IPDPS
St. Petersburg, Florida USA
May 15-19, 2023

COMPSYS'23

European Commission | Horizon 2020 European Union funding for Research & Innovation

nephele

On the implications of Heterogeneous Memory Tiering on Spark In-memory Analytics

Manolis Katsaragakis[*+], **Dimosthenis Masouros**[*], Lazaros Papadopoulos[*], Francky Catthoor[+α], Dimitrios Soudris[*]

[*]Microprocessors and Digital Systems Laboratory, ECE, National Technical University of Athens(NTUA), Greece
[+]Katholieke Universiteit Leuven(KUL), Belgium
[α]IMEC, Leuven, Belgium

{mkatsaragakis, dmasouros, lpapadop, dsoudris}@microlab.ntua.gr
francky.catthoor@imec.be



Introduction

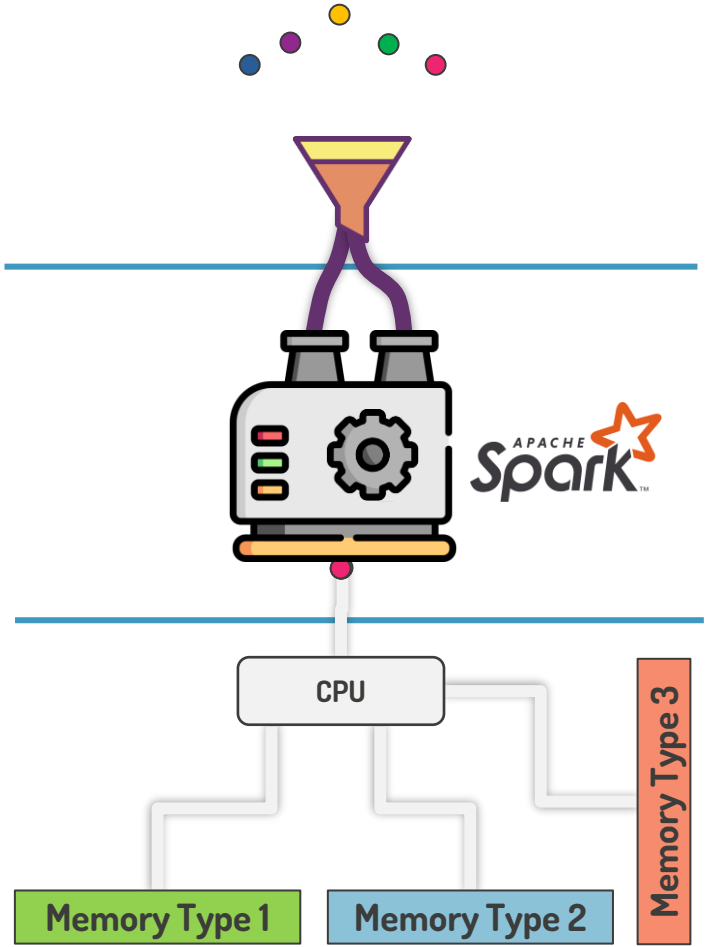
- Over **2.5 quintillion bytes generated** daily[1]

End Users

- Adoption of novel **in-memory processing frameworks** for large scale data analytics

Providers

- Integration of **heterogeneous** memory technologies and **multi-tier memory** architectures.
  - DRAM along with PMEM on the same server
  - Disaggregated DRAM



