DON'T CRY OVER SPILLED RECORDS
Memory elasticity of data-parallel applications and its application to cluster scheduling

Călin Iorgulescu (EPFL), Florin Dinu (EPFL), Aunn Raza (NUST Pakistan), Wajih Ul Hassan (UIUC), Willy Zwaenepoel (EPFL)

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

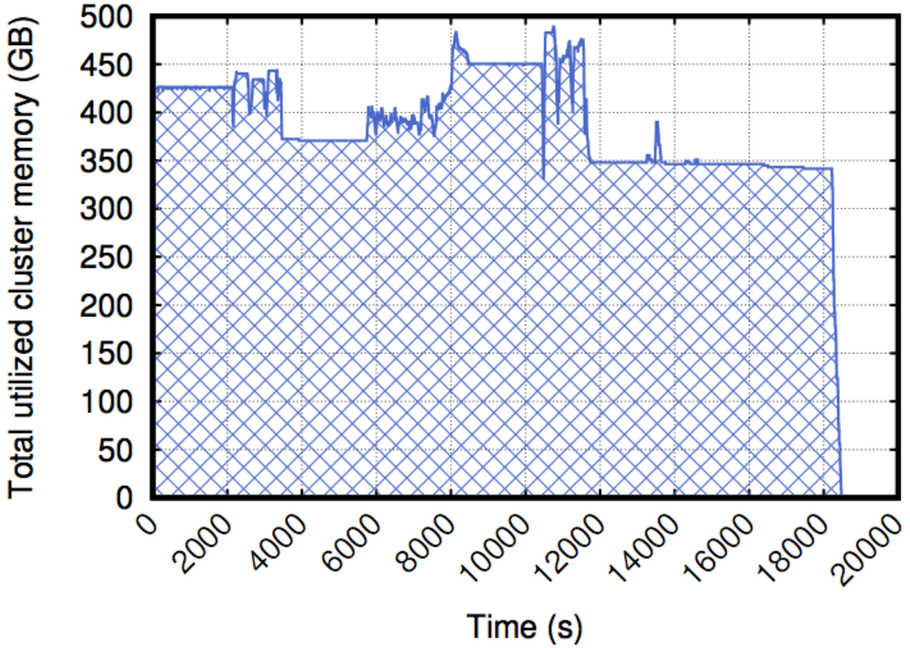
Cluster operators care about resource utilization

✓ Best bang for your buck!

✓ Maximize performance of data-parallel applications

- Idea: Efficient resource utilization through **under-provisioning**

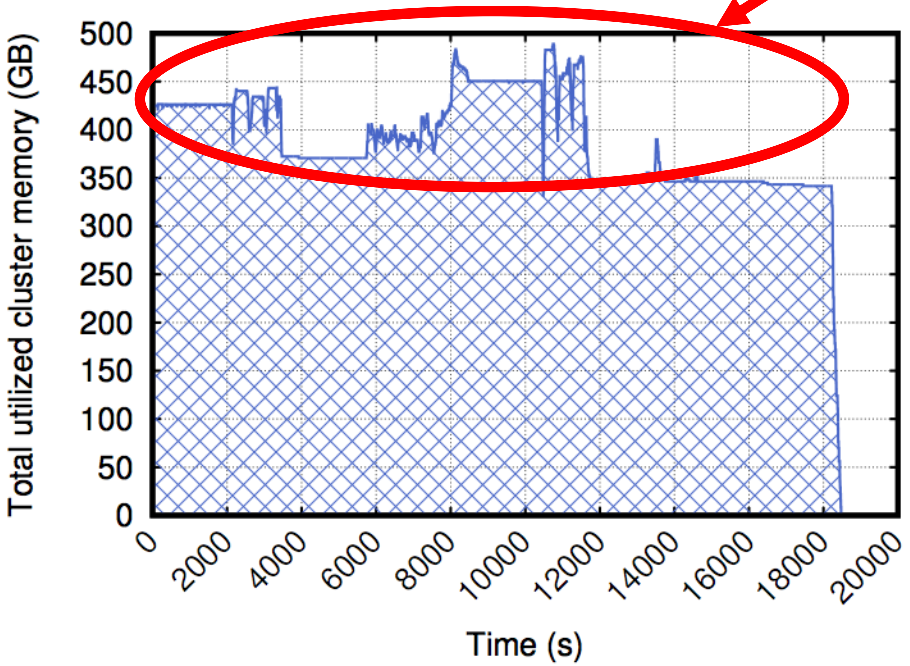
Cluster memory is under-utilized!



Avg. mem. utilization: **78%**

Cluster memory is under-utilized!

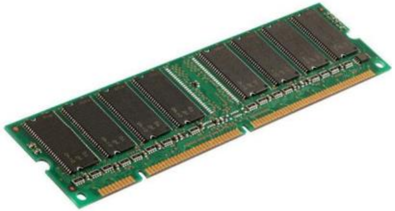
Leverage this idle memory!



Avg. mem. utilization: **78%**

Impact of memory constraining applications

- Conventional wisdom: do not touch memory!



- Risks:
  - crashes
  - severe performance degradation (e.g., thrashing)

Can we safely, deterministically, and with modest impact constrain memory?

Context: batch jobs and their memory usage

Context: batch jobs and their memory usage

Input

Context: batch jobs and their memory usage

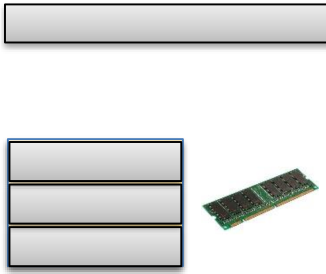
Input

Ideal memory

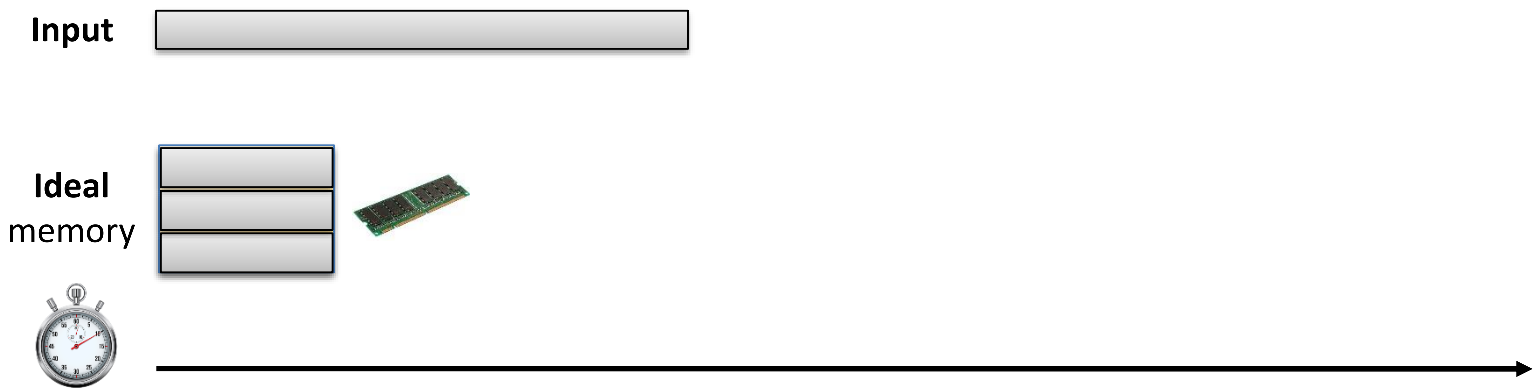
Context: batch jobs and their memory usage

Input

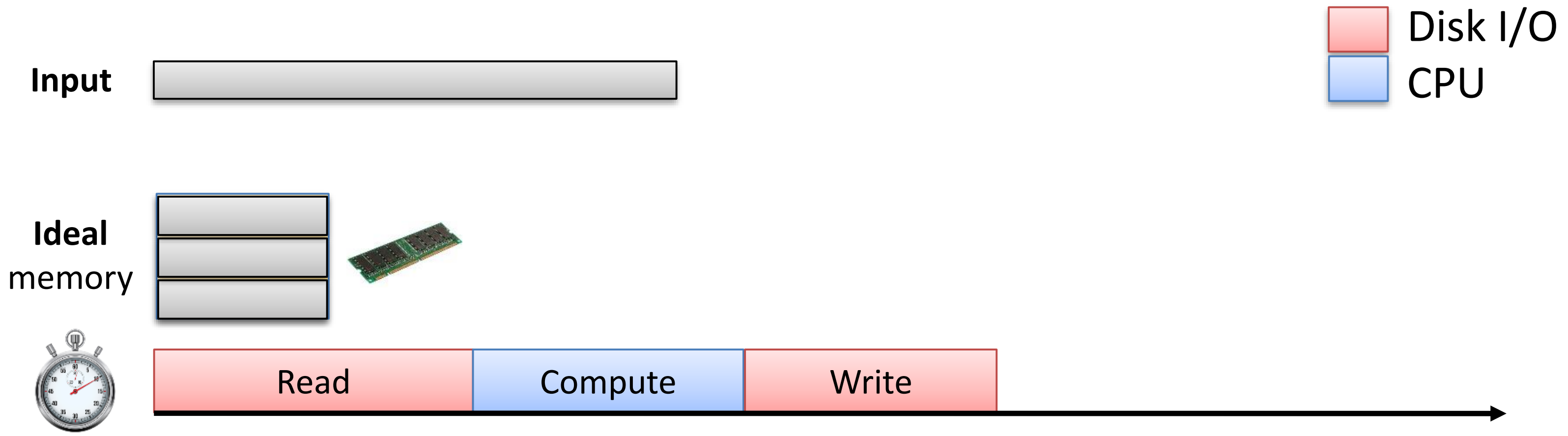
Ideal memory



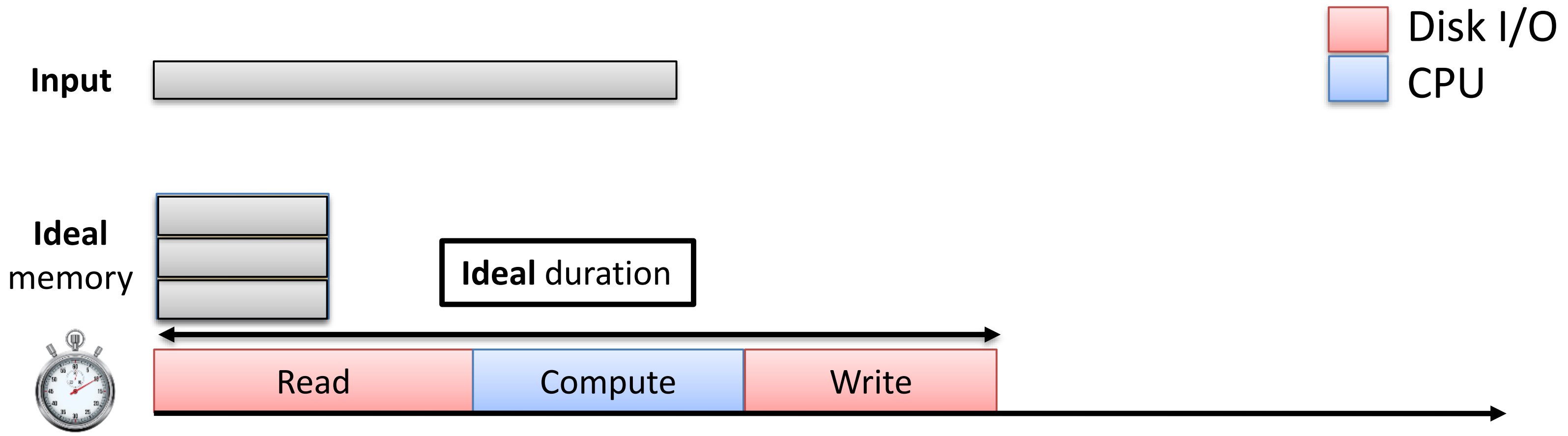
Context: batch jobs and their memory usage



