FID3008 Advanced Topics in Distributed Systems

Review of Presented Papers

David Gureya

KTH Royal Institute of Technology
daharewa@kth.se

This report represents the review of the papers presented in the *Advanced Topics in Distributed Systems* course.

1 Selected Papers

1. J. Park, S. Park, and W. Baek. *CoPart: Coordinated partitioning of last-level cache and memory bandwidth for fairness-aware workload consolidation on commodity servers.* EuroSys'19.
2. S. Chen, C. Delimitrou, and F. Martínez. *PARTIES: Qos-aware resource partitioning for multiple interactive services.* ASPLOS'19.
3. X. Wang, S. Chen, J. Setter and J. F. Martínez. *SWAP: Effective Fine-Grain Management of Shared Last-Level Caches with Minimum Hardware Support.* HPCA'17

2 Justification

The business models for cloud and datacentre computing emphasizes on reducing infrastructure costs. *Cost efficiency* is typically achieved through *workload consolidation*, i.e., by co-locating workloads from multiple users on the same physical host to maximize resource utilization. Despite its benefits, workload consolidation poses critical challenges. The main challenge is the performance interference among the consolidated workloads, which is mainly caused by the contention over shared hardware resources. Many resources within modern multi-core platforms used in datacentres are shared including network, CPUs, last-level caches (LLC), memory bandwidth (BW), etc.

To prevent the performance interference among the consolidated workloads, the importance of efficient partitioning techniques for shared hardware resources is expeditiously increasing. Recent commodity CPUs have introduced hardware support for managing shared resources. The chosen papers incentivize and provide a general guide to implement robust policies to achieve maximum resource utilization by effectively employing the resource partitioning mechanisms provided by the hardware.

3 Introduction

Prior work has extensively explored architectural and system software techniques to tackle interference in a multi-tenant environment. These techniques can be grouped into three broad approaches. The first approach is to simply avoid sharing resources with performance sensitive applications. This preserves the quality of service (QoS) of the latency critical (LC) applications, but lowers the resource efficiency of the underlying system. The second approach is to avoid co-scheduling of applications that may interfere with each other. Although this approach improves resource utilization, it limits the options of applications that can be co-scheduled and may require some offline/prior knowledge.

Finally, interference can be eliminated altogether by partitioning resources among consolidated applications. This approach (1) maximize resource utilization and system throughput, or trade off throughput vs. fairness; (2) provide QoS for LC applications; (3) protect the system from timing channel attacks, where a malicious program can still secure information, such as encryption keys, by sharing LLC. All the three papers fall under this approach.

4 CoPart

4.1 Motivation

While prior work has investigated various resource partitioning techniques, coordinated partitioning of LLC and memory BW in commodity servers is still unexplored. To bridge this gap, the authors proposed CoPart, coordinated partitioning of LLC and memory bandwidth for *fairness-aware* workload consolidation on commodity servers. CoPart dynamically analyses the application characteristics and partitions LLC and memory BW in a coordinated manner.

4.2 Contribution

CoPart relies on recent commodity CPUs that have introduced hardware support for managing shared hardware resources. As part of its latest Xeon scalable processors, Intel has released *Intel Resource Director Technology* (RDT), a framework that monitors and manages the shared LLC and memory BW. RDT includes *Cache Allocation Technology* (CAT), a hardware support for LLC partitioning which is based on *way partitioning*. With way partitioning, the cache is partitioned "vertically" in different ways and different cache ways are assigned to different cores. This achieves perfect isolation and low re-partition overhead. For memory BW, RDT provides a similar technology called *Memory Bandwidth Allocation* (MBA). MBA is a per-core mechanism that controls the traffic between the L2 cache and the LLC. CoPart implements a robust policy that achieves the overall fairness by effectively employing the LLC and memory BW partitioning mechanisms provided by the hardware (i.e., CAT and MBA).

4.3 Solution

The resource allocation problem of CoPart was formulated as the *Hospitals/Residents* (HR) problem, which is one of the most extensively-studied and widely-applied problem in economics. In HR problem, they are H hospitals and R medical students with preference lists. Each hospital admits a certain number of students as residents and each student applies to a certain number of hospitals. The objective is to find a stable match that contains no blocking pairs. To formulate resource allocation problem as the HR problem, CoPart considers the resource types (LLC, memory BW) to be supplied as the Hospitals and the demanding applications (consumers) as the medical students. The objective is to find a stable match between the resources to be supplied and the demanding applications.

