PANDA: NOISE-RESILIENT ANTAGONIST IDENTIFICATION IN PRODUCTION DATACENTERS

Sixiang Zhou [1] **Nan Deng** [2] **Krzysiek Rządca** [2] **Xiaojun Lin** [1] **Y. Charlie Hu** [1]

[1]Purdue University [2]Google Inc.

ABSTRACT

Modern warehouse-scale datacenters commonly collocate multiple jobs on shared machines to improve resource utilization. However, such collocation often leads to performance interference caused by antagonistic jobs that overconsume shared resources. Existing antagonist-detection approaches either rely on offline profiling, which is costly and unscalable, or use a sample-from-production approach, which suffers from noisy measurements and fails under multi-victim scenarios. We present PANDA, a noise-resilient antagonist identification framework for production-scale datacenters. Like prior correlation-based methods, PANDA uses cycles per instruction (CPI) as its performance metric, but it differs by (i) leveraging global historical knowledge across all machines to suppress sampling noise and (ii) introducing a machine-level CPI metric that captures shared-resource contention among multiple co-located tasks. Evaluation on a recent Google production trace shows that PANDA ranks true antagonists far more accurately than prior methods—improving average suspicion percentile from 50–55% to 82.6%—and achieves consistent antagonist identification under multi-victim scenarios, all with negligible runtime overhead.

1 INTRODUCTION

Modern warehouse-scale datacenters improve resource utilization by collocating multiple jobs from different users on the same machine (Verma et al., 2015; Garefalakis et al., 2018). However, such collocation often incurs performance interference, as some jobs aggressively contend for shared resources and degrade others' performance. This interference is especially detrimental to latency-sensitive (LS) jobs—such as user-facing services—that must satisfy strict quality-of-service (QoS) guarantees. Therefore, to maximize the benefits of collocation without compromising performance, it is crucial to identify antagonistic jobs that over-consume shared resources and harm their co-runners.

A widely adopted approach in prior work for identifying antagonists is *offline screening* (Delimitrou and Kozyrakis, 2013; 2014; Zhang et al., 2014; Yang et al., 2013; Mars et al., 2011). This method builds dedicated test environments to measure detailed interference metrics by co-running each job with others on shared machines. Every job is profiled in this controlled setting before being admitted into the datacenter job pool. With such fine-grained profiling, offline approaches can accurately identify antagonistic behaviors and proactively mitigate interference through scheduling. However, such an approach suffers limited scalability: modern datacenters may host millions of jobs, making exhaustive profiling impractical. Furthermore, job behaviors often change dynamically over time, necessitating repeated re-profiling and exacerbating the scalability challenge.

Another common strategy is to sample performance data directly from production machines and identify antagonists through correlation analysis (Zhang et al., 2013; Kannan et al., 2018; Wang et al., 2022). We refer to this as the *sample-from-production* approach. Most existing studies rely on hardware performance counters, particularly cycles per instruction (CPI), as the key metric. This choice is motivated by two reasons: (i) CPI can be collected using existing hardware counters with minimal software modifications, and (ii) it closely reflects the application-level behavior of latency-sensitive (LS) jobs. Once victims (*i.e.,* LS tasks exhibiting abnormal CPI values) are detected, prior work (Zhang et al., 2013; Kannan et al., 2018; Wang et al., 2022) identifies their corresponding antagonists among colocated tasks, typically through CPU-usage-based correlation analysis.

However, the sample-from-production approach suffers severely from *sampling noise* in the collected performance data. The sampling program is typically software-based and runs as a co-located job alongside production workloads. To minimize its interference with production performance, it must use a small sampling window, which in turn introduces significant measurement noise. For example, our twin-task analysis on CPI samples from a production trace (Google,

2019) reveals substantial variability across identical tasks (see detailed analysis in Section 2). Similar issues have been reported by Proctor et al. (Kannan et al., 2018), who describe this as a distorted performance metric. Such noise can cause false victim alarms and, more critically, incorrect antagonist identification. Although prior works (Zhang et al., 2013; Kannan et al., 2018; Wang et al., 2022) address the false-alarm issue through various design choices, none effectively mitigate the misidentification of antagonists.

In addition, identifying antagonist in modern datacenter also faces a *multi-victim challenge*. As hardware capacity continues to increase, the number of colocated tasks per machine has grown rapidly. For example, (Zhang et al., 2013; Kannan et al., 2018) report an average of around ten colocated tasks per machine, whereas recent production traces (Google, 2019) show machines hosting 50–100 tasks simultaneously. Since these tasks share the same hardware resources, contention can create *global interference effects*, causing multiple victim tasks to appear simultaneously. However, these victims may be affected by shared contention to different degrees. As a result, analyzing each victim independently, as done in (Zhang et al., 2013; Kannan et al., 2018), can lead to inconsistent antagonist identification and, consequently, confusion in diagnosis. An analysis of a production trace (Google, 2019) further illustrates this issue: among all machines that contain at least one victim, 43% host multiple victims, and reapplying the method of (Zhang et al., 2013) to any two colocated victims yields different identified antagonists in 69% of the cases.

