Appendix for SwFormer

A Discussions for Portability and Complexity

In this section, we compare the various tiling and scheduling optimizations in SwFormer and analyze their online and offline overhead.

Table 1. Comparison and Overhead Analysis of Optimizations in SwFormer

Method	Portability	Type	Time Complexity
Intra-op Tiling	Architecture Specific	Pad-free Tiling	$O(mn_1)$
		Tile-fusion	$O(m(n_1+n_2))$
Inter-op Scheduling	General	Dependent	$O(mn_1)$
		Independent-OPT2	$O(mn_1^2)$
		Independent-KA	$O(mn_1)$

A.1 Portability

As shown in Table 1, intra-op tiling optimizations are architecture-specific, as they are closely related to the hardware characteristics of the SW26010pro processor. In contrast, SwFormer's inter-op scheduling is more like a general-purpose optimization. The issue of underutilization during the execution of foundation models is not unique to the Sunway Supercomputer. It is also observed on other architectures such as GPUs. For example, cuSync [23] highlights that GEMM operators in large language models face challenges when partitioning tiles of the final wave across multiple thread blocks. FlashAttention-2 [29] also notes that limited attention heads can lead to parts of SMs idle.

Considering that our OPT2 strategy and KA strategy utilize independent operators to fill the idle portions of GEMM and attention operators, we believe that SwFormer's inter-op scheduling optimizations provide valuable insights for other architectures.

A.2 Overhead Analysis

Only offline overhead needs to be considered in the overhead analysis. The online overhead, which includes the runtime BFS-based graph traversal in the OPT2 strategy and the KA strategy, has a time complexity linear to the number of nodes in the computation graph. As the number of nodes in foundation models is relatively small, the runtime overhead can be negligible.

The offline overhead analysis of intra-op tiling optimizations is shown in Table 1. In Algorithm 1, a profile for GEMM of shape (M, N, K) requires both M and N to be divisible by the numbers in *Div_List*. Although the numbers in *Div_List* usually range from 1 to 128, the number of valid pad-free tiling types for a GEMM is limited to 50-100 due to the shape constraints of the micro-kernel. As for the tile-fusion strategy, the number of valid fused tiles can be even smaller for the following reasons. First, tile-fusion cannot be applied when the number of pad-free tiles is fewer than six such as $M_{\text{div}} = 2$ and $N_{\text{div}} = 2$. Second, the heuristic tile-fusion strategy has not been applied to some uncommon pad-free tiling shapes such as $(M_{\text{div}}, N_{\text{div}}) = (16, 16)$ and $(M_{\text{div}}, N_{\text{div}}) = (16, 32)$.

Here, we use n_1 to represent the number of valid pad-free tiles of each GEMM, n_2 to represent the number of valid tile-fusion strategies for each GEMM, and m to indicate the number of different GEMM shapes with target parallel configurations. Apparently, the time complexity of profiling for pad-free tiling is $O(mn_1)$, while the time complexity of profiling for tile-fusion is $O(m(n_1 + n_2))$.

For inter-op scheduling, the variable m similarly represents the number of GEMMs with different shapes. Besides, only pad-free tiles are employed in the inter-op scheduling. The reasons are as follows. First, our task queue cannot accurately determine the placement of each GEMM tile at runtime. Second, evaluation results in Section 5 show that tile-fusion provides limited performance improvement (around 5%). Therefore, not applying tile-fusion has minimal impact. Specifically, for dependent scheduling, where two GEMMs must use the same tile size, the complexity is $O(mn_1)$. In contrast, the OPT2 strategy, which allows GEMMs to use different tile sizes, has a complexity of $O(mn_1^2)$. Thus, as the number of independent operators scheduled simultaneously increases, the search space expands correspondingly. Potential approaches for reducing the profiling overhead include machine learning-based operator performance models [30] or leveraging multiple SW26010pro processors for parallel profiling.

The primary profiling overhead of the KA strategy involves filling the idle CGs of the attention operator by adjusting the tile sizes of GEMM and other operators. During the BFS in Algorithm 2, each GEMM is traversed only once. Therefore, the time complexity of the KA strategy remains $O(mn_1)$.

