POSTER: FineCAT: Instruction-Count-Driven Fine-Grained LLC Management for Co-Location Scenarios

Yanqi Kan
Lei Wang✉
Fanda Fan
kanyanqi23@mails.ucas.ac.cn
wanglei_2011@ict.ac.cn
fanfanda@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
Haidian, Beijing, China
University of Chinese Academy of Sciences
Haidian, Beijing, China

Yikang Yang
Wanling Gao
Chunjie Luo
Jianfeng Zhan
yangyikang23s@ict.ac.cn
gaowanling@ict.ac.cn
luochunjie@ict.ac.cn
zhanjianfeng@ict.ac.cn
Institute of Computing Technology, Chinese Academy of Sciences
Haidian, Beijing, China

Abstract

Workload co-location improves server utilization but creates last-level cache (LLC) contention. While Intel cache allocation technology (CAT) enables LLC partitioning, existing wall-clock-driven managers can become misaligned with workload phases under co-location interference, limiting performance isolation. We present FineCAT, which uses retired-instruction progress as the common unit for monitoring, prediction, and CAT reconfiguration. FineCAT samples fine-grained LLC-demand features, predicts instruction-count-indexed phases, and adjusts low-priority (LP) allocations at phase boundaries while protecting high-priority (HP) workloads. On one socket of a two-socket Intel server, FineCAT improves mean HP performance by 6.9% over No-CAT in eight-workload co-location; for LLC-sensitive HP workloads, the mean and peak gains are 13.6% and 52.0%, outperforming SOTA schemes by 6.9% on average and up to 25.6%. Mean LP degradation is 2.8% versus No-CAT.

CCS Concepts

• **Computer systems organization** → *Parallel architectures*; • **Software and its engineering** → *Allocation / deallocation strategies.*

Keywords

workload co-location, instruction-count-based management

ACM Reference Format:
Yanqi Kan, Lei Wang, Fanda Fan, Yikang Yang, Wanling Gao, Chunjie Luo, and Jianfeng Zhan. 2026. POSTER: FineCAT: Instruction-Count-Driven Fine-Grained LLC Management for Co-Location Scenarios. In *International Conference on Parallel Architectures and Compilation Techniques (PACT '26), October 19-22, 2026, Chicago, IL, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3838684.3846848

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PACT '26, Chicago, IL, USA
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846848

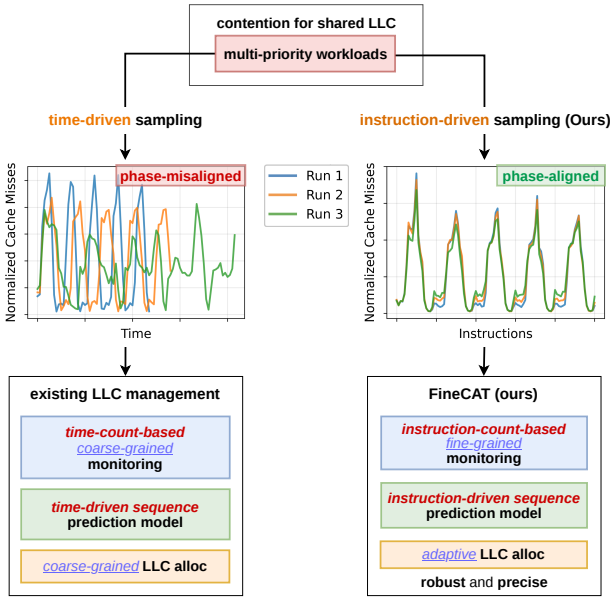


Figure 1: Overview of FineCAT.

1 Introduction and Motivation

Co-locating workloads of different priorities raises utilization, but shared-LLC contention undermines performance isolation for both high-priority (HP) and low-priority (LP) applications. Intel CAT enables way-level LLC partitioning, while existing CAT managers typically use wall-clock-driven control with offline profiles, runtime feedback, or demand prediction [1, 3, 9, 11]. Under co-location interference, such control can become phase-misaligned, making profiles hard to reuse and allocation decisions untimely.