To tackle both the noise problem and the multi-victim problem, we propose PANDA (**P**roduction **AN**tagonist **D**etection and **A**nalysis), a noise-resilient antagonist identification framework for production-scale datacenters. Like prior correlation-based approaches (Zhang et al., 2013; Kannan et al., 2018), PANDA uses cycles per instruction (CPI) as its primary performance metric, but it differs fundamentally in scope and robustness. Instead of inferring *task-level* antagonists locally—based only on short-term observations from the same machine that the victim resides— PANDA aggregates information globally across all machines and time windows, and performs analysis at the *job level*. By leveraging this global knowledge, PANDA produces more stable and accurate inferences, substantially mitigating the effects of sampling noise.

Achieving such large-scale, job-level aggregation, however, introduces new scalability challenges. To address the multi-victim problem and the scalability challenge, PANDA introduces a *machine-level CPI* metric that captures the overall contention level of shared hardware resources, enabling coordinated analysis when interference simultaneously affects multiple colocated tasks.

We evaluate PANDA using the large-scale Google cluster trace (v3) (Google, 2019), which contains roughly 1 M tasks executed across 80 K production machines. The trace provides detailed per-task resource-usage and performance statistics, enabling realistic assessment of interference behaviors in real-world datacenters.

Our evaluation examines two questions: (i) when true antagonists colocate with victims, how effectively does PANDA rank them compared to prior correlation-based methods (Zhang et al., 2013; Kannan et al., 2018)? (ii) how consistent are its antagonist predictions in multi-victim scenarios?

Across both dimensions, PANDA delivers substantially more accurate detection. PANDA assigns true antagonists an average suspicion rank of 82.6 percentile, compared to only 50–55 percentile for prior approaches, demonstrating the benefit of PANDA 's global view. In addition, PANDA achieves full consistency in multi-victim settings, whereas prior methods often disagree across victims. Finally, PANDA incurs negligible runtime overhead, making it practical for production-scale deployment.

2 MOTIVATION

To fully utilize hardware resources, modern production datacenters colocate multiple tasks on shared machines, which inevitably leads to colocation interference (Lu et al., 2023; Verma et al., 2015; Garefalakis et al., 2018). Such interference can cause significant performance degradation, especially for latency-sensitive (LS) workloads that must meet strict quality-of-service requirements. The key to mitigating this problem is to accurately identify antagonists—tasks that overuse shared resources and cause other tasks to suffer. Precisely identifying antagonists allows datacenter operators to swiftly control interference sources and maintain service quality.

2.1 Limitations of Prior Approaches

Accurately identifying antagonists in production datacenters remains challenging. Existing approaches can be broadly classified into two categories: the offline-screening approach and the sample-from-production approach.

The *offline-screening* approach (Zhang et al., 2014; Yang et al., 2013; Mars et al., 2011; Delimitrou and Kozyrakis, 2013; 2014) builds dedicated testing environments where tasks are executed and profiled individually to evaluate their behavior on shared resources. Through comprehensive task screening and profiling, these methods enable schedulers to make informed placement decisions and proactively avoid colocation interference. However, modern datacenters host millions of jobs (Google, 2019), making such exhaustive profiling prohibitively expensive. Moreover, because task behavior can change dynamically over time, offline

approaches require periodic re-profiling, which further exacerbates their scalability limitations.

The *sample-from-production* approach (Zhang et al., 2013; Wang et al., 2022; Kannan et al., 2018) identifies antagonists through correlation analysis on real-time performance samples collected from production machines. When a victim task—one suffering from colocation interference and exhibiting abnormal behavior—is detected, the system gathers both its performance metrics and the resource usage statistics of colocated tasks, then computes correlation scores to infer which task is the likely antagonist.

These methods typically rely on performance counter (PC) metrics, such as cycles per instruction (CPI), because they can be obtained using existing hardware with minimal software changes. Prior work (Zhang et al., 2013) shows that CPI correlates well with the application-level performance of latency-sensitive (LS) workloads, making it an attractive choice for interference detection.

Among online approaches, (Wang et al., 2022) collects detailed cache metrics and constructs a contention graph across multiple cache layers. While this design improves visibility into cache interference, it depends on specialized cache-monitoring hardware, making deployment across an entire datacenter costly. Moreover, it focuses exclusively on cache contention, ignoring other potential bottlenecks such as memory bandwidth, I/O, and network.

In contrast, (Zhang et al., 2013; Kannan et al., 2018) infer antagonists directly from correlation between the victim's performance degradation and the colocated tasks' resource usage, assuming that antagonists' usage patterns fluctuate in tandem with victims' performance. However, relying solely on CPI and local correlation introduces significant vulnerability to noise: transient fluctuations in performance counters can easily distort correlation results and lead to false or inconsistent antagonist identification. Further, partial observability and multi-task contention also blur these correlations. Below, we use real-world datacenter traces to explain why the sample-from-production approach remains fragile and prone to both false positives and incorrect antagonist attribution.

2.2 Challenges in Production Environments

The noise problem. When sampling performance counter (PC) metrics on production machines, it is necessary to prevent the monitoring program from interfering with production workloads. To minimize its impact, the sampling window for performance metrics is typically very short. For example, in a production trace (Google, 2019), the reported CPI value represents the average over a 10-second window within every 5-minute interval. As a result, the collected samples contain substantial measurement noise, which can trigger false victim alarms and lead to incorrect antagonist identification. Similar issues were also reported by Proctor et al. (Kannan et al., 2018), who observed corrupted PC samples in their dataset. To further quantify the extent of this noise, we perform a twin-task analysis on the production trace (Google, 2019).