A.3 Experiment for Profiling Overhead

SwFormer's profiling overhead for the intra-op optimization is shown in Table 2, which corresponds to the experiment in Fig.**??**. For all parallel configurations in Fig.**??**, we first collect GEMM operators with different shapes, and the profiling time represents the total time required to profile all 168 GEMMs. For GPT-3 13B and 6.7B models, the profiling times of pad-free tiling are 86 seconds and 54 seconds, respectively. When the tile-fusion strategy is incorporated, the times increase to 100 seconds and 63 seconds, respectively. Overall, the average search space size of all GEMM operators is around 50, which enables efficient determination of GEMM tiling strategies for various parallel configurations. Besides, our heuristic tile-fusion strategy delivers performance improvements without introducing much additional overhead.

Table 2. Profiling Overhead of Intra-op GEMM Tiling

Model	GEMM Number	Pad-free Tiling		Tile-fusion	
		Space Size	Time (s)	Space Size	Time (s)
GPT-3 13B	168	6879	86	1881	100
GPT-3 6.7B	168	5966	54	1773	63

The profiling overhead of the inter-op optimization is shown in Table 3, which corresponds to the experiment in Fig.**??**. We do not present the profiling times for the inter-op dependent scheduling, as it only involves two consecutive linear layers in the MLP layer and requires little overhead. Compared with intra-op profiling, the search space sizes of the OPT2 strategy increase to about 45000. However, the profiling time does not increase much. The reason is that intra-op profiling involves large GEMM operators in the logit layer [15], which contributes significantly to the profiling time. Additionally, the profiling times and search space sizes of the KA strategy are similar to those of intra-op profiling, which demonstrates that SwFormer's profiling-based approach can efficiently determine tiling and scheduling strategies.

Table 3. Profiling Overhead of Inter-op Scheduling

Model	OPT2		KA	
	Space Size	Time (s)	Space Size	Time (s)
GPT-3 13B backward	54248	215	8232	65
GPT-3 6.7B backward	34416	99	6479	36
MMGPT 13B forward	49812	178	9210	58
MMGPT 6.7B forward	38290	100	8434	40

References

[1] Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X H, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, Uszkoreit J, Houlsby N. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929, 2020. https://arxiv.org/abs/2010.11929, Jan. 2025.

[2] Brown T B, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler D M, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D. Language models are few-shot learners. In *Proc. the 34rd International Conference on Neural Information Processing Systems*, Dec. 2020, pp.1877-1901. DOI:

10.5555/3495724.3495883.

[3] Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M A, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, Lample G. Llama: Open and efficient foundation language models. arXiv:2302.13971, 2023. https://arxiv.org/abs/2302.13971, Jan. 2025.

[4] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser Ł, Polosukhin I. Attention is all you need. In *Proc. the 31st International Conference on Neural Information Processing Systems*, Dec. 2017, pp.6000-6010. DOI: 10.5555/3295222.3295349.

[5] Gao J G, Zheng F, Qi F B, Ding Y J, Li H L, Lu H S, He W Q, Wei H M, Jin L F, Liu X, Gong D Y, Wang F, Zheng Y, Sun H H, Zhou Z, Liu Y, You H T. Sunway supercomputer architecture towards exascale computing: analysis and practice. *Science China Information Sciences*, 2021, 64(4): 141101. DOI: 10.1007/s11432-020-3104-7.

[6] Zhao X C, Li M F, Xiao Q, Chen J S, Wang F, Shen L, Zhao M J, Wu W H, An H, He L X, Liang X. AI for quantum mechanics: high performance quantum many-body simulations via deep learning. In *Proc. the 2022 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2022, pp. 1-15. DOI: 10.1109/SC41404.2022.00053.