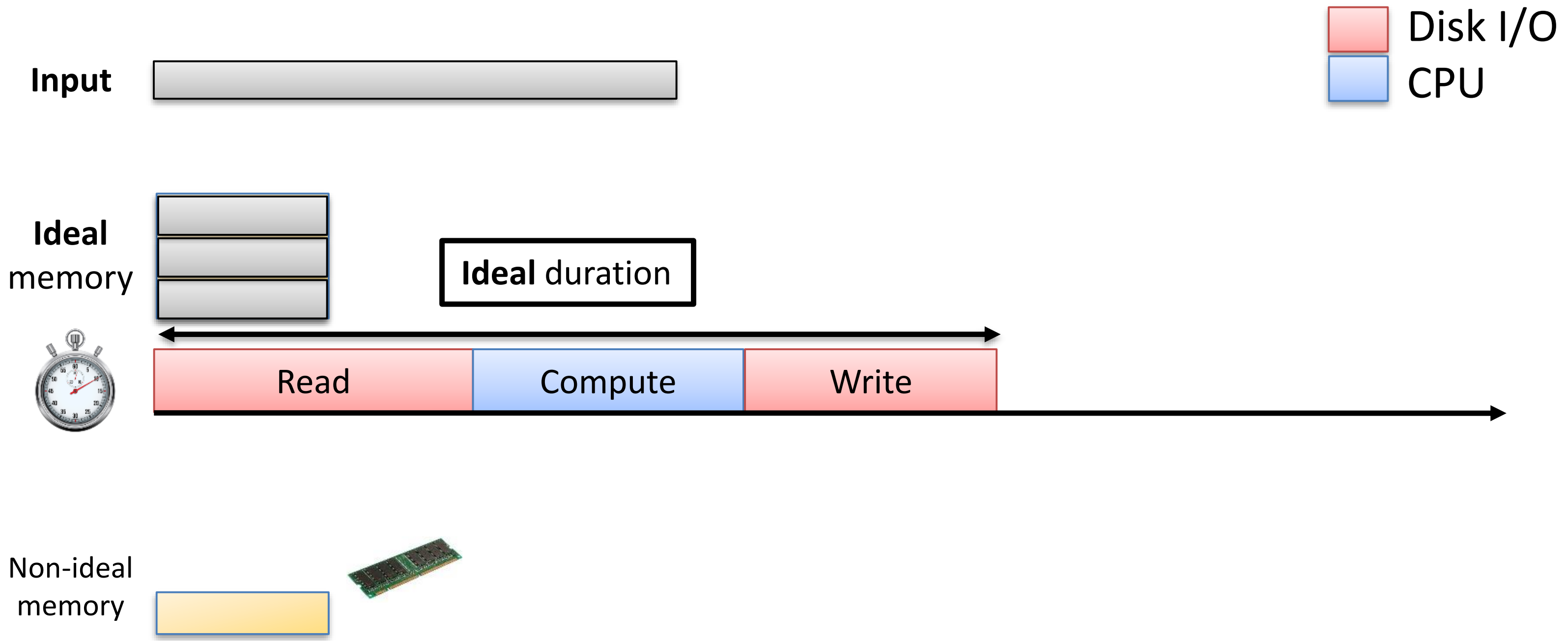
Context: batch jobs and their memory usage



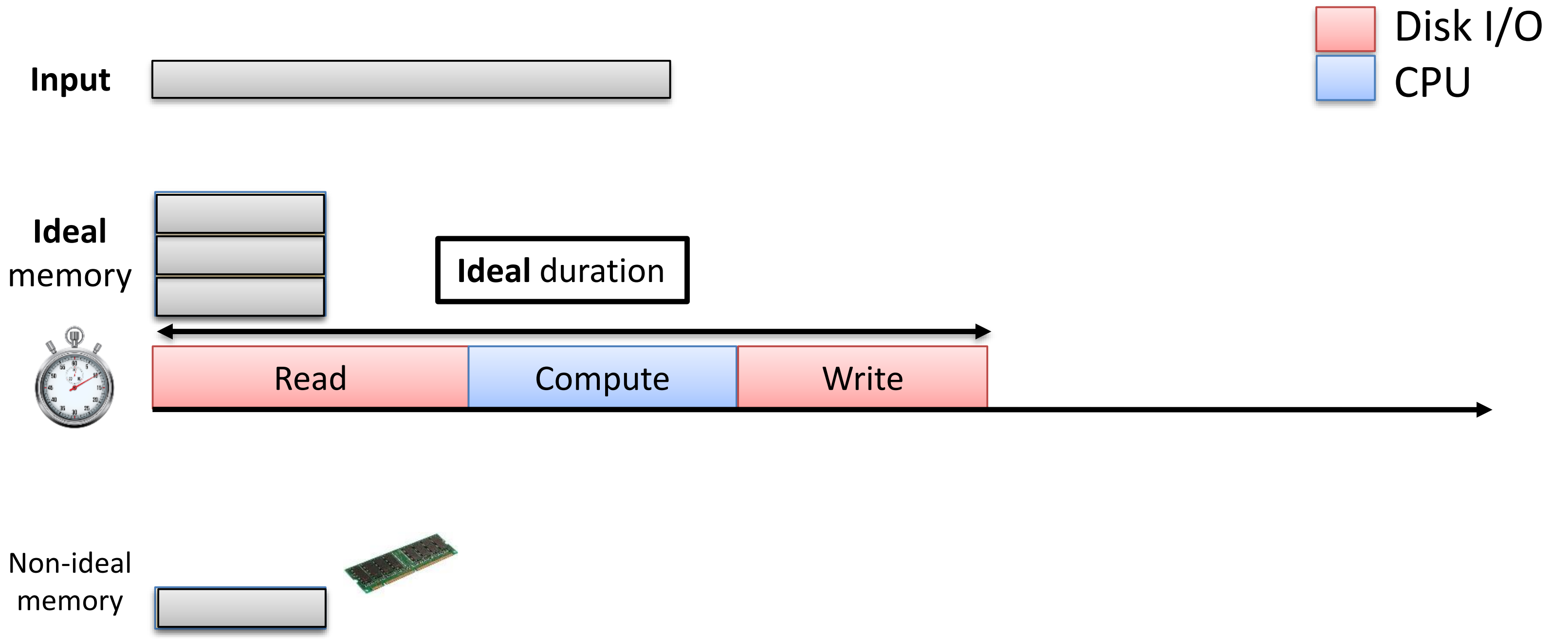
Context: batch jobs and their memory usage



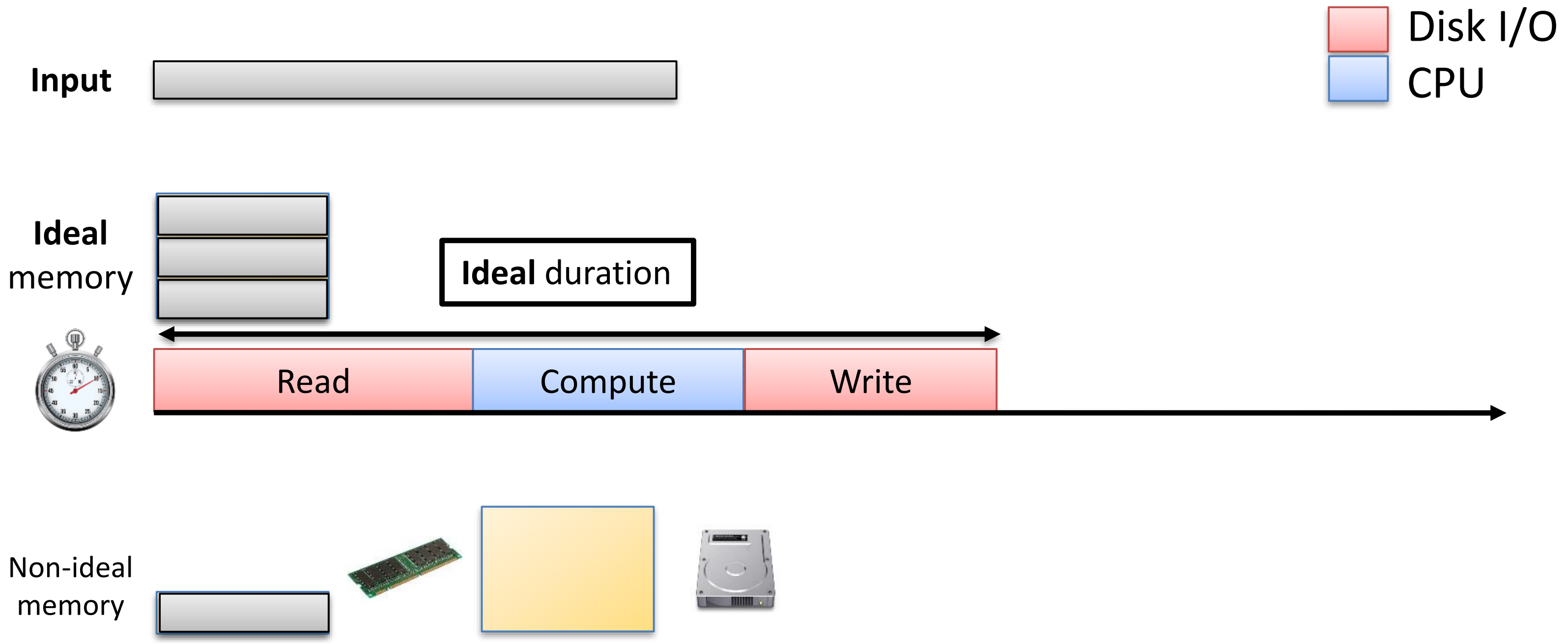
Context: batch jobs and their memory usage



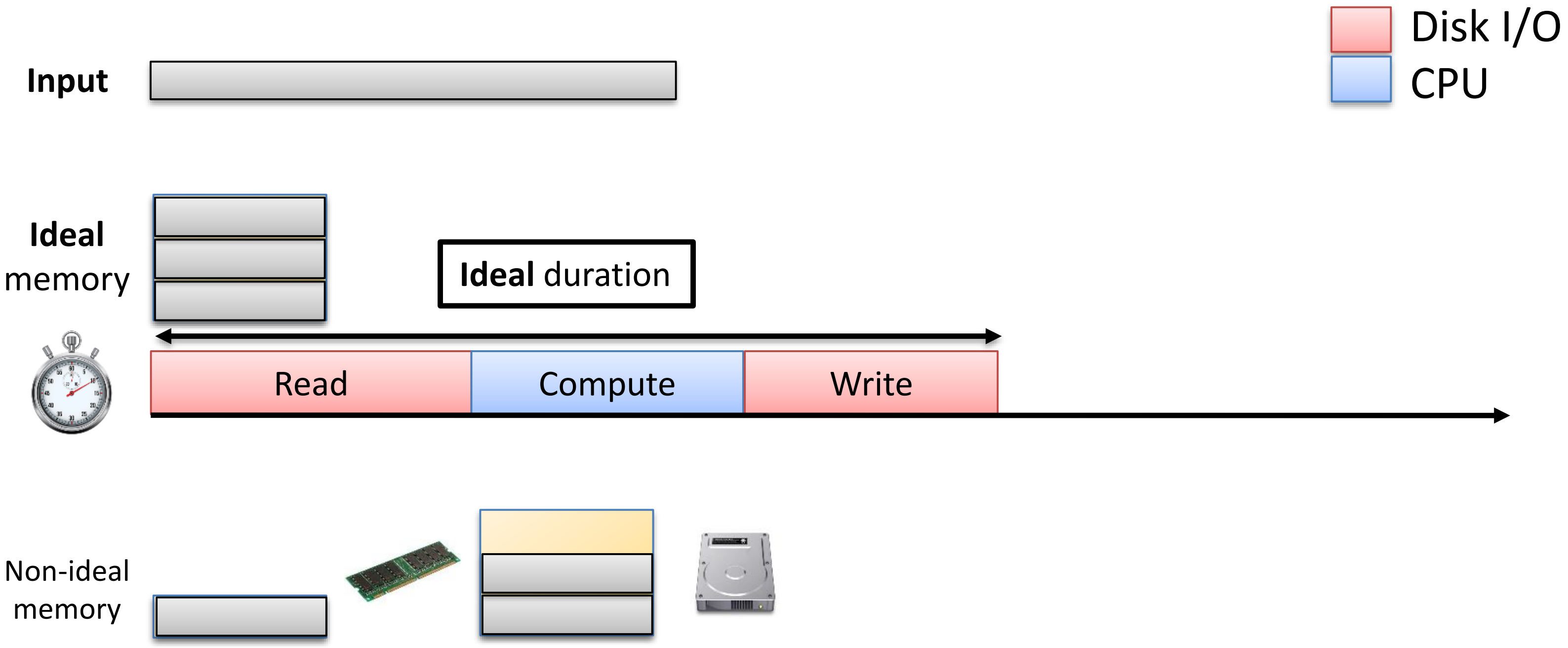
Context: batch jobs and their memory usage



