Comparison of Request Admission Based Performance Isolation Approaches in Multi-tenant SaaS Applications

Rouven Kreb[1] and Manuel Loesch[2]
[1]SAP AG, Walldorf, Germany
[2] FZI Research Center for Information Technology, Karlsruhe, Germany

Keywords: Multi-tenancy, Performance, Isolation, SaaS, QoS.

Abstract: In the Software-as-a-Service model one single application instance is usually shared between different tenants to decrease operational costs. However, sharing at this level may lead to undesired influence from one tenant onto the performance observed by the others. Intentionally, the application does not manage hardware resources and the responsible OS is not aware of application level entities like tenants. Consequently, it is difficult to control the performance of individual tenants to make sure they are isolated. In this paper we present an overview and classification of methods to ensure performance isolation based on request admission control. Further, informational requirements of these methods are discussed.

1 INTRODUCTION

In the cloud computing scenario the providers leverage economies of scale and resource sharing to reduce costs of their service offerings. Multi-tenancy enables one application instance to be shared between different tenants, including all underneath layers. This architectural style is widely used in Software-as-a-Service (SaaS) to decrease costs. A tenant is defined as a group of users sharing the same view on an application. A view includes the data they access, the configuration, the user management, particular functionality, and non-functional properties (Krebs et al. 2012). Typically, a tenant represents a legal entity like a company. Thus, multi-tenancy provides every tenant a dedicated view and share of the instance which is isolated from other shares.

Since Multi-tenant Applications (MTA) share the hardware, operating system, and application instance there are potential performance influences between different tenants and unreliable performance is one of the major concerns for potential cloud users. Consequently, one of the cloud provider's goals is to provide the best possible isolation, with regards to the performance the tenants observe.

Performance isolation exists if for customers working within their quotas, the performance is not affected when aggressive customers exceed their quotas (Krebs et al. 2012).

Besides performance isolation, product diversification is an important economic factor. Diversification could be achieved by additional functionality or tenant specific Quality-of-Service (QoS), since customers have a divergent willingness to pay for performance. At the application level the guaranteed performance usually refers to response time and throughput as long as a defined request rate (quota) is not exceeded.

In MTAs where different tenants share one single application instance, the intended abstraction between the operating system that provides resource management and the application that serves multiple tenants makes performance isolation harder to achieve. This is due to the fact that the operating system is not aware of entities like tenants and the application has no resource control, which is essential for controlling performance.

Various approaches to ensure performance isolation in MTAs were already discussed in the literature (Li et al. 2008; Krebs et al. 2012). In this paper we focus on approaches which apply a request based admission control. These methods delay, or reject requests from a certain tenant before they are processed by the application server. Thus, it is possible to influence the performance for each tenant and to reduce the impact of one tenant's requests onto the others as the amount of competing requests in the application server can be limited.

In this paper we present an overview of 5 generic

Kreb R. and Loesch M..
Comparison of Request Admission Based Performance Isolation Approaches in Multi-tenant SaaS Applications.
DOI: 10.5220/0004948804330438
In *Proceedings of the 4th International Conference on Cloud Computing and Services Science* (CLOSER-2014), pages 433-438
ISBN: 978-989-758-019-2

methods which can be used to establish such an admission control and estimate their capabilities to ensure isolation. Further, we describe the concrete information requirements that have to be fulfilled to realize the different approaches. This helps developers who aim for performance isolation to find the most suitable approach for their scenario.

The reminder of this paper is structured as follows. In Section 2 we provide an overview and classification of potential methods to ensure performance isolation in MT environments based on admission control, and discuss their pros and cons. Section 3 summarizes the informational requirements of the introduced classes. An overview about related work in the area of performance isolation is presented in Section 4. Finally, Section 5 concludes with a summary and an outlook on future work.

2 PERFORMANCE ISOLATION METHODS

In this chapter, five different classes of admission control measures for performance isolation and QoS differentiation are presented. The separation into classes is done based on the kind of required information to make a particular method work.

2.1 Static Approaches

Static approaches make decisions without considering further runtime information such as perceived response times. Usually these approaches have a constant behavior over time. The basic idea for static approaches is illustrated in Figure 1.

