International Journal of Engineering Research & Management Technology

ISSN: 2348-4039

July-2018 Volume-5, Issue-4

Email: editor@ijermt.org www.ijermt.org

IMPACT OF PROGNOSTICATION FRAMEWORK ON SERVER PERFORMANCE

Monika Sainger
Research Scholar (CSE)
Mewar University, Rajasthan

Prof.(Dr.) Sarvottam Dixit
Advisor to Chairman
Mewar University, Rajasthan

ABSTRACT

Adaptive allocation of resources based on variations in workload, improves the overall resources utilization of the system. In this paper we have described the impact of prognostication framework that captures the changes in workload demand, on performance of the server. This framework provides a cost effective allocation of resources which is beneficial for both cloud user as well as cloud provider. Here the option of an application specific prognostication engine may be argued but it is a challenge for an application developer to develop an application specific prognostication engine. However, if the IaaS framework provides some provision for a standard prognostication engines many applications will benefit. Therefore a prognostication engine generic enough like this will be able to cater many applications needs.

KEYWORDS: Prognostication Engine, variable workload, cost-effective allocation.

INTRODUCTION

Enabling elastic resource allocation in IaaS is useful for applications with highly variable workload. Current provisioning techniques in IaaS perform static allocation of resources which sometimes results in under-allocation of resources leading to performance degradation or over-allocation of resources leading to wastage of resources making user pay for resources that might not be used by the application. The main challenge is to allocate the resources in such a way that they are not over-allocated and at the same time ensure that performance doesn't get degraded. In order to achieve this, a forecast based approach must be used that predicts the user workload and takes an appropriate resource allocation decision a priori.

In our work a prognostication engine based on a cost model is introduced into the core layer of the Iaas architecture. As indicated earlier, the main function of this component is to host a prediction model for an application hosted on the cloud. It generates the appropriate workload prediction for the next scheduling cycle that is calculated using a cost model, which is then translated into appropriate resource requirement using Resource manager.

The proposed prognostication framework in our work consists of mainly two parts: 1) Prognostication engine, and 2) Cost Model. Prognostication engine predicts the user workload dynamically based on the past usage pattern which is then translated into resource requirement. It produces forecast bounds for a given confidence interval. Cost model further modifies the forecast in such a way that it tries to obtain the best tradeoff between over-allocation of resources and underperformance of application (SLA penalty). It does so by selecting the upper bound of the forecast and finding the appropriate value of confidence interval which produces a minimum positive value of effective cost.

METRICS FOR SERVER PERFORMANCE

REQUESTS RATE

This is the evaluation of how many requests per second are coming to a server. In other words, this metric is called Average Load and it allows one to understand that under what load the web application is currently working. Usually, this is calculated as a count of the requests received during a measurement period, where the period is represented in seconds.

ERROR RATES

Generally some errors may occur when processing requests, especially when under a big load. The Error Rate usually reflects how many response HTTP status codes indicate an error on the server including the requests that never get a response (timed out) knowing that web servers return an HTTP Status Code in the response header. Normal codes are usually 200 (OK) or something in the 3xx range, indicating a redirect on the server. Error rate is calculated as a percentage of problem requests relative to all requests. Common error codes are 4xx and 5xx, which mean the web server knows it has a problem fulfilling that request. Error Rate is a significant metric because it measures "performance failure" in the application and tells how many failed requests have occurred at a particular point in time. Normally, no one can define the tolerance for Error Rate in their web application. Some consider an Error Rate of less than 1% successful. However, normally one must try to minimize possible errors in order to avoid performance problems, and constantly work to eliminate them.

AVERAGE RESPONSE TIMES (ART)

By measuring the duration of every request/response cycle, it will be possible to evaluate how long it takes the target web application to generate a response. The ART takes into consideration every round trip request/response cycle during a monitoring period and calculates the mathematical mean of all the response times. The resulting metric is a reflection of the speed of the web application which is perhaps the best indicator of how the target site is performing, from the users' perspective. The recommended standard unit of measurement for ART is milliseconds.