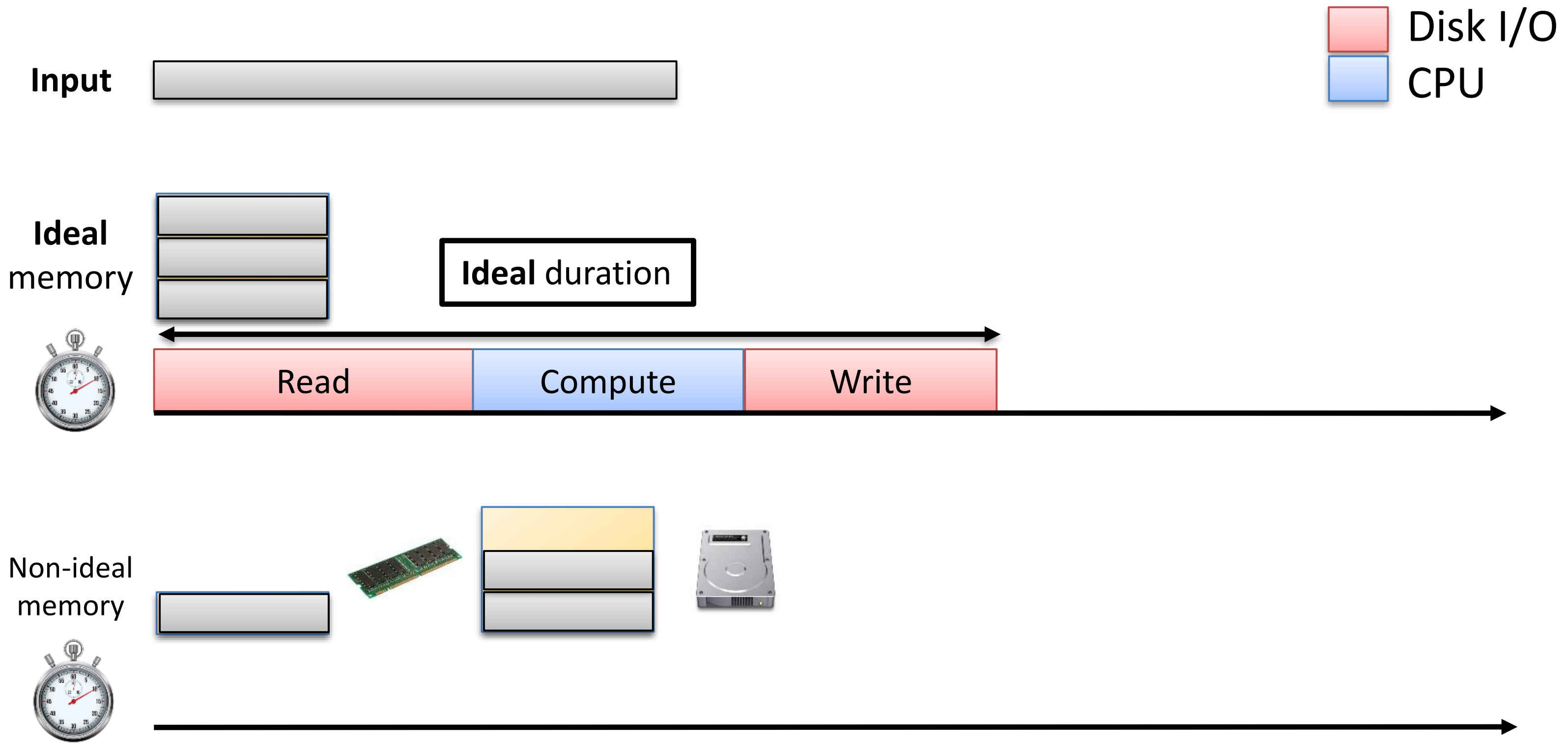
Context: batch jobs and their memory usage



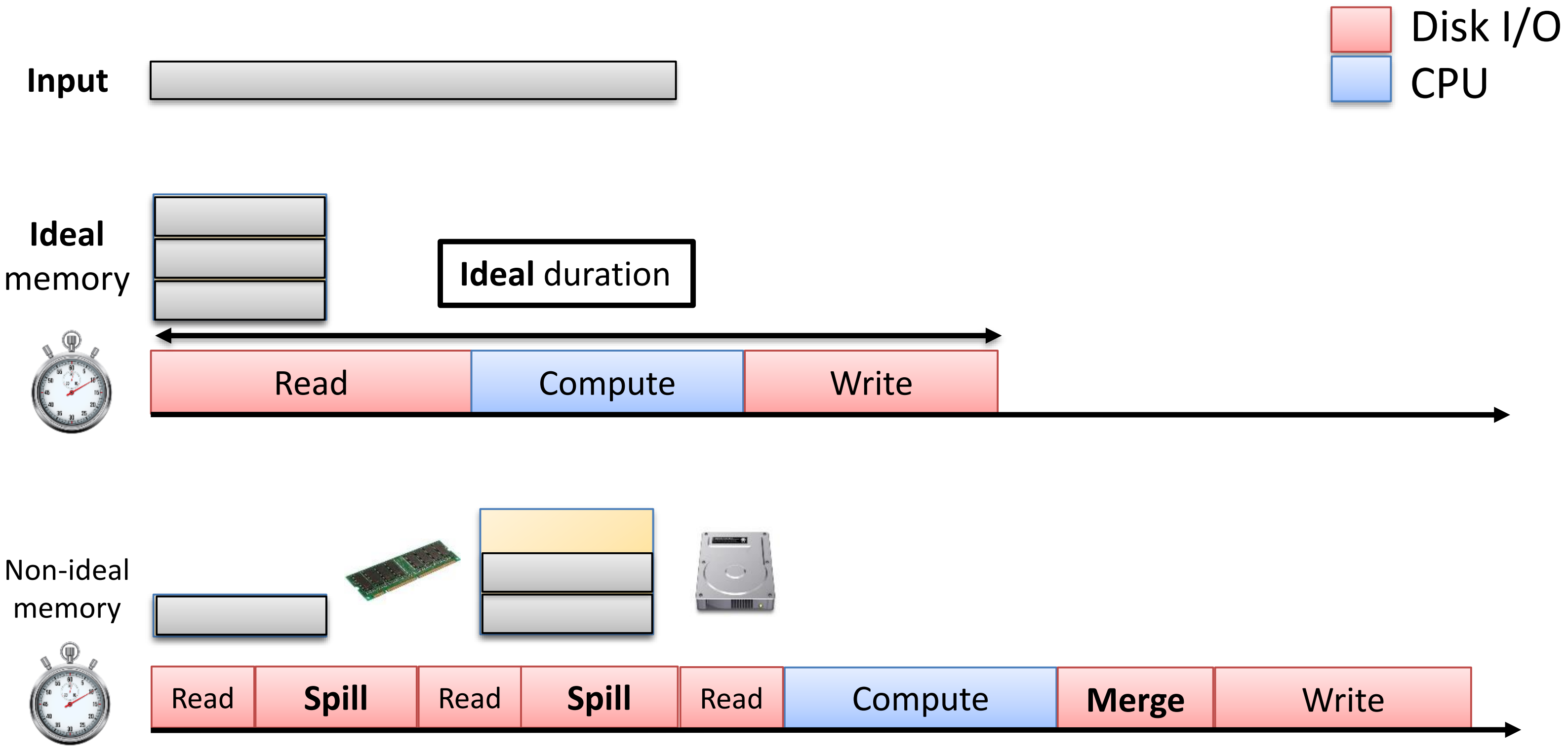
Context: batch jobs and their memory usage



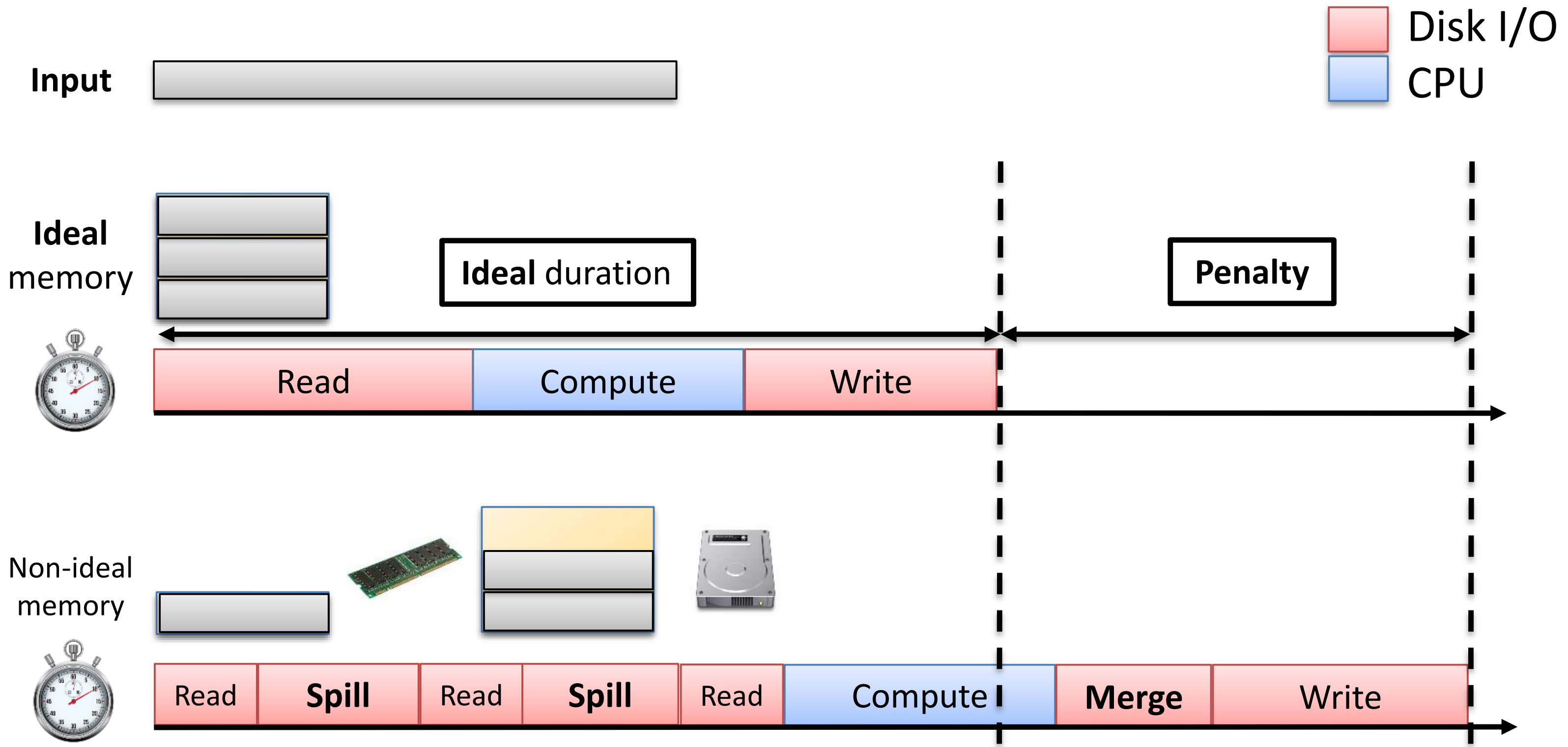
Context: batch jobs and their memory usage



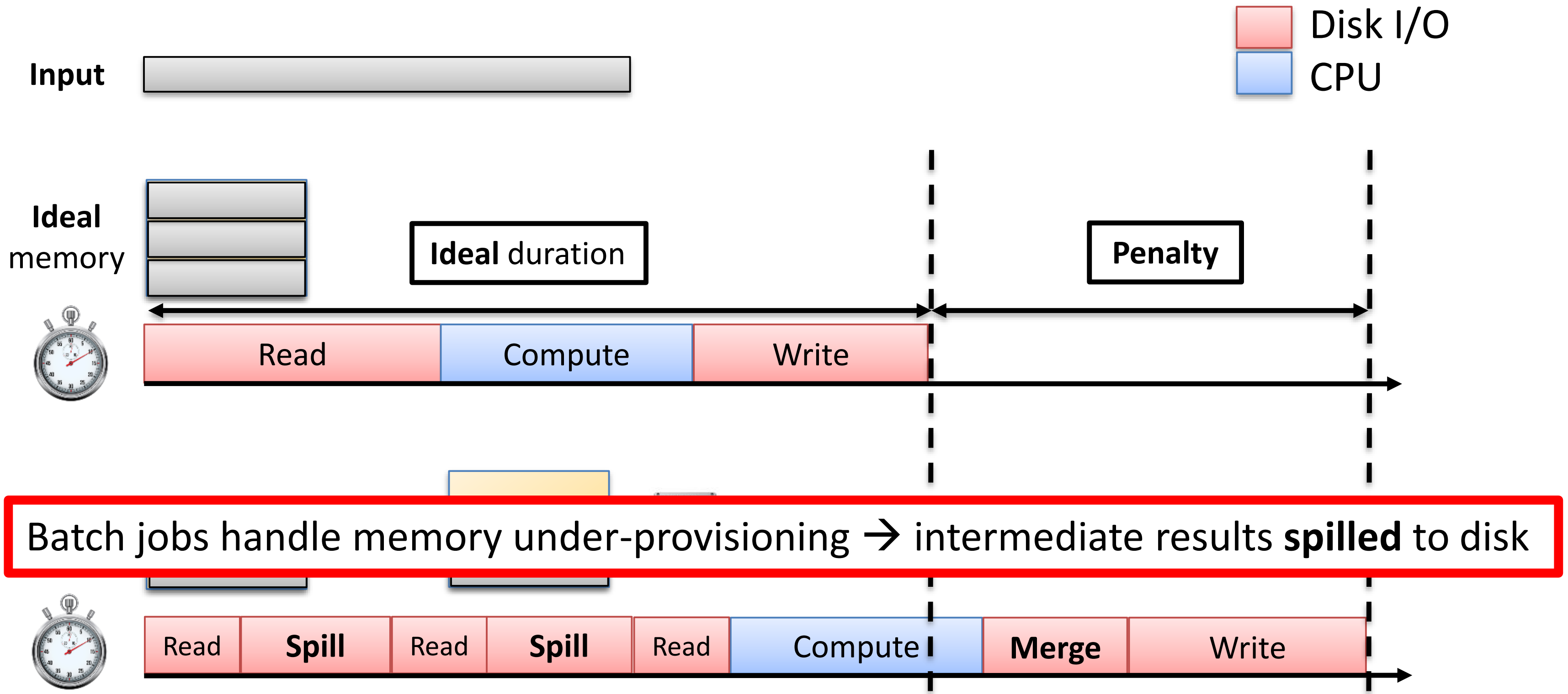
Context: batch jobs and their memory usage