1. Data never sleeps, https://www.domo.com/solution/data-never-sleeps-6

Introduction

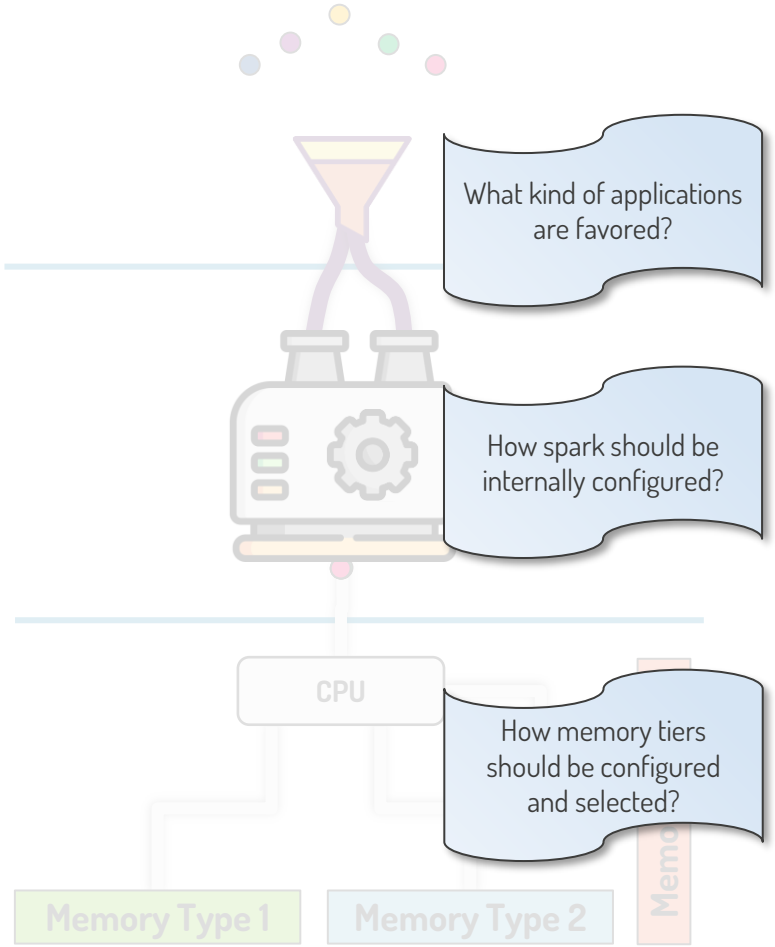
- Over **2.5 quintillion bytes generated** daily[1]

End Users

- Adoption of novel **in-memory processing frameworks** for large scale data analytics

Providers

- Integration of **heterogeneous** memory technologies and **multi-tier memory** architectures.
  - DRAM along with PMEM on the same server
  - Disaggregated DRAM



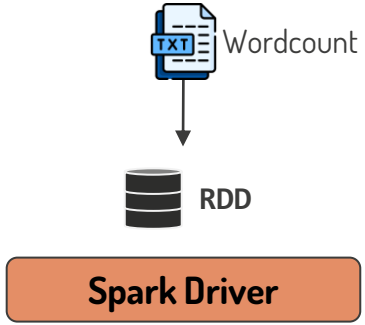
1. Data never sleeps, https://www.domo.com/solution/data-never-sleeps-6

Goal of this work

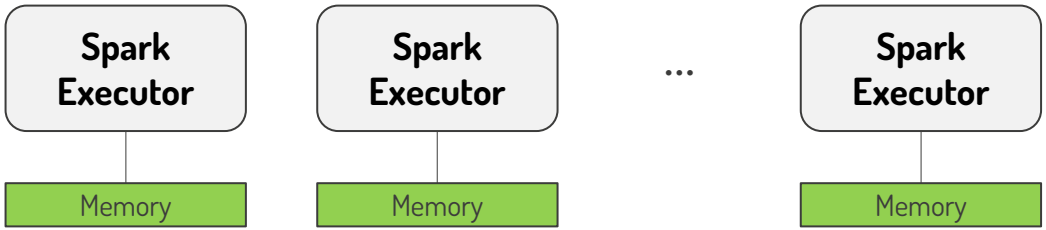
Provide an **exploration** and **performance analysis** of **Spark applications** over an **heterogeneous multi-tier** memory system

- Key questions w.r.t. the effect of memory tiering on Spark analytics
- Key takeaways in terms of:
  - Performance Implications
  - Performance Bottlenecks
  - Performance predictability

Spark (quick) Background

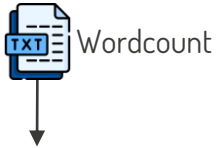


Orchestrator that determines the **tasks** to be performed based on a piece of code



Workers that actually **execute** the tasks

Spark (quick) Background

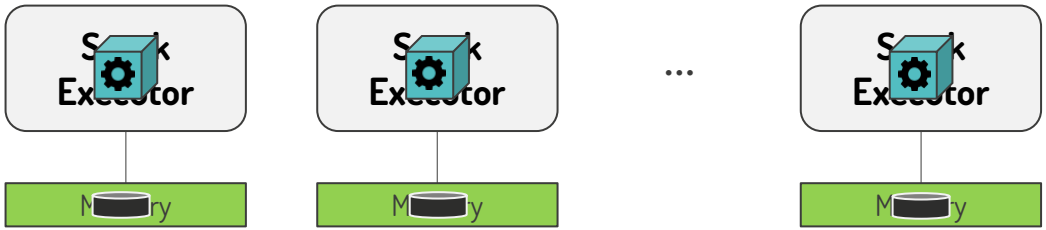


Spark Driver

Orchestrator that determines the **tasks** to be performed based on a piece of code

- Extremely efficient
- Requires huge amount of memory

Perfect candidate for multi-tier/disaggregated systems!



