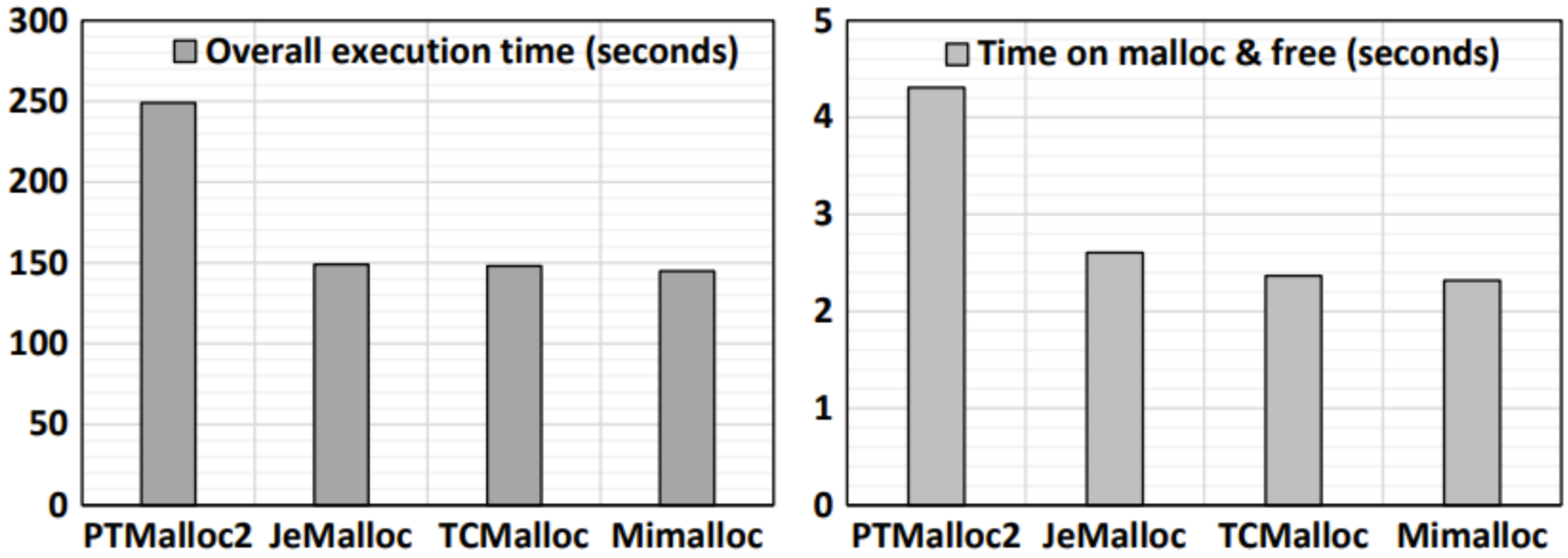
Enhancing Performance through Advanced Memory Allocation Techniques

CS575 Jiankun Dong

Link to presentation
https://drive.google.com/file/d/1YPor4N25kyNe1Lkqd_276k0fV_hRU5JR/view?usp=drive_link

