Poster: FECBench: An Extensible Framework for Pinpointing Sources of Performance Interference in the Cloud-Edge Resource Spectrum

Yogesh D. Barve, Shashank Shekhar, Ajay D. Chhokra, Shweta Khare,
Anirban Bhattacharjee and Aniruddha Gokhale
Dept of EECS, Vanderbilt University, Nashville, TN 37212, USA
Email:{yogesh.d.barve,shashank.shekhar,ajay.d.chhokra, shweta.p.khare,anirban.bhattacharjee,a.gokhale}@vanderbilt.edu

***Index Terms*—Cloud Computing; Performancing Monitoring; Benchmarks; Performance Interference; Resource Management**

I. Introduction

Effective resource management is critical in multi-tenant, virtualized cloud platforms to meet service level objectives (SLOs) of individual applications. Thus, cloud providers must be able to detect sources of performance bottlenecks and reliability problems. One such cause, which is the focus of this study, is *Performance Interference*, where applications collocated on the same physical resource influence each others' performance in a non-linear fashion. To control the adverse consequences of such collocation-caused interference, different resource usage statistics, models of application's sensitivity to other collocated applications, and knowledge of workload patterns are required [1]. This challenge is further amplified when the resources span the spectrum from centralized cloud resources all the way to the edge.

Obtaining resource statistics without unduly consuming cloud platform resources is a hard problem for a variety of reasons including having to deal with different virtualization techniques, heterogeneity and advances in hardware and operating systems, changing application mix with differing SLOs, dynamic workloads, and the tight coupling of the installed statistics collection strategies to hardware-specific, low-level statistics collection APIs. These factors make it hard for providers to reuse and extend existing resource usage metric collection capabilities particularly when hardware changes, and applications and their structure changes (e.g., a move from traditional 3-tier to microservices-based architectures).

Although resource usage statistics collection tools such as collectd (https://collectd.org/) and benchmarking frameworks, such as PARSEC (http://parsec.cs.princeton.edu/), YCSB [2], CloudSuite [3] and BigDataBench [4] exist, their objective is not about inferring performance interference on multi-tenant heterogeneous servers, which is a key determinant of performance delivered to the hosted service. Although iBench [1] attempts to quantify the data-center performance interference, it provides only some of the building blocks thus making the users responsible to design and integrate the capabilities and deal with the complexities stemming from making sense out of the collected low-level raw usage data.

The desired resource monitoring framework must be able to collect micro architectural resource statistics such as context switches, page faults, cache utilization, retired instructions per second (IPS), memory bandwidth, scheduler wait time and scheduler I/O wait time, and utilization of last level cache (LLC). At the same time, not all the micro architectural details may be necessary for every use case, and hence the framework's micro benchmarking capabilities must be able to pinpoint the dominant micro architectural statistics thereby allowing the user to configure the framework to collect only the important statistics.

To address these requirements, this poster presents ongoing work on an extensible performance interference benchmarking and modeling framework that we are developing called *FECBench (Fog/Edge/Cloud Bench).*

II. FECBench Design and Preliminary Results

Figure 1 shows the FECBench architecture. FECBench's **usage statistics collection** framework exploits the plugin architecture of the underlying *collectd* monitoring tool. We have developed several plugins to collect micro-architectural metrics. Our design allows additional plugins to be added. FECBench can collect virtual machine (VM) and Docker container-specific metrics. For data collection at a centralized location, an InfluxDB time-series database is utilized. Beyond its resource monitoring capabilities, FECBench also

