Poster Abstract - Application-Agnostic Batch Workload Management in Cloud Environments

Seyyed Ahmad Javadi, Shalini Bhaskara, Rahul Doshi, Prashanth Soundarapandian, Muhammad Wajahat, Anshul Gandhi
Stony Brook University -{sjavadi, shbhaskara, radoshi, psoundarapan, mwajahat, anshul}@cs.stonybrook.edu

ABSTRACT

We present Scavenger, a reactive batch workload manager that opportunistically runs containerized batch jobs next to customer Virtual Machines (VMs) in a public cloud like setting to improve utilization. Scavenger dynamically regulates the resource usage of batch jobs, including CPU usage, memory capacity, and LLC capacity, to ensure that the customer VMs' resource demand is met at all times. We experimentally evaluate Scavenger and show that it considerably increases resource usage without compromising on the resource demand of customer VMs. Importantly, Scavenger does so without requiring any offline profiling or prior information about the customer workloads.

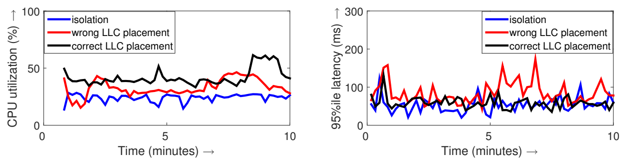
1 PROBLEM BACKGROUND

Servers in cloud data centers often have low resource utilization. A study focused on Amazon EC2 observed that cloud server usage is often below 10% [5]. To increase server utilization, prior works have proposed running provider workloads, such as Hadoop or Spark batch jobs, next to customer VMs to leverage idle resources [1, 3]. While effective, the key challenge with this approach is *interference* – the performance degradation of the colocated customer VMs due to resource contention with batch workloads at the underlying host server. This interference can be caused by contention for several resources simultaneously [4].

Ideally, in a public cloud, provider (or *background (bg)*) workloads should run next to customer (or *foreground (fg)*) workloads or VMs in such a way that their resource utilization complements that of the customer VMs. In particular, the dynamic demand of the customer VMs, across all resources, should be met at all times and the bg workloads should consume the remaining resources to do useful work. Thus, the goal is to maximize resource usage and bg workload throughput in a public cloud while satisfying the customer VM workloads' resource demands at all times.

While there has been considerable prior work in this important area, there are still several shortcomings that must be addressed. Existing solutions often either rely on historical usage patterns to predict the resource demand of fg VMs [6] or benchmark customer VM performance to carefully colocate bg workloads [2]; such solutions cannot always be deployed in public clouds where customer VMs should not be instrumented and there is often significant variation in VM loads [3, 4].

SoCC '18, October 11–13, 2018, Carlsbad, CA, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6011-1/18/10.
https://doi.org/10.1145/3267809.3275446



(a) Average CPU utilization of PMs. (b) 95%ile latency for fg (Pinot).

Figure 1: Colocation impact under CPU and LLC regulation

2 SCAVENGER DESIGN

Scavenger currently handles resource management of CPU, Memory Capacity (MemCap), and Last-Level-Cache (LLC). Scavenger leverages cgroups for CPU resource management. For LLC management, we use a threshold-based approach to estimate the cache sensitivity of fg workloads, thus informing the colocation of bg workloads on specific processor sockets. For MemCap management, we monitor the memory usage of fg workloads and reactively scale (up or down) the memory consumption of our batch job containers.

We implement Scavenger on top of KVM, Docker, and YARN. Our experimental results on a multi-server testbed show that Scavenger can dynamically allocate resources to batch jobs while having negligible effect on fg workload latencies. We show that while CPU resource regulation does not suffice by itself to address contention, when combined with LLC regulation, we can significantly improve resource utilization. For instance, Figure 1(a) shows the average CPU utilization across 4 host servers when running Pinot (popular OLAP system in use at LinkedIn and Uber) in isolation (blue), in colocation but with wrong placement of bg workloads (red), and with the correct colocation placement via Scavenger (black). We see that the increase in CPU utilization under Scavenger is considerably higher than under the wrong LLC placement. Figure 1(b) shows the impact of colocation on the 95%ile latency of Pinot. With the correct placement under Scavenger, the latency increases by about 10.6%, whereas under the wrong placement, the latency increases by about 59.9%. This shows that careful LLC regulation, as under Scavenger, can considerably increase the resource utilization under colocation without significantly impacting the performance of fg VMs.

REFERENCES

[1] A. Goder et. al. 2015. Bistro: Scheduling Data-parallel Jobs Against Live Production Systems. In *Proceedings of USENIX ATC*. Santa Clara, CA, USA, 459–471.
[2] C. Delimitrou et. al. 2013. Paragon: QoS-aware Scheduling for Heterogeneous Datacenters. In *Proceedings ASPLOS*. Houston, TX, USA, 77–88.
[3] D. lo et. al. 2015. Heracles: Improving Resource Efficiency at Scale. In *Proceedings of ISCA*. Portland, OR, USA, 450–462.
[4] S. A. Javadi et. al. 2017. DIAL: Reducing Tail Latencies for Cloud Applications via Dynamic Interference-aware Load Balancing. In *Proceedings of ICAC*. Columbus, OH, USA, 135–144.
[5] H. Liu. 2011. A Measurement Study of Server Utilization in Public Clouds. In *Proceedings of DASC*. Sydney, Australia, 435–442.
[6] Y. Zhang et. al. 2016. History-based Harvesting of Spare Cycles and Storage in Large-scale Datacenters. In *Proceedings of USENIX OSDI*. 755–770.