Journal2

Raja Kantheti

September 15, 2024

1 PART 1

The browse analysis yielded several valid results, as to how other researchers are tackling the problem. It also gave me a lot of perspective as how the problem evolved and also about the trade-offs that organizations have endured while trying to mitigate this problem. The part of the assignment where I went through journals or conferences gave me an idea of the pace of the current innovation and also determined the novelty factor of my approach towards solving this problem.

1.1 The three papers I chose:

https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4666
https://www2.eecs.berkeley.edu/Pubs/TechRpts/2017/EECS-2017-157.html
https://dl.acm.org/doi/10.1145/3566053

1.2 Rationale of why I chose these three papers:

I needed a compilation of generic and most used branch prediction techniques used in contemporary Architectures, and I also needed the latest Cores that are being researched to check for compatibility with my approach and also I needed to see the research on mitigations of side-channel attacks on the target architecture to see how various approaches towards that particular problem can improve my approach. I used the relevant keywords and found many papers to be relevant.

These Three papers gives me the history of Branch Prediction techniques until 2018, Recent Development on the mitigating the SPECTRE attacks which is a known consequence of speculative execution which is a possible result of a Branch Prediction miss and an insight into how modern ISA can be tested against my proposed theory of branch prediction.

2 PART 2

The five papers I skimmed out of the 25 papers are:

2.1 Paper 1:

Survey and Comparison of Pipeline of Some RISC and CISC System Architectures

Notes: Compares RISC and CISC, Super Pipelining: Two-stage super pipelined microprocessor, the task completed by each pipelined segment can be divided into two non overlapping parts and each part can be executed within half a clock cycle.

2.2 Paper 2:

The optimum pipeline depth for a microprocessor

Notes: Old Paper, Used a proprietary simulation system to simulate various versions of Pipeline for varying depths. tested mainly for SPEC benchmark suite. Need to study the section titled "Theory" more. Not really sure about how the methodology can relate to the modern workloads.

2.3 Paper 3:

Dynamic branch prediction for a VLIW processor

Notes: Selective Branch Prediction for VLIW Processors. mixinf predicted and delayed branches yielded performance improvements.

2.4 Paper 4:

Reducing the branch penalty in pipelined processors

Notes: "Branch bypass and multiple prefetch" this is what you are thinking along with multiple decode units. Read more about this as this contains some methods to quantify such hypothetical machines.

2.5 paper 5:

Branch Prediction — RISCV-BOOM documentation

Notes: Clear explanation aboout the BOOM Outof order RISCV core with emphasis on it's branch prediction strategy.

The Remaining 21 Papers are Group Cited *here* [1–21]

3 PART 3

3.1 Table of Contents

ACM Transactions on Architecture and Code Optimization1

Why I chose this Journal: I chose ACM TACO as the journal because of its diversity in topics and specialized focus on subjects like parallel computing, pipeline optimizations, Computer Architecture, etc.

3.2 My Learning process:

Browsing, Skimming, and Scanning of the research papers were very productive and liberating. The time elapsed between starting and ending the assignment was the most I had ever done since I thought of my thesis problem. The process of searching for papers and journals, which gave me more insight into what was happening in the field, gave me a broader perspective on my approach to solving the problem.

Although it was a tedious task at the beginning of the assignment, I got into flow in part two when I started skimming the 5 papers. I learned how to search papers and what to look for when trying to find relevance to my research. I saw how much I could expand my research to solve other problems and find new evaluation methods if I ever got results. I also understood how I could better express my research to my advisor.

3.3 Raw notes from critical reading three papers:

Paper 1: Delay-on-Squash: Stopping Microarchitectural Replay Attacks in Their Tracks

Notes: Critical REad, 45 sec, Scan it, Talks about halting/disallowing speculative execution after page miss, very closely related to your work.The poinnt is not to significantly impact the performance but to stop replay attacks by tracking the squashes. Bloom filters are suedd to deteect if a PC is ina set after hashing it if yes, then a bulk reset is the only way to reset the filter which will change the pc. approach is to ise two bloom fileters and switch between them. at ny point in time only fileter is tagged as active. the other one will wait to be cleard. after the the active filter is cleared the status is altered on both filters. take figure 4 for IPC metric.

Figure 1: Table of Contents.