Context: batch jobs and their memory usage



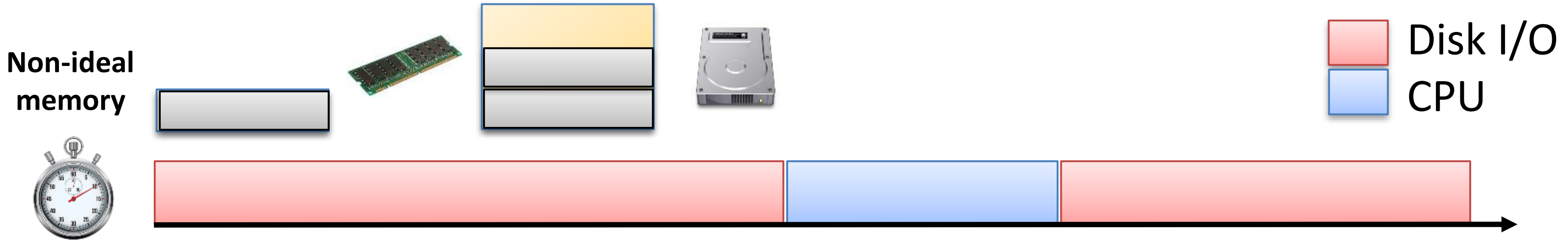
Context: batch jobs and their memory usage



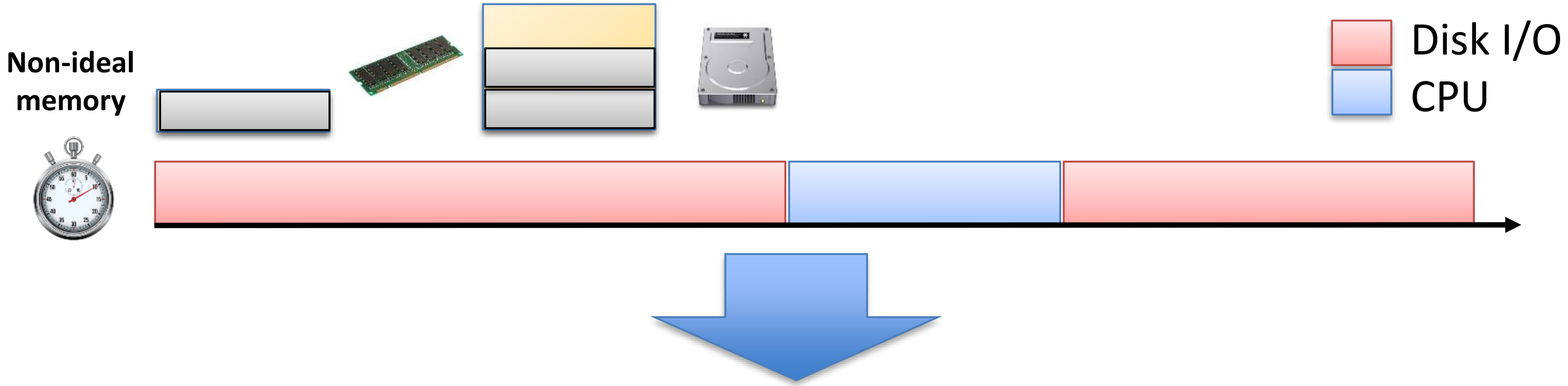
Batch jobs handle memory under-provisioning → intermediate results **spilled** to disk

MEMORY ELASTICITY

What is Memory Elasticity?

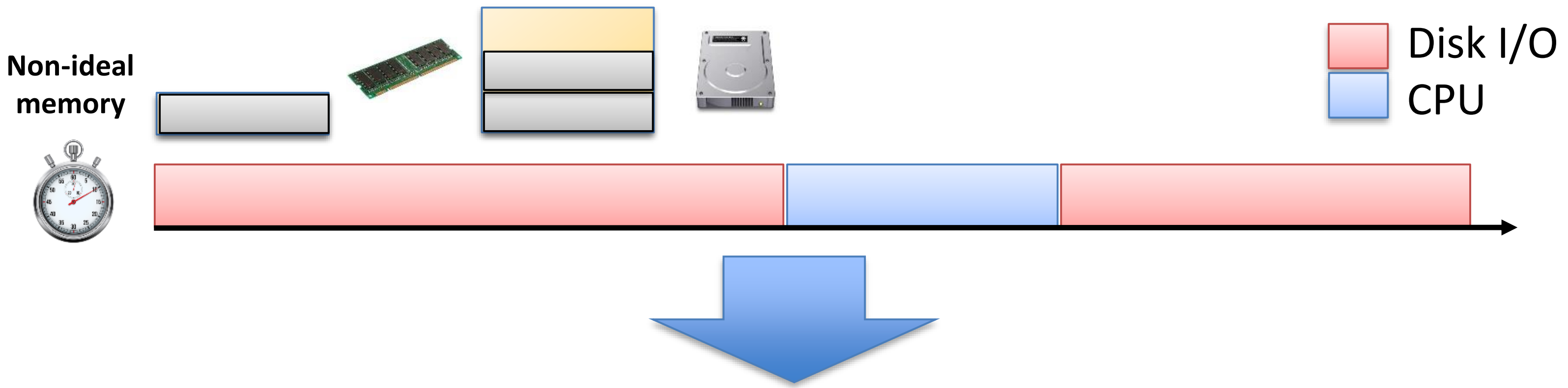


What is Memory Elasticity?



✓ **Safely** constrain memory

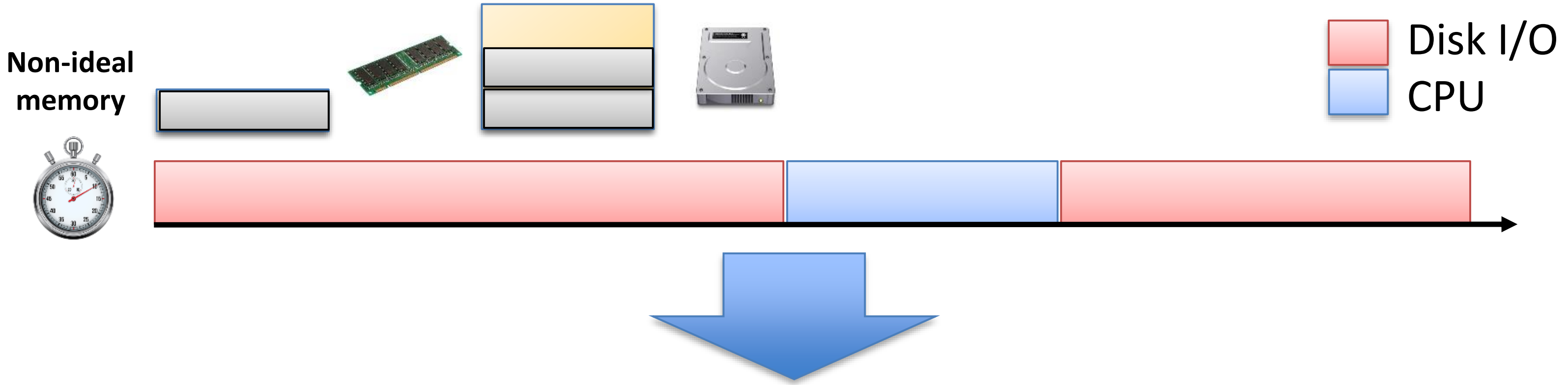
What is Memory Elasticity?



✓ **Safely** constrain memory

✓ Moderate **penalties**

What is Memory Elasticity?

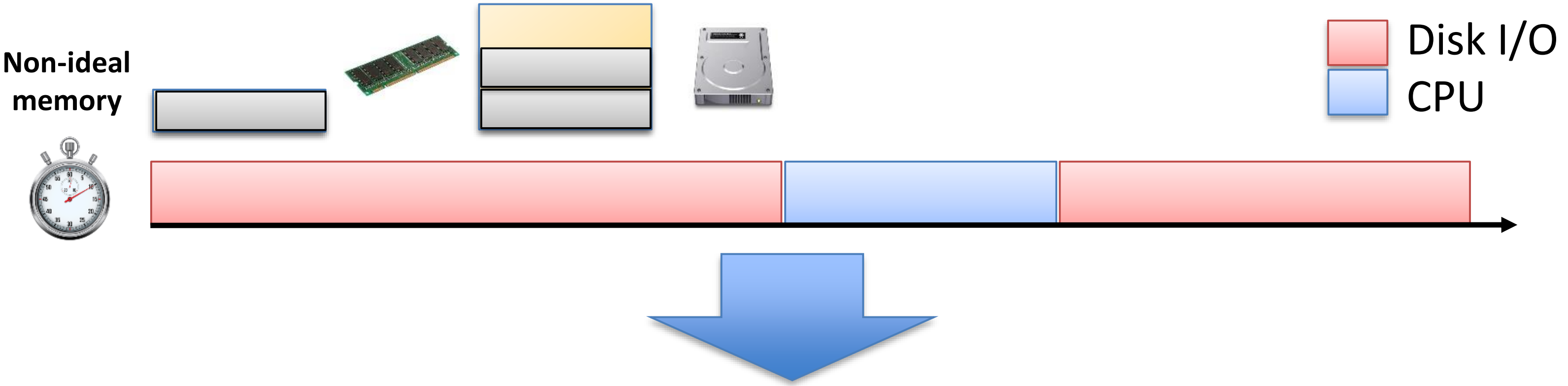


✓ **Safely** constrain memory

✓ Moderate **penalties**

✓ Highly **predictable**

What is Memory Elasticity?



✓ **Safely** constrain memory

✓ Moderate **penalties**

✓ Highly **predictable**

✓ **Ubiquitous** for most data-parallel apps

An empirical study of Memory Elasticity

- Analysis of 18 jobs across 8 different applications
- Constrain tasks' memory → measure **penalty**
- Bypass disk buffer cache (to not mask impact of spilling to disk)

Questions about Memory Elasticity

Questions about Memory Elasticity

- <u>Are the penalties large?</u>

Questions about Memory Elasticity

- Are the penalties large?
- Do penalties vary considerably w.r.t given memory?

Questions about Memory Elasticity

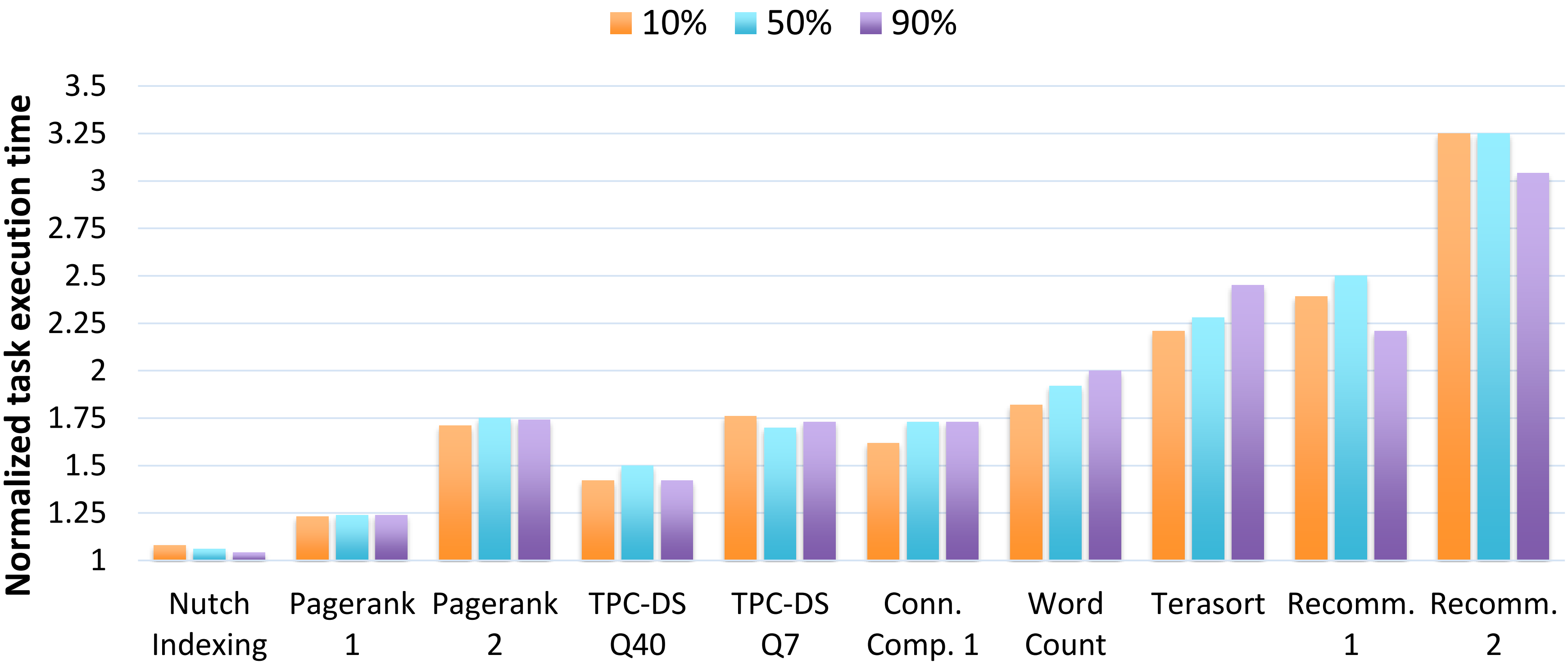
- Are the penalties large?
- Do penalties vary considerably w.r.t given memory?
- Does the additional I/O cause disk contention?