[7] Ma Z X, He J A, Qiu J Z, Cao H Q, Wang Y W, Sun Z B, Zheng L Y, Wang H J, Tang S Z, Zheng T Y, Lin J Y, Feng G Y, Huang Z Q, Gao J, Zeng A H, Zhang J W, Zhong R X, Shi T H, Liu S, Zheng W M, Tang J, Yang H X, Liu X, Zhai J D, Chen W G. BaGuaLu: targeting brain scale pretrained models with over 37 million cores. In *Proc. the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Apr. 2022, pp. 192-204. DOI: 10.1145/3503221.3508417.

[8] Zhao Y, Zheng J P, Fu H H, Wu W Z, Gao J, Chen M X, Zhang J X, Zhang L X, Dong R M, Du Z R, Liu S, Liu X, Zhang S Q, Yu L. SW-LCM: A Scalable and Weakly-supervised Land Cover Mapping Method on a New Sunway Supercomputer. In *Proc. the 2023 IEEE International Parallel and Distributed Processing Symposium*, May. 2023, pp. 657-667. DOI: 10.1109/IPDPS54959.2023.00071.

[9] Fang J R, Fu H H, Zhao W L, Chen B W, Zheng W J, Yang G W. swdnn: A library for accelerating deep learning applications on sunway taihulight. In *Proc. the 2017 IEEE International Parallel and Distributed Processing Symposium*, May. 2017, pp. 615-624. DOI: 10.1109/IPDPS.2017.20.

[10] Gao W, Fang J R, Zhao W L, Yang J Z, Wang L, Gan L, Fu H H, Yang G W. swatop: Automatically optimizing deep learning operators on sw26010 many-core processor. In *Proc. the 48th International Conference on Parallel Processing*, Aug. 2019, pp. 1-10. DOI: 10.1145/3337821.3337883.

[11] Tao X H, Zhu Y, Wang B Y, Xu J L, Pang J M, Zhao J. Automatically generating high-performance matrix multiplication kernels on the latest sunway processor. In *Proc. the 51th International Conference on Parallel Processing*, Aug. 2022, pp. 1-12. DOI: 10.1145/3545008.3545031.

[12] Liu F F, Ma W J, Zhao Y W, Chen D K, Hu Y, Lu Q L, Yin W W, Yuan X H, Jiang L J, Yan H, Li M, Wang H S, Wang X Y, Yang C. xMath2.0: a high-performance extended math library for SW26010-Pro many-core processor. *CCF Transactions on High Performance Computing*, 2023, 5(1): 56-71. DOI: 10.1007/s42514-022-00126-8.

[13] Liu S, Gao J, Liu X, Huang Z Q, Zheng T Y. Establishing high performance AI ecosystem on Sunway platform. *CCF Transactions on High Performance Computing*, 2021, 3(3): 224-241. DOI: 10.1007/s42514-021-00072-x.

[14] Shoeybi M, Patwary M, Puri R, LeGresley P, Casper J, Catanzaro B. Megatron-lm: Training multi-billion parameter language models using model parallelism. arXiv:1909.08053, 2019. https://arxiv.org/abs/1909.08053, Jan. 2025.

[15] Narayanan D, Shoeybi M, Casper J, LeGresley P, Patwary M, Korthikanti V, Vainbrand D, Kashinkunti P, Bernauer J, Catanzaro B, Phanishayee A, Zaharia M. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proc. the 2021 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2021, pp. 1-15. DOI: 10.1145/3458817.3476209.

[16] Li S G, Liu H X, Bian Z D, Fang J R, Huang H C, Liu Y L, Wang B X, You Y. Colossal-ai: A unified deep learning system for large-scale parallel training. In *Proc. the 52nd International Conference on Parallel Processing*, Aug. 2023, pp. 766-775. DOI: 10.1145/3605573.3605613.

[17] Wu R H, Zhu X Y, Chen J S, Liu S, Zheng T Y, Liu X, An H. SWattention: designing fast and memory-efficient attention for a new Sunway Supercomputer. *The Journal of Supercomputing*, 2024, 80(10): 13657-13680. DOI: 10.1007/s11227-024-05890-8.

[18] Dao T, Fu D Y, Ermon S, Rudra A, Ré C. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Proc. the 36rd International Conference on Neural Information Processing Systems*, Nov. 2022, pp.16344-16359. DOI: 10.5555/3600270.3601459.

