A Feedback Controlled Scheduler for Performance Isolation in Multi-tenant Applications

Rouven Krebs
TIP Core Research
SAP AG
Rouven.Krebs@sap.com

Arpit Mehta
TIP Core Research
SAP AG
Arpit.Mehta@sap.com

***Abstract*—Multi-tenancy is an architectural approach for sharing one application instance among different tenants. It is used to reduce operational costs of SaaS offerings. The major drawback of this approach is the potential mutual influence of one tenant onto the performance experienced by others. In this paper we present a combination of a traditional scheduling mechanism and a feedback control loop based controller to ensure performance isolation while efficiently utilizing the system and faster reaction to workload changes than a control cycle's duration.**

Keywords—multi-tenancy; SaaS; performance; isolation; SLA; QoS

I. INTRODUCTION

Cloud computing significantly decreases costs for IT through economies of scale. Furthermore, delivering computing resources as a service allows sharing among multiple customers and thus a temporarily higher workload of a customer can be covered in case of simultaneously lower workload of other customers. In Software as a Service (SaaS) offerings, often one single application instance is shared among several customers. Such an architectural style is called multi-tenancy and is motivated by large cost saving potential. A tenant is a group of users sharing the same view on an application they use. This view includes the data they access, the configuration, the user management, particular functionality and related non-functional properties [2].

A major obstacle in Cloud Computing is the unreliable performance [1] and providing performance guarantees is a major research issue. Especially in Multi-tenant Applications (MTA) it is likely that one tenant can significantly degrade the overall performance of all tenants due to the tight coupling of different tenants within one application instance and their abstraction from the actual control of the hardware resources. A system is said to be performance isolated, if for tenants working within their quotas the performance is within the (response time) SLA while other tenants exceed their quotas (e.g., request rate). One tenant is allowed to exceed its quota as long as other tenants do not observe a negative impact onto their SLAs. In this paper, we present an approach to performance isolate different tenants from the influence of others based on a request admission control and a control feedback loop. Subsequently, a brief overview on related work on which the concrete approach is based on and an overview of the results obtained by applying this approach is presented.

II. SYSTEM FOR PERFORMANCE ISOLATION

Several solutions (e.g., [3]) try to place tenants onto a defined set of nodes in a way such that their workload profiles do not interfere with others to avoid mutual performance influences. However, this is only possible if the workload profiles for the tenants are predictable. In [2] various scheduling mechanisms, e.g., Round Robin for forwarding the tenants requests were introduced and evaluated with concerns to their isolation capabilities. It was shown, that these approaches can provide a certain degree of isolation. However, they still lack in efficiency, especially when the system becomes overcommitted due to constant priorities for each tenant. If a system is overcommitted a tenant using his entire quota has to get a higher priority to utilize resources not used by other tenants. Lin et al. [3] use control feedback loops to ensure isolation. The authors observe the current response times for each tenant and regulate their throughput based on a comparison of the actual response time and the guaranteed reference response time. However, in situations where one tenant suddenly changes his workload a control feedback loop needs some time to react to these changes and to adapt the throughput.

Scheduling approaches have the benefit to react very fast to changes in the workload profile as they do not have to wait until a control cycle is completed, but they cannot dynamically adjust the priorities of the tenants to find an optimal setup. We combined a control feedback loop with a priority based scheduling mechanism to dynamically adjust the priorities of a tenant. This allows converging to optimal weights, and also the impact of potential SLA violations can be limited on a short term basis by the scheduler. Furthermore, if a tenant doesn't send as many requests as expected within one control cycle, the other tenants' requests are forwarded as the scheduler always select the next tenants queue in this situation. Thus, the system can be efficiently operated in terms of efficient resource utilization. Figure 1 shows the conceptual approach. Each tenant T_i is associated with an individual SLA which consists of an average response time R and the maximum quota of throughput X which corresponds to the request rate. Arriving request are queued in a tenant specific queue. Once the application server has free threads a Weighted Round Robin

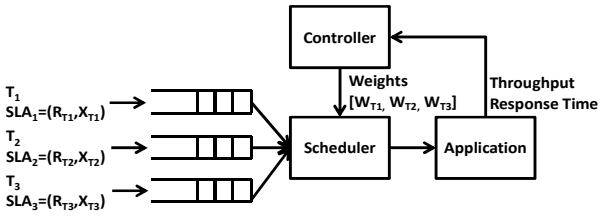

Figure 1: Conceptual Approach

(WRR) selects a request from the queues with pending request according to the given weights. Consequently, within a given period, the proportion of requests processed for one tenant as compared to the others corresponds to the proportion of their weight. If one tenant has no pending request, another tenants' request will be handled, and accounted, immediately without any delay. A Proportional Integral (PI) controller receives the throughput and the average response time information for each tenant and calculates the WRR weights for the next period. As basis for this an error is calculated by comparing the actual measured average response time and throughput with the SLAs. The proportional part of the PI controllers use the error signal from the last period and the integral part use the error value from the last n periods of time to calculate the control signal, i.e., the weights in our case. In this implementation the integral part was limited to the last two periods. These calculated weights are used by the WRR for the complete next period until the controller sends new updates.

