Improving Multi-tenant NPU Efficiency via Decoupled Tiling and Adaptive Memory Allocation

Sanghyeon Lee
KAIST
Daejeon, Republic of Korea
leesh6796@casys.kaist.ac.kr

Jaehyuk Huh
KAIST
Daejeon, Republic of Korea
jhhuh@kaist.ac.kr

***Abstract*—Neural Processing Units (NPUs) achieve efficient inference using systolic arrays and scratch-pad memory (SPM), but existing multi-tenant approaches incur high context-switch overhead or resource underutilization. We propose an architecture–scheduler co-design that decouples execution from allocation granularity and dynamically reallocates SPM resources. Experimental results show reduced turnaround times and DRAM accesses, balancing efficiency, fairness, and throughput.**

***Index Terms*—Neural processing unit, multi-tenancy, spatial sharing, scratch-pad memory, dynamic scheduling**

I. Introduction

Neural Processing Units (NPUs) achieve exceptional energy efficiency in cloud inference by coupling systolic arrays with scratch-pad memory (SPM) for optimized matrix multiplication [1], [4]. However, rigid tile sizes and static allocation cause compute underutilization in multi-tenant settings. Temporal approaches like PREMA [2] incur context-switch penalties from scratch-pad checkpoints, while spatial methods like Planaria [3] impose layer-level scheduling, resulting in idle resources and inter-model interference. These limitations reflect a fundamental trade-off between locality and context-switch responsiveness, worsening with larger tiles.

This paper introduces a co-designed architecture and scheduling framework that resolves this tension through granularity decoupling. Our approach nests fine-grained execution sub-tiles within coarse-grained allocation tiles, preserving locality while enabling rapid context switching. We complement this with dynamic capacity-aware SPM allocation that exploits heterogeneous memory sensitivity profiles, redistributing capacity from saturated to memory-bound workloads. Experimental evaluation demonstrates significant reductions in turnaround time and DRAM traffic, validating improved efficiency and fairness in multi-tenant neural processing.

II. Background and Motivation

Neural Processing Units and Multi-tenancy Challenges. Neural Processing Units (NPUs) achieve high efficiency by pairing systolic arrays with scratch-pad memory (SPM), streaming tiled matrices from DRAM for fast multiply-accumulate operations. However, this rigidly couples tile sizes and resource allocations to single workloads, causing underutilization of computation and memory bandwidth in diverse cloud environments. Existing multi-tenant approaches [2], [3] incur significant overhead, either from frequent context switches (temporal sharing) or persistent inter-model interference (spatial sharing).

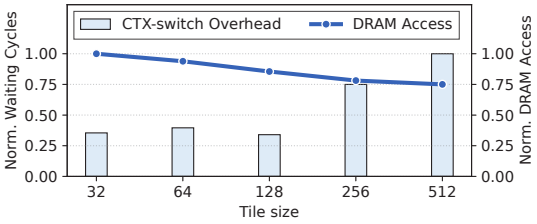


Fig. 1. Trade-off between context-switch overhead and DRAM accesses.

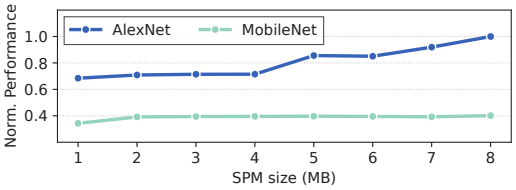
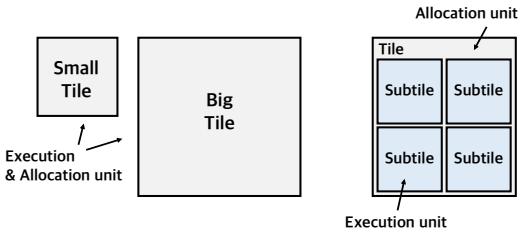


Fig. 2. Performance sensitivity of AlexNet and MobileNet to SPM capacity.

Spatial Sharing Limitations. Spatial multi-tenant NPUs partition systolic arrays into independent sub-arrays but suffer from layer-level scheduling granularity that creates efficiency bottlenecks. Each sub-array must complete entire layers before relinquishing control, forcing schedulers to wait for all sub-arrays before context switching. While this prevents mid-layer starvation, it strands early-finishing tenants and leaves cores idle during slowest layer drainage. The problem worsens with larger tiles, where cores either idle or checkpoint to DRAM, both adding latency and wasting resources.

Fundamental Trade-offs in Multi-tenant NPUs. Figure 1 shows that execution-tile width creates a critical trade-off between on-chip reuse and context-switch agility. As tile width grows from 32 to 512, DRAM transactions decrease by 25% but context-switch cycles more than double beyond width 256. Figure 2 reveals that static SPM partitioning wastes capacity—MobileNet saturates at 2 MB while AlexNet scales linearly in our 80 GB/s testbed. These findings motivate architecture-scheduler co-design that decouples execution from allocation tiles and dynamically reallocates SPM capacity from saturated to memory-hungry models.



(a) Unified Tile (b) Decoupled Tile

Fig. 3. Conceptual comparison of unified tiles versus decoupled tiles.

