTHB, Algiers, Algeria  
{msliem, nsalmi, mioualalen}@usthb.dz

<sup>2</sup> LISTIC, Université de Savoie, Annecy le Vieux, France

**Abstract.** Achieving efficient resource allocation is one of the most challenging problems faced by cloud providers. These providers usually maintain hosted web applications within multiple tiers over the cloud, leading to an overall increased complexity. To answer user requests, meet their Service Level Agreements (SLA) and reduce the energy cost of the data center, cloud systems are being enforced with self-adaptive features such as self-scaling, to efficiently automate the resource allocation process. However, the main concern is how to choose the best resource configuration to reach these objectives of Quality of Service (QoS) with a minimal amount of resources consumption. In this context, we target to use performance modelling and analysis, to forecast the system performances and deduce the most appropriate resource configuration to be applied by the autonomic manager. As a first work to define a modelling based resource allocation autonomic manager, we present, in this paper, the modelling and analysis process, allowing to predict the efficiency of the self-adaptive systems relating resource allocation in the context of multi-tiers cloud systems. We used Stochastic Petri Nets modelling, enforced with a reduction method to avoid a scalability issue. A set of experiments illustrates our approach starting from modelling to performance evaluation of the studied system.

**Keywords:** Cloud computing · Autonomic computing · Performance modelling · Resource allocation · Petri net

## 1 Introduction

Today's data centers are subject to an increasing demand for computing resources, as they host complex, large-scale and heterogeneous distributed systems. This is also true in Cloud systems. Particularly for multi-tiers applications, resource management is critical, as sometimes, performance bottlenecks may appear when insufficient resources are not available in one tier, leading to a decrease of the overall profit. This growing complexity stresses the challenging issue of appropriate resource management, especially when a cloud provider has to maintain Service Level Agreements (SLA) with its clients. The main concern to achieve efficient resource allocation is then to find the minimal amount

of resources that an application needs to meet the desired Quality of Service without degrading system performances.

To face this challenge, decisions made by a provider with regard to the deployment of a tier application in the cloud and resource allocation, can be strengthened with a scale-up/down operation. Scaling up/down consists of adding/removing resources to individual virtual machines. However, when the application is implemented following a multi-tier architecture, this will directly impact both the application performances and the providers operation cost. To overcome these issues, the concept of autonomic computing [1] has been proposed to dynamically build the adequate configuration on the monitored system. Autonomic computing commonly describes computing systems that are able to manage themselves. In the context of resources allocation, self-management usually provides two main properties: self-configuration to reconfigure the system according to high-level goals and self-optimization to dynamically optimizes the resource use.

However, a beforehand study of the self-adaptive resource allocation is required, to correctly adjust the scaling parameters of the autonomic manager. This helps to implements the system with more optimized performances. For this purpose, it is interesting if decisions of the autonomic manager are made on the basis of an “a priori” knowledge about predictive behaviour and performances of the chosen resource sharing configuration. IN this context, formal methods are strong tools for system performance prediction based on modelling. Mathematical models, such as Petri nets, are well suitable for modelling distributed complex systems.

We aim, in our work, to contribute in developing modelling and analysis approaches and tools that allows strengthening autonomic systems [8]. To reach our goal, we introduced in [8] a case study for modelling resources allocation in self-adaptive systems. In this paper, we extend the approach for a more detailed modelling of the workload behaviour and computer additional performances indices. We then, extended and dedicated our approach to autonomic multi-tiers cloud systems, addressing resource self-scaling. Our approach is based on a Stochastic Petri Nets (SPN) modelling, with the use of the GreatSPN tool [3] for performance analysis of obtained models. The scalability of our approach is, then, considered using a proposed Petri net reduction method.

The paper is organized as follows: Sect. 2 discusses related work. Then, Sect. 3 presents our case study. Our reduction method id then explained in Sects. 4 and 5 discusses some experimental results. Finally, Sect. 6 concludes the paper and gives some future work.