Twin-task analysis. We observe that several machines in (Google, 2019) host two or more instances of the same job running concurrently. These task instances share identical binaries, CPU and memory allocations, and colocated neighborhoods. We refer to any such pair as *twin tasks*. In the absence of noise, the performance-counter samples of twin tasks should be nearly identical, since they execute under the same conditions. However, our analysis reveals otherwise.

For each job, we compute the CPI sample difference between its twin tasks and compare it against the CPI sample difference between two randomly selected tasks of the same job that run on different machines and at different times (hereafter referred to as *random tasks*). Although the CPI differences of twin tasks are generally more concentrated than those of random tasks, they still exhibit substantial variation. Across all jobs with twin-task occurrences, the CPI difference of twin tasks has, on average, still $0.59\times$ the standard deviation observed for random tasks of the same job. Figure 1 shows the probability density functions (PDFs) of CPI differences for twin tasks and random tasks among the four jobs with the most twin-task occurrences, corroborating this observation.

In summary, while switching from random tasks to twin tasks reduces part of the variation—reflecting the true interference signal—the remaining variation due to noise remains dominant. This indicates that the signal-to-noise ratio in CPI-based inference is low.

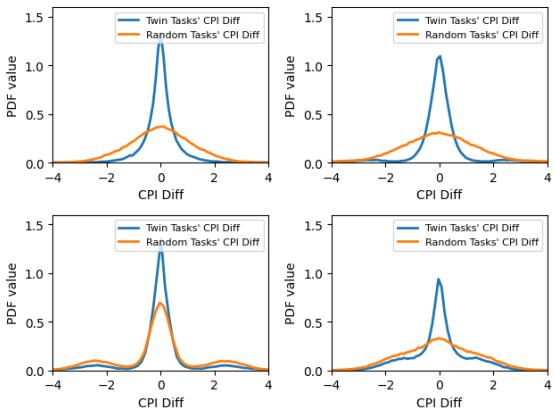

Figure 1: PDF of the absolute CPI sample difference between twin tasks and between random task pairs, shown for the four jobs with the highest number of twin-task occurrences.

The noise problem gives rise to the *incorrect antagonist identification* issue. Neither (Zhang et al., 2013) nor (Kannan et al., 2018) addresses the impact of noise on antagonist identification, and as a result, the inferred antagonists may be incorrect. Misidentifying antagonists can have serious operational consequences. The standard recovery procedure (Zhang et al., 2013; Kannan et al., 2018) involves iteratively identifying the suspected antagonist, evicting it, and then observing whether the victim's performance improves. If not, the process repeats with a new suspected antagonist. When noise leads to incorrect identification, this procedure may evict multiple innocent tasks, prolong the victim's degradation, and severely disrupt normal datacenter operations.

The multi-victim problem. Identifying antagonists for victim tasks in modern datacenters is further complicated by the frequent presence of *multiple victims*, which can lead to conflicting inferences. With the rapid growth of hardware capacity, machines now host far more concurrent tasks than before. For example, prior work (Zhang et al., 2013; Kannan et al., 2018) reports an average of around ten colocated tasks per machine, whereas recent production traces (Google, 2019) show machines hosting 50–100 tasks simultaneously.

As a result of this increased degree of task colocation, victims tend to appear in clusters: the presence of one victim often indicates aggressive shared-resource contention that can degrade multiple colocated tasks. Indeed, an analysis of (Google, 2019) shows that colocated tasks exhibit roughly $0.7\times$ the standard deviation of CPI values observed between randomly paired tasks, suggesting a shared interference effect. Consequently, multiple victims commonly arise on the same machine, making per-task analysis both computationally expensive and unreliable.

To further illustrate this challenge, we replicate the correlation-based antagonist identification method proposed in (Zhang et al., 2013) using the dataset (Google, 2019). Specifically, we identify latency-sensitive (LS) victim jobs and compute the CPU-usage–CPI correlation across all colocated tasks within the previous hour. Tasks with the highest correlation are considered potential antagonists, while *victims* are defined as jobs whose CPI exceeds the mean by more than two standard deviations. The results are revealing:

1. Among all per-machine, per-5-minute records containing at least one victim, 43% include multiple victims;
2. For any two colocated victims, there is a 69% probability that they accuse different antagonists.

These findings highlight a fundamental limitation of existing correlation-based antagonist identification approaches: when multiple victims coexist, noise and shared contention frequently lead to inconsistent antagonist attribution, rendering such methods unreliable in large-scale production environments.

In summary, existing approaches on antagonist identification suffer from several limitations, motivating the need for a new solution that meets the practical requirements of production-scale datacenters. Specifically, our goal is to design an approach that:

1. Scales efficiently and can be deployed seamlessly in modern production environments.
2. Robustly handles measurement noise and multi-victim scenarios, enabling accurate antagonist identification under realistic conditions.

3 Key Ideas

In this section, we present the insights and key ideas that form the foundation of our antagonist identification framework, followed by the main challenges that our design based on these ideas must address.

3.1 Key Insights

Our design is inspired by both datacenter domain knowledge and empirical observations from production traces. Specifically, we make the following observations:

1. **Task similarity within a job.** A single production job is typically divided into multiple tasks that execute distributedly across different machines. These tasks usually share the same binary and configuration, and therefore exhibit highly similar performance and resource-usage behavior.
2. **Repetitive workload patterns.** Datacenter workloads are highly repetitive over time. For example, dataset (Google, 2019) shows that more than 90% of jobs have tasks that reappear weekly within a one-month period.