BOSTON UNIVERSITY

Challenges for current system

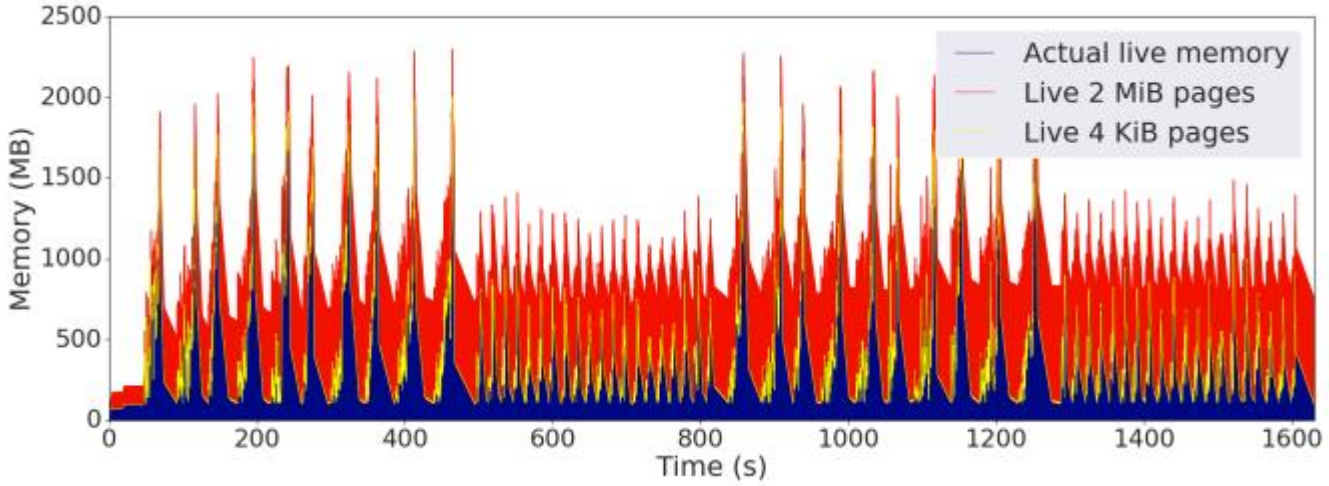


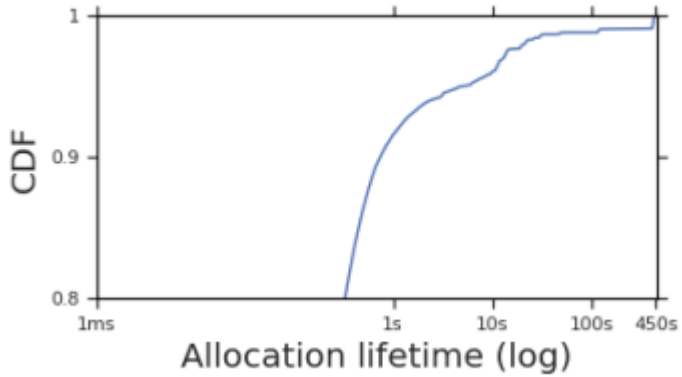
Challenges for current system

# of threads	1	2	4	8
cycles	4.333E+10	6.708E+10	1.148E+11	1.963E+11
instructions	2.387E+10	2.979E+10	3.893E+10	4.874E+10
LLC-load-misses	**1.223E+05**	**2.196E+05**	**2.477E+06**	**1.175E+07**
LLC-store-misses	**3.676E+06**	**4.231E+06**	**1.650E+07**	**5.402E+07**

Allocator	PTMalloc2	JeMalloc	TCMalloc	Mimalloc
cycles	**1.177E+12**	**7.115E+11**	**7.091E+11**	**6.959E+11**
instructions	1.282E+12	1.320E+12	1.264E+12	1.262E+12
LLC-load-misses	**4.059E+08**	**9.445E+07**	**1.016E+08**	**1.477E+08**
LLC-store-misses	**3.554E+08**	**1.630E+08**	**1.254E+08**	**1.321E+08**
dTLB-load-misses	**1.804E+09**	**1.482E+08**	**1.641E+08**	**1.628E+08**
dTLB-store-misses	3.669E+07	2.937E+07	2.591E+07	2.787E+07