## 2 Related Work

Modelling autonomic systems has gained many attention during the last decade, investigating mainly solutions for the distributed resource allocation problem, to satisfy client Service Level Agreements (SLAs). Among these proposals, some authors used mathematical and performance modelling/optimisation approaches

in the general context of autonomic computing, and optimized resource allocation algorithms for the special case of cloud systems.

Litoiu [7] used queuing networks instead of a single queue model. He investigates performance analysis techniques to be used by the autonomic manager. The workload complexity was studied, and algorithms were proposed for computing performance metric bounds for distributed transactional systems. Workloads are characterized by their intensity representing the total number of users in the system, and their mixes which depict the users number in each service class.

In the field of cloud computing, an SLA-based resource allocation for multi-tier applications was considered in [5]. An upper bound on the total profit was provided and an algorithm based on force-directed search was proposed to solve the problem. Processing, memory requirements and communication resources were considered as three dimensions in which optimization is performed.

In [4] identifies open issues in autonomic resource provisioning and presents innovative management techniques for supporting Software as a Service (SaaS) applications hosted in Clouds. The authors present a conceptual architecture and early results highlighting the benefits of Clouds autonomic management.

Most of the cited proposals are based on the use of formal modelling for an autonomic online optimization [2,6,9,10], basing generally on queuing models, or of optimization algorithms. Few work, however focused on the efficiency of autonomic components to achieve adequate resources management. In this direction, we use, for our modelling, SPN rather than queueing models, to be able to express most properties of nowadays complex systems. In addition, we model the complete autonomic system behaviour including the resource management and the self-adaptive component. This will allow to analyze the efficiency of the autonomic manager and identify the best configuration to apply.

### 3 A Self-Adaptive Resource Allocation Cloud Platform

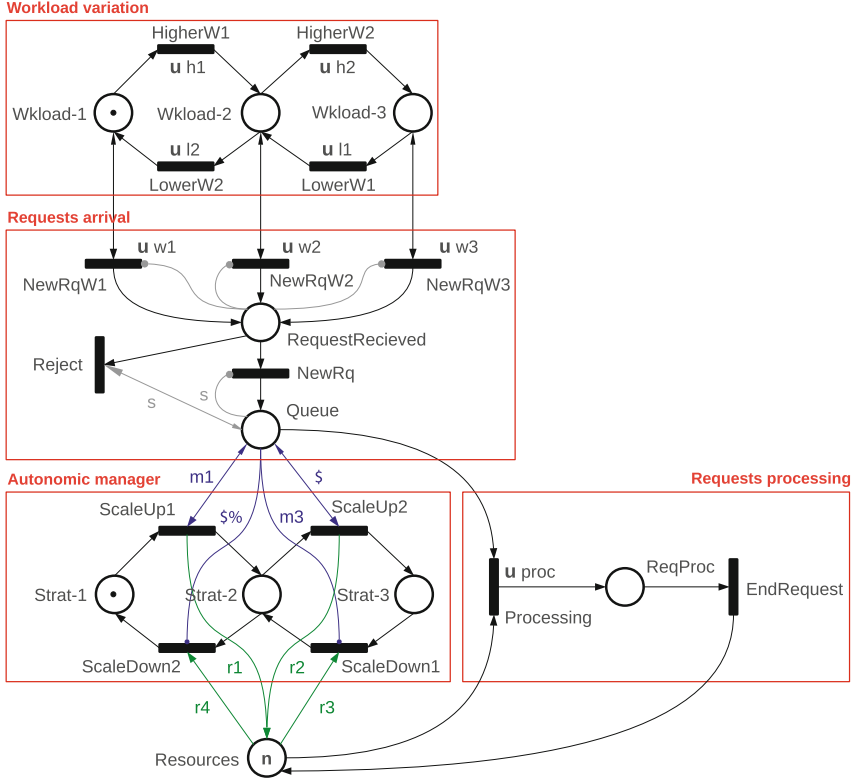
To illustrate our modelling and analysis approach, we present, in this section, a typical example of a resource allocation cloud system strengthened with a self-scaling property.