A Parameterized Admission Control mechanism uses static rules with constant priorities per tenant for the decision of which request is allowed to pass through to the *Multi-Tenant Application*. Thus, the requests can be accepted, refused, or delayed. The information used by such approaches are the SLA guarantees given to a tenant which are used to derive a constant priority for each tenant. In order to provide QoS differentiation, static priorities can be assigned to the tenants based on the SLA-guaranteed response time and quota. The benefit of such approaches is the rather simple implementation. However, they do not work well without feedback about the perceived performance and quota used by the tenants due to the combination of random load and a difficultly predictable system behavior, combined with the aim of sharing idle resources.

Different kinds of static isolation approaches and there pros and cons have already been discussed (Krebs et al. 2012). One concrete example is a Round Robin with an own FIFO-queue per tenant. In contrast to traditional, non-MTAs such an implementation does not only use one single queue for requests from all tenants, but one dedicated queue for each tenant. The queues are queried one after each other every time when a new request can be processed.

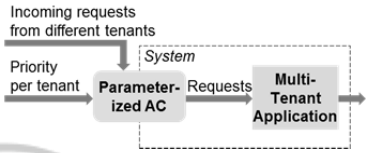

Figure 1: Static Approaches.

The approaches provide a very good isolation as long as the system is not over-committed; however, it is possible that one queue holds plenty requests and the respective tenant's average response time cannot be reached although it could be. Such a scenario is, e.g., given when a tenant has a small amount of requests in his queue which results in a low average response time for this tenant while other tenants may have SLA violations due to longer queues.

2.2 Response Time/Feedback Based Approaches

Static approaches come along with the discussed disadvantages in over-committed and high load scenarios. It is possible that a tenant with a high request rate perceives SLA violations although other tenants could have an average response time significantly below their SLA-guaranteed. In such a case, the average response time of the tenant with a high request rate could be reduced by degrading the tenants that are significantly below their SLA-guaranteed response times. To solve this issue runtime information have to be considered to dynamically adjust the priorities. According to our SLA definition, the perceived response times and the quota are of interest.

Hence, response time based approaches like (Lin et al. 2009) introduce a control loop using feedback about response times and the throughput as depicted in Figure 2.

A *Monitor* offers the perceived response times per tenant as well as the throughput per tenant. This allows the *AC Controller* to dynamically adjust priorities per tenant. They are enforced by the *Parameterized Admission Control*. Since SLA

guarantees are known by the *AC Controller*, the knowledge of the perceived response times allows a more efficient operation by solving the above-described issues.

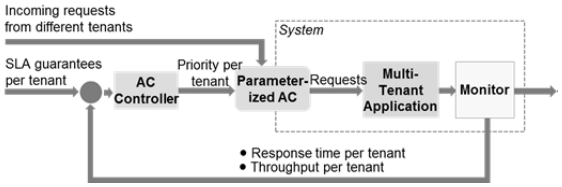

Figure 2: Response Time/Feedback Based approaches.

2.3 Resource Demand Based Approaches

Approaches like (Wang et al. 2012) (Krebs et al. 2014) try to directly control the resource consumption of a tenant to ensure application level performance guarantees. The entitled amount of resources is derived from the SLA guarantees. It can be expected that a higher quality of performance isolation can be achieved since resource demand approaches consider the root cause (consumption of required resources) for the perceived performance, rather than the implication (response times). The mapping of high-level SLAs such as guaranteed response times from the application layer to low-level resource requirements on the resource layer is a research topic that currently attracts a lot of attention. The idea of these approaches is illustrated in Figure 3.

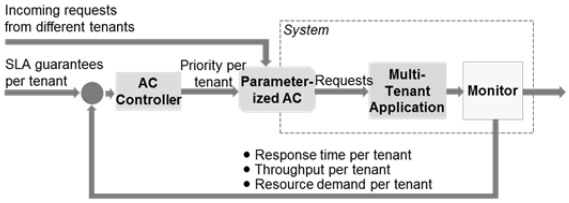

Figure 3: Resource Demand Based Approaches.

The *AC Controller* gives a priority per tenant based on the static SLA guarantees, the current resource demand per tenant as well as the response time and throughput per tenant. The *AC Controller* has two major tasks. First, the mapping of a tenant's SLA guarantee, its perceived response time and its perceived throughput to a fair resources demand per tenant. Second, to set priorities to enforce the tenant's calculated fair resource demands based on the deviation to his current resource demand. Besides more accurate performance isolation, such approaches allow alternative billing options in which tenants pay for the resources they are allowed to consume.