These insights, though simple, reveal an important property of datacenter workloads: they exhibit rich temporal and spatial redundancy—across tasks belonging to the same job and across recurring jobs over time. Exploiting this redundancy allows for more reliable inference even under noisy measurement conditions, forming the foundation of our key design principles for robust antagonist identification:

1. **Job-level inference.** Instead of identifying antagonistic tasks, we identify *antagonistic jobs*, since tasks of the same job share similar behavior and resource characteristics.

2. **Global knowledge aggregation.** Rather than relying solely on local information—such as recent history from a single machine—we aggregate global information across all machines and over time to achieve more stable and accurate inference.

Finally, to support online monitoring in production environments, we adopt performance counter (PC) metrics (i.e., CPI), as they can be collected efficiently on existing hardware with minimal software overhead.

3.2 Design Challenges

Implementing our antagonist identification method based on the above key ideas faces several important challenges.

Challenge 1: Strong noise in performance counter (PC) metric samples. To avoid interfering with colocated workloads, the sampling window for PC metrics on production machines is typically very short. For example, in the publicly available trace (Google, 2019), the CPI metric is sampled over a 10-second window for every 5-minute record. In addition, various external factors—such as transient network conditions, background system activities, and dynamic business workloads—can further affect task performance. As a result, data collected from production machines inevitably contain strong noise, and improper denoising can easily lead to incorrect antagonist identification. Addressing this issue represents a primary challenge throughout this work.

Challenge 2: Scalability of global job-level CPI analysis. Realizing our key ideas requires a global analysis performed at the job level, which introduces two major scalability challenges. First, leveraging global knowledge demands analyzing CPI samples with resource usage across a very large number of jobs. Such pairwise analysis becomes computationally prohibitive in modern production environments. For example, conducting this analysis on (Google, 2019) would require computing relationship between hundreds of thousands of LS jobs and the resource usage of millions of jobs.

Second, obtaining joint observations for a given job pair requires that both jobs have been colocated on the same machine during at least one time interval—a condition determined by datacenter scheduling policy rather than statistical sufficiency. As a result, data availability across job pairs can be highly uneven: even when aggregating across the full production history, many job pairs have insufficient co-location samples, making their antagonist inference particularly vulnerable to noise.

4 DESIGN

In this section, we present the detailed design of our antagonist identification framework, **P**roduction **AN**tagonist **D**etection and **A**nalysis (PANDA).

4.1 Features of PANDA

We first describe the key design features that enable PANDA to address the challenges discussed earlier and distinguish it from prior approaches.

Global view. Unlike prior work that infers antagonists using only short-term data from a single machine, PANDA is designed with a global view. It aggregates historical information from all machines across the datacenter, enabling inference over a much broader temporal and spatial scope. This holistic perspective allows PANDA to effectively suppress local noise and produce more accurate and stable antagonist identification results.

Machine-level performance metric. PANDA operates at the machine level, analyzing relations between each job's activity and the overall machine performance, such as the machine-wide aggregated CPI. Although aggregating to the machine level may smooth out certain abnormal behaviors from individual tasks, we consider this trade-off reasonable. When only a small subset of tasks exhibits irregular performance, such anomalies are likely due to non-interference factors—such as workload patterns or transient system fluctuations—rather than true resource contention. In contrast, genuine performance interference typically stems from severe shared-resource contention, which affects all colocated tasks to some degree. Consequently, the machine-level performance metric not only mitigates the multi-victim problem identified in Challenge 2 but also helps average out measurement noise, addressing the false alarm problem from Challenge 1 simultaneously.

We note that aggregating performance metrics at the machine level also changes the definition of an antagonist. In prior work, antagonists are typically defined as tasks that degrade the performance of their colocated tasks. In contrast, in our formulation, an *antagonist* refers to a task that causes degradation in the overall machine-level performance. This broader definition aligns with our use of aggregated metrics and reflects the global impact of resource contention within a shared hardware environment.

4.2 Design Overview

We first give an overview of the design of PANDA, our antagonist identification framework.

PANDA identifies antagonistic jobs by analyzing machine usage records—either from offline traces or real-time monitoring data. It uses cycles per instruction (CPI) as its primary performance metric, a well-established performance counter (PC) indicator of system-level interference in prior studies (Zhang et al., 2013; Kannan et al., 2018).

PANDA consists of two major components: an offline phase and an online phase.

Offline phase. In this phase, PANDA continuously collects performance and resource usage data from all machines across the datacenter. Specifically, for each machine and each time slot, it records:

- the list of tasks running on the machine,
- the average CPU usage of each task during that time slot, and
- the sampled CPI values of important (e.g., latency-sensitive) tasks.

Using this information, PANDA maintains an *antagonist score* for each job and updates it incrementally as new data arrive. The score is computed by aggregating the machine-level CPI experienced by the job's tasks, weighted by each task's average CPU usage. This weighting emphasizes periods in which a job is actively consuming resources and thus more likely to exert interference. By aggregating such signals across all machines and time, PANDA captures global interference patterns rather than relying solely on local, short-term correlations.

Online: Antagonist identification. In the online phase, PANDA passively monitors live machines and is triggered whenever a machine exhibits abnormal behavior—such as a CPI value significantly higher than its baseline. Upon detection, PANDA analyzes the set of colocated jobs running on that machine and leverages the globally maintained antagonist scores computed during the offline phase to determine the most likely source of interference. Specifically, among all colocated tasks, PANDA selects the batch job with the highest antagonist score as the predicted antagonist, allowing for quick and informed mitigation of performance degradation.