Questions about Memory Elasticity

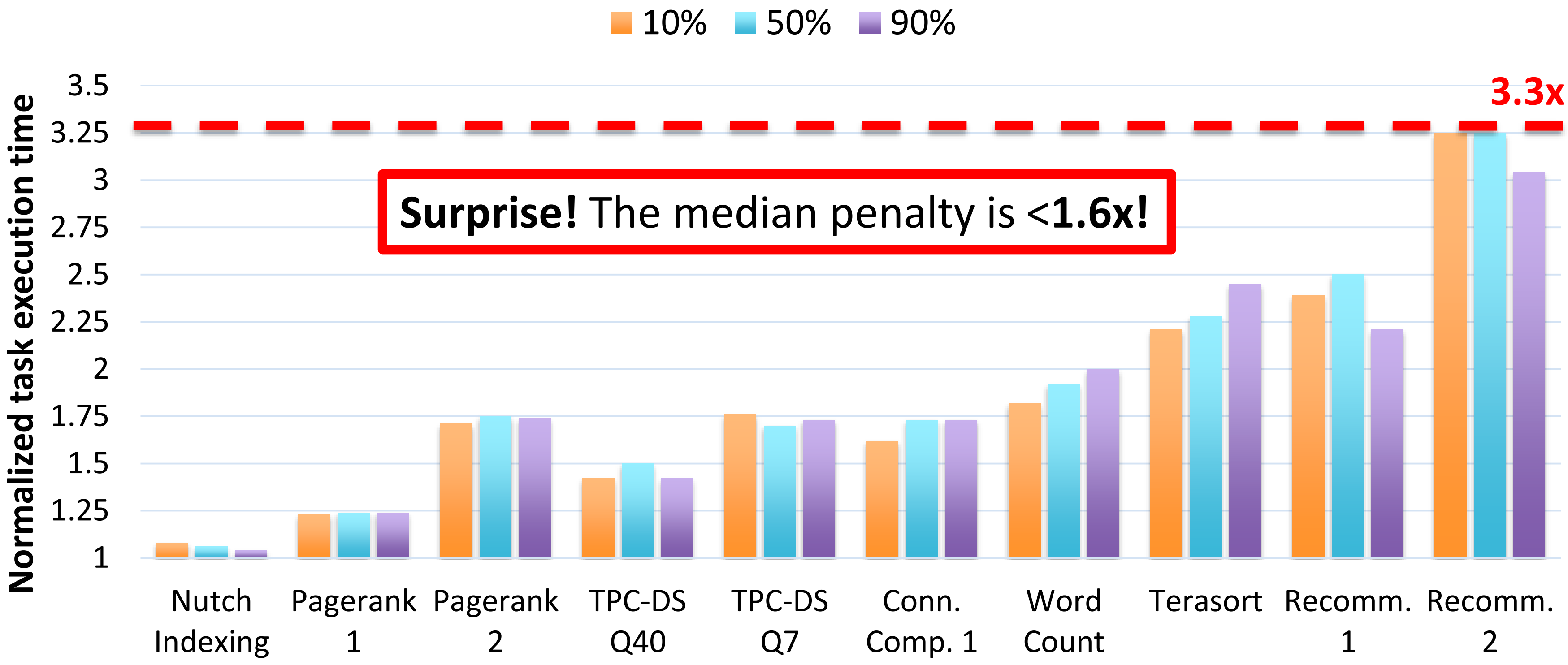
- Are the penalties large?
- Do penalties vary considerably w.r.t given memory?
- Does the additional I/O cause disk contention?

NOT SO MUCH!

Elasticity of Hadoop workloads: **Reducers**



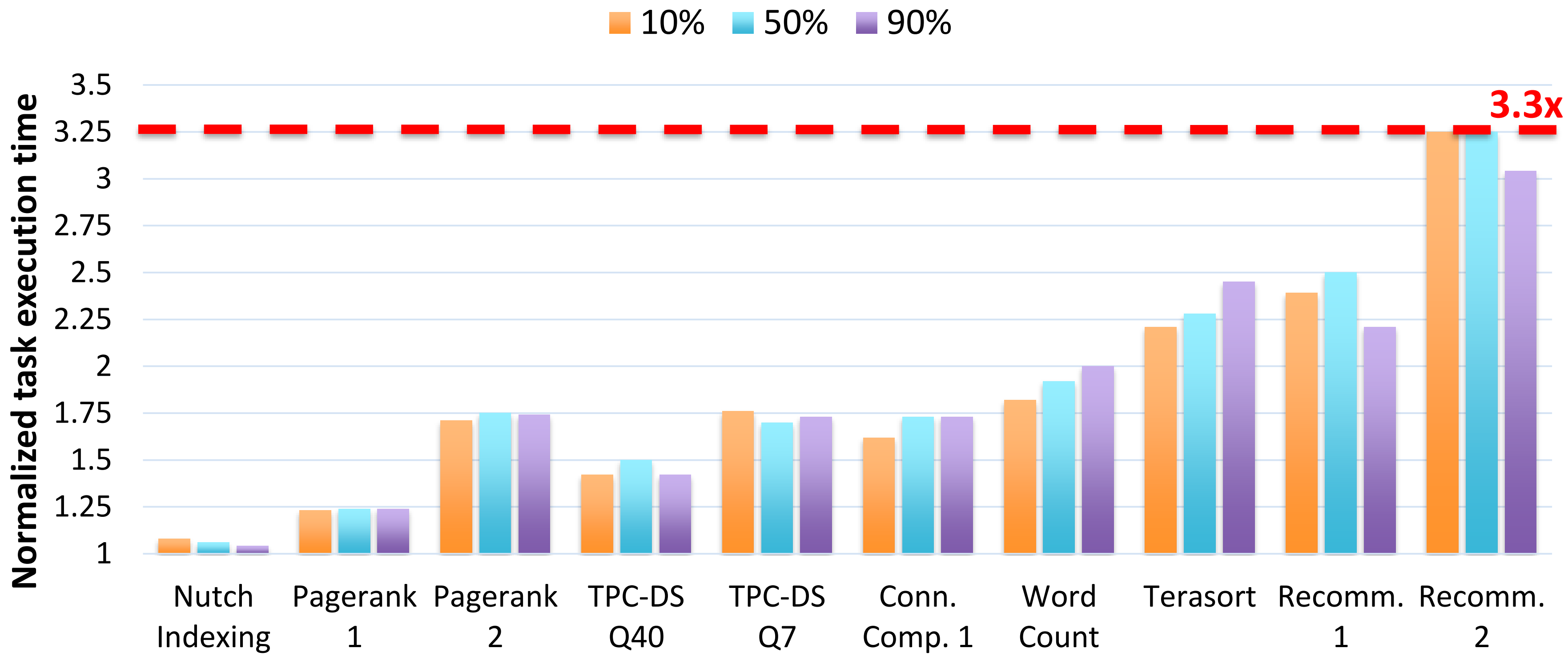
Elasticity of Hadoop workloads: **Reducers**



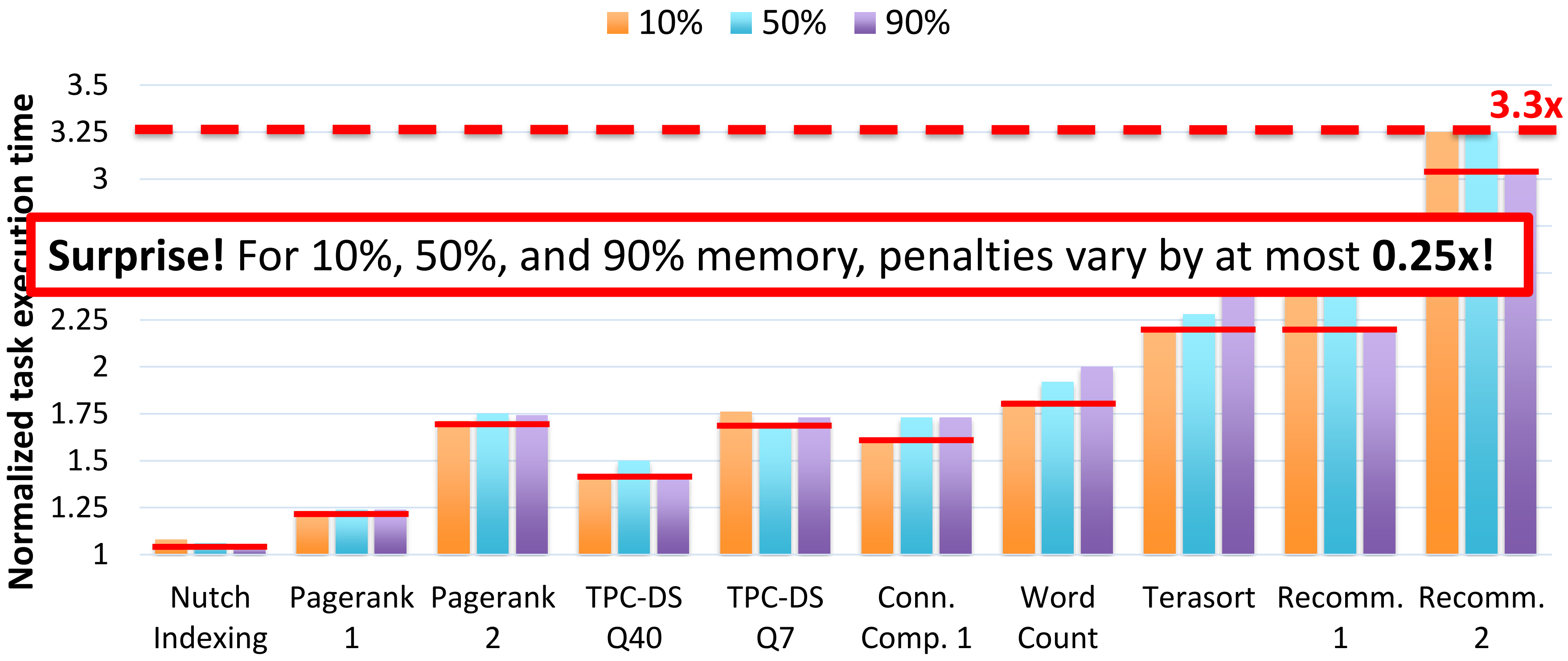
Why are the penalties so modest?

Data buffer – most of app. memory	• Memory pressure absorbed by data buffer
Sequential disk access	• Spilling records to disk is faster than OS paging
Logarithmic external merge algorithms	• Merge steps required << disk spills

Elasticity of Hadoop workloads: **Reducers**



Elasticity of Hadoop workloads: **Reducers**

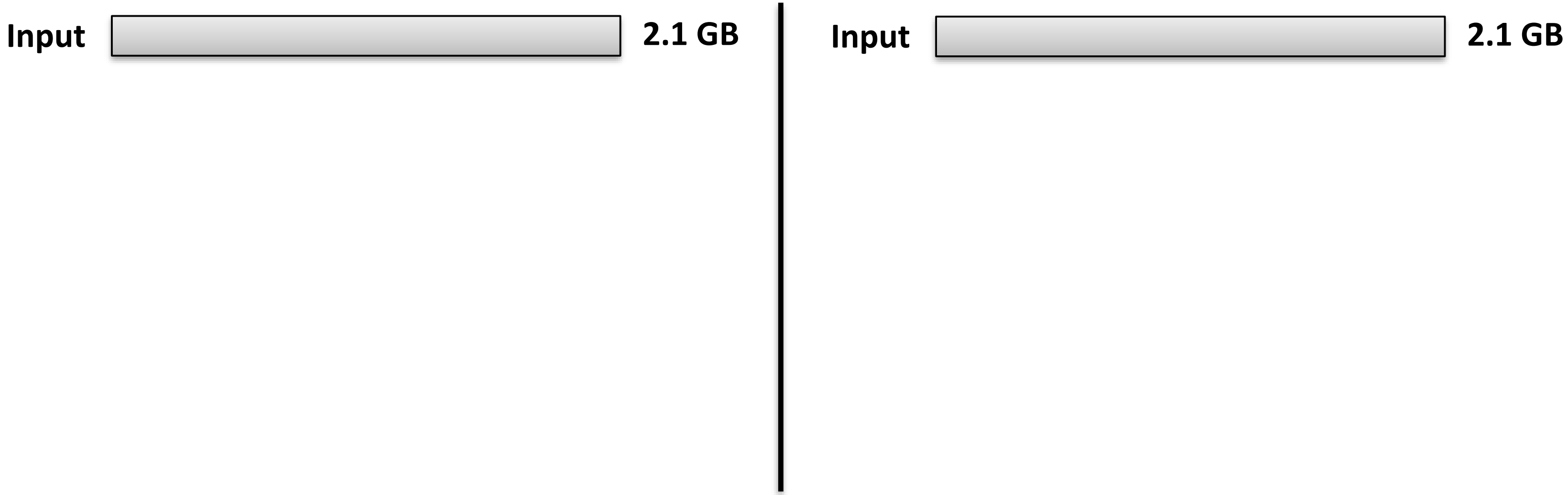


Why do penalties vary so little w/ memory?