Workers that actually **execute** the tasks

Spark Benchmarks

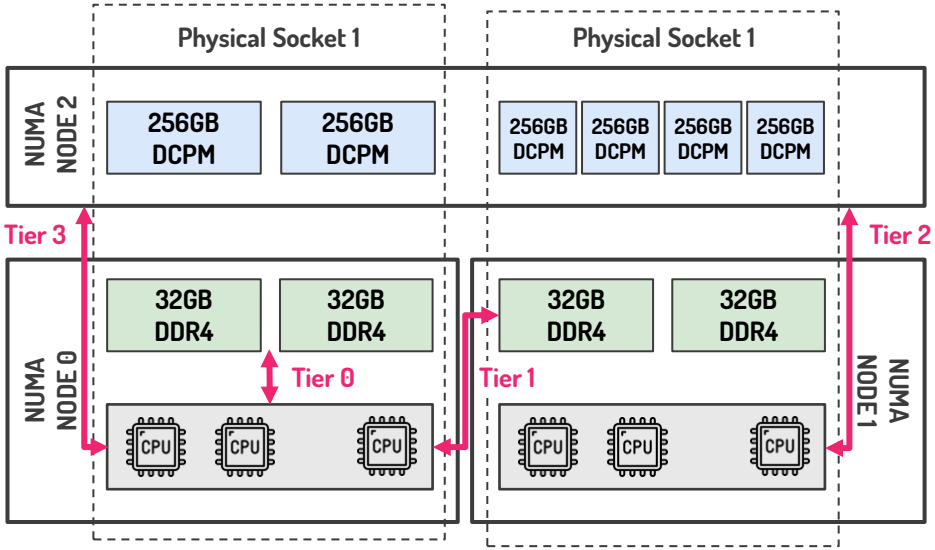
- Benchmarks derived from HiBench[1] suite:
  - Diverse domains
    - micro-operations, ML, web search
  - Diverse set of input workloads:
    - tiny, small, large
- Pseudo-distributed, standalone mode:
  - Spark driver and executors on the same node
  - HDFS file system

Application	Abbr.	Data size range (tiny,small,large)
Sorting of text input data	sort	32KB, 320MB, 3.2GB
Performs shuffle operations	repartition	3.2KB, 3.2MB, 32MB
Alternating Least Squares	als	100, 1.000, 10.000 (users) 100, 1.000, 10.000 (products) 200, 2.000, 20.000 (ratings)
Naive Bayes classification	bayes	25.000, 30.000, 100.000 (pages) 10, 100, 100 (classes)
Random forest	rf	10, 100, 1.000 (examples) 100, 500, 1.000 (features)
Latent Dirichlet Allocation	lda	2.000, 5.000, 10.000 (docs) 1.000, 2.000, 3.000 (vocabulary) 10, 20, 30 (topics)
PageRank	pagerank	50, 5.000, 500.000 (pages)

*https://github.com/Intel-bigdata/HiBench

Hardware Testbed

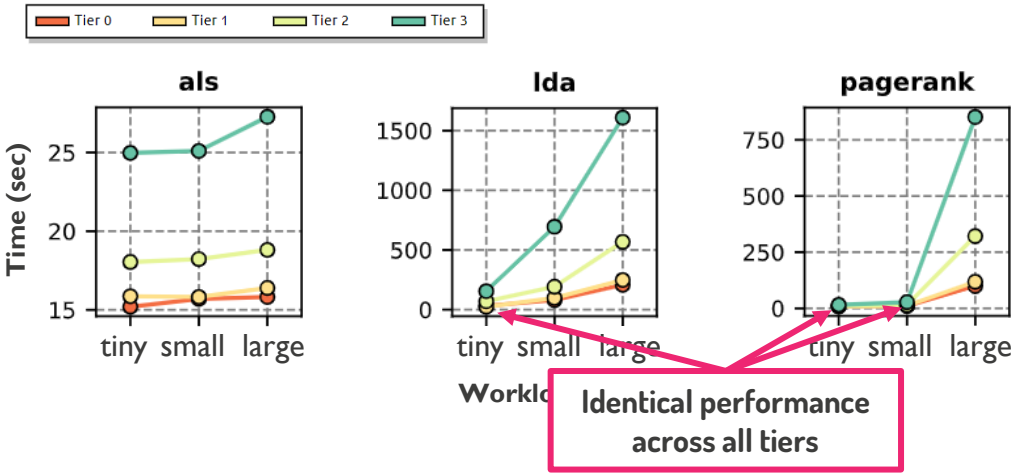
- Dual-socket Intel Xeon 5218R
  - 40 threads/socket
- Symmetric DRAM topology
  - 2x32GB DDR4 DRAM DIMMs per socket
- Assymetric Intel Optane DCPM topology
  - 2x256GB (socket 1) vs 4x256GB (socket 2)
  - App Direct mode
- 4 Memory tiers with difference latency and bandwidth
  - Tier binding through `numactl` command