By combining a global historical perspective with machine-level performance metrics, PANDA effectively filters out sampling noise and accurately identifies antagonists even in large, noisy production environments.

4.3 Offline Phase Design Details

In this subsection, we describe the detailed design of the offline phase, which consists of two main components: data preprocessing and antagonist score calculation.

4.3.1 Preprocessing

After collecting raw monitoring data, PANDA performs a series of preprocessing steps to transform it into model-ready input vectors. The preprocessing includes normalization, aggregation, and splitting, as detailed below.

Normalization. The first step is to normalize the CPI samples. A numerically high CPI value for a task can arise from two distinct causes: (1) Intrinsic factors, where a task is inherently computation-intensive, making a high CPI value normal for that task; and (2) Extrinsic factors, where the CPI increase results from interference or resource contention. PANDA focuses on the latter, as these reflect performance variability that should be minimized.

Following the method in (Zhang et al., 2013), PANDA applies **standard normalization** to all CPI samples. Specifically, for each job j, we compute its average CPI, denoted as $\overline{\text{CPI}}_j$, and its standard deviation $\sigma(\text{CPI}_j)$ using all CPI samples of its tasks. Let the CPI sample of task k from job j at time t be $CPI_{k,t}$. The *normalized CPI (nCPI)* for a task k from job j at time t, denoted as $nCPI_j\,[k,t]$, is then calculated as:

$$nCPI_j[k,t] = (CPI_{k,t} - \overline{CPI}_j)/\sigma(CPI_j). \tag{1}$$

Aggregation. After normalization, we aggregate the normalized CPI values (*nCPI*) into a machine-level metric, denoted as *mnCPI*. Specifically, *mnCPI* represents the average *nCPI* of all latency-sensitive (LS) jobs running on a given machine. Since LS jobs account for the most critical and performance-sensitive workloads in production datacenters, we consider *mnCPI* a meaningful indicator of the overall machine performance.

Moreover, such jobs are present frequently in real-world deployments. For example, in the Google trace (Google, 2019), cell `a`, each machine hosts on average 13.5 LS tasks, with 90.1% of machines running at least three such tasks concurrently.

We define *mnCPI* for each machine m at time t as follows. Let $K_{m,t}$ denote the set of all LS and high-priority tasks running on machine m at time t, each of which is denoted by (j,k), i.e., the k-th task from job j. Then,

$$mnCPI[m,t] = \frac{\sum_{(j,k)\in K_{m,t}} nCPI_j[k,t]}{|K_{m,t}|}. \tag{2}$$

Unlike (Zhang et al., 2013; Kannan et al., 2018), which focus on per-task performance metrics, PANDA is based on machine-level metric *mnCPI*. This design choice effectively mitigates the *multi-victim challenge* by capturing interference at the shared hardware level rather than from individual noisy task-measurements.

4.3.2 Calculating Antagonist Scores

Inspired by prior work (Zhang et al., 2013; Kannan et al., 2018), we design the *antagonist score* in PANDA under the assumption that a job's interference behavior is **positively**

and linearly correlated with its CPU utilization. Hence, we use the correlation between *mnCPI* and the colocated job's average CPU usage to produce antagonist scores. We now provide a precise definition of the *antagonist score*.

Let K_j be the set of all tasks from job j and let MT_n be the set of all machine-time pairs $[m, t]$ such that task $n \in K_j$ runs in it. Recall that $u^n_{m,t}$ is the average CPU usage of task n on $[m, t]$ and $\textit{mnCPI}[m, t]$ is as defined in (2). Then, the antagonist coefficient for each job j is defined as follows:

$$a_j = \frac{\sum_{n \in K_j} \sum_{[m,t] \in MT_n} u^n_{m,t} * \textit{mnCPI}[m, t]}{\sum_{n \in K_j} \sum_{[m,t] \in MT_n} (u^n_{m,t})^2} \quad (3)$$

Note that (3) can be viewed as the least-square estimate of the coefficient a_j in a linear model of the form $\textit{mnCPI}[m, t] = a_j u^n_{m,t}$, assuming that all other $[m, t]$ not in MT_n have $u^n_{m,t} = 0$. During the online phase, when a victim appears, say, on machine m_c at time t_c, we then multiply each collocated task k's average CPU usage with the corresponding job j's antagonist coefficient a_j, to compute this task k's antagonist score, which can be viewed as task k's contribution to the performance degradation.

5 Evaluation

Evaluating antagonist identification methods, in particular establishing reliable ground truth, poses significant challenges. Below, we first summarize how prior studies (Zhang et al., 2013; Kannan et al., 2018; Wang et al., 2022) addressed this issue, and then describe our own evaluation methodology.

5.1 Prior Methods for Establishing Ground Truth

Proctor et al. (Kannan et al., 2018) evaluated their approach using a simulated colocation environment, where both the workloads and antagonists were artificially generated. While this setup allows controlled testing, it fails to capture the complexity and heterogeneity of real production systems, limiting its representativeness.

In contrast, (Zhang et al., 2013) conducted experiments directly in Google's production environment. Their ground truth was established through an ablation-based study: after throttling the identified antagonists, they observed changes in the victim's CPI. If the CPI decreased, the identification was deemed correct; otherwise, it was considered incorrect. Although this approach provides strong validation, it involves active perturbation of production systems, which is risky and difficult to reproduce in practice.