PEAK RESPONSE TIMES (PRT)

PRT measures the longest round trip of request/response cycles. Generally, the PRT shows that at least one of the resources is potentially problematic which is leading to increase in response time. But, when the ART and PRT start becoming comparable, that indicates that undoubtedly there is a problem in the server. Sometimes it may reflect an anomaly in the application, or may be due to expensive database queries, etc. The standard measurement unit of PRT is recommended to be milliseconds.

UPTIME

Uptime is the amount of time that a server has stayed up and running properly. It reflects the reliability and availability of the server and, obviously, this value should be as large as possible. The value can be calculated as an absolute value or as a percentage of actual server uptime to ideal server uptime. For example, if a server is started on Aug 1, 2018 and checked for the uptime exactly 31 days later i.e. on Aug 31, 2018, then the whole duration is 31 days or 2,678,400 seconds. If the server has been stopped during that period for 1,000 seconds, then the uptime percentage (availability) will be $100 * (1 – (1000 / 2678400)) = 99.963\%$. Usually, if your server is in production, a value less than 99% should lead to attention and less than 95% to troubling.

CPU UTILIZATION

CPU Utilization is the amount of CPU time used by the web application while processing a request. Usually, it is the percentage of CPU usage that is calculated, which indicates how much of the processor's capacity is currently in use by your application. When the percentage of CPU usage begins to max out at 100%, additional action may need to be taken because that points to the existence of some problem in your application, or to a capacity deficiency of the host machine.

MEMORY UTILIZATION

Memory Utilization refers to the amount of memory used by a web application while processing a request. Usually, it is calculated as the process's percentage of memory utilization, which is a ratio of the Resident Set Size to the physical memory whereas the Resident Set Size (space for text, data, stack) is a real occupied memory size.

THE COUNT OF THREADS

As usual, a web application can generate a lot of threads to process requests. The number of threads is an important metric because the number of threads per process is normally limited by the system. So if an application generates too many threads then it can be an indicator that there is a problem in the application. Obviously, the count of existing threads is proportional to the load and inversely proportional to the processing time of the requests.

THE COUNT OF OPEN FILES DESCRIPTORS

A file descriptor is an object that a process uses to read or write to an open file and to open network sockets. Generally, an operating System places limits on the number of file descriptors that a process may open. The non-availability of file descriptors can cause a wide variety of symptoms which are not always easily traced back to. The Open Files Descriptors (OFD) provides a count of the total number of file descriptors that are currently allocated and open for processing. The percentage of the total number of open file descriptors with respect to the maximum allowed count of descriptors for processing is a good metric for evaluating the health of a web application.

REQUEST RATE AND RESPONSE TIME

In our work, we have considered I/O workloads as they have the potential to exercise most of the resources of a system. They provide a useful case study as they are highly prevalent workloads in clouds [2]. I/O workloads are characterized by spurts of CPU usage followed by I/O activity. Examples of I/O workloads are read/write operations to disk, interactive network I/O workloads such as request-response sequence of web server or mail server etc.. The request rate is the number of requests per unit time. In this work, since the scheduling decisions are assumed to be taken on a per hour basis, request rate represents number of requests per hour. Hence the metric that is used to measure the performance of the server is the response time that it takes to service these requests. For each request, there is a response time associated.

SERVER PERFORMANCE USING RESPONSE TIME

In this section we discuss the SLA penalty occurred due to under-performance of the application, when the resources are allocated based on prediction. Basically, when the resources are under-allocated, SLA violations occur because of increase in response time of the system. Hence, to observe the SLA violation in the form of a penalty, response time of the system needs to be measured when the resources are allocated based on the prediction and the system receives the actual workload. Once the system behavior is known, it can then be used to find the response time of the system at each point where the allocated resources based on prediction are not sufficient to fulfill the demand.

RESPONSE TIME WITH UNPREDICTED LIMITED RESOURCES

To find the system's performance behavior, response time of the system is measured after providing limited resources to the system without predicting the resource usage. For a web server application, response time of the system is calculated while restricting CPU allocated to the VM and CPU limit is varied from 0 to 80% for different request rates. In this work, it is assumed that the other VM resources including memory, network bandwidth etc. and hypervisor resources are not constrained. However, these can also be the reason for SLA violations. That means the SLA violation due to these is not considered here as it not within the scope of this study.