This system consists of an autonomic cloud server receiving user requests and distributing a set of resources to these requests through a resource allocation process. Requests in our example, are served following a FIFO policy, so conflicting requests are served according to their arrival order. As a first work, we consider here only one service class for user requests. Studying several service classes requires more investigated models, namely high level models. However, the same modelling methodology can be used to predict performances, considering, in addition, specific colour classes.

The main managed element in this cloud autonomic system is the resource. A resource may be any element used or invoked during the processing of a user request: it may be a server, a processor cycle, a memory space, a virtualized memory, a used device, a network bandwidth, and so on.

The received requests are put by the system in a queue for their processing. The requests processing is carried out by first taking a request from the requests

queue, and allocating it the required amount of resources from the resources pool. Each user request consumes, hence, resources for its operation. If the needed number of resources isn't available, the current requests are kept pending in the requests queue, waiting for the release of the required amount of resources.



**Fig. 1.** Final model

Once the processing finished, the resource is available again, to be allocated for other purposes. When the workload is high, a saturation state may be reached quickly, making the system unable to process new request arrivals. We assume, then, that the system uses an autonomic scaling up/down strategy, to increase the system scalability, implemented in an autonomic manager.

Meanwhile, the self-adaptive manager adjusts the amount of available resources for the processing of the queued requests. For this, the main system and its environment are continuously monitored to analyze the workload variation and the number of pending requests. When the number of available resources is inappropriate in relation to the current demand (too low/ too high), the self-adaptive subsystem switches the current configuration to a more suitable one. This process is carried out by adding/removing resources from the resources pool. In addition, we want to plug a modelling analysis module into

this self-adaptive manager, to choose the best suitable resource configuration. This will allow to obtain a more trusted and optimized system with regard to performances and resources consumption cost.

We build a model for each component of the platform, using stochastic Petri nets. The obtained sub-models are combined to build a global model of the whole system behaviour. Finally, on this global model are directed a set of experiments for predicting performances of the configuration under study.

## 4 Analysis of a Multi-tiers Autonomic System Model

When dealing with several tiers, the obtained model becomes rapidly important which make it difficult and even impossible to analysis. To overcome this issue, we propose an analysis process based on a Petri net reduction principle:

Let  $S$  be the autonomic system under analysis, being composed of  $n$  tiers  $T_i$ ,  $i = 1..N$ . To analyze this system, we propose to perform this task gradually in several steps. Each step studies a tier model, connected to “abstract views” of other models, rather than connecting the complete tiers models. Let us, first, define what is an “abstract view” of a tier.

### 4.1 Abstract View of a Tier

The *abstract view* of a tier is a macro representation or *macro-view* that aggregates its behavior to a minimal equivalent Petri net keeping interactions with other tiers.

**Definition 1 (Abstract view of a tier).** *Let  $M_i$  be the SPN model of a tier  $T_i$ . The abstract view  $A(M_i)$  of the tier  $T_i$  is composed of a timed transition *TierProcessing*, modelling the activities of the tier, this transition summarizes all the Petri net connected between the place *Queue* and transition *Processing* in Fig. 1. This technique has as benefit to significantly reduce the state space of the SPN.*

*To the transition is associated a firing rate inversely proportional to the response time of the  $n$  requests marking the input place, that means:*

$$\theta(t, M) = 1 \div \text{ResponseTime}(M)$$

### 4.2 Analysis Process

The analysis process proceeds as follows:

- First, we build the model of each tier  $T_i$ . We denote the obtained model  $M_i$ .
- We isolate the model  $M_N$  of the last tier. A complete analysis of this model is, then, performed to compute the mean response time of the tier under different initial markings and transitions rates.
- We move to the analysis of the model  $M_{N-1}$  of the tier  $T_{N-1}$ . To do that:

- We isolate the model  $M_{N-1}$  of the tier  $N-1$ , beginning from the *Requests* place of the tier, as if it is the 1st tier.
- The analyzed tier's model  $M_N$  is replaced by its abstract view  $A(M_N)$ .
- The abstract model  $A(M_N)$  is connected to the model  $M_{N-1}$ , according to the system specification.