III. EVALUATION

The presented approach was applied in the context of SAP HANA Cloud to ensure performance isolation in an enhanced version of the widely accepted TPC-W benchmark which supports Multi-tenancy (MT TPC-W) [5]. The system under test (SUT) runs a SLES11 SP2 as host system on a server with 16GB memory and 16x2GHz CPUs. This system hosts a Virtual Machine with 1x2GHz and 2GB memory with an SLES11 SP2 operating system. The system runs a SAP HANA Cloud application runtime container which serves the MT TPC-W. The WRR was directly embedded into the application runtime container, the controller was deployed on a separate machine and the SUT was also running on a separate environment. In both experiments the server was configured to run with 100 threads. Figure 2 shows the results where the respective SLA was set to R=2500ms and X=3100 requests per minute. The abiding tenant (creating less than the allowed requests) and the disruptive tenant (creating more than the allowed requests) were started with 500 users each. Once these users were started, another 500 users were added to the disruptive one to turn it disruptive since it will then send more requests then allowed. Overall, after 600 seconds all users where started and the controller could start to adjust the weights at a constant load. At 900 seconds it was observed, that the SLA for the abiding one was no longer violated and the disruptive tenant has significantly higher response times because of higher queuing delay due to low priority. Figure 3 shows a setup where the disruptive tenant has 1000 users and the abiding one 500 users at the beginning. In this scenario the

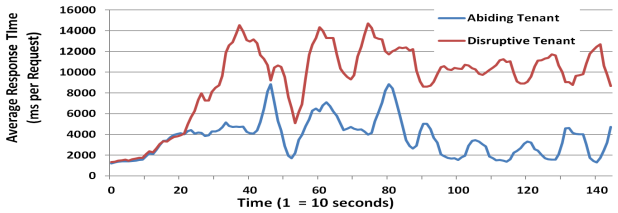

Figure 2: Isolation Capability of the System

SLAs were configured to R=4000ms and X=3100 for each tenant. In this test case, the throughput for the abiding tenant did not exceed the SLA, as the amount of users was too low. The throughput of the disruptive tenant was also limited by the controller to avoid negative impact onto the performance. At 1210 seconds the load of the abiding tenant was immediately reduced by 300 users. Thus, the overall system utilization was reduced; hence the resource became free which could be used by the disruptive tenant. Therefore, the throughput for that tenant increased very fast between 1210 and 1270 seconds.

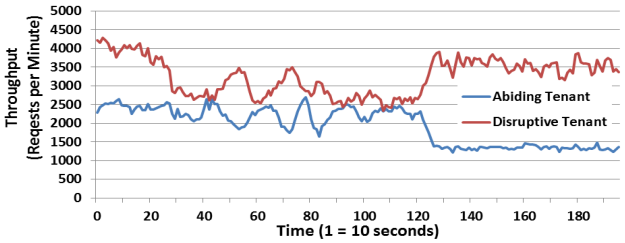

Figure 3: Capability of the System to Shift Free Resources

IV. CONCLUSION

This paper combines a Weighted Round Robin (WRR) scheduling mechanism with a control feedback loop to ensure performance isolation. Such combination enables the system to react very fast to load fluctuations and also ensures that the weights for the WRR converge to an optimal setup in which the overall SLAs are fulfilled and the system runs in an efficient resource utilization state.

REFERENCES

[1] Bitcurrent, "Bitcurrent Cloud Computing Survey 2011", bitcurrent, 2011

[2] R. Krebs, C. Momm, and S. Kounev, "Metrics and Techniques for Quantifying Performance Isolation in Cloud Environments", Proceedings of the 8th ACM SIGSOFT International Conference on the Quality of Software Architectures (QoSA 2012), 2012.

[3] H. Lin, K. Sun, S. Zhao, and Y. Han, "Feedback-Control-Based Performance Regulation for Multi-Tenant Applications", Proceedings of the 2009 15th International Conference on Parallel and Distributed Systems, IEEE Computer Society, 2009.

[4] Y. Zhang, Z. Wang, B. Gao, C. Guo, W. Sun, and X. Li, "An Effective Heuristic for On-line Tenant Placement Problem in SaaS", Web Services, IEEE International Conference on, IEEE Computer Society, 2010.

[5] TPC BENCHMARK W, Transaction Processing Performance Counci., 2002.