Finally, (Wang et al., 2022) evaluated their method on a production trace, using human expert labeling to determine ground truth. Considering its balance between practicality and realism, we adopt a similar trace-based evaluation methodology in our work.

Table 1: Summary statistics of the dataset used in our evaluation (Google, 2019). LS: latency-sensitive.

Statistic	Value
Trace duration	31 days
Total number of jobs	∼5.2 million
Total task execution records	∼5 billion
Average number of colocated tasks per machine	50–100
Job types	LS and batch
Average CPU util. per machine	45–65%
Average memory utilization per machine	60–70%
Sampling interval for performance counters (CPI)	10s/5 min

5.2 Methodology

We next describe our evaluation methodology.

Human-expert labeling. Because all data in (Google, 2019) are anonymized, the true antagonists are not directly observable. Therefore, similar to (Wang et al., 2022), we rely on *human experts* to provide ground-truth labels; specifically, we ask Google domain experts to identify jobs they know to be antagonists. However, human-expert labeling is inherently *one-sided*. Experts can confidently confirm that a job is an antagonist if they have repeatedly observed it causing interference, but it is difficult for them to assert that a job is *not* an antagonist with the same level of certainty. As a result, the available ground truth is incomplete, and standard classification metrics such as recall cannot be directly applied. Indeed, (Wang et al., 2022) faced the same limitation and evaluated only precision. Following this practice, our evaluation also focuses on precision.

Dataset. Table 1 summarizes the statistics of cell `a` of the dataset (Google, 2019) used in our evaluation. The trace contains 31 days of resource-usage records collected from more than 10,000 machines in a production Google datacenter. LS and batch jobs in (Google, 2019) are classified according to their priority and scheduling class: LS jobs has priority ≥ 120 and scheduling class ≥ 2, and the rest are batch jobs.

In our experiments, the offline phase of PANDA updates antagonist coefficients once per day. Thus, whenever the online phase is triggered, PANDA relies on the coefficients computed using data up to the previous day.

Because human-expert labeling is one-sided, we only have access to a subset of true antagonists. Therefore, in our evaluation we restrict analysis to records hosting tasks belonging to jobs within this expert-identified antagonist set.

Baselines. To evaluate the effectiveness of PANDA, we

compare it against representative state-of-the-art antagonist identification approaches that cover the two major categories of prior work:

- **CPI^2** (Zhang et al., 2013): a correlation-based method that detects antagonists from real-time production data by analyzing the correlation between victim's CPI and colocated tasks' CPU usage within a recent window. As a representative *sample-from-production* approach, CPI^2 serves as the most directly comparable baseline to PANDA.
- **Proctor** (Kannan et al., 2018): an enhanced variant of **CPI^2** that analyzes the correlation between a victim's CPI and the CPU usage of colocated tasks using a random subsample within a recent time window. A Chi-square test is applied to ensure that the subsample is statistically representative. As a recent *sample-from-production* approach, Proctor provides another meaningful comparison to PANDA.
- **PANDA-Local**: a localized variant of PANDA that restricts analysis to the full history of a *single* machine rather than leveraging global information across the cluster. Comparing PANDA against PANDA-Local isolates the benefit of PANDA's global view and highlights its improvement in antagonist-identification accuracy.

This comprehensive comparison allows us to evaluate PANDA 's robustness to noise, scalability across large traces, and ability to handle multi-victim interference in production-scale environments. We note that (Wang et al., 2022) relies on dedicated cache-monitoring hardware, making it difficult to deploy broadly in production datacenters; therefore, it is not included as a baseline.

Specific setups. We treat a task as a victim if it belongs to a high-priority or latency-sensitive (LS) job and exhibits $nCPI \geq 2$. For fair comparison, we align the online triggering conditions of PANDA and all baselines: antagonist identification begins only when (i) the machine-level *mnCPI* exceeds its 99th percentile, and (ii) at least one victim is present for three consecutive time slots. To ensure sufficient historical coverage for estimating PANDA 's antagonist coefficients, our accuracy evaluation focuses on victim events occurring on or after day 20 of the trace. For baseline configuration, CPI^2 computes correlations using the past two hours of data from the local machine, while Proctor selects a random one-hour subsample from the same two-hour window, using three victim-CPI categories and a Chi-square threshold of 1.

Metrics. Since our ground truth is one-sided, classical metrics, such as precision and recall, can be unrepresentative. However, we observe that both PANDA and the baselines naturally produce *scores* for all colocated tasks and then select the task with the highest score as the antagonist. This scoring behavior induces a natural *ranking*, *i.e.*,, the scores provide an ordered list of colocated tasks based on their suspected likelihood of being antagonistic.

Accordingly, we measure performance using the *average percentile rank* of all expert-identified antagonists among their colocated tasks. This metric evaluates how highly a method ranks the true antagonists within each candidate set, thereby reflecting the effectiveness of the underlying scoring mechanism used for antagonist identification.

We note that a rank of r out of n will be converted to a percentile p by

$$p = \frac{n - r}{n - 1}. \tag{4}$$

5.3 Evaluation Results

In this subsection, we present the evaluation results of PANDA on cell `a` of (Google, 2019). We first summarize the overall average ranking percentiles of PANDA and baselines, then analyze their performance, and finally discuss case studies and insights derived from our evaluation.