Our measurements reveal two insights (Fig. 1). First, across repeated co-located runs, retired instruction counts vary by less than 1% (CV), while execution time varies by over 5% on average, motivating instruction-count-based phase alignment. Second, LLC demand

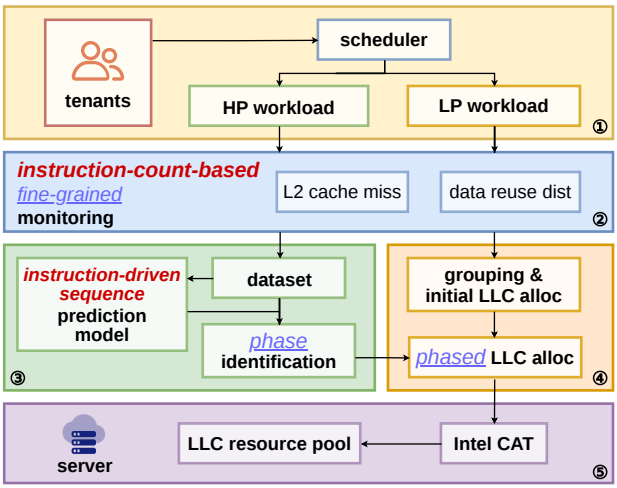


Figure 2: FineCAT's priority-aware workflow.

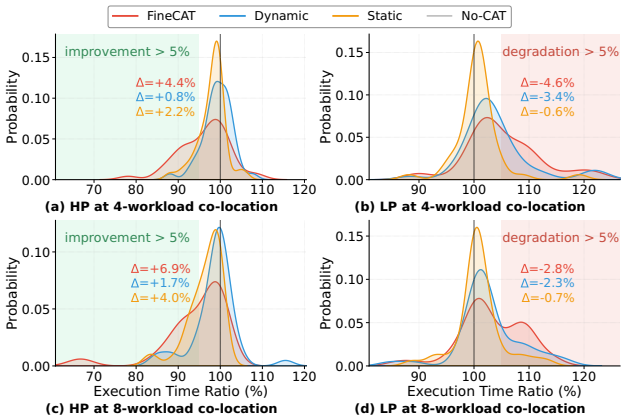


Figure 3: Normalized performance ratios (No-CAT = 100%); lower is better. Δ denotes mean change.

varies at macro (10^{11} instructions) and micro (10^8 instructions) scales, so coarse sampling can miss short-term changes.

Based on these insights, FineCAT aligns monitoring, prediction, and allocation with retired-instruction progress. Its main contributions are: 1) instruction-driven prediction robust to co-location noise; 2) fine-grained, priority-aware CAT allocation based on predicted HP demand; and 3) improved HP performance with modest mean LP degradation in the evaluated scenarios.

2 FineCAT Design

Figure 2 summarizes FineCAT. HP gets a dedicated CAT class with full LLC access, while LP workloads share baseline partitions based on LLC-sensitive reuse-distance ratios.

Instruction-driven monitoring. FineCAT configures Linux `perf_event_open` to interrupt after every $\Delta I = 10^8$ retired instructions. For HP workloads, it records L2 cache misses and computes $\rho_t = 1000 M_t / \Delta I$, where M_t is the miss count in interval t. The resulting L2 misses-per-kilo-instruction (MPKI) sequence tracks relative LLC-demand changes. For LP workloads, lightweight hardware-assisted address sampling estimates reuse-distance distributions [7]; the fraction of accesses whose reuse distances fall between effective L2 and LLC capacities determines a baseline LLC-sensitive ratio.

Prediction and phase identification. From n HP samples $\{\rho_t\}_{t=k-n+1}^{k}$ ending at interval k, FineCAT predicts the next h MPKI values. Its best-predictor selection ensemble (BPSE) combines the decomposition-based DLinear [10] with historical-average and last-value predictors, selecting the previous round's lowest-error predictor. Online fine-tuning adapts DLinear to distribution shifts. Predictions are partitioned into phases; $\hat{\rho}_{i,80}$ denotes the 80th-percentile predicted MPKI in phase i.

Phase-aware allocation. For LP group j, let $R_{\text{base},j}$ and $R_{i,j}$ denote its baseline and phase-i LLC-way allocations. FineCAT scales the baseline inversely with predicted HP demand:

$$R_{i,j} = \text{clip}_{[1,T]}\left(\left\lfloor \alpha \frac{\bar{\rho}}{\hat{\rho}_{i,80}} R_{\text{base},j} \right\rfloor\right), \tag{1}$$

where $\bar{\rho}$ is the historical mean of the observed HP MPKI sequence ρ_t, $\alpha > 0$ is a dimensionless LP capacity scaling factor, and T is the number of LLC ways. Floor and clip yield an integer in $[1, T]$. LP groups receive more capacity at low HP demand and less near peaks. FineCAT merges adjacent phases with identical or near-identical allocations, updating CAT only at remaining boundaries to limit overhead and synchronize monitoring, prediction, and action with program progress.