Paper 2: SpecTerminator: Blocking Speculative Side Channels Based on Instruction Classes on RISC-V

Notes: Critical READ,45 seec, scan it, talks about bblocking SPECTRE attacks in RISCV. Need to get the grip on different types of SPECTRE attacks. You may actually be mitigating only one based on the text. Delayed execution strategies, TLB request ignoring, Classifiation of LOAD instructions in ISA whic are speculative depended baased on the architectural changes., TAble 1 has existing hardware defecnecs against spectre attacks, need to critically read those in hte coming days. Experimental setup is unique instead of using genric simiulation like gem5 using FPGA simulation platform will give more access towards the architectural changes ? this doesn't mean it is biased, it is required for the analysis of the experiment results. I donno BOOM might not have been the best processor to choose, a multiple issue inorder may be would've yielded different results ? Finally, SPECTerminator has a lowe performance ocverheard than other hardwaare or OS based techniques.

Paper 3: The Impact of Page Size and Microarchitecture on Instruction Address Translation Overhead

Notes: Critical rEad, 2:00, Scan it, Talks about how page size and pipeline can impact hardware overhead of address translation. Quatification of different TLB organization based on overhead. TLB overhead is about 13.44

References

[1] Augustus K. Uht. Disjoint Eager Execution: What It Is / What It Is Not. *ACM SIGARCH Cmputer Architecture News*, 30(1):12–14, March 2002.

[2] Linknath Surya Balasubramanian. *Towards No-Penalty Control Hazard Handling in RISC Architecture Microcontrollers*. Thesis, Purdue University Graduate School, September 2024.

[3] 3. Branch Prediction Unit — MARSS-RISCV 4.1a documentation. https://marss-riscv-docs.readthedocs.io/en/latest/sections/branch-pred.html.

[4] B. Calder and D. Grunwald. Fast and accurate instruction fetch and branch prediction. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, ISCA '94, pages 2–11, Washington, DC, USA, April 1994. IEEE Computer Society Press.

[5] P. G. Emma and E. S. Davidson. Characterization of branch and data dependencies on programs for evaluating pipeline performance. *IEEE Trans. Comput.*, 36(7):859–875, July 1987.

[6] Marius Evers, Po-Yung Chang, and Yale N. Patt. Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches. *SIGARCH Comput. Archit. News*, 24(2):3–11, May 1996.

[7] Dmitry Evtyushkin, Ryan Riley, Nael CSE and ECE Abu-Ghazaleh, and Dmitry Ponomarev. BranchScope: A New Side-Channel Attack on Directional Branch Predictor. *SIGPLAN Not.*, 53(2):693–707, March 2018.

[8] A. Hartstein and Thomas R. Puzak. The optimum pipeline depth for a microprocessor. *SIGARCH Comput. Archit. News*, 30(2):7–13, May 2002. Old Paper

Used a proprietary simulation system to simulate various versions of Pipeline for varying depths. tested mainly for SPEC benchmark suite.

Need to study the section titled "Theory" more.

Not really sure about how the methodology can relate to the modern workloads.

[9] Yan He and Xiangning Chen. Survey and Comparison of Pipeline of Some RISC and CISC System Architectures. In *2023 8th International Conference on Computer and Communication Systems (ICCCS)*, pages 785–790, April 2023. Compares RISC and CISC

Super Pipelining: Two-stage super pipelined microprocessor, the task completed by each pipelined segment can be divided into two non overlapping parts and each part can be executed within half a clock cycle.

[10] John D. Johnson. Branch predication using large self history. Technical Report, Stanford University, Stanford, CA, USA, November 1992.

[11] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *Commun. ACM*, 63(7):93–101, June 2020.

[12] Esmaeil Mohammadian Koruyeh, Khaled N. Khasawneh, Chengyu Song, and Nael Abu-Ghazaleh. Spectre Returns! Speculation Attacks using the Return Stack Buffer. In *12th USENIX Workshop on Offensive Technologies (WOOT 18)*, 2018.

[13] Konstantin I. Kostromitin, Boris N. Dokuchaev, and Danila A. Kozlov. Analysis of the Most Common Software and Hardware Vulnerabilities in Microprocessor Systems. In *2020 International Russian Automation Conference (RusAutoCon)*, pages 1031–1036, September 2020.