[19] Ma L X, Xie Z Q, Yang Z, Xue J L, Miao Y S, Cui W, Hu W X, Yang F, Zhang L T, Zhou L D. Rammer: Enabling holistic deep learning compiler optimizations with rtasks. In *Proc. the 14th USENIX Conference on Operating Systems*

Design and Implementation, Nov. 2020, pp.881-897. DOI: 10.5555/3488766.3488816.

[20] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z M, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J J, Chintala S. PyTorch: An imperative style, high-performance deep learning library. In *Proc. the 33rd International Conference on Neural Information Processing Systems*, Dec. 2019, pp.8026-8037. DOI: 10.5555/3454287.3455008.

[21] Shi Y N, Yang Z, Xue J L, Ma L X, Xia Y Q, Miao Z M, Guo Y X, Yang F, Zhou L D. Welder: Scheduling deep learning memory access via tile-graph. In *Proc. the 17th USENIX Conference on Operating Systems Design and Implementation*, Jul. 2023, pp.701-718.

[22] Ng K K W, Demoulin H M, Liu V. Paella: Low-latency model serving with software-defined gpu scheduling. In *Proc. the 29th Symposium on Operating Systems Principles*, Oct. 2023, pp.595-610. DOI: 10.1145/3600006.3613163.

[23] Jangda A, Maleki S, Dehnavi M M, Musuvathi M, Saarikivi O. A Framework for Fine-Grained Synchronization of Dependent GPU Kernels. In *Proc. the 2024 IEEE/ACM International Symposium on Code Generation and Optimization*, Mar. 2024, pp.93-105. DOI: 10.1109/CGO57630.2024.10444873.

[24] He K M, Zhang X Y, Ren S Q, Sun J. Deep residual learning for image recognition. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.770-778. DOI: 10.1109/CVPR.2016.90.

[25] Qi P H, Wan X Y, Huang G X, Lin M. Zero Bubble (Almost) Pipeline Parallelism. In *Proc. the 12th International Conference on Learning Representations*, May. 2024.

[26] Radford A, Kim J W, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I. Learning transferable visual models from natural language supervision. In *Proc. the 2021 International conference on machine learning*, Jul. 2021, pp.8748-8763.

[27] Huang J, Zhang Z, Zheng S, Qin F, Wang Y D. DISTMM: Accelerating Distributed Multimodal Model Training. In *Proc. the 21st USENIX Symposium on Networked Systems Design and Implementation*, Apr. 2024, pp.1157-1171.

[28] Gupta K, Stuart J A, Owens J D. A study of persistent threads style GPU programming for GPGPU workloads. In *Proc. the 2012 Innovative Parallel Computing*, May. 2012, pp.1-14. DOI: 10.1109/InPar.2012.6339596.

[29] Dao T. Flashattention-2: Faster attention with better parallelism and work partitioning. arXiv:2307.08691, 2023. https://arxiv.org/abs/2307.08691, Jan. 2025.

[30] Yu G X, Gao Y B, Golikov P, Pekhimenko G. Habitat: A Runtime-Based computational performance predictor for deep neural network training. In *Proc. the 2021 USENIX Annual Technical Conference*, Jul. 2021, pp.503-521.

[31] Fu H H, Liao J F, Yang J Z, Wang L N, Song Z Y, Huang X M, Yang C, Xue W, Liu F F, Qiao F L, Zhao W, Yin X Q, Hou C F, Zhang C L, Ge W, Zhang J, Wang Y G, Zhou C B, Yang G W. The Sunway TaihuLight supercomputer: system and applications. *Science China Information Sciences*, 2016, 59: 1-16. DOI: 10.1007/s11432-016-5588-7.

[32] Lin R F, Yuan X H, Xue W, Yin W W, Yao J N, Shi J D, Sun Q, Song C B, Wang F. 5 ExaFlop/s HPL-MxP Benchmark with Linear Scalability on the 40-Million-Core Sunway Supercomputer. In *Proc. the 2023 International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov. 2023, pp. 1-13. DOI: 10.1145/3581784.3607030.