		Idle Latency (ns)	Bandwidth (GB/s)
Tier	Tier 0	77.8	39.3
	Tier 1	130.9	31.6
	Tier 2	172.1	10.7
	Tier 3	231.3	0.47

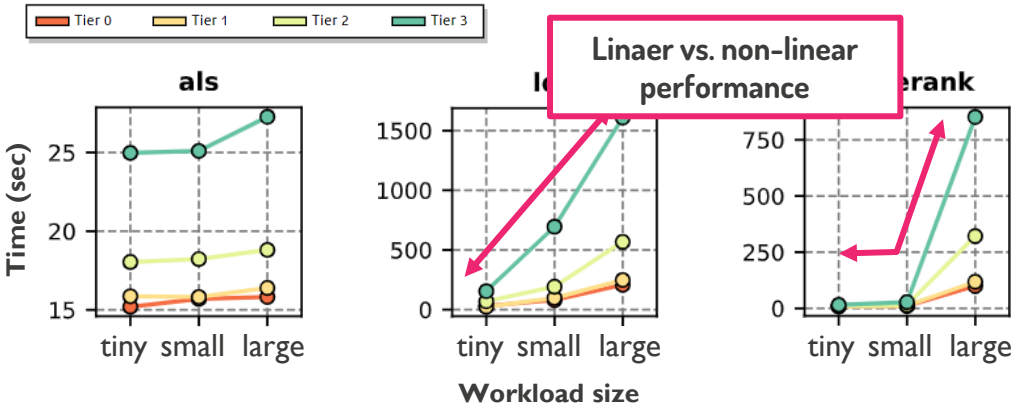
Performance Implications of Memory Tiering

How do applications perform on different tiers?



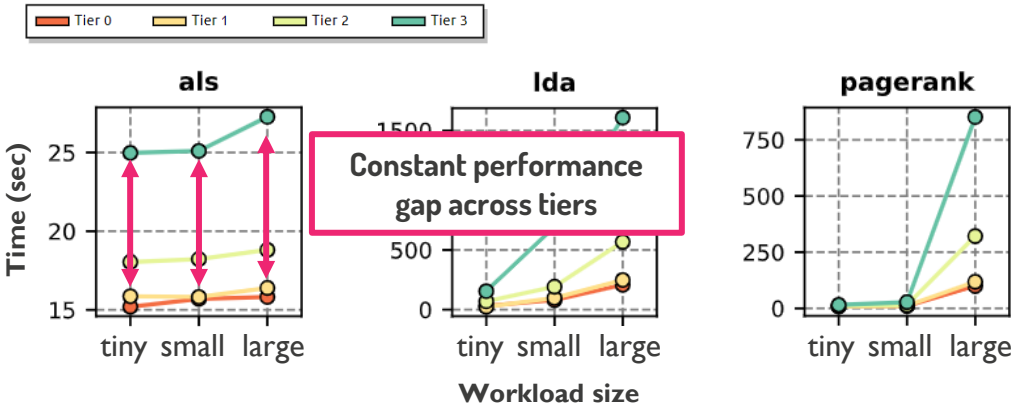
Performance Implications of Memory Tiering

How do applications perform on different tiers?



Performance Implications of Memory Tiering

How do applications perform on different tiers?