[14] Peinan Li, Lutan Zhao, Rui Hou, Lixin Zhang, and Dan Meng. Conditional Speculation: An Effective Approach to Safeguard Out-of-Order Execution Against Spectre Attacks. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 264–276, February 2019.

[15] Chit-Kwan Lin and Stephen J. Tarsa. Branch Prediction Is Not a Solved Problem: Measurements, Opportunities, and Future Directions. https://arxiv.org/abs/1906.08170v1, June 2019. Branch Prediction Championship ?

Classified certain control flows as rare branches and concluded rare branches have worst statistics.

used TAGE as a reference and tested on several benchmark suites which are relatively new.

Talks about how incresing the branch prediction capacity of a pipeline doesn't yield significantly better results on some intel processor.

Shows how several BP techniques scale in the Itnel processors.

[16] Giorgi Maisuradze and Christian Rossow. Speculose: Analyzing the Security Implications of Speculative Execution in CPUs. https://arxiv.org/abs/1801.04084v1, January 2018.

[17] Milad Mohammadi, Song Han, Tor M. Aamodt, and William J. Dally. On-Demand Dynamic Branch Prediction. *IEEE Computer Architecture Letters*, 14(1):50–53, January 2015.

[18] Static methods in hybrid branch prediction | IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/abstract/document/727254.

[19] Viktor Vitek. Validating Side Channel models in RISC-V using Model-Based Testing.

[20] David W Wall. Speculative Execution and Instruction-Level Parallelism.

[21] Ruijian Zhang. *A Hybrid Branch Prediction Method: An Integration of Software and Hardware Techniques*. PhD thesis, University of Houston, USA, 2002.

[22] Branch Prediction — RISCV-BOOM documentation. https://docs.boom-core.org/en/latest/sections/branch-prediction/. Clear explanation aboout the BOOM Outof order RISCV core with emphasis on it's branch prediction strategy.

[23] J. Hoogerbrugge. Dynamic branch prediction for a VLIW processor. In *Proceedings 2000 International Conference on Parallel Architectures and Compilation Techniques (Cat. No.PR00622)*, pages 207–214, October 2000. Selective Branch Prediction for VLIW Processors. mixinf predicted and delayed branches yielded performance improvements.

[24] D. J. Lalja. Reducing the branch penalty in pipelined processors. *Computer*, 21(7):47–55, July 1988. "Branch bypass and multiple prefetch" this is what you are thinking along with multiple decode units. Read more about this as this contains some methods to quantify such hypothetical machines.

[25] Muhammad Waqar Azhar, Madhavan Manivannan, and Per Stenström. Approx-RM: Reducing Energy on Heterogeneous Multicore Processors under Accuracy and Timing Constraints. *ACM Trans. Archit. Code Optim.*, 20(3):44:1–44:25, July 2023. 15 secs, Trashit, Not related.

[26] Hadjer Benmeziane, Hamza Ouarnoughi, Kaoutar El Maghraoui, and Smail Niar. Multi-objective Hardware-aware Neural Architecture Search with Pareto Rank-preserving Surrogate Models. *ACM Trans. Archit. Code Optim.*, 20(2):29:1–29:21, April 2023. 20 secs, Trashit, Not related.

[27] Dongwei Chen, Dong Tong, Chun Yang, Jiangfang Yi, and Xu Cheng. FlexPointer: Fast Address Translation Based on Range TLB and Tagged Pointers. *ACM Trans. Archit. Code Optim.*, 20(2):30:1–30:24, March 2023. 50 secs, Scan it, Address translation methodology in TLB.

[28] Ruobing Chen, Haosen Shi, Jinping Wu, Yusen Li, Xiaoguang Liu, and Gang Wang. Jointly Optimizing Job Assignment and Resource Partitioning for Improving System Throughput in Cloud Datacenters. *ACM Trans. Archit. Code Optim.*, 20(3):34:1–34:24, July 2023. 40secs, Trashit, Not related.

[29] Zhangyu Chen, Yu Hua, Luochangqi Ding, Bo Ding, Pengfei Zuo, and Xue Liu. Lock-Free High-performance Hashing for Persistent Memory via PM-aware Holistic Optimization. *ACM Trans. Archit. Code Optim.*, 20(1):5:1–5:26, November 2022. 30 sec, Trash it, Not related.