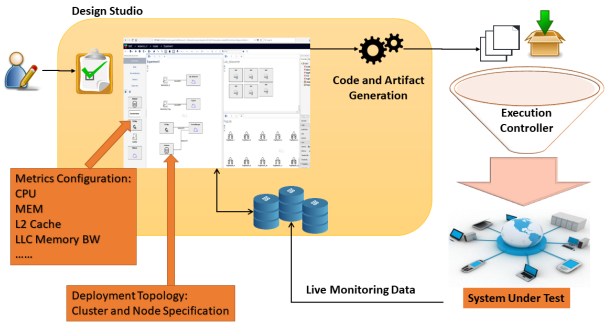

Fig. 1. FECBench Architecture

provides a **benchmarking component**. This consists of a number of latency-sensitive, client-server target applications, such as image processing, machine learning applications, web-search application, etc, which we have integrated from existing frameworks. In addition, there are workload applications that are used to cause performance interference on the target

applications. To make it intuitive to use FECBench, it supports the configuration of metrics collection and infrastructure provisioning using visual **domain specific modeling language (DSML)**. We have used DSMLs for deployment and configuration of software systems [5, 6, 7]. FECBench's **performance modeling** component, provides application performance modeling capabilities using different machine learning models. We have used FECBench for resource management across the cloud, fog, edge resource spectrum [8, 9].

Next, we describe two key building blocks of FECBench.

Performance Modeling using Surrogate Modeling: To build application performance models, we must first understand the cumulative effect of all the resources on the application's performance. This needs a strategy to stress different system resources such as CPU, memory bandwidth, L2 bandwidth, L3 bandwidth, context switches, network and disk I/O, and measure the application performance. Tools such as stress-ng, iperf, and bonnie are used to create resource stress for a single resource at different stress levels. However, they lack the ability to generate stress across different resources at user-specified levels. To address this difficulty, FECBench supports an extensible knowledge base of colocated applications that produces different stress levels on the system's resource. To our knowledge, presently there is no known benchmarking suite that can stress all the system resources from zero to 100 percent along different resource limits in parallel.

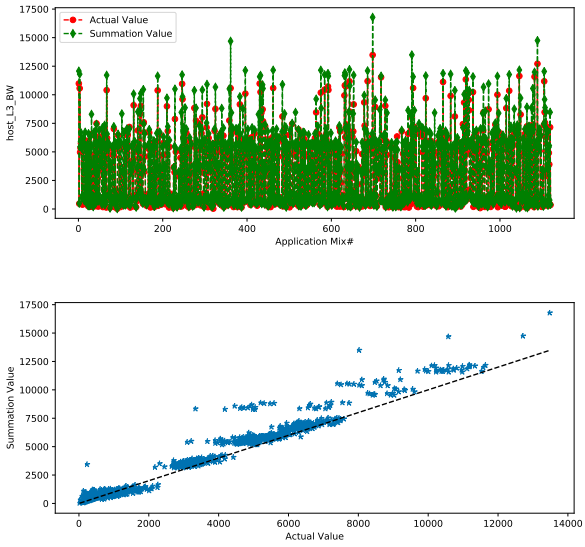

Fig. 2. Observed L3 Bandwidth utilization using different colocated application mix

We can obtain an application collocation combination that will produce the desired stress on the resources. Our hypothesis is that if we colocate applications, then the final resource utilization of the system will follow some weightage summation of the individual application resource utilization. To test this hypothesis, a three application colocation pattern is executed on an Intel Xeon E5-2620v3 processor. Figure 2 shows our initial findings indicating a near linear relationship between the summation of individual application resource utilization and the collocated applications' aggregated resource utilization.

Analysis and Visualization: We are building an automated pipeline for data analysis and visualization that will help performance engineers get deeper insights into application's performance characteristics. In our preliminary study, we focused on an image processing application as our target application. Due to server multi-tenancy, the executing background applications cause performance degradation of the target image processing application. The CDF response time of the job completion is shown in Figure 3. Our automated analysis pipeline is able to identify the dominant sources of interference in terms of micro architectural metrics as shown in the second part of Figure 3.