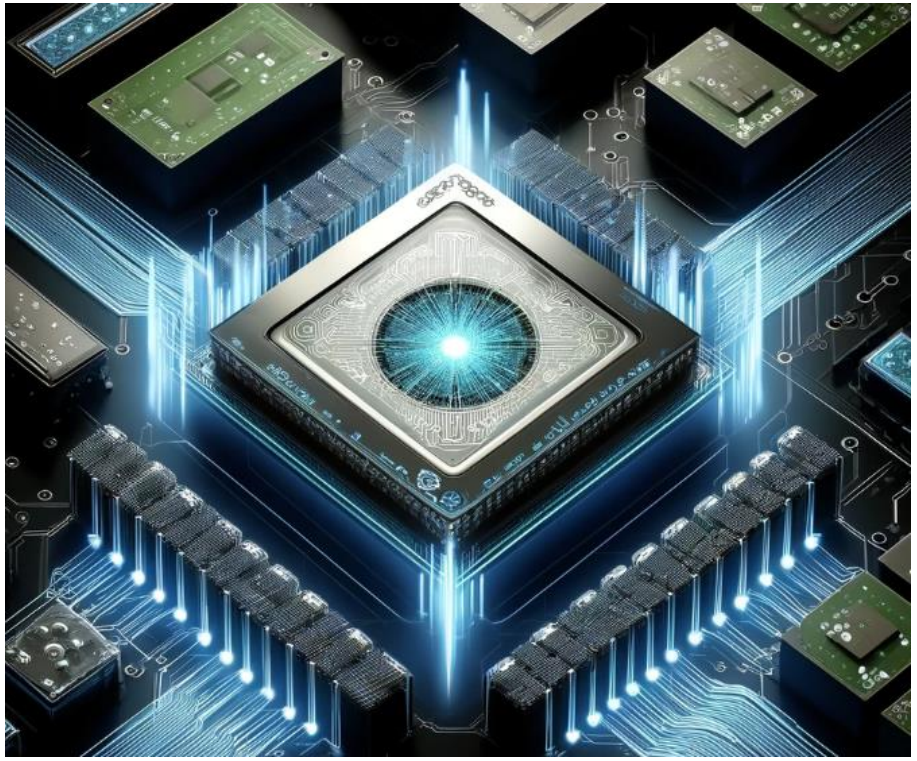
Challenges for current system



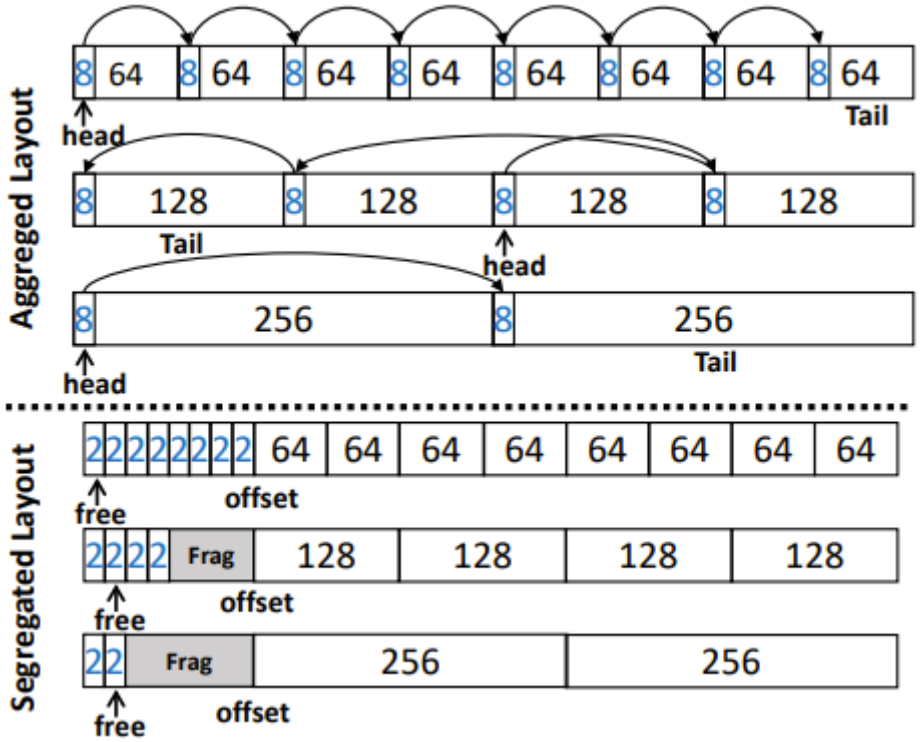


BOSTON UNIVERSITY

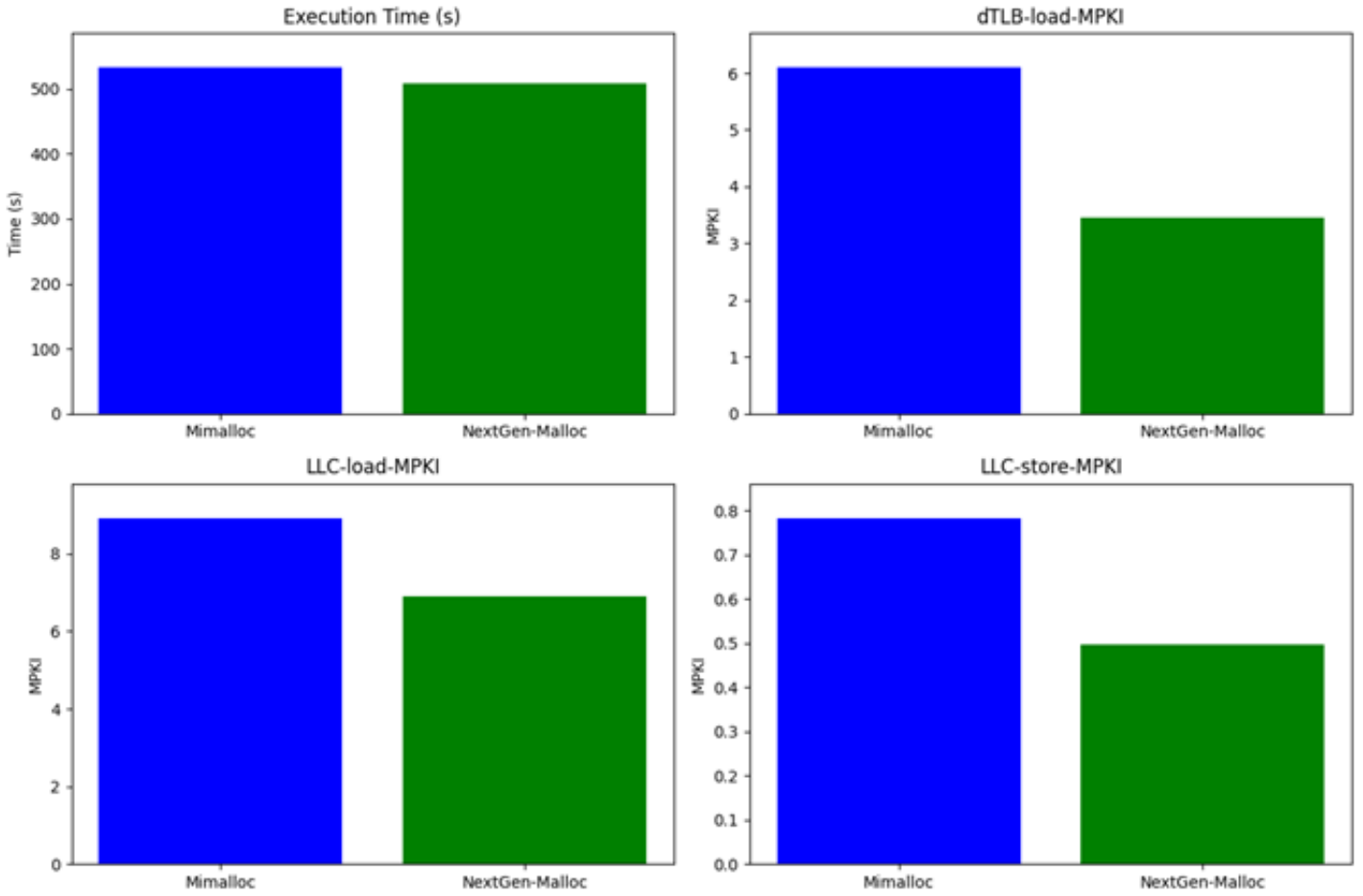
Nextgen-Malloc



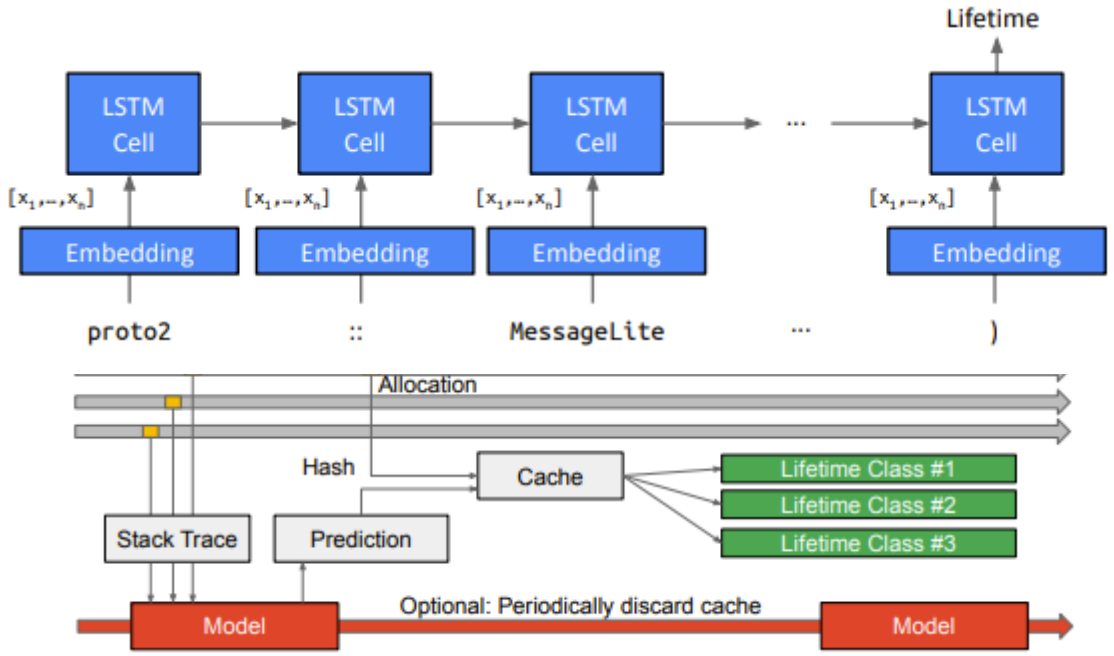
Nextgen-Malloc



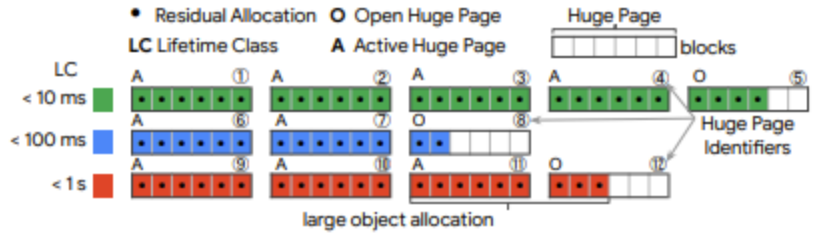
Nextgen-Malloc



Llama

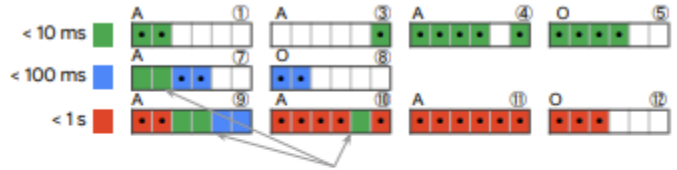


