Towards Interference-aware Application Co-locations

Ioannis A. Papadakis
ypap@cslab.ece.ntua.gr
National Technical University of Athens
Athens, Greece

Nectarios Koziris
nkoziris@cslab.ece.ntua.gr
National Technical University of Athens
Athens, Greece

Georgios Goumas
goumas@cslab.ece.ntua.gr
National Technical University of Athens
Athens, Greece

Abstract

Interference among co-located applications is a major concern among Cloud Service Providers that aim to maximize the utilization of their resources, while maintaining the quality of service for their tenants. Prior work relies on modelling techniques that predict interference and support co-location decisions, but fails to provide a practical solution that can accommodate co-locations beyond pairwise placements, avoid per-application offline profiling and seamlessly incorporate new, unseen applications.

In this work, we propose a prediction model that can drive multi-wise co-locations of applications within a CPU socket through the use of unsupervised learning techniques. By clustering applications based on resource access patterns and leveraging co-execution results, our approach constructs inferred heatmaps that facilitate co-location decisions across multiple degrees.

CCS Concepts: • **Computer systems organization** → **Cloud computing**; Processors and memory architectures; • **General and reference** → *Performance.*

Keywords: cloud computing, clustering, unsupervised learning

1 Introduction

Cloud computing is a dominant paradigm, relying on large-scale data centers that entail significant acquisition and operational costs while offering substantial computing capabilities. Maximizing resource utilization by co-locating workloads is essential for both profitability and user satisfaction. However, the architecture of modern servers presents significant challenges in resource allocation, since they incorporate multiple cores and share resources in the memory hierarchy.

The main goal of a Cloud Service Provider (CSP) is to co-locate as many applications as possible on every available CPU socket to maximize resource utilization. Consequently, limiting co-locations to pairwise placements fails to fully exploit the capabilities of a compute node. However, the shared nature of the resources (CPU, memory channels and network interfaces) can cause degraded and unpredictable performance [1, 8], potentially leading to Service Level Objective (SLO) violations.

Despite the extensive prior work on this field [1, 2, 5, 7, 8], we still lack a practical solution for effective prediction of the performance of applications in a co-location scenario. A contention-unaware approach overlooks the interference patterns between co-located applications, increasing the likelihood of SLO violations. Existing state-of-the-art approaches rely on per-application offline profiling, fail to support the co-location of multiple applications within a single node, or require extensive retraining to handle unseen applications.

In this work, we propose a proactive model for application characterization that relies on unsupervised machine learning methods. Specifically, this model (a) can drive co-location decisions of several co-location degrees beyond pairwise placements, (b) can be deployed online, since it requires solely the collection of resource utilization metrics from isolated executions, avoiding per-application offline profiling and (c) can trivially handle unseen applications without the necessity for extensive retraining.

2 Background

Aligned with prior research on the field [1, 2, 5, 6] we consider that the set of SLOs includes the deviation of the application performance when executed in a shared resource environment, compared to the performance achieved when executed in isolation. The performance indicator depends on the nature of each application and can vary among throughput, tail latency, execution time and others. We use *slowdown* as Service-Level Indication (SLI), defined as follows:

$$slowdown = \frac{performance_{isolation}}{performance_{co-execution}}$$

Prior research on avoiding interference in co-location scenarios through characterization of applications can be categorized into two approaches. Several works focus on on predicting the performance degradation of applications in co-location scenarios [7, 8], while others classify applications based on their resource usage and their likelihood to cause or suffer from interference [2, 5].

A brute force model for slowdown prediction in a scenario of k co-located applications would look up the slowdown of each application in a k-dimensional matrix, called heatmap, derived by offline executions of the same co-locations. However, this approach is unrealistic since it requires N^k offline co-executions (N: number of applications hosted in the data center) for every degree of co-location and would require extensive profiling to accommodate unseen applications.

Zacarias et al. [8] proposed a regression model that aims to converge to the brute-force model with pairwise co-locations by predicting the performance degradation of each application in a co-location scenario. The characterization requires

the collection of performance metrics in an interference-free environment, enabling the immediate deployment of each application upon arrival. However, the information from these predictions fails to provide insight on multi-wise co-locations, as the pairwise results cannot be trivially generalized to a higher co-location degree.

In Pythia [7], Xu et al. accommodate multi-wise co-locations by extending pairwise approaches with models that predict the degradation of each application in these scenarios. They also assert the inaccuracy of simplistic additive models when assessing slowdown in multi-wise co-locations. However, the need for extensive retraining of the entire model when encountering an unseen application limits its practicality for deployment in real-world production environments.