This step gives a new aggregated model, denoted  $AGGR_{N-1}$  representing the two tiers  $T_N$  and  $T_{N-1}$ . The analysis of this aggregated model will result in a reduced number of states.

- We analyze the obtained model  $AGGR_{N-1}$ , and compute the mean response time of the tier  $T_{N-1}$  under different initial markings.
- We move to the analysis of the model  $M_{N-2}$  of the tier  $T_{N-2}$ . The reduction is applied again on the model  $AGGR_{N-1}$ , reducing the two last tiers with a new abstract model  $A(M_{N-1})$ . We repeat the construction step of the aggregated model of the tier  $T_{N-2}$ , then the analysis step.
- The previous steps are repeated until all tiers are considered. The recursive reduction of each analyzed part of the system ensures to keep a reasonable state space of the obtained Petri net, regardless of the total number of tiers in the system. The final aggregated model analysis (of  $AGGR_1$ ) gives then performance and reliability metrics of the whole system, avoiding a combinatorial explosion of the state space number.

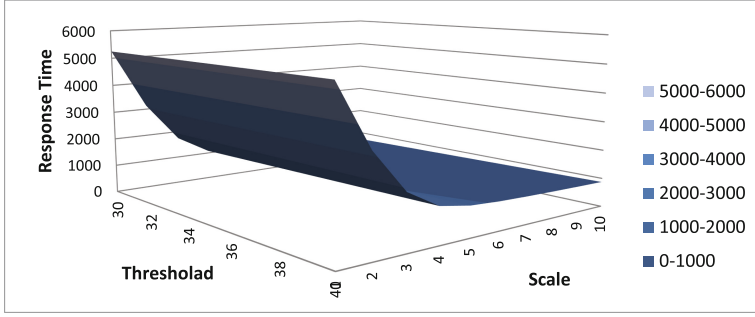
This analysis technique ensures the scalability of our methodology, and guarantees faithful results with regard to the unreduced model. In the next section, we illustrate our methodology with a set of experiments applied to an illustration example, the targeted.

## 5 Illustration

We directed a set of experiments on the constructed model for a 2-tiers system. The analysis of the obtained model was done using the GreatSPN package [3], on an Ubuntu linux 12.4 LTS workstation with 4 GO of RAM. Perl scripting was used to automate the tests with GreatSPN under different configurations, by varying the workload intensity, the available resources number, the threshold of self-adaptation launching and the amount of resources scaled.

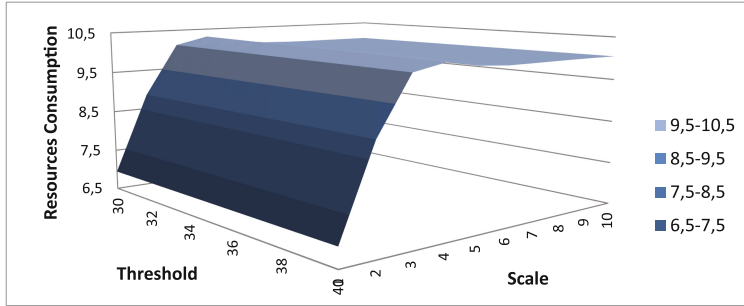
Figure 2 shows the mean response time of requests processed by the system, under the variation of *param 1* and *param 2*. The threshold of monitored requests and the amount of scaled resources represent the two main parameters influencing the autonomic manager efficiency. The figure shows an inversely logarithmic evolution of the response time, as the number of scaled resources increases, while the obtained results are slightly more important for a lower reconfiguration threshold. A stagnation is then noticed at the final part of the graph where the response time remains slightly the same for the different threshold values. This metric helps in identifying the configuration points where no impact is induced on the system performances.