It has to be considered that MTAs are typically composed of three tiers (client, application, database) (Koziolek 2011). Each tier could consist of several instances (Loesch & Krebs 2013). This has to be considered by resource demand based approaches whereas in the previous approaches these internal details could be ignored. Further, different kinds of resources for both the application and the database tier exist. It also is worth mentioning that some resources such as RAM allow determining a current demand represented in Byte whereas resources like CPU only know the two states busy and idle. For them, a well-defined time frame has to be considered for specifying their utilization.

Finally, resource demand based approaches have to consider that the resource demand per request type can vary between different tenants due to tenant-specific customization of the application. The resource demand is also not static over time. For example, increasing database sizes or changing configurations of the application might have an impact.

2.3.1 Direct and Indirect Resource Demand Estimation

For estimating a tenant's resource demand, two fundamental different approaches can be separated: Direct and Indirect Resource Demand Estimation (RDE).

Direct RDE leverages operating system / platform functionality that allows monitoring the resource demand on a tenant's basis at the application-layer. It is the approach depicted in Figure 3. The tenant-specific resource demand is then directly reported to the *AC Controller* which in turn is responsible for adjusting the admission control parameters. This is based on the system's capability to measure the resource demand on a fine-grained level that delivers the amount of required resources per user request (e.g., the service time spent at CPU and required Bytes of RAM). If such monitoring functionality is available it often comes along with a significant overhead (Kuperberg et al. 2011) or lacks in monitoring capabilities among different processes.

Indirect RDE uses algorithms to determine the resource demand per tenant without internal knowledge about tenants by using information that can be measured from outside the tenant-aware application. To estimate the resource demand,

indirect RDE approaches mostly make use of the resource utilization, throughput and response time. Since we are interested in the resource demand per tenant, the throughput must be reported per tenant. For implementing indirect RDE different techniques can be applied such as linear regression or Kalman filter as realized by (Wang et al. 2012). The control loop for indirect RDE approaches is depicted in Figure 4.

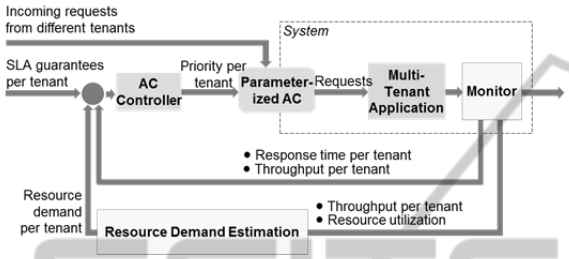

Figure 4: Indirect Resource Demand Estimation.

2.4 Enhanced Resource Demand Approaches

It is possible that the perceived response time is bad due to a high utilization of a single resource that has to be used in order to complete a request. We assume it is possible to build a system where the resource demand of a certain request type is known (Krebs et al. 2014). Together with information about the resource utilization it may allow improving admission decisions. This approach is illustrated in Figure 5.

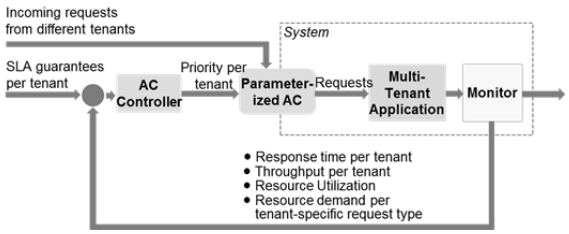

Figure 5: Control Loop for Enhanced Resource Demand based Approaches using direct RDE.

In the previous presented Resource Demand Approaches the resource utilization is only used in case of indirect Resource Demand Estimation. In this kind of approaches, the resource utilization is always delivered to the *AC Controller*. Furthermore, in the previous presented approach, the resource demand was considered on a tenant-base (directly, or indirectly via the *Resource Demand Estimation* component). In this approach, the resource demand can be gathered directly or indirectly as well, however it has to be delivered per tenant-specific request type. Using this information, the *AC Controller* can make better admission decisions. By knowing the resource demand per request type as well as the current utilization of the available resources, requests can be allowed to pass through, even if other resources are highly utilized.