Figure 1 shows the response times observed at different request rates, the VM CPU is restricted from 2% to 80%. Initially response time is under the defined performance SLA limit (where SLA violations do not occur, generally, which is, approximately 70 ms for web server application). Till a certain point, for example, at about 5000 requests per hour, response time is well under 150 ms after about 40% of VM CPU allocation. This point is named as Safe Resource Allocation Point (SRAP). This SRAP increases with increase in the request rate as can be observed from the Figure 1.

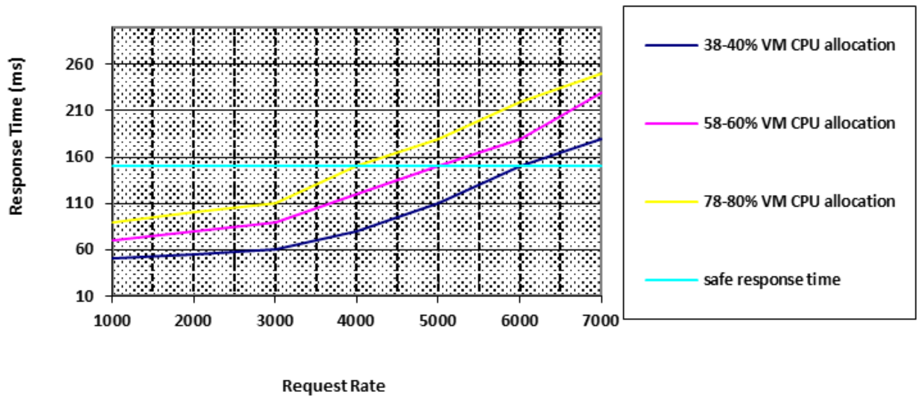

Fig. 1 Response Time with varying request rates with unpredicted resources

RESPONSE TIME USING PREDICTED RESOURCES

Using the response time behavior of the system under restricted resources, the response time of the system for simulated workload can be calculated if the predicted resources are allocated to the system. When the resources are allocated using the predicted value (0% confidence interval), the response time behavior for all of the cases is shown in Fig 2. The x-axis denotes request rate (workload) and y-axis denotes response time. Here, dotted line denotes the safe response time for the system corresponding to that application. Dark line represents response time of the system when the resources are allocated based on prediction. All of the points where the response time exceeds the safe limit correspond to SLA violation points. SLA penalty starts increasing if the response time of the system increases beyond the safe limit and saturates after a long point where in the service becomes meaningless to the user.

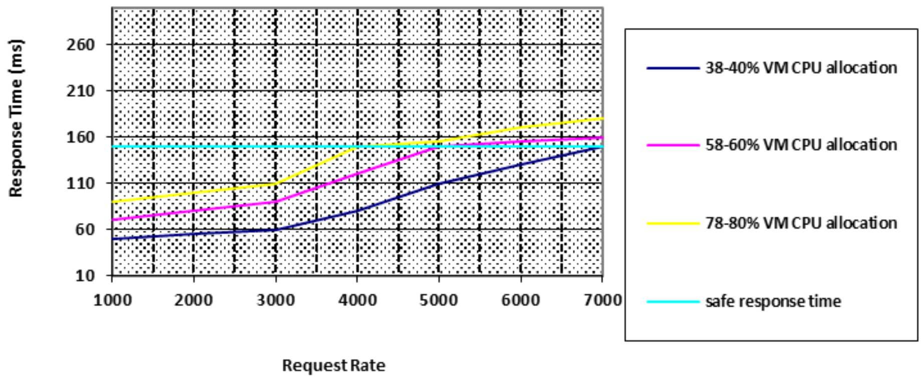

Fig. 2 Response Time with varying request rates with predicted resources

Minimizing Effective Cost

Effective cost can be found as following:

$$C_{Effective} = C_{Over\ allocation}(\alpha)$$

$$\text{And} \quad C_{Over\ allocation} = \alpha * ER$$

Here, ER denotes the extra resources allocated than required. As we have discussed earlier, the upper bound of the forecast for a given confidence interval is used to provision resources. With the increase in confidence interval, the upper bound of the forecast increases. This leads to increase in over-allocation cost for all of the points where allocation is more than the required resources. On the other hand, for all of the points where allocation is less than the required resources, increasing the confidence interval decreases SLA penalty. Hence, it turns out that the effective cost function is the function of confidence interval. The objective is to find the minimum effective cost or a value of confidence interval which minimizes the effective cost.