5.3.1 Average Ranking Percentiles

We begin by average ranking percentiles of PANDA with baselines on the suspicious of the expert-identified antagonists. Table 2 reports the overall average ranking percentiles of the true antagonists suspicious level on the trace. PANDA achieves consistently higher percentile, demonstrating its robustness against sampling noise and multi-victim interference. Consider usually $\sim$30 colocated batch jobs, PANDA on average rank the true antagonists in top 5, while baselines on average give a slight-above-median rank.

Table 2: average ranking percentiles of antagonist identification methods.

CPI^2	**Proctor**	**PANDA**	**PANDA-Local**
0.543	0.556	0.826	0.602

5.3.2 Impact of Global View

To evaluate the effectiveness of the global aggregation strategy, we compare PANDA with PANDA-Local, a local-only variant that uses machine-specific historical data. As shown in Table 2, PANDA achieves higher ranking percentiles than PANDA-Local, confirming that leveraging historical data across all machines helps suppress noise and improve inference stability. Further, PANDA-Local obtain higher ranking percentiles than the baselines which are limited to only recent history, demonstrating the effectiveness of past history in improving antagonist identification.

5.3.3 *Handling the Multi-Victim Scenario*

We next evaluate PANDA and the baselines under multi-victim conditions, defined as cases where two or more victims are colocated on the same machine at the same time. Table 3 reports the *antagonist-consistency rate*, i.e., the probability that pairwise colocated victims identify the same antagonist. By design, PANDA consistently attributes interference to the same job across colocated victims and therefore achieves a consistency rate of 1 in all cells. In contrast, the baselines exhibit varying degrees of disagreement, highlighting their susceptibility to antagonist divergence in multi-victim scenarios.

Table 3: Antagonist-consistency rate regarding to the multi-victim case of antagonist identification methods.

CPI^2	Proctor	PANDA
0.314	0.376	1

5.3.4 *Summary of Findings*

In summary, our evaluation demonstrates that:

- PANDA achieves higher ranking in the antagonist score than prior correlation-based methods.
- The global inference design effectively suppresses sampling noise.
- The machine-level performance metric enables reliable analysis in multi-victim environments.

6 Related Work

A large body of prior research has explored performance isolation between latency-sensitive (LS) jobs and background jobs in production datacenters (Lu et al., 2023; Yang et al., 2013; Fried et al., 2020; Patel and Tiwari, 2020; Lo et al., 2014; Schwarzkopf et al., 2013; Lo et al., 2016; Delimitrou and Kozyrakis, 2013; Chen et al., 2019; Iorgulescu et al., 2018; Zhang et al., 2014; Delimitrou and Kozyrakis, 2014; Kasture et al., 2015; Mars et al., 2011; Zhang et al., 2013). These studies can be broadly categorized into three groups.

Reactive resource partitioning. The first category, exemplified by (Chen et al., 2019; Fried et al., 2020; Patel and Tiwari, 2020), focuses on reactive mechanisms that respond to contention signals by partitioning specific hardware resources—such as last-level cache (LLC) partitioning via Intel Cache Allocation Technology (CAT). While effective in controlled environments, these approaches depend on specialized hardware support, limiting their deployment in heterogeneous production datacenters. Moreover, resource partitioning technologies like CAT offer only coarse-grained control and support a limited number of partitions, making them difficult to scale to modern servers that host over 100 colocated tasks.

Offline screening and profiling. The second category, represented by (Zhang et al., 2014; Yang et al., 2013; Mars et al., 2011; Delimitrou and Kozyrakis, 2013; 2014; Lu et al., 2023), employs dedicated environments to screen jobs and profile their interference behavior. Each job is executed in isolation or with controlled co-runners to build an interference profile, which is later incorporated into the scheduler to guide interference-aware placement decisions. This approach can proactively prevent colocation interference and achieve strong performance isolation under ideal conditions. However, applying such methods to production-scale datacenters is prohibitively expensive—profiling millions of jobs would impose substantial overhead in both computation and storage.

Sampling from production. The third category of work, exemplified by (Zhang et al., 2013; Kannan et al., 2018; Wang et al., 2022), directly samples performance metrics from production machines to identify performance culprits, which we refer to as *antagonist jobs*. This approach is simple, scalable, and well-suited for deployment in production datacenters. Once antagonists are correctly identified, colocation interference can be quickly mitigated through targeted actions such as task throttling or eviction. However, sampling in production environments typically relies on short observation windows to avoid performance overhead, which introduces significant measurement noise. Such noise can severely degrade the accuracy of antagonist identification. None of the existing works has systematically addressed this issue, which directly motivates our work.

7 Conclusion

We presented PANDA, a noise-resilient and scalable framework for identifying antagonist jobs in production datacenters. Unlike prior task-level approaches that rely on noisy, locally sampled data, PANDA leverages a global view across all machines and time windows, combining machine-level performance metrics with historical correlation analysis to achieve robust inference. Our evaluation on a production datacenter trace demonstrates that PANDA significantly improves the precision and stability of antagonist identification, even under noisy and multi-victim conditions. By enabling accurate and low-overhead detection of interference sources, PANDA paves the way for more adaptive and interference-aware scheduling in large-scale production environments.

Acknowledgements

This work is supported in part by NSF grant 2113893.