Llama

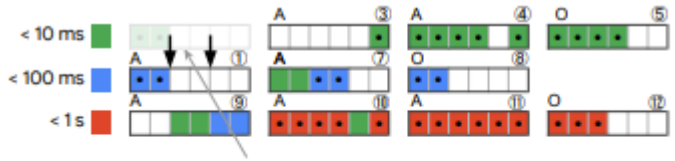


(a) Initial allocations. Huge pages are bump-pointer allocated into LC regions. Each huge page is first filled with same LC blocks, marked residual with a dot.

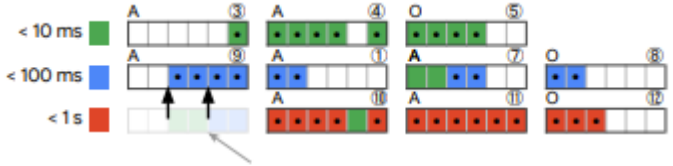
(b) After objects free, some blocks and huge pages are free (white). LLAMA immediately returns free huge pages to the OS to control maximum heap size.



(c) Subsequent allocations of shorter LC small objects first fill free blocks in the highest LC in A(ctive) huge pages 9 and 10, and then blocks in huge page 7. These blocks are not residual (no dot) and expected to be freed before the residual blocks. O(pen) pages 5, 8, and 12 are ineligible for such allocation.