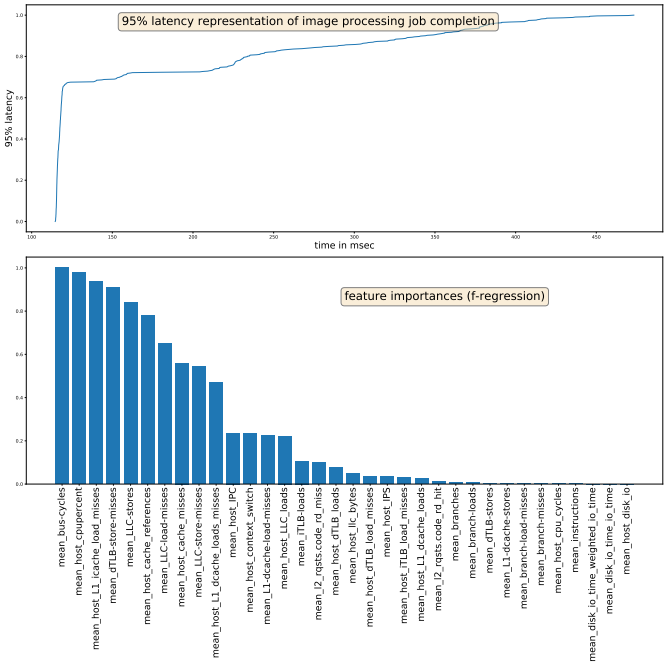

Fig. 3. CDF graphs showing Image processing latency variation and the identified dominant features that cause performance interference

Ongoing and Future Work: We are exploring an approach to find different sampling points that cover our design space across different resource dimensions using the *Design of Experiments (DOE)* method. For each such sampling point, using a heuristic algorithm, we can find that application combination which will stress the resources near the sampling values' limit. Using these sampling points, approximate surrogate models of application performance can then be built.

ACKNOWLEDGMENTS

This work is supported in part by NSF US Ignite 1531079, AFOSR DDDAS FA9550-18-1-0126 and NIST 70NANB17H274. Any opinions, findings, and conclusions or recommendations expressed here are those of the author(s) and do not necessarily reflect the views of NSF, AFOSR or NIST.

REFERENCES

[1] C. Delimitrou and C. Kozyrakis, "ibench: Quantifying interference for datacenter applications," in *Workload Characterization (IISWC), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 23–33.

[2] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *1st ACM symposium on Cloud computing*. ACM, 2010, pp. 143–154.

[3] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: a study of emerging scale-out workloads on modern hardware," in *ACM SIGPLAN Notices*, vol. 47, no. 4. ACM, 2012, pp. 37–48.

[4] L. Wang, J. Zhan, C. Luo, Y. Zhu, Q. Yang, Y. He, W. Gao, Z. Jia, Y. Shi, S. Zhang *et al.*, "Bigdatabench: A big data benchmark suite from internet services," in *High Performance Computer Architecture (HPCA), 2014 IEEE 20th International Symposium on*. IEEE, 2014, pp. 488–499.

[5] Y. D. Barve, P. Patil, A. Bhattacharjee, and A. Gokhale, "Pads: Design and implementation of a cloud-based, immersive learning environment for distributed systems algorithms," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 20–31, 2018.

[6] A. Bhattacharjee, Y. Barve, A. Gokhale, and T. Kuroda, "(wip) cloudcamp: Automating the deployment and management of cloud services," in *2018 IEEE International Conference on Services Computing (SCC)*. IEEE, 2018, pp. 237–240.

[7] Y. D. Barve, P. Patil, and A. Gokhale, "A cloud-based immersive learning environment for distributed systems algorithms," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 1. IEEE, 2016, pp. 754–763.

[8] S. Shekhar, A. D. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, "Indices: exploiting edge resources for performance-aware cloud-hosted services," in *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*. IEEE, 2017, pp. 75–80.

[9] S. Shekhar, Y. Barve, and A. Gokhale, "Understanding performance interference benchmarking and application profiling techniques for cloud-hosted latency-sensitive applications," in *Proceedings of the10th International Conference on Utility and Cloud Computing*. ACM, 2017, pp. 187–188.