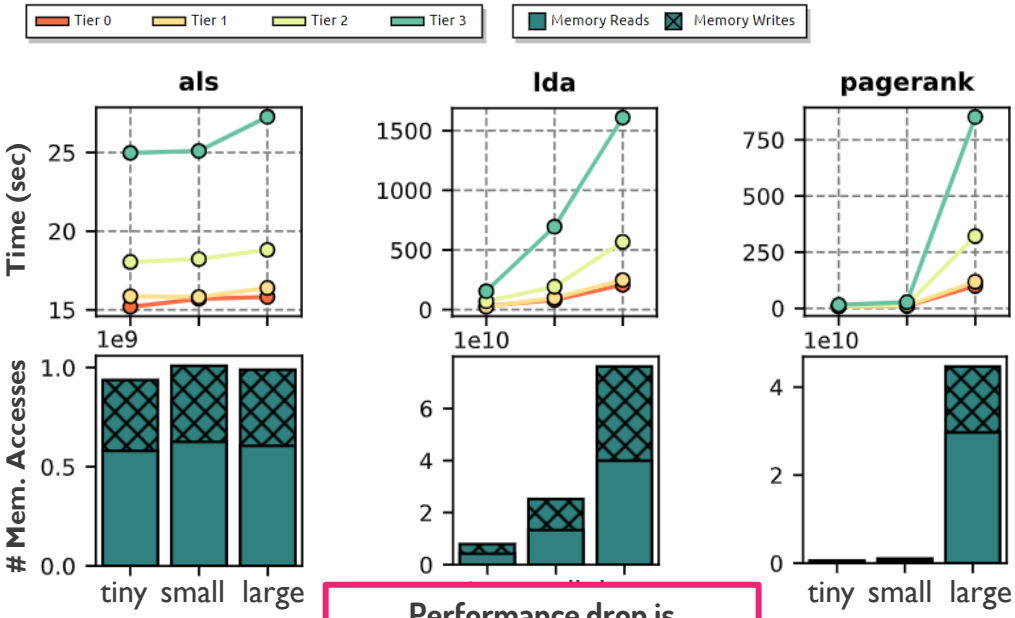


Constant performance gap across tiers

Takeaway: Performance degradation depends on the nature of each application and input workload size

Performance Implications of Memory Tiering

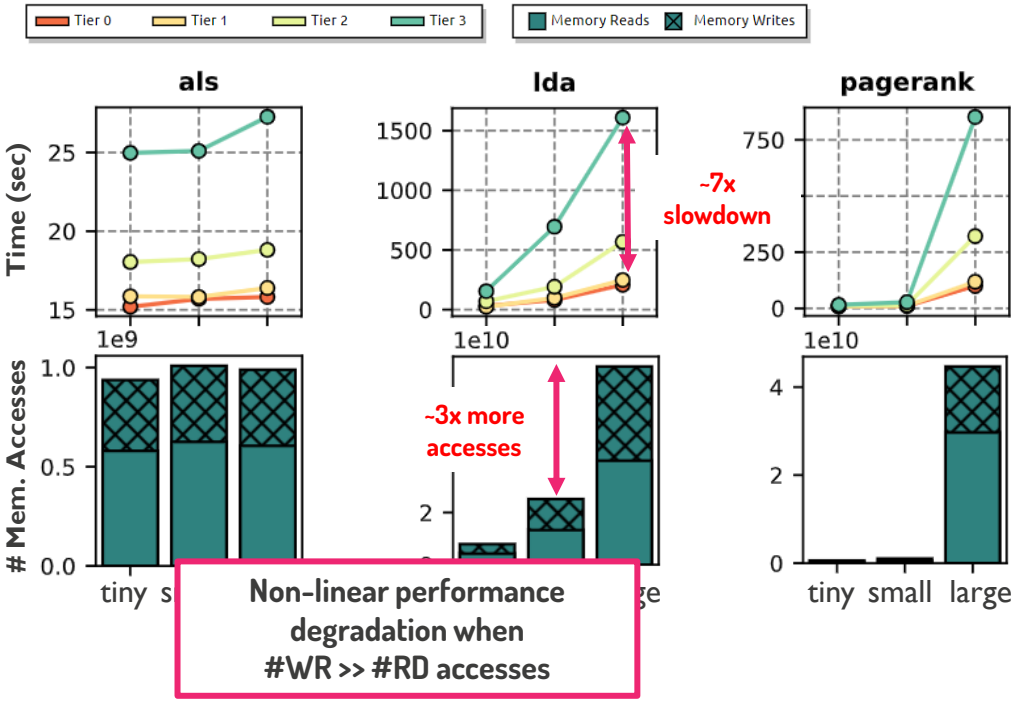
What is the core bottleneck of performance degradation?