- Static spilling threshold → comparable data spilling for 90% and 10% of memory

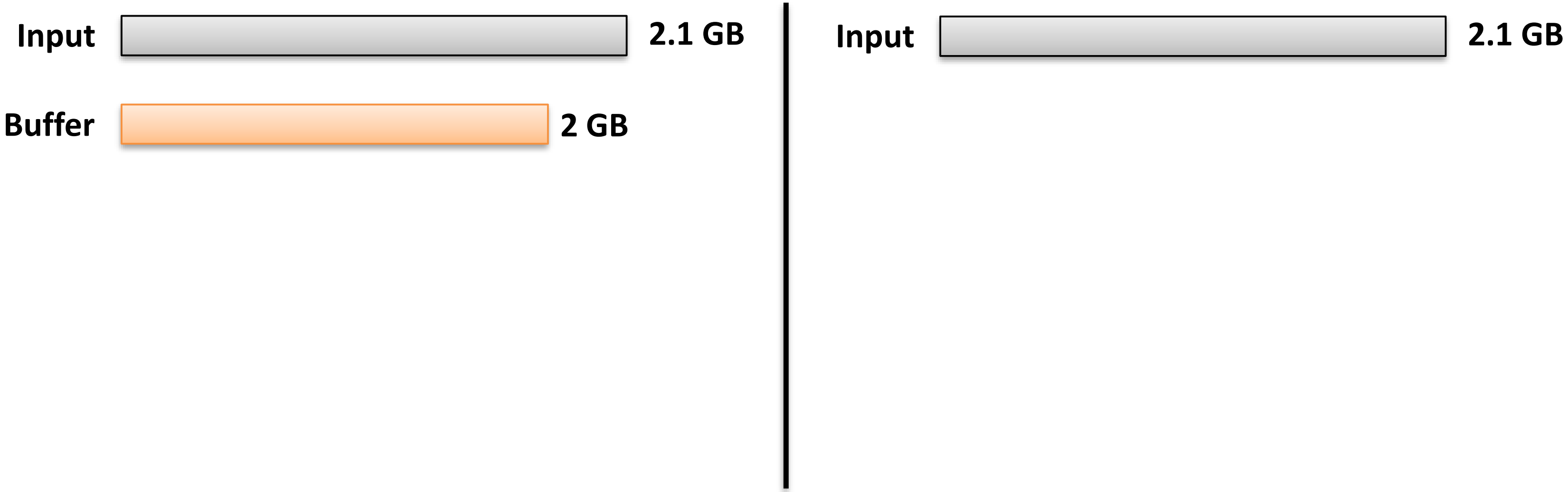
Why do penalties vary so little w/ memory?

- Static spilling threshold → comparable data spilling for 90% and 10% of memory



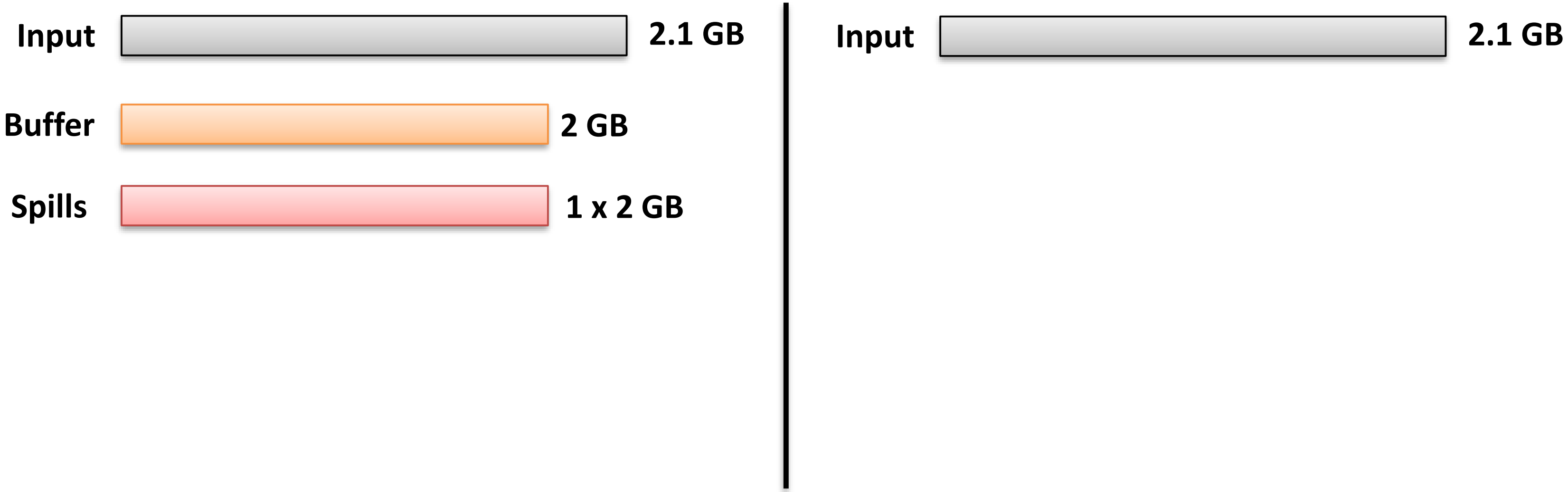
Why do penalties vary so little w/ memory?

- Static spilling threshold → comparable data spilling for 90% and 10% of memory



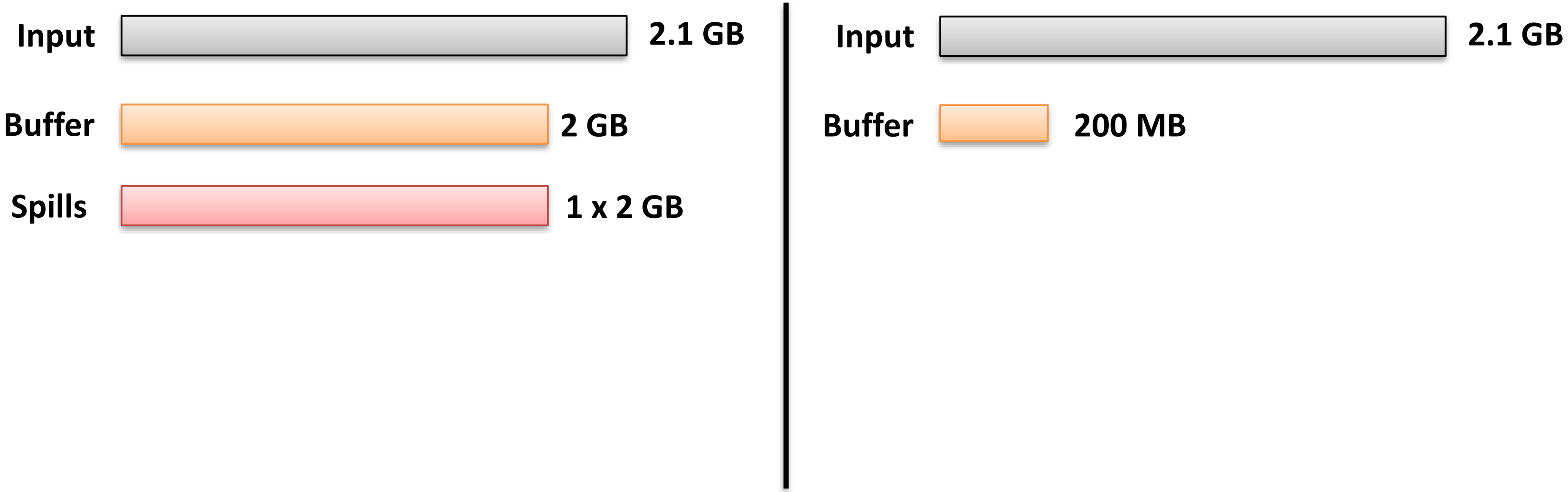
Why do penalties vary so little w/ memory?

- Static spilling threshold → comparable data spilling for 90% and 10% of memory



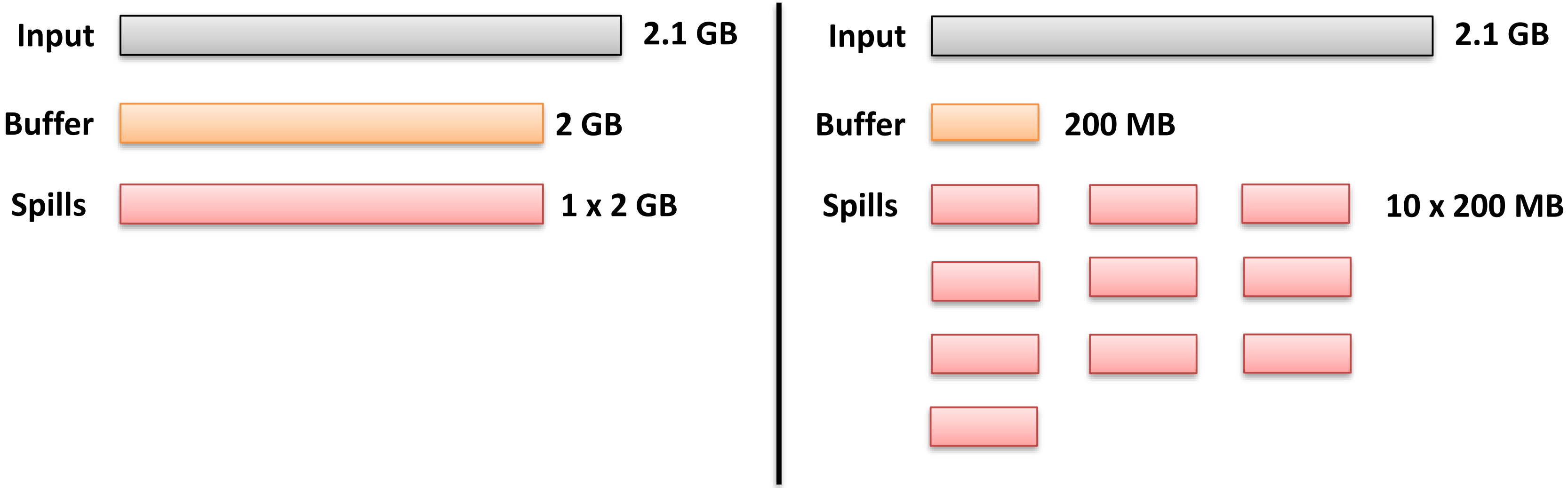
Why do penalties vary so little w/ memory?

- Static spilling threshold → comparable data spilling for 90% and 10% of memory



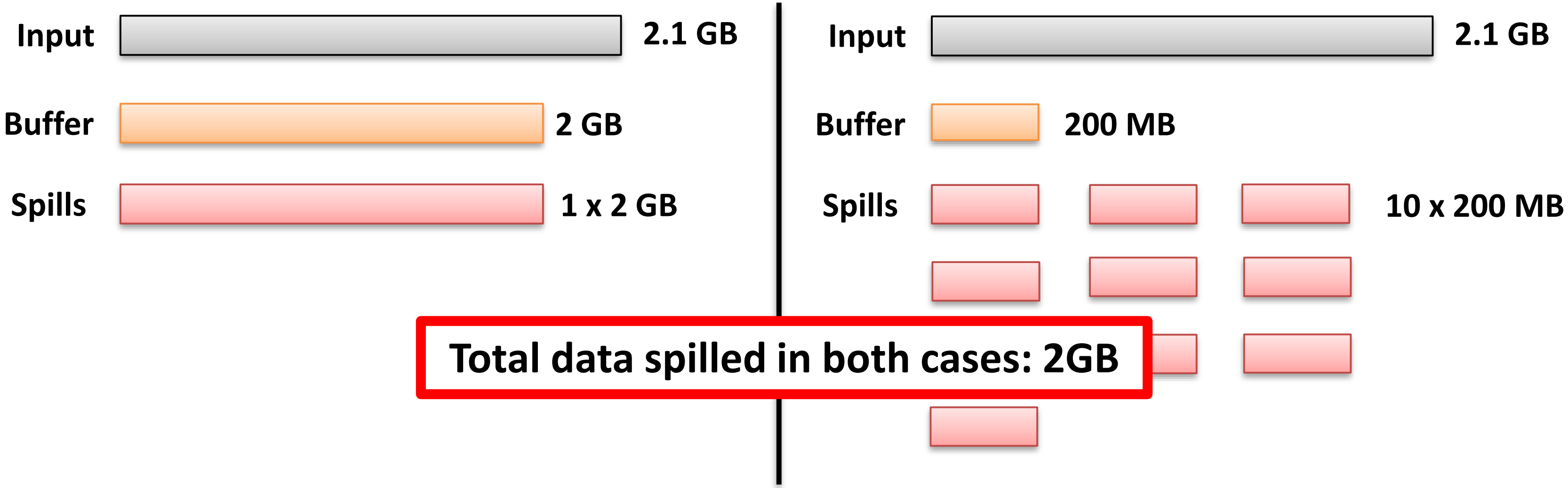
Why do penalties vary so little w/ memory?

