#714 Hestia: An Attention-guided Hyperthread-level Scheduling Framework for Interference Mitigation among Co-located Cloud Services

Email notification
Select to receive email on updates to reviews and comments.

▼ PC conflicts
Jingwen Leng
Shuangchen Li
Xiaoyao Liang

Reject

Submission (1.1MB) Nov 22, 2023, 9:30:23 AM UTC · 966f8121

▼ Abstract

To improve resource efficiency, a server usually deploys multiple latency-sensitive (LS) instances. However, this co-location may cause performance degradation due to resource contention. Numerous studies have proposed interference-aware scheduling methods or utilized resource isolation such as Linux cpuset to mitigate interference. However, there exists no fine-grained binding core strategy to guide scheduling in production. In this study, we analyze trace data of LS services from a production cluster consisting of 32,408 instances deployed on 3,132 servers and recognize two interference patterns: sharing-core (SC) and sharing-socket (SS) interference. Based on our findings, we introduce Hestia, an attention-guided hyperthread-level interference mitigation framework to complement existing schedulers. Hestia includes a CPU utilization predictor employing a self-attention mechanism considering SC and SS neighbors, as well as workloads and hardware heterogeneity, to

▶ Authors (anonymous)

J. Dai, D. Yang, S. Qian, J. Cao, G. Xue [details]

▶ Topic Areas and options

accurately predict the CPU utilization of co-located instances. Moreover, Hestia incorporates an interference scoring approach to estimate the potential interference of instances before scheduling. To evaluate the effectiveness of Hestia, we evaluate its performance in a production cluster with 19,760 cores. The results demonstrate that Hestia can reduce the total cluster CPU usage by 2.27% and decrease the 95th percentile service latency by 13.9%. We also compare Hestia with baselines based on large-scale data-driven simulations and it can increase the performance of LS services by up to 20.89%.

	PosRebOve	OveMer	RevExp	RevCon	WriQua	Nov	ExpMet
Review #714A	1	1	4	3	4	2	2
Review #714B	2	2	4	3	2	3	1
Review #714C	2	2	3	3	2	2	2
Review #714D	1	1	4	3	2	1	1

You are an **author** of this submission.

Edit submission Add comment

Reviews in plain text

Review #714A

Mar 6

Paper summary

This paper proposes Hestia, a scheduler for colocated workloads that considers interference from SC and SS. Compared to prior work, Hestia selects not only a single server, but also a set of HTs for each incoming instance. The key observation is that application performance correlates very well

with CPU utilziation. Therefore, Hestia uses an attention-based mechanism to predict CPU utilization and infer the level of interference.

Strengths

- Tackles an important area
- Scheduling at HT granularity, instead of host granularity
- Comprehensive evaluation

Weaknesses

- Lack of clarification on the underlying assumptions of the model
- Only considers interference from cores

Post-Rebuttal Overall Merit

1. Reject

Overall merit

1. Reject

Reviewer expertise

4. Expert

Reviewer confidence

3. High

Writing quality: is the paper easy to read and understand?

4. Well-written

Novelty

2. Incremental improvement

Experimental methodology

2. Fair: Reasonable methodology, needs more results

Comments for authors

Thanks for submitting to ISCA. This paper tackles an important area. The system is built on the observation that CPU utilization is a great indicator of service latency, and therefore builds a transformer-based model to predict CPU utilization and infer interference. The paper is generally well written, but there are a couple of underlying assumptions that may stop the system from generalizing to other scenarios.

First, the paper works on a very crowded area. Interference has been studied over a decade. There are numerous ways to predict performance and a variety of resource interference (other than core interference) has been extensively studied. I find that the related work section seems to (purposely) underestimate the applicability of prior approaches. SC and SS interference are very well known. Table 1 shows that prior work do not tackle SC and SS interference, but this is not true. Despite not explicitly claiming to support this type of interference, prior work can be easily extended to support interference and latency prediction. For instance, CPI2 proposes metric CPI, and CPI can definitely reflect interference from SC and SS. In reality, prior approaches can accomodate more types of interference (such as cache and memory), while this paper only targets core interference. This makes the paper lack of novelty in the first place.