3 Evaluation

Methodology. We evaluate FineCAT on one socket of a two-socket Intel Xeon Silver 4109T server using SPEC CPU 2017 [6], AIoT-Bench [4], and Redis workloads with diverse LLC sensitivities. Four- and eight-workload scenarios contain one HP and three or seven LP workloads, respectively. Each experiment is repeated five times, with FineCAT compared against No-CAT and SOTA static (Static) and predictive dynamic (Dynamic) CAT schemes, using execution time for batch workloads and 90th-percentile latency for Redis.

End-to-end performance. Figure 3 shows the HP–LP trade-off. In eight-workload scenarios, FineCAT improves mean HP performance by 6.9% over No-CAT, rising to 13.6% for LLC-sensitive HP workloads and up to 52.0% at peak. Mean LP degradation is 2.8% relative to No-CAT, below 3% in this scenario. Benefits are smaller for LLC-insensitive workloads, consistent with FineCAT helping most when partitioning is beneficial.

Why FineCAT works. Ablation results show that instruction-driven sampling improves prediction accuracy across all evaluated predictors, such as DLinear, iTransformer [2], PatchTST [5], TimeXer [8], and BPSE. MSE falls by 1.9–31.7% and MAE by 6.0–28.4% compared with time-driven sampling. For FineCAT's BPSE, the reductions are 7.3% and 11.1%, supporting instruction-count-based phase alignment. A granularity sweep further shows that FineCAT's 10^8-instruction interval balances phase sensitivity and overhead, while LP execution time varies by at most 0.9% in LLC-sensitive scenarios.

Acknowledgments

This work was supported by the Innovation Funding of ICT, CAS, China under Grant No. E661110.

References

[1] Bin Li, Yipeng Wang, Ren Wang, Charlie Tai, Ravi R. Iyer, Zhu Zhou, Andrew J. Herdrich, Tong Zhang, Ameer Haj-Ali, Ion Stoica, and Krste Asanović. 2020. RLDRM: Closed Loop Dynamic Cache Allocation with Deep Reinforcement Learning for Network Function Virtualization. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, Piscataway, NJ, USA, 335–343. doi:10.1109/NETSOFT48620.2020.9165471

[2] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *Proceedings of the International Conference on Learning Representations*. OpenReview.net.

[3] David Lo, Liqun Cheng, Rama K. Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. 2015. Heracles: improving resource efficiency at scale. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, June 13-17, 2015*, Deborah T. Marr and David H. Albonesi (Eds.). ACM, New York, NY, USA, 450–462. doi:10.1145/2749469.2749475

[4] Chunjie Luo, Xiwen He, Jianfeng Zhan, Lei Wang, Wanling Gao, and Jiahui Dai. 2020. Comparison and Benchmarking of AI Models and Frameworks on Mobile Devices. arXiv:2005.05085 [cs.LG] https://api.semanticscholar.org/CorpusID:218581849

[5] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A Time Series is Worth 64 Words: Long-Term Forecasting with Transformers. *arXiv preprint arXiv:2211.14730* (2022).

[6] Standard Performance Evaluation Corporation. 2017. SPEC CPU 2017 Benchmark Suite. https://www.spec.org/cpu2017.

[7] Qingsen Wang, Xu Liu, and Milind Chabbi. 2019. Featherlight Reuse-Distance Measurement. In *25th IEEE International Symposium on High Performance Computer Architecture, HPCA 2019, Washington, DC, USA, February 16-20, 2019*. IEEE, Piscataway, NJ, USA, 440–453. doi:10.1109/HPCA.2019.00056

[8] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. 2024. TimeXer: Empowering Transformers for Time Series Forecasting with Exogenous Variables. In *Advances in Neural Information Processing Systems*.

[9] Meng Xu, Robert Gifford, and Linh Thi Xuan Phan. 2019. Holistic multi-resource allocation for multicore real-time virtualization. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*. ACM, New York, NY, USA, 1–6. doi:10.1145/3316781.3317840

[10] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 9 (2023), 11121–11128. doi:10.1609/aaai.v37i9.26317

[11] Chuanqi Zhang, Xueqi Li, Ninghui Sun, Yungang Bao, and Sa Wang. 2024. LazyCAT: Efficient Fine-Grained Cache Partitioning with Two Boundaries. In *2024 IEEE International Conference on High Performance Computing and Communications (HPCC)*. IEEE, Piscataway, NJ, USA, 111–118. doi:10.1109/HPCC64274.2024.00025