Takeaway: Performance degradation depends on the nature of each application and input workload size

Performance drop is proportional to the number of RD+WR accesses

Performance Implications of Memory Tiering

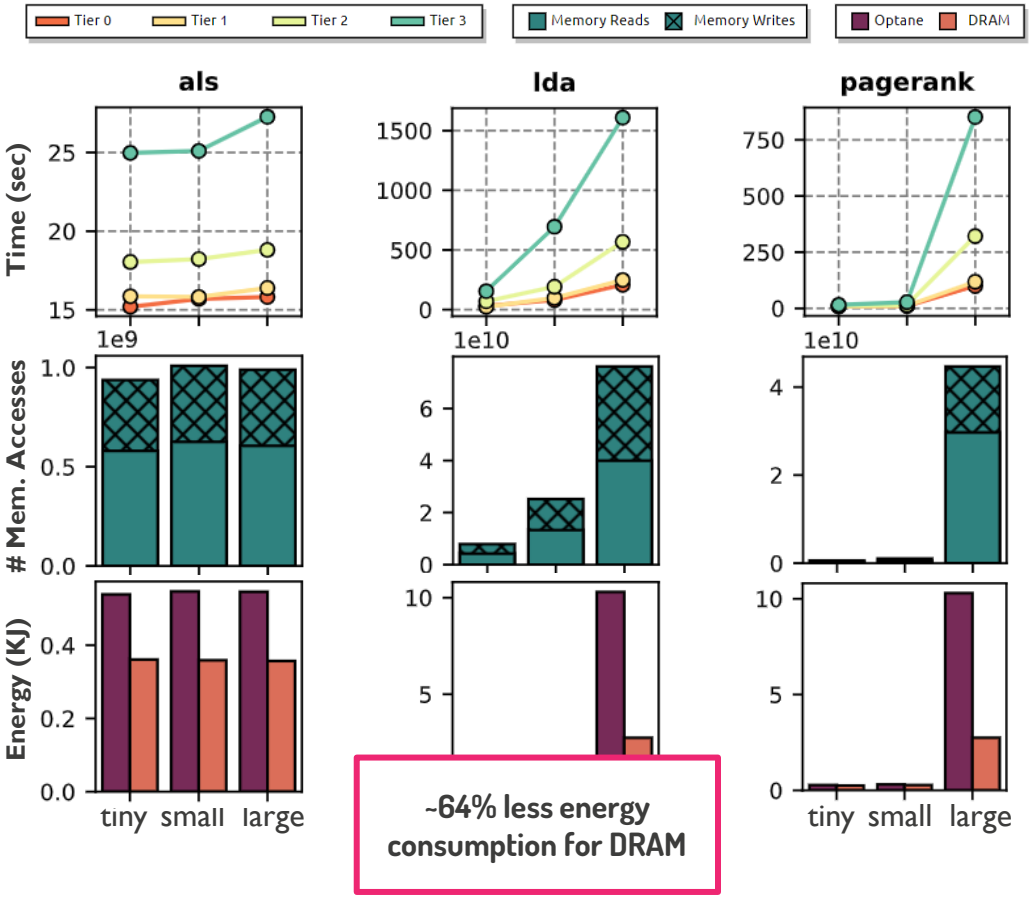


What is the core bottleneck of performance degradation?

Takeaway: Performance degradation depends on the nature of each application and input workload size

Takeaway: Performance is highly affected by the number of RD and WR operations on PMEM, with the latter having even more impact by design.

Energy Implications of Memory Tiering



~64% less energy consumption for DRAM

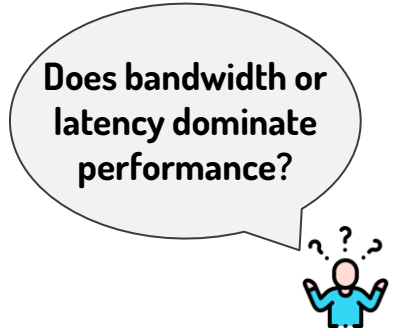
How about energy consumption?

Takeaway: Performance degradation depends on the nature of each application and input workload size

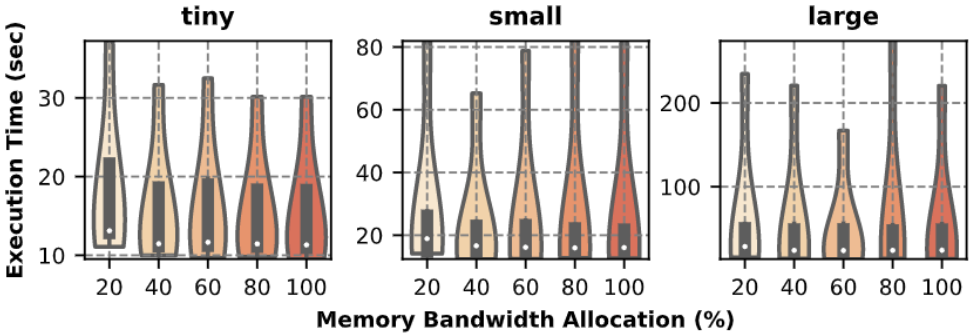
Takeaway: Performance is highly affected by the number of RD and WR operations on PMEM, with the latter having even more impact by design.