Second, I am confused by observation 2 in Section IV. I wonder if the correlation of CPU utilization comes from intinsic application-level logic. For instance, all the instances running on the same server experience similar load fluctuations. For instance, one instance runs cart and one instance runs ordering. A load increase in cart implies more costomers shopping online, and therefore the load of the ordering service also increases. So, CPU utilization of both instances increase or decrease at the same time. So the correlation does not represent the actual interference, but simply load fluctuation. If so, the authors should state clearly about these assumptions. Also, if two instances share the same physical cores (but not HTs), can the hardware accurately show HT-level CPU utilization?

Third, I'm not sure why self-attention is needed. There is no insight or qualitative discussion on why this method is chosen, and how it compares with simpler approaches (like simply using the average CPU utilization in the history) or more complex approaches (like LSTM).

Finally, the evaluation is first done in simulator. It is not clear what the simulator actually simulates. How does it simulate the latency degradation from workload colocation?

Questions to address in rebuttal/revision

- Please clearly clarify the underlying assumptions of your workloads, and the generality of the correlation of CPU utilization.
- Please introduce the insight on the choice of self-attention.
- Please clarify more details of the simulator.

Review #714B

Mar 4

Paper summary

The paper presents machine learning based techniques — attention mechanisms — to estimate interference for colocated workloads on hyperthreaded multiprocessors. It describes embeddings and encoders that accurately score the performance of available hyperthreads based on intererference within a processor core and within a processor socket.

Strengths

The use of attention mechanisms seems novel and may encourage researchers to look more closely at this class of techniques.

Weaknesses

The paper does not compare against a very large body of work in this space. The machine learning techniques are presented in such a way that it may be difficult for readers to understand precisely the model architecture and dataset used to train it.

Post-Rebuttal Overall Merit

2. Weak reject

Overall merit

2. Weak reject

Reviewer expertise

4. Expert

Reviewer confidence

3. High

Writing quality: is the paper easy to read and understand?

2. Needs improvement

Novelty

3. New contribution

Experimental methodology

1. Poor: Unacceptable methodology and/or results

Comments for authors

Studying interference and optimizing workload colocation is an important problem. The idea of using attention mechanisms is also intriguing. However, the paper seeks to make a contribution in crowded problem space without comparing, conceptually or empirically against a large body of prior work. This prevents the reader from assessing the merits of the idea.

Specifically, researchers have used machine learning to predict performance and interference (e.g., see work from Christina Delimitrou, Lingjia Tang, Jason Mars, Ben Lee, and others). None of these prior studies used attention mechanisms so there could be indeed an interesting opportunity to advance the state-of-the-art. But it is not possible to assess the goodness of attention mechanisms without a direct comparison against, for example, recommendation systems or heuristic scoring.

On attention mechanisms, the paper would be stronger if it could detail more precisely the dataset features and the model architecture. Page 7 describes the broad contours of embeddings and encoders, but provides little detail about the practical aspects of collect datasets, identifying effective features, and how the model architecture is tuned for this system setting. These operational insights would allow the paper to more effectively influence how researchers use machine learning and attention mechanisms in their own system settings or problem domains.

Review #714C Mar 3

Paper summary

This paper's goal is to reduce the interference suffered by latency sensitive (LS) container instances in cloud datacenters, where the servers are composed of two sockets and cores are multithreaded. To do so, the paper proposes what appears to be an elaborate neural network model that predicts the utilization of individual cores. The assumption is that the core utilization is a

predictor of the interference. The paper evaluates the model in a real cloud environment and with simulations.

Strengths

Interesting experimental data.

Weaknesses

The writing in the paper is poor.

There are many things that are unclear.