(d) When huge page 1's deadline expires, residual blocks are still live (mis-prediction). LLAMA increases the huge page's LC by one, from 10 to 100 *ms*. Residual blocks remain residual; their expected lifetime is now at least 100 *ms*.

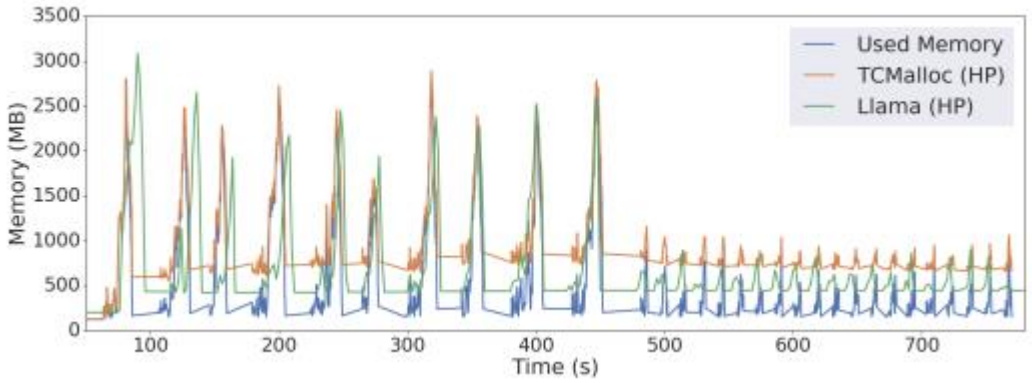


(e) Huge page 9 only contains non-residual blocks and consequently, LLAMA decreases its LC. It marks all live blocks residual since they match or are less than the huge page's LC.

BOSTON UNIVERSITY

Llama

Workload	Prediction Accuracy		Final Steady-state Memory			Fragmentation
	Weighted	Unweighted	TCMalloc	LLAMA	Live	reduction
Image Processing Server	96%	73%	664 MB	446 MB	153 MB	43%
TensorFlow InceptionV3 Benchmark	98%	94%	282 MB	269 MB	214 MB	19%
Data Processing Pipeline	99%	78%	1964 MB	481 MB	50 MB	78%
Redis Key-Value Store	100%	94%	832 MB	312 MB	115 MB	73%



BOSTON UNIVERSITY

Reference

- [1] Ruihao Li, Qinzhe Wu, Krishna Kavi, Gayatri Mehta, Neeraja J. Yadwadkar, and Lizy K. John. 2023. NextGen-Malloc: Giving Memory Allocator Its Own Room in the House. In Proceedings of the 19th Workshop on Hot Topics in Operating Systems (HOTOS '23). Association for Computing Machinery, New York, NY, USA, 135–142. https://doi.org/10.1145/3593856.3595911

- [2] Hermann Schweizer, Maciej Besta, and Torsten Hoefler. 2015. Evaluating the cost of atomic operations on modern architectures. In 2015 International Conference on Parallel Architecture and Compilation (PACT). IEEE, 445–456.https://doi.org/10.48550/arXiv.2010.09852

- [3] Svilen Kanev, Juan Pablo Darago, Kim Hazelwood, Parthasarathy Ranganathan, Tipp Moseley, Gu-Yeon Wei, and David Brooks. 2015. Profiling a warehouse-scale computer. In 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA). 158–169. https://doi.org/10.1145/2749469.2750392

- [4] Svilen Kanev, Sam Likun Xi, Gu-Yeon Wei, and David Brooks. 2017. Mallacc: Accelerating Memory Allocation. In Proceedings of the TwentySecond International Conference on Architectural Support for Programming Languages and Operating Systems (Xi'an, China) (ASPLOS '17).

- [5] Martin Maas, David G. Andersen, Michael Isard, Mohammad Mahdi Javanmard, Kathryn S. McKinley, and Colin Raffel. 2020. Learning-based Memory Allocation for C++ Server Workloads. In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '20). Association for Computing Machinery, New York, NY, USA, 541–556. https://doi.org/10.1145/3373376.3378525