Takeaway: Energy consumption is inline with execution time and DRAM (despite less power-) is more energy-efficient

Bandwidth vs. Latency



- Limit cores' available bandwidth to memory and execute on Tier 2
  - Intel's Memory Bandwidth Allocation(MBA) tool*
  - 20, 40, 60, 80, 100%



- Average execution time and variance are **not** affected by available bandwidth
- Our applications do **not** saturate bandwidth

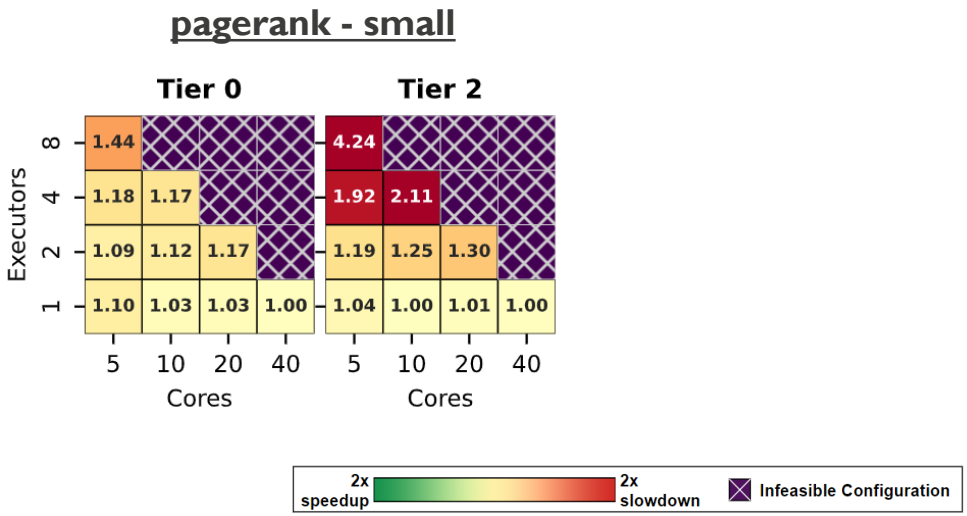
Takeaway: Performance is dominated by latency and **bandwidth is not saturated**

*https://github.com/intel/intel-cmt-cat

Spark "Sizing" vs. Performance

- Different number of executors and cores/executor
  - Executor colocation with concurrent access to memory
- Baseline (default execution) ➔ single executor, 40 cores

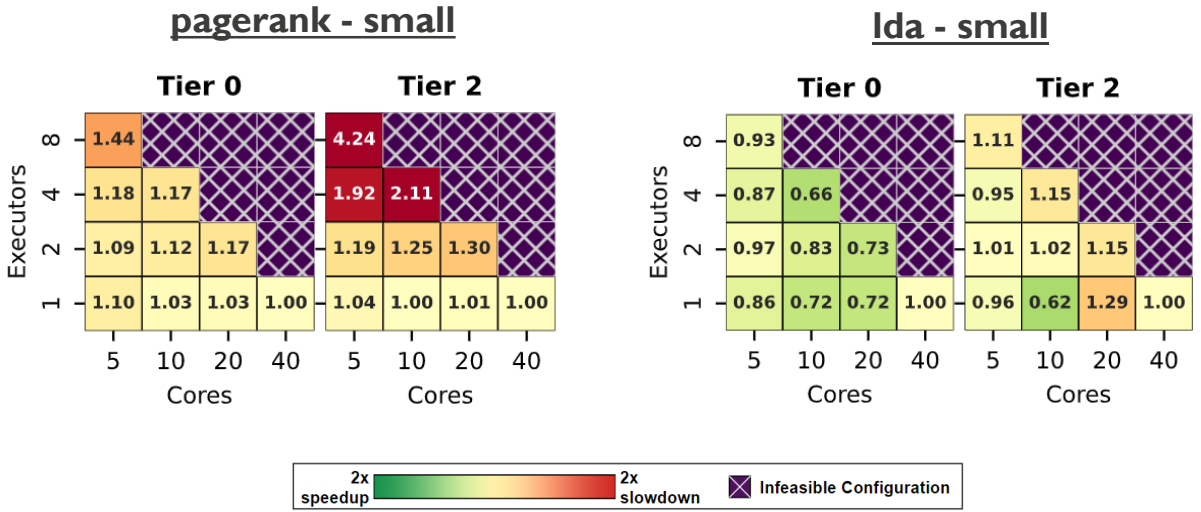
Takeaway: Increased number of **executors that compete over shared memory resources leads to further performance degradation**, with persistent memory being even more susceptible to resource contention.