The subsequent scenario illustrates the benefit of these kinds of approaches. For example, a resource demand estimation mechanism may determine that a certain request type requires no resources on the database server. If the database server's resources are high utilized and the application server has a low utilization, the controller can adjust the admission control parameters in a way that this request type is accepted since enough of the crucial resource is available. In contrast to all previous approaches, this requires the Admission Control to be not only tenant- but also request type-specific. Depending on the variety of resource requirements of different requests, this may significantly increase the throughput but again increases complexity.

2.5 Model-based Approaches

Model-based approaches make use of performance prediction models which are based on workload forecasts. Foreseeing performance problems has two major advantages. (1) Adaptation decision can be made pro-actively before problems occur. (2) The impact of different adaptation decisions can be simulated which allows choosing the best adaptation decisions.

Respective performance predictions are driven by the transition towards Cloud Computing platforms and are especially addressed by the research area of self-aware systems engineering (Kounev et al. 2010). Performance prediction models are usually based on the detection of workload patterns and Hidden Markov Models that allow predicting variations in such patterns. Based on such performance prediction models, Model-based Approaches work as depicted in Figure 6. The Performance Prediction Model allows the prediction of response times based on the current AC parameters.

This way, the AC Controller can make adaptation decisions before performance problems occur. Further, it is possible to simulate the impact of different adaptation decisions. Therefore, the *AC Controller* could create multiple adaptation decisions (e.g., multiple sets of tenant priorities) that are expected to optimize the performance isolation.

By using the *Performance Prediction Model*, the

influence of different options onto the performance could then be simulated. The decision that comes along with the best performance isolation can be chosen based on a fitness function for the overall SLA compliance and maybe resource utilization (cf. Section 2.3 and Section 2.4).

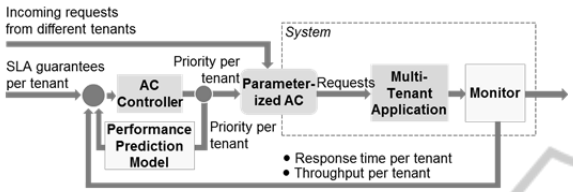

Figure 6: Control Loop for Model-Based Approaches.

3 OVERVIEW

Table 1 depicts an overview about the elaborated informational requirements of the introduced classes of performance isolations approaches. The specific advantages that are reasoned by the increased amount of information used have been discussed in the previous sections as well as the increased effort to gather this information and to implement the presented classes.

Table 1: Informational Requirements.

	1	2	3	4	5
Informational Requirement					
SLA guarantees, per tenant	x	x	x	x	x
Response time, per tenant		x	x	x	x
Throughput, per tenant		x	x	x	x
Resource demand - per tenant - per request type			x	 x	 opt.
Resource utilization				x	opt.
Model for performance predictions					x
Admission Control Parameters					
Specific to tenant	x	x	x	x	x
Specific to request type				x	opt.

It can be recognized that *Static Approaches (1)* only use static SLA information without considering runtime information at all. To overcome the problem of these approaches, we introduced *Response Time based Approaches (2)* that use a control loop leveraging response times and throughputs as runtime information. *Resource Demand based Approaches (3)* try to take advantage by making admission decisions based on the tenant's resource demands which are the root cause for their perceived performance. The table shows the informational requirements for direct RDE approaches where the resource demand is gathered through direct measurements. If the resource demand should not be measured directly, it can be estimated based on the tenant's throughput and the resource's utilization (cf. Section 2.3). The *Enhanced Resource Demand based Approaches (4)* enable a performance isolation that allows preferring requests that can be processed by free resources. This will increase the overall utilization and hence the economic efficiency. Therefore, the resource utilization is required and the resource demands have to be specific for the tenant and the tenant-specific request-type. It is worth to mention again, if the resource demands should be estimated indirectly, this can be done based on the request types' throughputs, response times and the resource's utilization. *Model-based Approaches (5)* are able to anticipate performance issues and simulate different adaptation decisions. Therefore, they require a model for performance predictions, usually based on workload forecasts. For them, it is optional to consider resource demands. When considering resource utilization and the resource demands on basis of request types, free resources can be utilized more efficiently by giving precedence to respective request types as in the case of Enhanced Resource Demand based Approaches.