- Static spilling threshold → comparable data spilling for 90% and 10% of memory

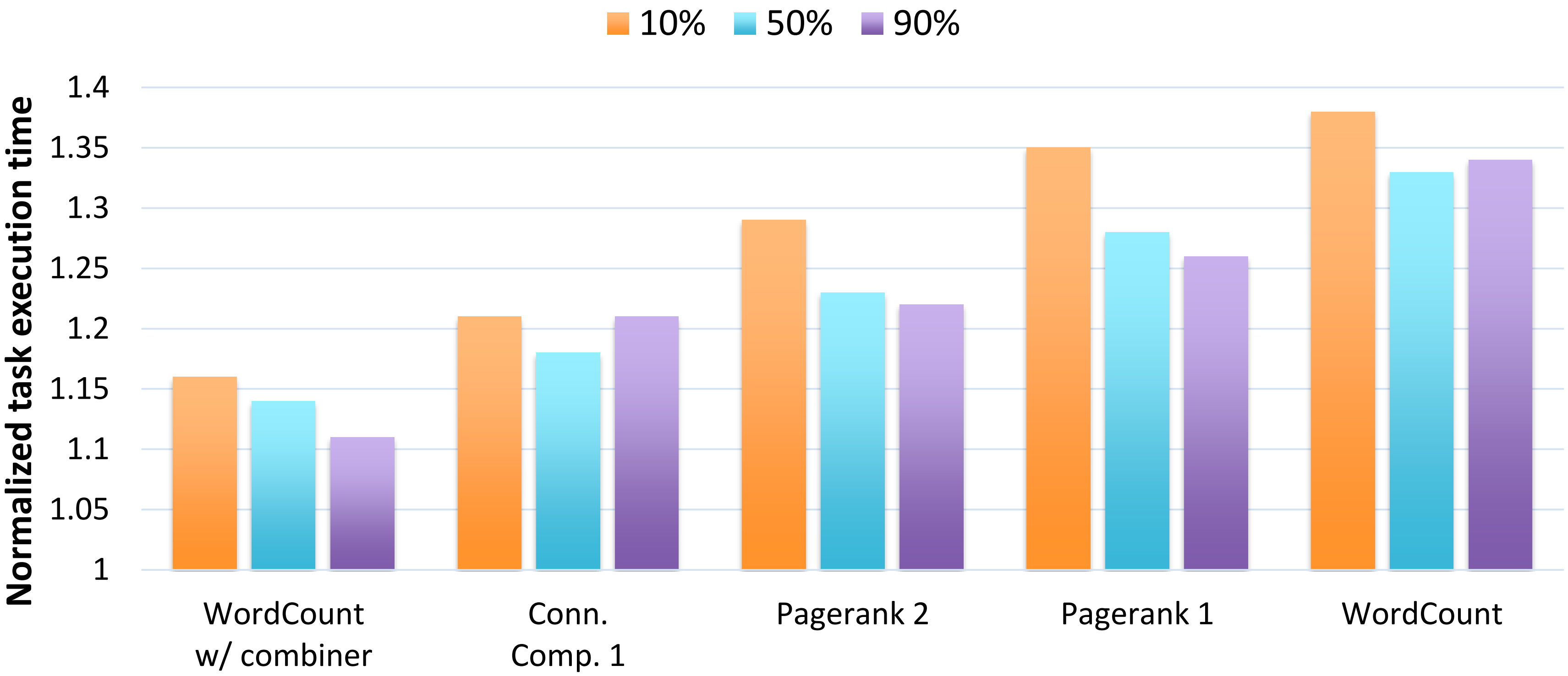


Why do penalties vary so little w/ memory?

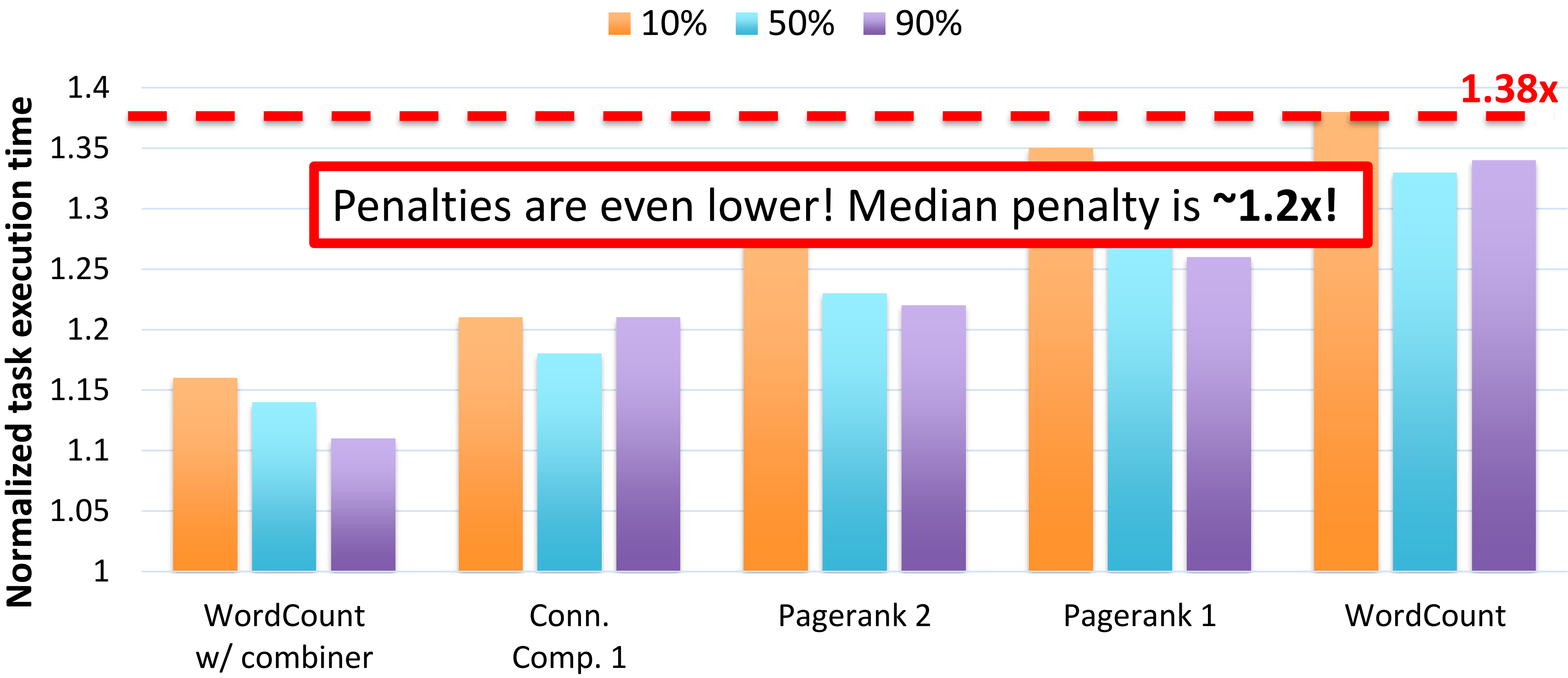
- Static spilling threshold → comparable data spilling for 90% and 10% of memory



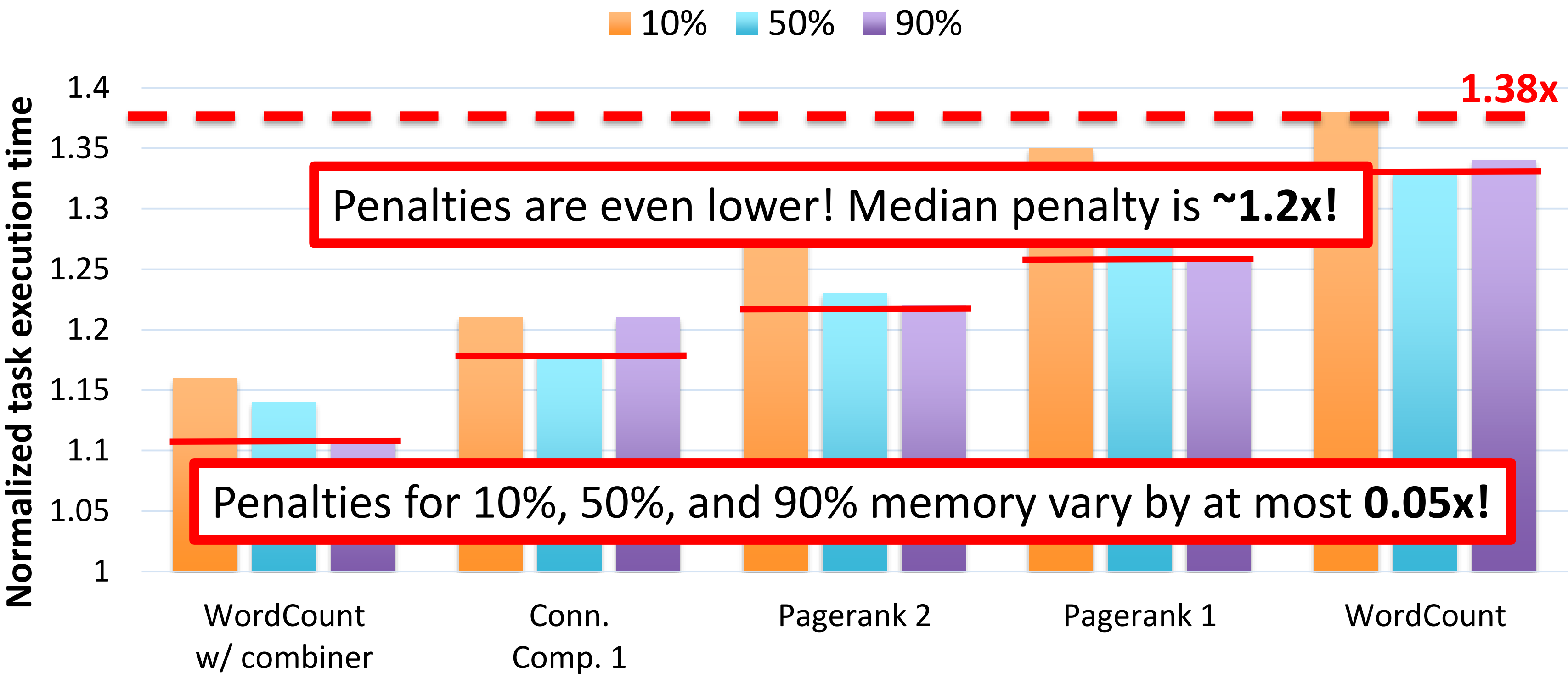
Elasticity of Hadoop workloads: **Mappers**



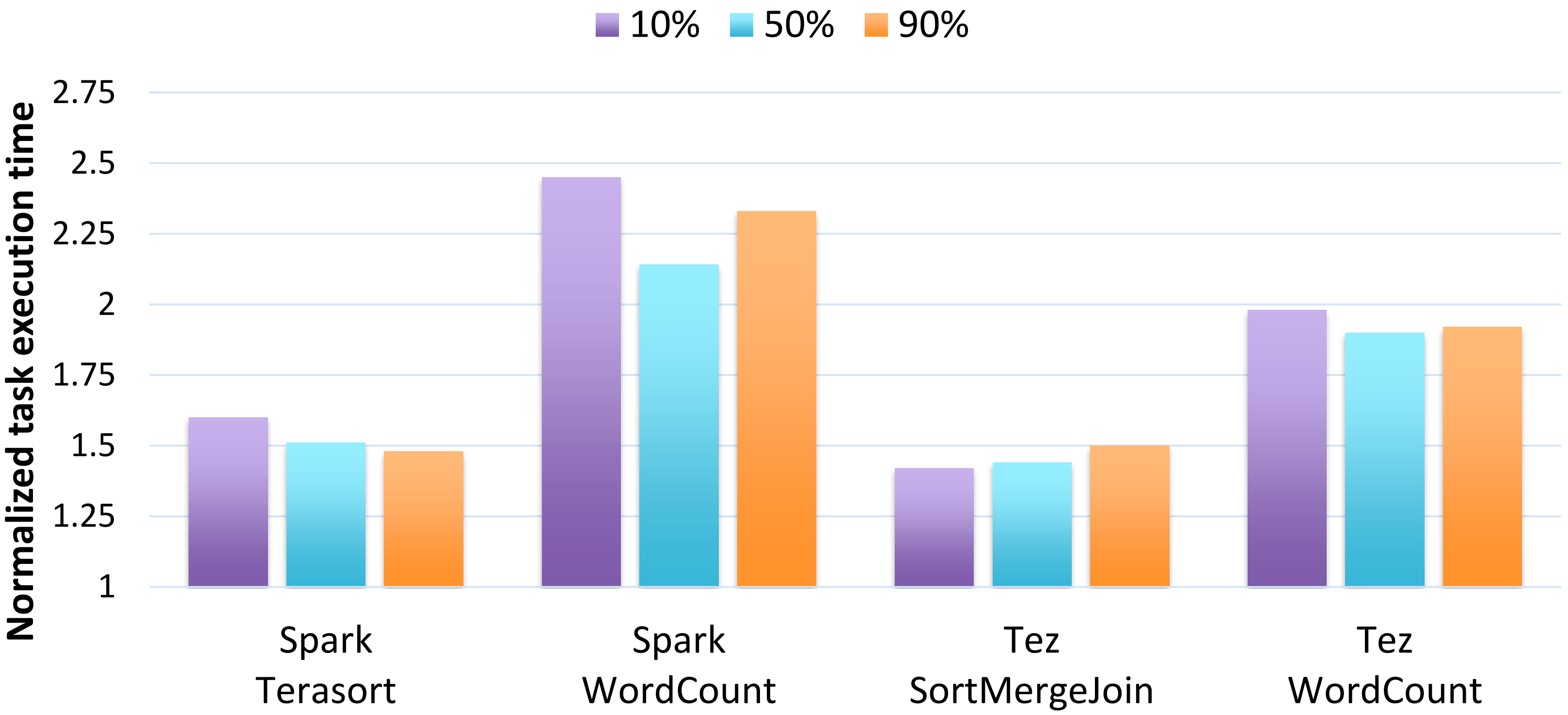
Elasticity of Hadoop workloads: **Mappers**