Post-Rebuttal Overall Merit

2. Weak reject

Overall merit

2. Weak reject

Reviewer expertise

3. Knowledgeable

Reviewer confidence

3. High

Writing quality: is the paper easy to read and understand?

2. Needs improvement

Novelty

2. Incremental improvement

Experimental methodology

2. Fair: Reasonable methodology, needs more results

Comments for authors

The writing in the paper is poor. The ideas do not come out clearly; the writing is confused and this makes it hard to understand the paper.

The core of the paper is to propose a sophisticated neural network model to predict the utilization of cores, and since the assumption is that core utilization is a predictor of interference, use the model to predict the contexts, cores, and sockets that will suffer most interference and avoid them.

There are many things that are unclear:

1) Why this emphasis on predicting the utilization of cores? Is there a better way to estimate the interference that does not rely on utilization? Despite some motivating data, this reviewer is not convinced that, to estimate interference, one has to look at utilization. As it is mentioned somewhere in the paper, what if the interference happens in the memory system or I/O? This has nothing to do with core utilization.

2) Do we really need this complicated neural network model to predict core utilization? Why do we need an initial embedding layer, then a self-attention encoder, etc... This is not justified. Try a simple technique to estimate core utilization. Then compare.

3) After all this elaborate model, the results are a bit underwhelming, I feel: the processor utilization is reduced by 2.5% and the tail latency by 13.9%. How significant is this?

Questions to address in rebuttal/revision

Respond to 1-3

Review #714D Mar 2

Paper summary

This paper proposes a scheduler for co-locating latency-critical workloads on a single server. The authors show sharing-core and sharing-socket-based interference that causes performance degradation. Based on the new proposed scheduler "Hestia", the authors can predict this interference and efficiently schedule the services.

Strengths

++ Interesting production-scale testing and experiments

Weaknesses

-- fairly trivial observations -- Many production-scale hardware-isolations techniques are missing

Post-Rebuttal Overall Merit

1. Reject

Overall merit

1. Reject

Reviewer expertise

4. Expert

Reviewer confidence

3. High

Writing quality: is the paper easy to read and understand?

2. Needs improvement

Novelty

1. Published before or openly commercialized

Experimental methodology

1. Poor: Unacceptable methodology and/or results

Comments for authors

I thank the authors for submitting the paper to ISCA. Interference between workloads, especially latency-critical workloads, is well-known. This paper tries to predict interference among latency-critical services before scheduling them. They use an attention-guided CPU utilization predictor to make better scheduling decisions and reduce interference. However, I have some basic concerns about the baseline system that the authors are using.

1. Why not allocate physical cores to the service? Why bind and allocate at hyperthreads? This introduces interference which is hard to control and not used in large-scale production usually. Moreover, there are multiple hardware-based isolation techniques like memory bandwidth allocation and last-level cache partitioning schemes that are used to reduce this interference among co-located services. If you include these basic knobs, the interference detector has to change drastically. This is my major concern about the baseline being employed in this study. Any large-scale deployment will first use these techniques to improve their utilization then use any kind of predictor to figure out interference and make scheduling decisions.

2. Why do predictions on interference? A workload in a large-scale deployment is movable or even can be killed on one server and launched on the other. Interference detection is a lot easier than interference prediction in general. Once you detect interference, you can move or launch the service on some other container that seems to be free or has a lesser load.

3. Since I have questions about the baseline scheduling and isolation techniques employed, it is hard to appreciate the novelty of the work. All the observations at scale seem to be straightforward. I want authors to bring out the novelty of the work and why is this important. Otherwise, this can be a better candidate for an industry-track paper.

Overall, I feel like the problem area is interesting from a company standpoint. However, the experiments and baselines are not designed scientifically and do not reflect efficient large-scale deployments. I would implore authors to upgrade their baseline and provide data in support of a predictor-based scheduler rather than rescheduling on an interference detection.

Questions to address in rebuttal/revision

Please the detailed comments section above.