[30] Jingwen Du, Fang Wang, Dan Feng, Changchen Gan, Yuchao Cao, Xiaomin Zou, and Fan Li. Fast One-Sided RDMA-Based State Machine Replication for Disaggregated Memory. *ACM Trans. Archit. Code Optim.*, 20(2):31:1–31:25, April 2023.

[31] Furkan Eris, Marcia Louis, Kubra Eris, José Abellán, and Ajay Joshi. Puppeteer: A Random Forest Based Manager for Hardware Prefetchers Across the Memory Hierarchy. *ACM Trans. Archit. Code Optim.*, 20(1):19:1–19:25, December 2022. 30 secs, Scan it, a new IF block block design for each level of memory, related to your approach of solving your problem of branch prediction.

[32] Vinicius Espindola, Luciano Zago, Hervé Yviquel, and Guido Araujo. Source Matching and Rewriting for MLIR Using String-Based Automata. *ACM Trans. Archit. Code Optim.*, 20(2):22:1–22:26, March 2023. 2:00, Trash it, talks about MLIR in the perspective of compilers, and translators.

[33] Ashish Gondimalla, Jianqiao Liu, Mithuna Thottethodi, and T. N. Vijaykumar. Occam: Optimal Data Reuse for Convolutional Neural Networks. *ACM Trans. Archit. Code Optim.*, 20(1):12:1–12:25, December 2022. 12 secs, Trash it, Not related.

[34] Dong Huang, Dan Feng, Qiankun Liu, Bo Ding, Wei Zhao, Xueliang Wei, and Wei Tong. SplitZNS: Towards an Efficient LSM-Tree on Zoned Namespace SSDs. *ACM Trans. Archit. Code Optim.*, 20(3):45:1–45:26, August 2023. 1:15. Trashit, Related to SSDs.

[35] Suyeon Hur, Seongmin Na, Dongup Kwon, Joonsung Kim, Andrew Boutros, Eriko Nurvitadhi, and Jangwoo Kim. A Fast and Flexible FPGA-based Accelerator for Natural Language Processing Neural Networks. *ACM Trans. Archit. Code Optim.*, 20(1):11:1–11:24, February 2023. 10 secs, Trash it, Not related.

[36] Jiazhi Jiang, Zijian Huang, Dan Huang, Jiangsu Du, Lin Chen, Ziguan Chen, and Yutong Lu. Hierarchical Model Parallelism for Optimizing Inference on Many-core Processor via Decoupled 3D-CNN Structure. *ACM Trans. Archit. Code Optim.*, 20(3):42:1–42:21, July 2023. 2:00, trashit, not related and you couldn't make sense of it.

[37] Hai Jin, Zhuo He, and Weizhong Qiang. SpecTerminator: Blocking Speculative Side Channels Based on Instruction Classes on RISC-V. *ACM Trans. Archit. Code Optim.*, 20(1):15:1–15:26, February 2023. Critical READ,45 seec, scan it, talks about bblocking SPECTRE attacks in RISCV. Need to get the grip on different types of SPECTRE attacks. You may actually be mitigating only one based on the text. Delayed execution strategies, TLB request ignoring, Classifiation of LOAD instructions in ISA whic are speculative depended baased on the architectural changes., TAble 1 has existing hardware defecnecs against spectre attacks, need to critically read those in hte coming days. Experimental setup is unique instead of using genric simiulation like gem5 using FPGA simulation platform will give more access towards the architectural changes ? this doesn't mean it is biased, it is required for the analysis of the experiment results. I donno BOOM might not have been the best processor to choose, a multiple issue inorder may be would've yielded different results ? Finally, SPECTerminator has a lowe performance ocverheard than other hardwaare or OS based techniques.

[38] Ivan Korostelev, João P. L. De Carvalho, José Moreira, and José Nelson Amaral. YaConv: Convolution with Low Cache Footprint. *ACM Trans. Archit. Code Optim.*, 20(1):18:1–18:18, February 2023. 2:00, Trash it, Kinda related but talks about a new algorithm to compute convolution in cnns.

[39] Alexander Krolik, Clark Verbrugge, and Laurie Hendren. rNdN: Fast Query Compilation for NVIDIA GPUs. *ACM Trans. Archit. Code Optim.*, 20(3):41:1–41:25, July 2023. 30 secs, Trash it, Not related.

