Tiered Memory Management Beyond Hotness

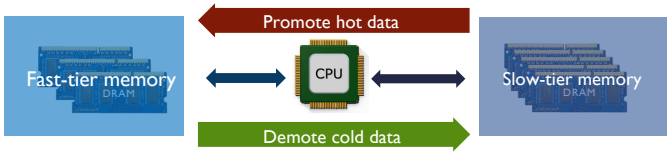
VIRGINIA TECH

Jinshu Liu, Hamid Hadian, Hanchen Xu, Huaicheng Li

OSDI 25

Memory – OS – Arch – Storage – MOAT LAB

Memory Tiering



Traditional Approach:

1. Common assumption: *Performance = Hotness*
2. Page migration: promote hot data to the fast-tier
3. First-touch allocation

Research Questions:

Why cannot hotness represent performance?

Which metrics should be used to guide tiering?

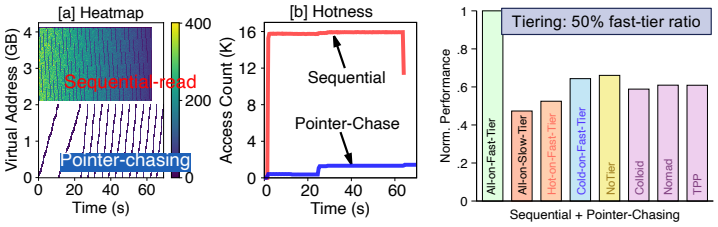
How to apply the new metrics on memory allocation and migration?

Overview

Memory allocation/migration for tiered memory *beyond hotness*

1. AOL: Amortized Offcore Latency
   - Key factor for accurate performance prediction
   - Quantifies the impact of memory accesses on performance slowdown
2. SOAR: Static Object Allocation based on Ranking
   - Near-optimal object placement after profiling based on AOL
3. ALTO: AOL-based Layered Tiering Orchestration
   - Regulate page migration for hot but less performance-critical pages
4. Evaluation
   - Compare SOAR/ALTO with TPP/Nomad/NUMA-Balancing/Colloid
   - Evaluated under memory contention on the fast-tier

Hotness != Performance

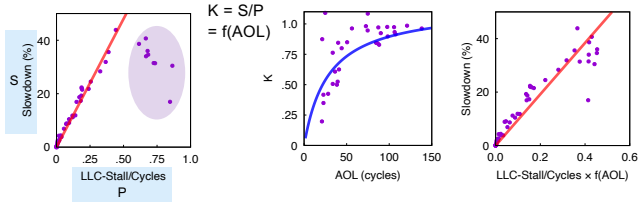


Hot-on-fast-tier gives worse (34%) performance than Cold-on-fast-tier !

Reasoning:

1. LLC stalled cycles
2. Memory Level Parallelism (MLP) can mask latency penalty

Amortized Offcore Latency



AOL = Latency / MLP

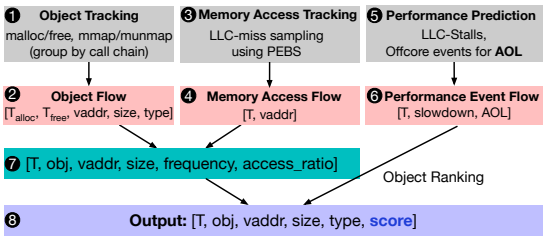
MLP itself is not enough to model how performance slowdown is impacted by parallel accesses

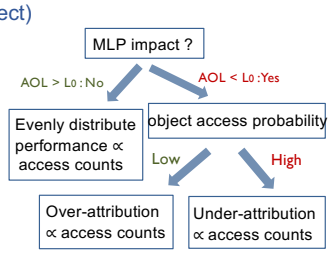
AOL quantifies how increased LLC stalls on slow-tier are amortized by MLP

SOAR: Static Object Allocation based on Ranking

Assign predicted performance to each object each time-period

Adjust slowdown attribution for objects based on AOL (MLP effect)





ALTO: AOL-based Layered Tiering Orchestration

Unnecessary page migration

1. Promote hot but non-performance-critical pages results in minimal or negative performance gain
2. Overheads caused by ineffective page migration

For each time period t_i, t_{i+1} :

> AOL_{high}	Enable page promotion
[AOL_{low}, AOL_{high}]	Partial page promotion *Scale <- f(AOL)*
< AOL_{low}	Disable page promotion

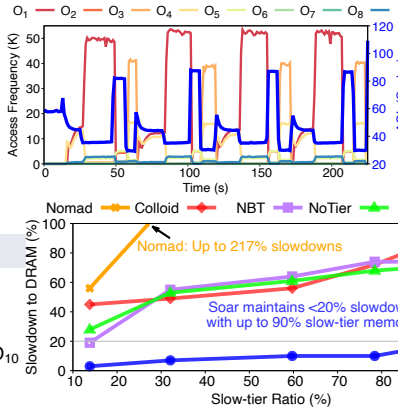
Evaluation

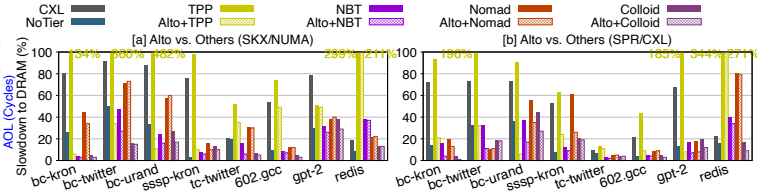
Object Ranking (bc-urand)

Object	Size	First-touch	Freq	SOAR
O_1	536MB	8	1	1
O_2	536MB	6	6	2
O_3	536MB	10	2	3
O_4	1073MB	5	3	4
O_5	1073MB	3	4	5
O_6	536MB	4	7	6
O_7	536MB	7	8	7
O_8	17GB	2	5	8
O_9	327KB	9	9	9
O_{10}	1073MB	1	10	10

Object placement (50% fast-tier ratio)

	Fast-tier	Slow-tier
First-touch	O_{10}, O_8	O_1-O_7, O_9
Freq	O_1, O_3-O_5, O_8	O_2, O_6-O_8, O_9-O_{10}
SOAR	O_1-O_8	O_8-O_{10}





ALTO/SOAR under memory bandwidth contention (bc-urand)

# of MLC Threads	Soar	NoTier	NBT	Alto+NBT	Nomad	Alto+Nomad	Colloid	Alto+Colloid
9	2%	6%	81%	70%	186%	105%	30%	33%
6	21%	28%	83%	52%	65%	38%	44%	35%
3	22%	55%	101%	78%	103%	70%	61%	50%
0	14%	55%	86%	60%	95%	65%	61%	58%

Paper Code

https://github.com/MoatLab/SoarAlto

Appears in the 19th USENIX Symposium on Operating Systems Design and Implementation (OSDI'25)