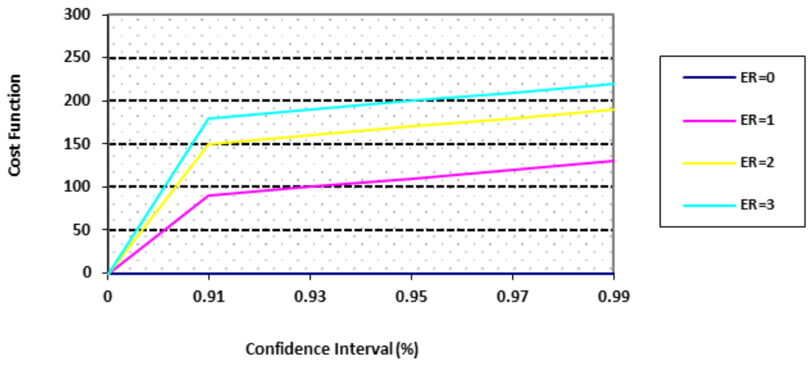

Fig. 3 Cost Function (over-allocation cost) with different confidence intervals

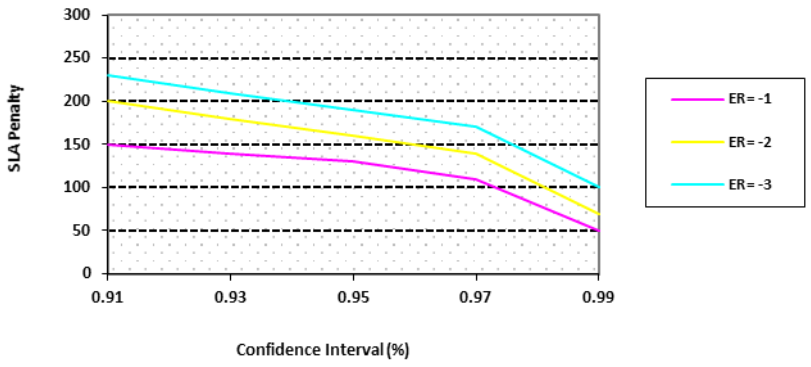

Fig. 4 Cost Function SLA Penalty (under-performance of application) with different confidence intervals

Figure 3 depicts the variation of effective cost function with confidence interval for our data using different values of ER. The x-axis denotes the confidence interval values ranging from 0 to 99% and y-axis denotes the value of effective cost function derived for the system when resources are allocated using the upper bound of the forecast at corresponding confidence intervals. In fig 4 it has been shown that at higher confidence interval, SLA violations will be less (i.e. less SLA penalty) and over-allocation of resources will be more. Hence, one can prioritize SLA violations over over-allocation and vice-versa based on the user's requirement.

IMPROVEMENT USING THE PROGNOSTICATION FRAMEWORK

Using the proposed prognostication framework, the resources can be allocated as per the upper bound of the confidence interval which minimizes the effective cost. This section identifies the improvement in terms of reduction in the resource allocation while keeping the performance intact, using the proposed framework.

• **Reduction in the resource allocation:** The reduction in the resource allocation is achieved using the proposed prognostication framework. An improvement of almost 50% is observed in terms of resource utilization efficiencydepending on the choice of confidence interval.

• **Few SLA violations:** SLA violations occur at those points where the allocated resources are less than the required resources, since the response time of the server can reach beyond the specified safe limit. These are mostly the points where a sudden hike is observed in the workload and which was not in the past patterns from history. Using the dynamic allocation by proposed framework, the SLA violations are very less (about 3.85%).