4 RELATED WORK

Lin et al. (Lin et al. 2009) focus on providing different QoS for different tenants by regulating response times. They make use of a regulator which is based on feedback control theory to achieve this. Li et al. (Li et al. 2008) first predict performance anomalies and identify an aggressive tenant who is responsible for the incident. Based on this, their approach applies an adoption strategy reducing the influence of the aggressive tenant on the others to ensure isolation. Wang et al. (Wang et al. 2012) developed a tenant-based resource demand estimation technique using Kalman filters. This approach predicts the current resource usage of a tenant and uses this information to control the admission of incoming requests based in a predefined quota. Four static mechanisms to realize performance isolation were described and evaluated in (Krebs et al. 2012). One approach is based on thread pool management and three other approaches leverage admission control.

These papers focus on the concrete algorithms and do not discuss the general approach nor classify

their approach or evaluate general informational requirements.

Guo (Guo et al. 2007) discuss multiple isolation aspects including performance isolation for MTAs on a conceptual level. They propose Resource Partitioning, Resource Reservation and Request-based Admission Control as mechanisms to maintain a certain QoS for each tenant. In comparison, we provide a classification and potential solutions within Request-based Admission Control which was not covered in detail. In (Loesch & Krebs 2013) the authors describe where a request-based admission control has to be realized in a distributed MTA scenario. However, a classification or concrete measures to achieve performance isolation are not described. Koziolek (Koziolek 2011) evaluated several existing MTAs and derived a common architectural style. However, Koziolek's architectural style does not discuss performance isolation at all.

5 CONCLUSIONS

Multi-tenancy is an architectural approach which shares one single application instance between several customers. This allows increasing the efficiency of SaaS applications. However, due to the abstraction of the application which is tenant aware, from the actual resources, controlled by the non-tenant aware OS, it is complicated to ensure the isolation of the performance observed by different tenants. In this paper we presented five conceptual approaches with increasing capabilities to control performance to the detriment of the complexity and need for detailed information about the system at runtime. All presented approaches overcome the described layer discrepancy by applying a request-based admission control. The simplest approach is based on a static admission control like a Round Robin which successively selects tenants' requests from tenant specific queues. In contrast, the most complex approach uses performance prediction models to find a suitable admission control method by proactively reacting to forecasted performance incidents that might happen in the future. We further discussed specific advantages of these approaches, followed by a list and comparison of their information requirements. This helps developers of multi-tenant applications to find an appropriate method.

REFERENCES

Guo, C. J. et al., 2007. A Framework for Native Multi-Tenancy Application Development and Management. In: *Proc. of the 9th IEEE Int. Conf. on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services.*

Kounev, S. et al., 2010. Towards Self-Aware Performance and Resource Management in Modern Service-Oriented Systems. In: *Proceedings of the IEEE International Conference on Services Computing.*

Koziolek, H., 2011. The SPOSAD Architectural Style for Multi-tenant Software Applications. In: *Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture.*

Krebs, R., Momm, C. & Kounev, S., 2012. Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments. In: *Proc. of the 8th international ACM Conference on Quality of Software Architectures*.

Krebs, R. et al., 2014. Resource Usage Control In Multi-Tenant Applications. In *Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.*

Kuperberg, M. et al., 2011. Defining and Quantifying Elasticity of Resources in Cloud Computing and Scalable Platforms. Karlsruhe Institute of Technology.

Li, X. H. et al., 2008. SPIN: Service Performance Isolation Infrastructure in Multi-tenancy Environment. In: *Proceedings of the 6th Int. Conference on Service-Oriented Computing.*

Lin, H. L. H. et al., 2009. Feedback-Control-Based Performance Regulation for Multi-Tenant Applications. In: *Proceedings of the 5th International Conference on Parallel and Distributed Systems (ICPADS).*

Loesch, M. & Krebs, R., 2013. Conceptual Approach for Performance Isolation in Multi-Tenant Systems. In: *Proceedings of the 3rd International Conference on Cloud Computing and Services Science.*

Wang, W. et al., 2012. Application-Level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications. In: *Proceedings of the IEEE 5th International Conference on Cloud Computing*.