[40] Yi Liang, Shaokang Zeng, and Lei Wang. Quantifying Resource Contention of Co-located Workloads with the System-level Entropy. *ACM Trans. Archit. Code Optim.*, 20(1):10:1–10:25, February 2023. 30 secs, Trash it, Resource utilization improvement in key-value like data wrt Datacenters.

[41] Qiaoyi Liu, Jeff Setter, Dillon Huff, Maxwell Strange, Kathleen Feng, Mark Horowitz, Priyanka Raina, and Fredrik Kjolstad. Unified Buffer: Compiling Image Processing and Machine Learning Applications to Push-Memory Accelerators. *ACM Trans. Archit. Code Optim.*, 20(2):26:1–26:26, March 2023. 2:00, Treshit, Not related.

[42] Thomas Luinaud, J. M. Pierre Langlois, and Yvon Savaria. Symbolic Analysis for Data Plane Programs Specialization. *ACM Trans. Archit. Code Optim.*, 20(1):1:1–1:21, November 2022. 1:00, traash it, Processing datapackets using FPGA.

[43] Wenjing Ma, Fangfang Liu, Daokun Chen, Qinglin Lu, Yi Hu, Hongsen Wang, and Xinhui Yuan. An Optimized Framework for Matrix Factorization on the New Sunway Many-core Platform. *ACM Trans. Archit. Code Optim.*, 20(2):23:1–23:24, March 2023. 2:00, Trashit, Talsk about sunway processor as a whole.

[44] Aristeidis Mastoras, Sotiris Anagnostidis, and Albert-Jan N. Yzelman. Design and Implementation for Nonblocking Execution in GraphBLAS: Tradeoffs and Performance. *ACM Trans. Archit. Code Optim.*, 20(1):6:1–6:23, November 2022. 35 secs, scan it, could be useful as you can see how the pipeline reacts to multi threading and it's execution flow.

[45] Francesco Minervini, Oscar Palomar, Osman Unsal, Enrico Reggiani, Josue Quiroga, Joan Marimon, Carlos Rojas, Roger Figueras, Abraham Ruiz, Alberto Gonzalez, Jonnatan Mendoza, Ivan Vargas, César Hernandez, Joan Cabre, Lina Khoirunisya, Mustapha Bouhali, Julian Pavon, Francesc Moll, Mauro Olivieri, Mario Kovac, Mate Kovac, Leon Dragic, Mateo Valero, and Adrian Cristal. Vitruvius+: An Area-Efficient RISC-V Decoupled Vector Coprocessor for High Performance Computing Applications. *ACM Trans. Archit. Code Optim.*, 20(2):28:1–28:25, March 2023. 2:00, Definite Scan, HPC through modification of ISA RISC-Vfor Exascale systems.

[46] Chandra Sekhar Mummidi and Sandip Kundu. ACTION: Adaptive Cache Block Migration in Distributed Cache Architectures. *ACM Trans. Archit. Code Optim.*, 20(2):25:1–25:19, March 2023. 45 sec, SCan it, Could be potentially used to mitigate Spec. Execution.

[47] Ataberk Olgun, Juan Gómez Luna, Konstantinos Kanellopoulos, Behzad Salami, Hasan Hassan, Oguz Ergin, and Onur Mutlu. PiDRAM: A Holistic End-to-end FPGA-based Framework for Processing-in-DRAM. *ACM Trans. Archit. Code Optim.*, 20(1):8:1–8:31, November 2022. 58sec, Scan it, Near memory computing paradigm using simulators, could be an interesting study.

[48] Bo Peng, Yaozu Dong, Jianguo Yao, Fengguang Wu, and Haibing Guan. FlexHM: A Practical System for Heterogeneous Memory with Flexible and Efficient Performance Optimizations. *ACM Trans. Archit. Code Optim.*, 20(1):13:1–13:26, December 2022. 1:30, scan it, Talks about memory optimization for performance gains using Flexhm in optane memory system and NVM. Diversity in testing with benchmarks. Experimental set up not biased.