Hence, using the proposed prognostication framework, significant improvement in the terms of efficient resource allocation has been observed keeping the SLA violations at a minimal level. It can be noted that the improvement in the resource allocation and SLA violations are related to each other i.e. depend on the choice of value of α. Table 1 shows the improvement in resource utilization and SLA violations at the confidence intervals at which the effective cost function is minimized for all of the cases. In summary, predicting workload to obtain resources close to requirement, and minimum SLA violations, conflicts with one another and needs to obtain the best tradeoff between the two. However, both can be achieved simultaneously only when the prediction accuracy is 100% which is practically difficult to achieve.

Workload	Improvement in utilization	SLA Violations
Web Server (Gaussian Process with periodic covariance function)	112.6528%	6.21118%
Web Server (Gaussian Process with linear periodic covariance function)	112.7945%	6.29016%

Table 1: Improvement using proposed prognostication framework

CONCLUSION

Predicting the varying workload and obtaining the resource requirement which is close to the actual requirement in the current IaaS architecture along with preserving performance of the applications hosted is the main idea of our work. The cost model aims at providing the optimal balance between the two opposite goals of improving resource utilization and preserving application's performance.

This paper provides the cost model and evaluates the proposed prognostication framework in [1]. Using the simulation, the response time of the system is tested when the resources are allocated as per the forecast. Further, using different values of confidence interval, effective cost is found for the system and that value is chosen for the confidence interval which minimizes the effective cost.

The results show the significant improvement in the resource utilization achieved using adaptive provisioning of resources based on variations in workload, over the static allocation that is used currently in IaaS clouds. At the same time, it ensures the guaranteed performance to the user by restricting SLA violations to a minimum.

REFERENCES

1. M. Sainger, K.P. Yadav, H.S.Sharma, "Framework for Application Service Behavior Prognostication with Cost-Effective Provisioning In a Utility Cloud", Int. Journal of Engineering Research and Application ISSN : 2248-9622, Vol. 8, Issue3, March2018.
2. D. Mosberger And T. Jin, "Httperf-A Tool For Measuring Web Server Performance," SIGMETRICS Perform. Eval. Rev., Vol. 26, No. 3, Pp. 31– 37, Dec 1998. [Online]. Available: Http://Doi.Acm.Org/10.1145/306225. 306235.

3. A. Anand, "Adaptive Virtual Machine Placement supporting performance SLAs" Master's thesis, Supercomputer Education and Research Center, Indian Institute of Science, 2013.
4. A. Anand, M. Dhingra, J. Lakshmi, and S. K. Nandy, "Resource usage monitoring for kvm based virtual machines," in Proceedings of the 18th annual International Conference on Advanced Computing and Communications (ADCOM 2012), To Be Published, dec. 2012.
5. M. Dhingra, J. Lakshmi, and S. K. Nandy, "Resource usage monitoring in clouds,"in Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, ser. GRID '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 184–191. [Online]. Available: http://dx.doi.org/10.1109/Grid.2012.10
6. K. Boloor, R. Chirkova, T. Salo, and Y. Viniotis, "Analysis of response time per- centile service level agreements in soa-based applications," in Global Telecommuni- cations Conference (GLOBECOM 2011), 2011 IEEE, Dec. 2011.
7. Y. Mei, L. Liu, X. Pu, S. Sivathanu, and X. Dong, "Performance analysis of network i/o workloads in virtualized data centers," Services Computing, IEEE Transactions on, vol. 6, no. 1, pp. 48–63, 2013.
8. G. Reig and J. Guitart, "On the anticipation of resource demands to fulfill the qos of saas web applications," in Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on, sept. 2012, pp. 147 –154.
9. Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, "Cloudscale: elastic resource scaling for multi-tenant cloud systems," in Proceedings of the 2nd ACM Symposium on Cloud Computing, ser. SOCC '11. New York, NY, USA: ACM, 2011, pp. 5:1–5:14. [Online]. Available: http://doi.acm.org/10.1145/2038916.2038921
10. "Scalr Cloud Management," 2013. [Online]. Available: http://scalr.com/
11. "Amazon Auto Scaling," 2013. [Online]. Available: http://aws.amazon.com/
12. R. R. Nikolas Roman Herbst, Samuel Kounev, "Elasticity in cloud computing: What it is, and what it is not," in ICAC 2013, To be published, 2013.