For the same parameters, Fig. 3 gives the amount of resources used by each system configuration. This metric is a main value that should be optimized in



**Fig. 2.** Mean response time varying the autonomic manager parameters

a resource allocation system, as a high number of active resources increases the energy consumption and the system operation cost. First, a higher scaling implies obviously more resources use; Although, from a certain threshold of scaling, the consumption decrease and stagnate at a same point, this is due to the improvement of performances that leads to less reconfigurations, which implies less resources consumption. In the other hand, the consumption decrease linearly according to the reconfiguration threshold increasing.



**Fig. 3.** Resources consumption varying the autonomic manager parameters

These experiments show that the use of our modelling approach helps in identifying such well suitable adjustments, according to particular workloads intensities and parameters adjustments. The computed metrics can then be used to efficiently configure the targeted system to ensure an appropriate solution between costs and quality of service.

## 6 Conclusion

Quality of service and energy management is one of the main issue of today's cloud applications, especially web applications usually designed following a multi-tier architecture. In this direction, we presented, in this paper, a modelling

and analysis approach to be used in an autonomic manager, to find the best balance between performance and resources consumption.

This contribution helps designers in forecasting the most appropriate configuration (and parametrization) to apply on a system before its deployment. For this purpose, we used SPN modelling along with a reduction method in order to model the main components of such systems and predict performances and ensure the system scalability.

To show how to reach the performance prediction goal, the obtained models were analyzed. Experimental results highlighted how to identify the most suitable parameters for the self-scaling component depending on the workload evolution.

Finally, our modelling approach requires us to develop appropriate tools implementing the proposed performance modelling and analysis, to be done automatically. This will allow to exploit concretely our modelling to build trustworthy and reliable systems.

## References

1. Autonomic Computing: Principles, Design and Implementation. Springer International Publishing (2013)
2. Amoretta, M., Zanichellib, F., Conte, G.: Efficient autonomic cloud computing using online discrete event simulation. *J. Parallel Distrib. Comput.* **73**(1), 4–13 (2013)
3. Baarir, S., Beccuti, M., Cerotti, D., Pierro, M.D., Donatelli, S., Franceschinis, G.: The greatspn tool: Recent enhancements (2009)
4. Buyya, R., N.Calheiros, R., Li, X.: Autonomic cloud computing: Open challenges and architectural elements (2012)
5. Goudarzi, H., Pedram, M.: Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems. In: *IEEE 4th International Conference on Cloud Computing* (2011)
6. Hu, Y., Wong, J., Iszlai, G., Litoiu, M.: Resource provisioning for cloud computing. *Future Gener. Comput. Sys.* **25**(6), 599–616 (2012)
7. Litoiu, M.: A performance analysis method for autonomic computing systems. *ACM Trans. Auton. Adapt. Sys.* **2**(1), 3 (2007)
8. Sliem, M., Salmi, N., Ioualalen, M.: An approach for performance modelling and analysis of multi-tiers autonomic systems. In: *The 28th IEEE International Conference on Advanced Information Networking and Applications* (2014)
9. Tchanaa, A., Tranb, G.S., Broto, L., DePalmaa, N., Hagimont, D.: Two levels autonomic resource management in virtualized iaas. *Future Gener. Comput. Sys.* **29**(6), 1319–1332 (2013)
10. Yuan, H., Bi, J., Li, B.H., Chai, X.: An approach to optimized resource allocation for cloud simulation platform. In: *14th International Conference on Systems Simulation* (2014)



Internet and Distributed Computing Systems

8th International Conference, IDCS 2015, Windsor, UK,

September 2-4, 2015. Proceedings

Di Fatta, G.; Fortino, G.; Li, W.; Pathan, M.; Stahl, F.;

Guerrieri, A. (Eds.)

2015, XIII, 306 p. 113 illus., Softcover

ISBN: 978-3-319-23236-2