[49] Víctor Pérez, Lukas Sommer, Victor Lomüller, Kumudha Narasimhan, and Mehdi Goli. User-driven Online Kernel Fusion for SYCL. *ACM Trans. Archit. Code Optim.*, 20(2):21:1–21:25, March 2023. 1:00, Trasahit, Talks about the parallel programming appeoch from kernel's perpective.

[50] Sooraj Puthoor and Mikko H. Lipasti. Turn-based Spatiotemporal Coherence for GPUs. *ACM Trans. Archit. Code Optim.*, 20(3):33:1–33:27, July 2023. 1:30, trashit, spatiotemporal coherence in caches, a very interesting read, you should try.

[51] Gokul Subramanian Ravi, Tushar Krishna, and Mikko Lipasti. TNT: A Modular Approach to Traversing Physically Heterogeneous NOCs at Bare-wire Latency. *ACM Trans. Archit. Code Optim.*, 20(3):35:1–35:25, July 2023. 1:30, Not sure about this, talks about esthablishing a communication networks between the many networks of exa scale systems.

[52] Benjamin Reber, Matthew Gould, Alexander H. Kneipp, Fangzhou Liu, Ian Prechtl, Chen Ding, Linlin Chen, and Dorin Patru. Cache Programming for Scientific Loops Using Leases. *ACM Trans. Archit. Code Optim.*, 20(3):39:1–39:25, July 2023. 1:30, treashit, talks about securing the cache in GPU and CPU using leasing approach. Very interesting.

[53] Abdul Rasheed Sahni, Hamza Omar, Usman Ali, and Omer Khan. ASM: An Adaptive Secure Multicore for Co-located Mutually Distrusting Processes. *ACM Trans. Archit. Code Optim.*, 20(3):32:1–32:24, July 2023. 2:00, scanit, secure orocessor ddesign for evaluating the execution processes in distributed computingn paradigm and virtualizzation too.

[54] Christos Sakalis, Stefanos Kaxiras, and Magnus Själander. Delay-on-Squash: Stopping Microarchitectural Replay Attacks in Their Tracks. *ACM Trans. Archit. Code Optim.*, 20(1):9:1–9:24, November 2022. Critical REad, 45 sec, Scan it, Talks about halting/disallowing speculative execution after page miss, very closely related to your work.The poinnt is not to significantly impact the performance but to stop replay attacks by tracking the squashes. Bloom filters are suedd to deteect if a PC is ina set after hashing it if yes, then a bulk reset is the only way to reset the filter which will change the pc. approach is to ise two bloom fileters and switch between them. at ny point in time only fileter is tagged as active. the other one will wait to be cleard. after the the active filter is cleared the status is altered on both filters. take figure 4 for IPC metric.

[55] Manuela Schuler, Richard Membarth, and Philipp Slusallek. XEngine: Optimal Tensor Rematerialization for Neural Networks in Heterogeneous Environments. *ACM Trans. Archit. Code Optim.*, 20(1):17:1–17:25, December 2022. 1:12, Trash it, talks about memory effeciency in tensors.

[56] Nilesh Rajendra Shah, Ashitabh Misra, Antoine Miné, Rakesh Venkat, and Ramakrishna Upadrasta. BullsEye : Scalable and Accurate Approximation Framework for Cache Miss Calculation. *ACM Trans. Archit. Code Optim.*, 20(1):2:1–2:28, November 2022. 1:47, Scan it, Talks about cache miss calculations in the perspective of math resulting in an approach called BULLSEYE, could be useful to evaluate BTB for virtual addresses.

[57] Parth Shah, Ranjal Gautham Shenoy, Vaidyanathan Srinivasan, Pradip Bose, and Alper Buyuktosunoglu. TokenSmart: Distributed, Scalable Power Management in the Many-core Era. *ACM Trans. Archit. Code Optim.*, 20(1):4:1–4:26, November 2022. 1:12, Trash it, Talks about power management in multi-core systems.

[58] Sarabjeet Singh, Neelam Surana, Kailash Prasad, Pranjali Jain, Joycee Mekie, and Manu Awasthi. HyGain: High-performance, Energy-efficient Hybrid Gain Cell-based Cache Hierarchy. *ACM Trans. Archit. Code Optim.*, 20(2):24:1–24:20, March 2023. 1:00, Traash it, NOt relatd.