REFERENCES

Shuang Chen, Christina Delimitrou, and José F. Martínez. 2019. PARTIES: QoS-Aware Resource Partitioning for Multiple Interactive Services. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems* (Providence, RI, USA) *(ASPLOS '19)*. Association for Computing Machinery, New York, NY, USA, 107–120. doi:10.1145/3297858.3304005

Christina Delimitrou and Christos Kozyrakis. 2013. Paragon: QoS-aware scheduling for heterogeneous datacenters. In *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (Houston, Texas, USA) *(ASPLOS '13)*. Association for Computing Machinery, New York, NY, USA, 77–88. doi:10.1145/2451116.2451125

Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: resource-efficient and QoS-aware cluster management. *SIGPLAN Not.* 49, 4 (feb 2014), 127–144. doi:10.1145/2644865.2541941

Joshua Fried, Zhenyuan Ruan, Amy Ousterhout, and Adam Belay. 2020. Caladan: Mitigating Interference at Microsecond Timescales. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 281–297. https://www.usenix.org/conference/osdi20/presentation/fried

Panagiotis Garefalakis, Konstantinos Karanasos, Peter Pietzuch, Arun Suresh, and Sriram Rao. 2018. Medea: scheduling of long running applications in shared production clusters. In *Proceedings of the Thirteenth EuroSys Conference* (Porto, Portugal) *(EuroSys '18)*. Association for Computing Machinery, New York, NY, USA, Article 4, 13 pages. doi:10.1145/3190508.3190549

Google. 2019. Google cluster-usage traces v3. https://github.com/google/cluster-data/blob/master/ClusterData2019.md.

Calin Iorgulescu, Reza Azimi, Youngjin Kwon, Sameh Elnikety, Manoj Syamala, Vivek Narasayya, Herodotos Herodotou, Paulo Tomita, Alex Chen, Jack Zhang, and Junhua Wang. 2018. PerfIso: Performance Isolation for Commercial Latency-Sensitive Services. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 519–532. https://www.usenix.org/conference/atc18/presentation/iorgulescu

Ram Srivatsa Kannan, Animesh Jain, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2018. Proctor: Detecting and Investigating Interference in Shared Datacenters. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 76–86. doi:10.1109/ISPASS.2018.00016

Harshad Kasture, Davide B. Bartolini, Nathan Beckmann, and Daniel Sanchez. 2015. Rubik: Fast analytical power management for latency-critical systems. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 598–610. doi:10.1145/2830772.2830797

David Lo, Liqun Cheng, Rama Govindaraju, Luiz André Barroso, and Christos Kozyrakis. 2014. Towards energy proportionality for large-scale latency-critical workloads. In *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. 301–312. doi:10.1109/ISCA.2014.6853237

David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. 2016. Improving Resource Efficiency at Scale with Heracles. *ACM Trans. Comput. Syst.* 34, 2, Article 6 (may 2016), 33 pages. doi:10.1145/2882783

Chengzhi Lu, Huanle Xu, Kejiang Ye, Guoyao Xu, Liping Zhang, Guodong Yang, and Chengzhong Xu. 2023. Understanding and Optimizing Workloads for Unified Resource Management in Large Cloud Platforms. In *Proceedings of the Eighteenth European Conference on Computer Systems* (Rome, Italy) *(EuroSys '23)*. Association for Computing Machinery, New York, NY, USA, 416–432. doi:10.1145/3552326.3587437

Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. 2011. Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In *2011 44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 248–259.

Tirthak Patel and Devesh Tiwari. 2020. CLITE: Efficient and QoS-Aware Co-Location of Multiple Latency-Critical Jobs for Warehouse Scale Computers. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 193–206. doi:10.1109/HPCA47549.2020.00025

Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. 2013. Omega: flexible, scalable schedulers for large compute clusters. In *SIGOPS European Conference on Computer Systems (EuroSys)*. Prague, Czech Republic, 351–364.

Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer*

Systems (Bordeaux, France) *(EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, Article 18, 17 pages. doi:10.1145/2741948.2741964

Kangjin Wang, Chuanjia Hou, Ying Li, Yaoyong Dou, Cheng Wang, Yang Wen, Jie Yao, and Liping Zhang. 2022. Cache Antagonists Identification: A Practice from Alibaba Colocation Datacenter. In *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 7–12. doi:10.1109/ISSREW55968.2022.00031

Hailong Yang, Alex Breslow, Jason Mars, and Lingjia Tang. 2013. Bubble-flux: precise online QoS management for increased utilization in warehouse scale computers. In *Proceedings of the 40th Annual International Symposium on Computer Architecture* (Tel-Aviv, Israel) *(ISCA '13)*. Association for Computing Machinery, New York, NY, USA, 607–618. doi:10.1145/2485922.2485974

Xiao Zhang, Eric Tune, Robert Hagmann, Rohit Jnagal, Vrigo Gokhale, and John Wilkes. 2013. CPI2: CPU performance isolation for shared compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems* (Prague, Czech Republic) *(EuroSys '13)*. Association for Computing Machinery, New York, NY, USA, 379–391. doi:10.1145/2465351.2465388

Yunqi Zhang, Michael A. Laurenzano, Jason Mars, and Lingjia Tang. 2014. SMiTe: Precise QoS Prediction on Real-System SMT Processors to Improve Utilization in Warehouse Scale Computers. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture* (Cambridge, United Kingdom) *(MICRO-47)*. IEEE Computer Society, USA, 406–418. doi:10.1109/MICRO.2014.53