III. Design

A. Decoupled Tiling Architecture

Our approach separates allocation granularity from execution granularity to resolve the latency-locality trade-off in multi-tenant NPUs. Figure 3 contrasts conventional unified tiles with our decoupled design that nests fine-grain execution sub-tiles within coarse-grain allocation tiles. While unified tiling forces a choice between quick preemption (small tiles) and good locality (large tiles), our approach maintains large parent tiles in SPM for locality while executing smaller sub-tiles that enable context switches at finer granularity, simultaneously preserving high locality and minimizing context-switch overhead. The scheduler can preempt execution after any sub-tile completes without flushing the entire parent tile from SPM, enabling rapid tenant switching while preserving accumulated intermediate results. For example, during matrix multiplication, operand blocks loaded into the parent tile can be reused across multiple sub-tile computations before eviction, significantly reducing DRAM traffic compared to conventional approaches that reload data for each small tile.

B. Capacity-Aware SPM Allocation

DNN models exhibit vastly different SPM sensitivity profiles, with some networks saturating at few megabytes while others scaling almost linearly. We quantify these sensitivities through offline grid searches over core count and SPM capacity for benchmark workloads, producing a lookup table mapping ⟨core count, SPM size⟩ pairs to expected throughput. From this data, we construct an SPM Yielding Matrix that guides runtime allocation: the scheduler yields surplus capacity from saturated layers to those with high marginal gains. Despite potential underestimation of runtime overheads, capacity-aware reallocation consistently improves aggregate performance, establishing scratch-pad sensitivity as a powerful scheduling dimension for multi-tenant NPUs.

IV. Evaluation

A. Experimental Setup

Hardware platform. Our cycle-accurate simulator models a Planaria-style [3] spatial NPU at 1 GHz, comprising four 32×32 PE arrays arranged in a 4×4 configuration backed by Fission Pods. The chip integrates 12 MB SPM using weight-stationary dataflow with 16-bit precision, providing 80 and 200 GB/s bandwidth with 100-cycle DRAM latency.

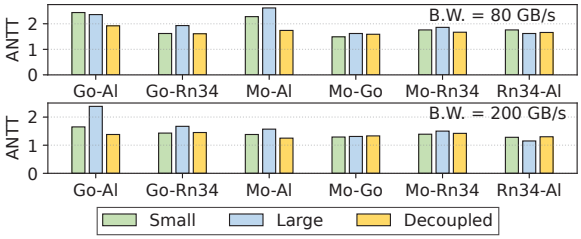


Fig. 4. ANTT across six workload pairs with small, large, and decoupled tiles under 80 GB/s and 200 GB/s memory bandwidth.

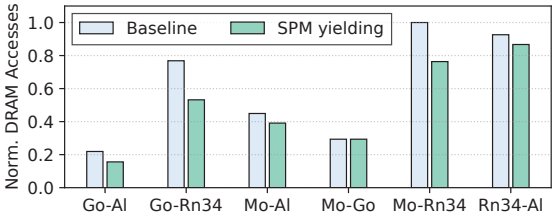


Fig. 5. Normalized DRAM accesses under baseline partitioning versus SPM-yielding.

Metric. We report Average Normalized Turnaround Time (ANTT) = $\frac{1}{n}\sum_{i=1}^{n}\frac{C_i^{\text{multi}}}{C_i^{\text{single}}}$, where C_i^{single} and C_i^{multi} are cycle counts for model i under single-tenant and multi-tenant execution. Lower ANTT indicates better aggregate quality of service. We additionally measure total DRAM accesses.

Workloads. Four convolutional networks comprise our benchmark: AlexNet (61M params), GoogLeNet (5M), MobileNet (4.2M), and ResNet-34 (63.5M).

B. Performance

Figure 4 evaluates three tiling strategies—Small, Large, and Decoupled—across dual-model workloads. At 80 GB/s, decoupled tiling reduces ANTT by 6% on average with maximum gains for reconfiguration-heavy pairs (*Go–Al*, *Mo–Al*) while trailing conventional approaches by only 4% in worst cases. At 200 GB/s, mean benefits decrease to 0.8% as DRAM penalties diminish, yet decoupling still provides up to 16% improvements for bandwidth-sensitive workloads with maximum degradation of 12%.

Figure 5 shows that dynamically reallocating SPM capacity reduces DRAM accesses by 17% on average, peaking at 31% for Go–Rn34. The Mo–Go pair shows negligible improvement as both models saturate at modest SPM sizes. These bandwidth-independent reductions translate directly to lower DRAM energy and increased tenant headroom, though latency benefits depend on double-buffering effectiveness.

V. Conclusion

We propose a multi-tenant NPU architecture–scheduler co-design that decouples execution from allocation granularity and dynamically manages SPM, improving efficiency.

VI. Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) funded by the Ministry of Science and ICT, Korea (RS-2024-00402898).

REFERENCES

[1] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[2] Y. Choi and M. Rhu, "PREMA: A predictive multi-task scheduling algorithm for preemptible neural processing units," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020.

[3] S. Ghodrati, B. H. Ahn, J. K. Kim, S. Kinzer, B. R. Yatham, N. Alla, H. Sharma, M. Alian, E. Ebrahimi, N. S. Kim *et al.*, "Planaria: Dynamic architecture fission for spatial multi-tenant acceleration of deep neural networks," in *53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020.

[4] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture (ISCA)*, 2017.