[59] Mitali Soni, Asmita Pal, and Joshua San Miguel. As-Is Approximate Computing. *ACM Trans. Archit. Code Optim.*, 20(1):3:1–3:26, November 2022. 2:00, Scan it, Approximation of execution output and talks about architecture, potential interesting read.

[60] Nicolas Tollenaere, Guillaume Iooss, Stéphane Pouget, Hugo Brunie, Christophe Guillon, Albert Cohen, P. Sadayappan, and Fabrice Rastello. Autotuning Convolutions Is Easier Than You Think. *ACM Trans. Archit. Code Optim.*, 20(2):20:1–20:24, March 2023. 30 secs, trash it, Not related.

[61] Xinfeng Xie, Peng Gu, Yufei Ding, Dimin Niu, Hongzhong Zheng, and Yuan Xie. MPU: Memory-centric SIMT Processor via In-DRAM Near-bank Computing. *ACM Trans. Archit. Code Optim.*, 20(3):40:1–40:26, July 2023. 2:00, Scan it, Relatedd to the other paper in issue 2 with near memory computoing.

[62] Weizhi Xu, Yintai Sun, Shengyu Fan, Hui Yu, and Xin Fu. Accelerating Convolutional Neural Network by Exploiting Sparsity on GPUs. *ACM Trans. Archit. Code Optim.*, 20(3):36:1–36:26, July 2023. 30 secs, Trash it, Not related.

[63] Yemao Xu, Dezun Dong, Dongsheng Wang, Shi Xu, Enda Yu, Weixia Xu, and Xiangke Liao. SSD-SGD: Communication Sparsification for Distributed Deep Learning Training. *ACM Trans. Archit. Code Optim.*, 20(1):7:1–7:25, December 2022. 2:00, scan, Contains a theory about Data Parallelism might be useful.

[64] Ahmet Caner Yüzügüler, Canberk Sönmez, Mario Drumond, Yunho Oh, Babak Falsafi, and Pascal Frossard. Scale-out Systolic Arrays. *ACM Trans. Archit. Code Optim.*, 20(2):27:1–27:25, March 2023. 25, Trashit, Not related.

[65] Qiang Zhang, Lei Xu, and Baowen Xu. RegCPython: A Register-based Python Interpreter for Better Performance. *ACM Trans. Archit. Code Optim.*, 20(1):14:1–14:25, December 2022. 1:10, Trash it, register heavy Byte-code generation for python.

[66] Jin Zhao, Yu Zhang, Ligang He, Qikun Li, Xiang Zhang, Xinyu Jiang, Hui Yu, Xiaofei Liao, Hai Jin, Lin Gu, Haikun Liu, Bingsheng He, Ji Zhang, Xianzheng Song, Lin Wang, and Jun Zhou. GraphTune: An Efficient Dependency-Aware Substrate to Alleviate Irregularity in Concurrent Graph Processing. *ACM Trans. Archit. Code Optim.*, 20(3):37:1–37:24, July 2023. 30 secs, Trash it, Not related.

[67] Yuwen Zhao, Fangfang Liu, Wenjing Ma, Huiyuan Li, Yuanchi Peng, and Cui Wang. MFFT: A GPU Accelerated Highly Efficient Mixed-Precision Large-Scale FFT Framework. *ACM Trans. Archit. Code Optim.*, 20(3):43:1–43:23, July 2023. 55secs, Trashit, Noe related.

[68] Tuowen Zhao, Tobi Popoola, Mary Hall, Catherine Olschanowsky, and Michelle Strout. Polyhedral Specification and Code Generation of Sparse Tensor Contraction with Co-iteration. *ACM Trans. Archit. Code Optim.*, 20(1):16:1–16:26, December 2022. 1:30, trash it, Related to a different compilation technique.

[69] Yufeng Zhou, Alan L. Cox, Sandhya Dwarkadas, and Xiaowan Dong. The Impact of Page Size and Microarchitecture on Instruction Address Translation Overhead. *ACM Trans. Archit. Code Optim.*, 20(3):38:1–38:25, July 2023. Critical rEad, 2:00, Scan it, Talks about how page size and pipeline can impact hardware overhead of address translation. Quatification of different TLB organization based on overhead. TLB overhead is about 13.44% of execution cycles. taken experimental setups XEon and Ryzen Free BSD OS. table 1 for TLB structures. Fig 1 for comparisions with differnt workloads.