Other works [2, 5] classify applications by co-executing them with several micro-benchmarks of tunable intensity. While such models can accommodate unseen applications, the co-execution with these micro-benchmarks necessitates a profiling step before the deployment of the application, since the risk for SLO violations is high and inherent in the methodology.

3 Our approach

In this work, we demonstrate a model for application characterization that leverages unsupervised machine learning techniques to perform interference-aware application co-locations. The model employs clustering and matrix completion methods, in the attempt to approximate the ideal brute force model, while avoiding its unrealistic and computationally expensive pre-deployment phase.

Given a set of N applications that are expected to be hosted in the CSP premises, the initial step is the collection of resource utilization metrics for each application in an isolated environment, such as a single CPU socket without co-located tenants. These metrics, obtained through performance counters, include CPU utilization, Instructions per Cycle, cache misses at all levels, Last-Level Cache occupancy, memory bandwidth usage, TLB misses, and incoming and outgoing network traffic. These metrics capture the behavior and access patterns of the application, and can be collected while the application is running, due to the low risk of SLO violations, eliminating the need for offline profiling.

After collecting the resource utilization metrics, a clustering method is applied to group applications with similar resource access patterns. Among various clustering techniques (KMeans clustering, Hierarchical Clustering or Gaussian Mixture), we choose Correlation Clustering with incremental updates [3, 4] due to its ability to incorporate new records without requiring extensive retraining. This approach allows for seamless integration of previously unseen applications.

The quality of the clustering results is dependent on the similarity between the members of the same cluster and the distance between the clusters, and can be measured through different scoring methods that include Silhouette score, Davies-Bouldin index and Calinski-Harabasz index.

The number of clusters that the clustering method yields, indicates the number of distinct groups of applications with similar resource access behavior. Supposing that the applications have been grouped into n clusters, we can choose one application per cluster as its representative. Using these n representative applications, we construct k heatmaps, one for each degree of co-location. Each element $i_1, i_2, ..., i_k$ of a heatmap indicates the slowdown of application i_1 when co-located with applications $i_2, ..., i_k$.

The required number of co-executions has a cost of $O(n^k)$. Although this approach is significantly cheaper than evaluating all N applications of the CSP, which would incur a cost of $O(N^k)$, it remains computationally expensive and grows exponentially with n. To further mitigate this cost, we employ matrix completion techniques, utilizing either Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) to estimate missing values within a matrix. Specifically, only a subset of the heatmap is executed, while the remaining values are inferred through matrix completion. This approach is expected to significantly reduce the cost of heatmap construction up to an order of magnitude.

References

[1] Christina Delimitrou and Christos Kozyrakis. 2013. Paragon: QoS-aware scheduling for heterogeneous datacenters. In *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS (ASPLOS '13)*. ACM, New York, NY, USA, 77–88. https://doi.org/10.1145/2451116.2451125

[2] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: Resource-efficient and QoS-aware cluster management. In *International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS (ASPLOS '14)*. ACM, New York, NY, USA, 127–143. https://doi.org/10.1145/2541940.2541941

[3] Micha Elsner and Warren Schudy. 2009. Bounding and Comparing Methods for Correlation Clustering Beyond ILP. (2009), 19–27.

[4] Anja Gruenheid, Eth Zurich, Xin Luna Dong, and Divesh Srivastava. 2014. Incremental Record Linkage. (2014).

[5] Shivam Kundan, Theodoros Marinakis, Iraklis Anagnostopoulos, and Dimitri Kagaris. 2022. A Pressure-Aware Policy for Contention Minimization on Multicore Systems. *ACM Transactions on Architecture and Code Optimization* 19 (2022), 1–26. Issue 3. https://doi.org/10.1145/3524616

[6] Daniel Mendoza, Francisco Romero, Qian Li, Neeraja J Yadwadkar, and Christos Kozyrakis. 2021. Interference-Aware Scheduling for Inference Serving. *EuroMLSys* 21 (2021). https://doi.org/10.1145/3437984.3458837

[7] Ran Xu, Subrata Mitra, Jason Rahman, Peter Bai, Bowen Zhou, Greg Bronevetsky, and Saurabh Bagchi. 2018. Pythia: Improving datacenter utilization via precise contention prediction for multiple co-located workloads. In *Proceedings of the 19th International Middleware Conference, Middleware 2018*. Association for Computing Machinery, Inc, New York, New York, USA, 146–160. https://doi.org/10.1145/3274808.3274820

[8] Felippe Vieira Zacarias, Vinicius Petrucci, Rajiv Nishtala, Paul Carpenter, and Daniel Mossé. 2021. Intelligent colocation of HPC workloads. *J. Parallel and Distrib. Comput.* 151 (may 2021), 125–137. https://doi.org/10.1016/j.jpdc.2021.02.010