Spark "Sizing" vs. Performance

- Different number of executors and cores/executor
  - Executor colocation with concurrent access to memory
- Baseline (default execution) ➔ single executor, 40 cores

How do different deployment approaches affect performance?



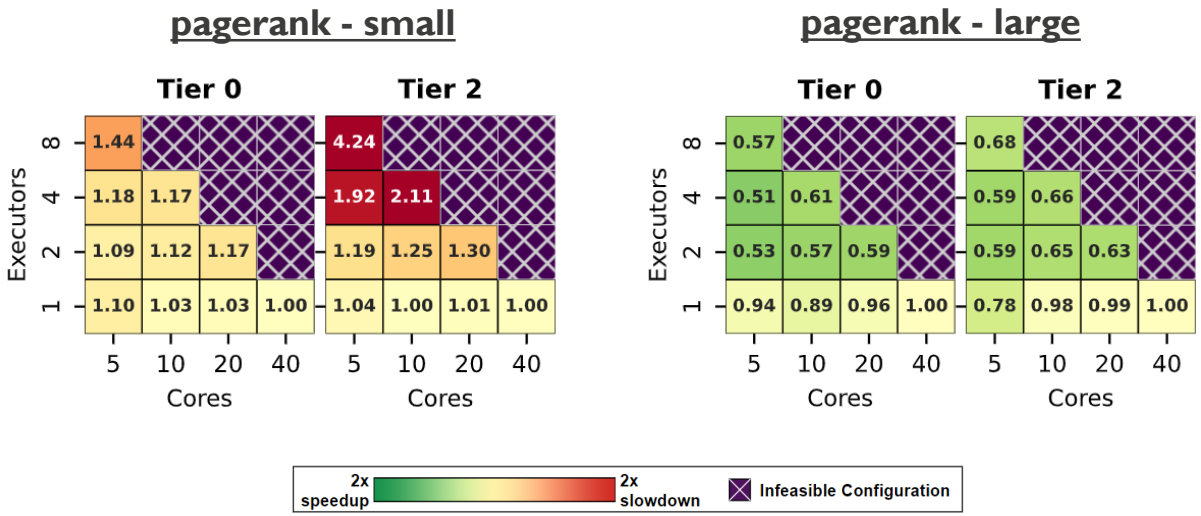
Takeaway: Increased number of **executors that compete over shared memory resources leads to further performance degradation**, with persistent memory being even more susceptible to resource contention.

Takeaway: Certain benchmarks are not affected by altering deployment's sizing

Spark "Sizing" vs. Performance

- Different number of executors and cores/executor
  - Executor colocation with concurrent access to memory
- Baseline (default execution) ➔ single executor, 40 cores

How do different deployment approaches affect performance?

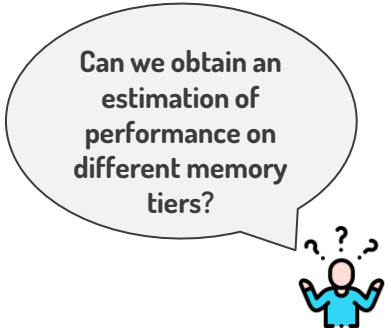


Takeaway: Increased number of **executors that compete over shared memory resources leads to further performance degradation**, with persistent memory being even more susceptible to resource contention.

Takeaway: Certain benchmarks are not affected by altering deployment's sizing

Takeaway: Bigger workload size can lead to performance boost due to **amortization of interference degradation from parallel processing**

Performance Predictability



- Pearson Correlation
- How execution time correlates with:

1) System-level events (e.g., IPC, LLC misses) ?
   - **No linear correlation** for the majority of the benchmarks
     ➔ Complex ML models needed

2) Hardware specs of each tier (Latency/Bandwidth) ?
   - **Very high linear correlation** for all benchmarks
     ➔ Linear models can be utilized

Conclusions

- In-memory applications + Multi-tier memory architectures emerging
- Spark perfect candidate
  - In-memory computations
  - Vast amount of memory requirements

In this work :

- Performance analysis of Spark applications over heterogeneous multi-tier memory system
- Key takeaways
  - Spark applications highly affected by slower memory tiers (due to latency)
  - Slower memory tiers can be utilized without performance drop in certain cases
  - Promising signs for performance predictions using ML

Q & A

{mkatsaragakis,dmasouros}@microlab.ntua.gr