4.4 Evaluation

Regarding the experiments and evaluation, the authors generated various workload mix with different characteristics and compared CoPart with different resource allocation policies.

5 PARTIES

5.1 Motivation

PARTIES partitions shared resources considering a different problem. Multitenancy in current datacentres is limited to running single LC interactive service alongside a number of Best effort (BE) jobs. However, cloud applications are progressively shifting from batch to low-latency services. Furthermore, cloud applications are undergoing a major redesign from large, monolithic services (that encompass an entire application) to hundred or thousand of loosely-coupled micro-services, where each of this micro-service must meet much tighter latency constrains.

5.2 Contribution

Just like CoPart, PARTIES also characterized the sensitivity of various LC services to different resource allocations and showed that resource isolation is essential. Based on the characterization results, the authors proposed PARTIES, the first *QoS-aware* resource manager for co-location of many LC services. Unlike CoPart, this paper provides dynamic partitioning of nine shared resources (*Hyperthread, CPU, Power, LLC Capacity, LLC Bandwidth, Memory Bandwidth, Memory Capacity, Disk Bandwidth and Network Bandwidth*). PARTIES also relies on various hardware and software isolation mechanism per resource to minimize interference among the consolidated applications.

5.3 Solution

The key to the effectiveness of PARTIES is the observation that resources are *fungible*, i.e., they can be traded for each other. This reduces the time Parties needs to find an allocation that satisfies QoS for two reasons: More flexibility in resource allocation and a simplified resource manager, as it is sufficient to find *one* satisfactory resource allocation. PARTIES consist of two components - monitoring and resource allocation. It uses monitoring data to determine appropriate resource allocations and enforces them using the existing isolation mechanisms. More specifically, PARTIES dynamically adjusts resource allocations between consolidated LC applications using fine-grained monitoring and resource partitioning, with the objective to meet all applications' QoS constraints.

5.4 Evaluation

Since their experimental platform did not support MBA, PARTIES used Intel CAT to indirectly regulate memory BW. As shown in the CoPart paper, different applications have different sensitivity to LLC and memory BW, hence this approximation may not work for every application (i.e., a coordinated partitioning of LLC and memory BW is needed). PARTIES was compared to three baselines: (1) Oracle – offline profiling, always finds a global optimum; (2) Heracles – a system from google, tailored for one LC application and many BE applications; (3) Unmanaged - no isolation mechanisms are used.

6 SWAP

6.1 Motivation

The SWAP paper exclusively focuses on the LLC shared resource. It addresses one of the key shortcomings of cache way partitioning. The granularity of way partitioning is too coarse to allow for separate partitions for more than a handful of applications. For instance, Intel's v4 multiprocessors allow for no more than 20 different partitions across 22 cores. Another approach to achieving cache partitioning is *page coloring*. Unlike way partitioning, page coloring partitions the cache "horizontally" by sets and assigns different cache sets to different cores. Page coloring is an OS-level technique and although it also achieves perfect isolation, the re-partitioning overhead is high, since pages have to be migrated from the old to new color. Similarly, page coloring also does not scale beyond a handful of colors.

6.2 Contribution

To address the scalability challenge, the authors proposed SWAP, a fine-grained cache partitioning mechanism that combines the cache way (hardware) and set (OS) partitioning. SWAP is able to divide the LLC into literally hundreds of regions, providing sufficiently fine granularity for the upcoming many-core processors.

6.3 Solution

SWAP divides the cache in a two-dimensional manner. Way partitioning divides the cache vertically, while page coloring divides the cache horizontally. For instance, in their experimental platform, the number of cache ways and possible page colors was 16 each. Therefore, the shared LLC can be partitioned into 256 independent regions. Although combining set and way partitioning to enable fine-grained cache partitions may be intuitive, in practice the authors have to address several important challenges to make it practical. These challenges include: What's the shape of the partition? How are partitions placed with each other? and how to minimize re-partition overhead?

6.4 Evaluation

The experimental results show that SWAP improves system throughput (weighted speedup) by $13.9\%, 14.1\%, 12.5\%$ and 12.5% on average for 16-, 24-, 32- and 48-application bundles with respect to unmanaged cache. This represent $2\times$ speedup compared to using only way partitioning.

7 Conclusion

Overall I thought these were good papers. They were obviously well motivated, very clearly written, and experiments show significant improvements over baselines. The main contributions are more from the combination of existing ideas than from new ideas. All the three papers also had a clear problem statement with solid evidence that existing approaches do not work well for relevant scenarios.