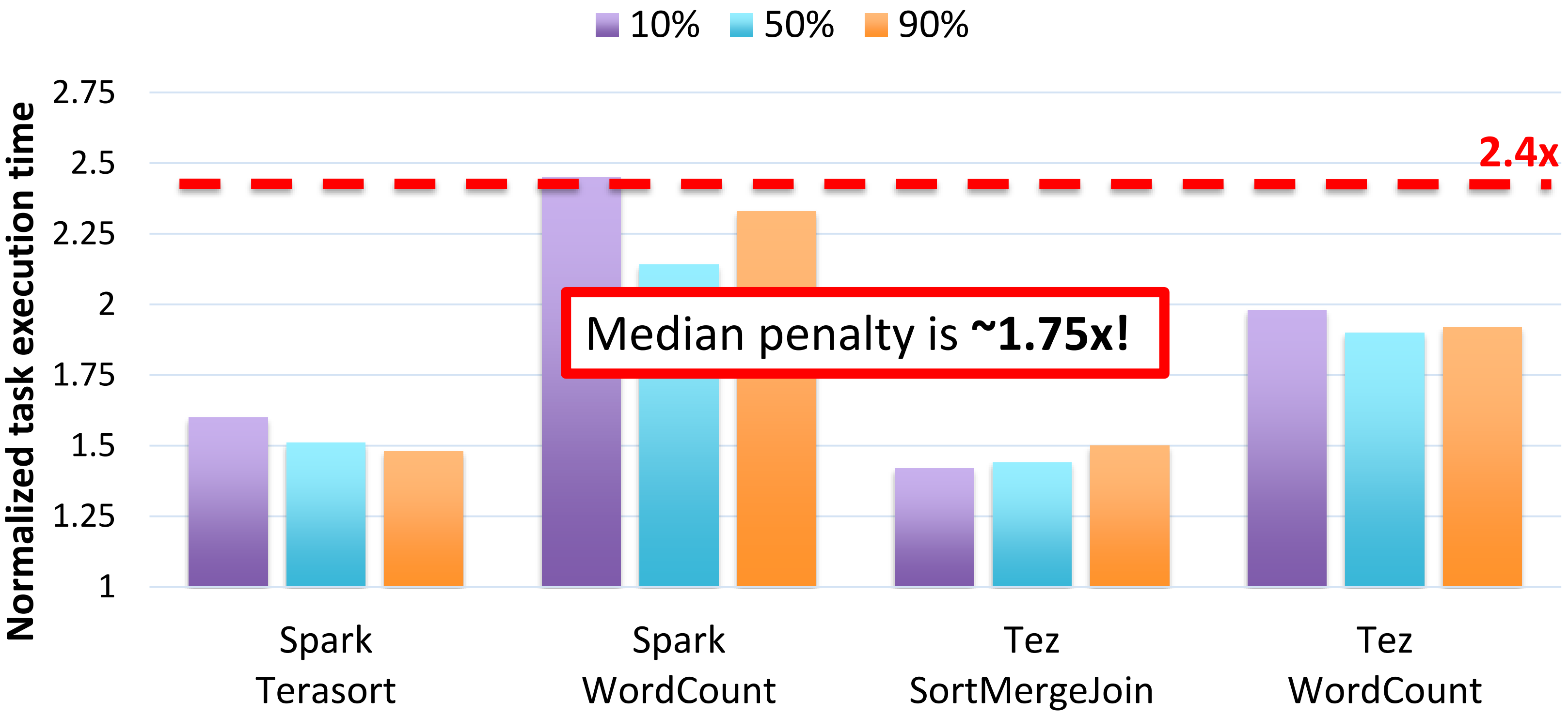
Elasticity of Hadoop workloads: **Mappers**



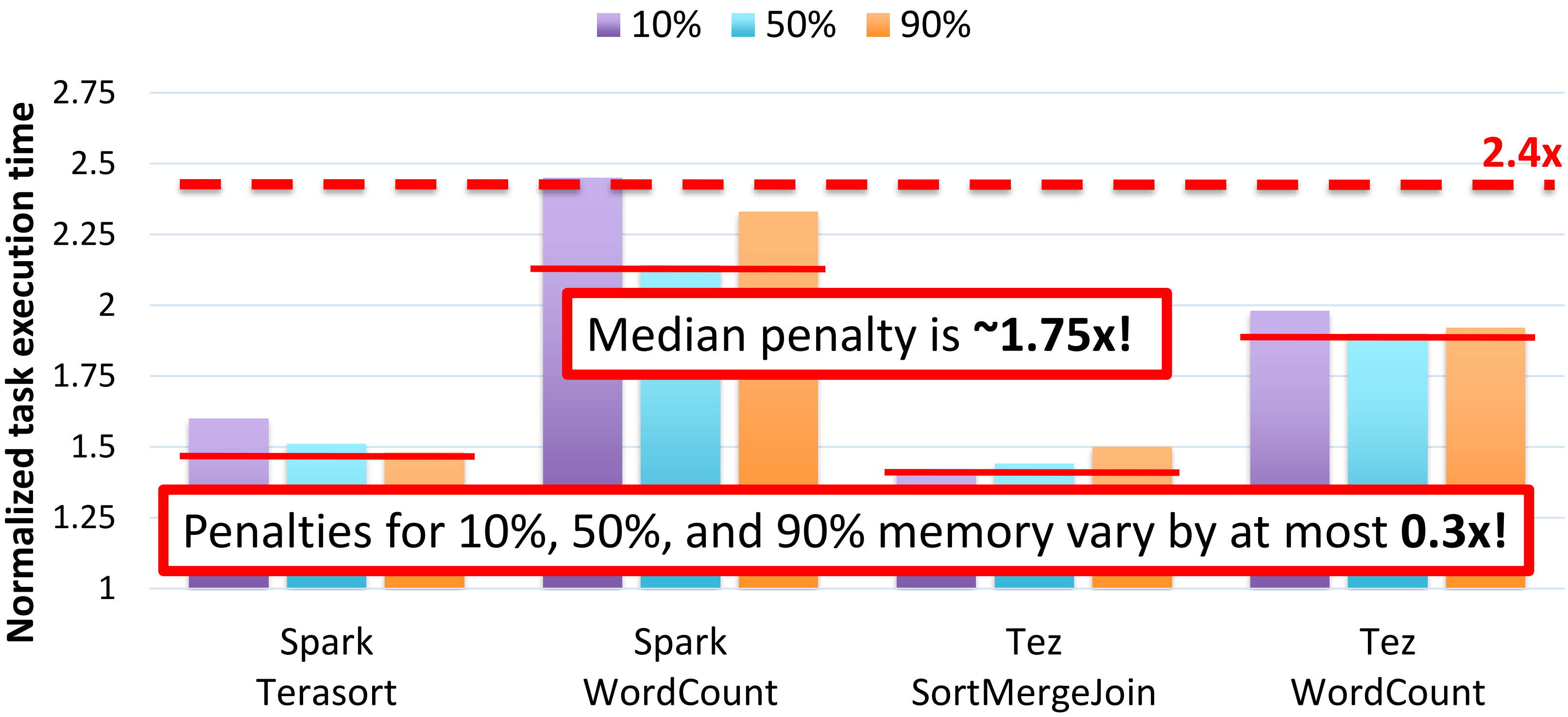
Elasticity of Spark and Tez workloads



Elasticity of Spark and Tez workloads

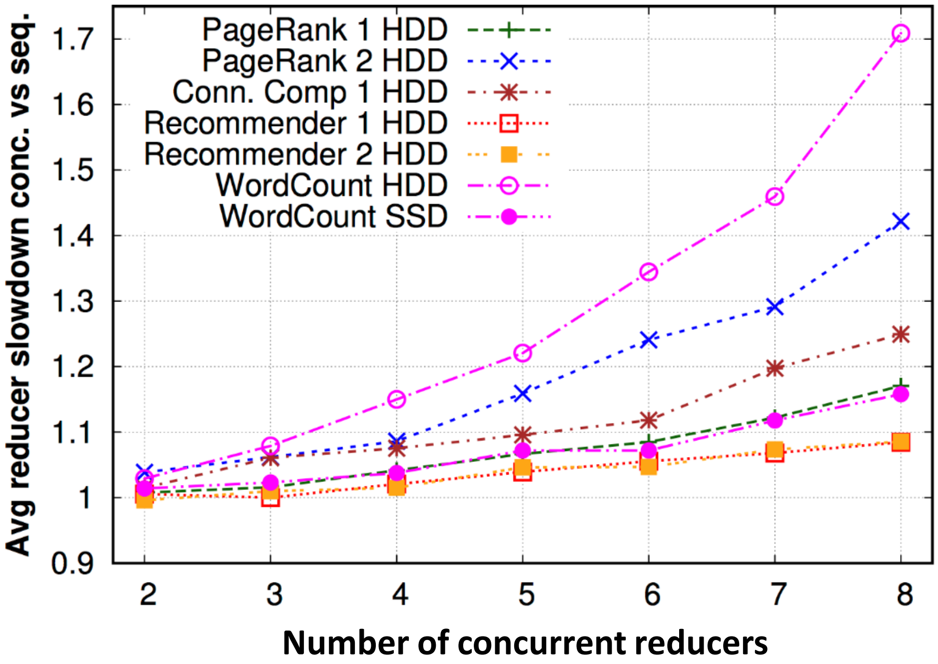


Elasticity of Spark and Tez workloads



Does the additional I/O cause disk contention?

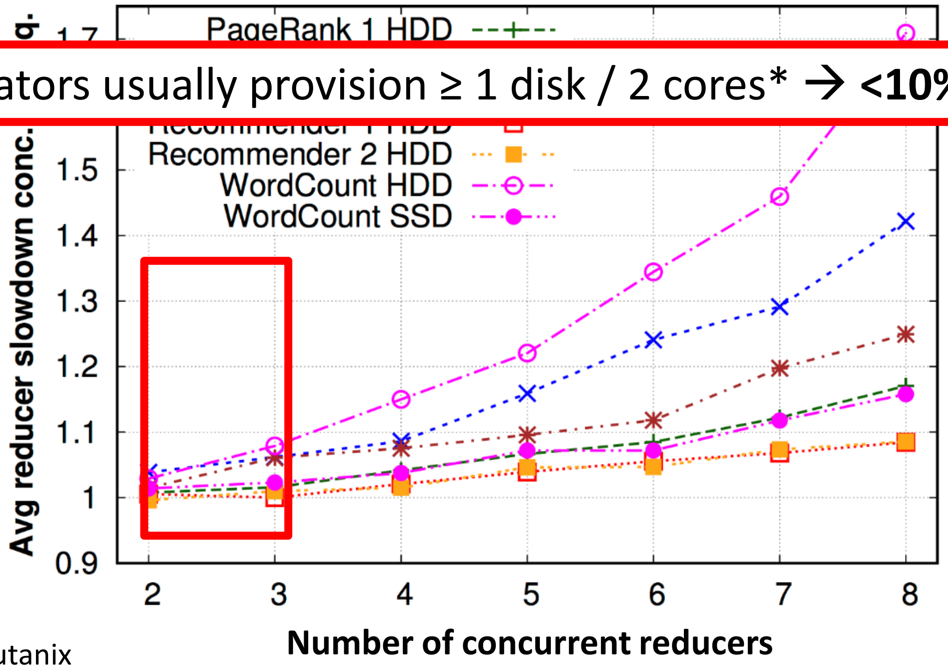
- Measure slowdown of <u>elastic</u> tasks on same machine spilling to the same disk



Does the additional I/O cause disk contention?

- Measure slowdown of <u>elastic</u> tasks on same machine spilling to the same disk

Cluster operators usually provision ≥ 1 disk / 2 cores* → **<10%** slowdown!



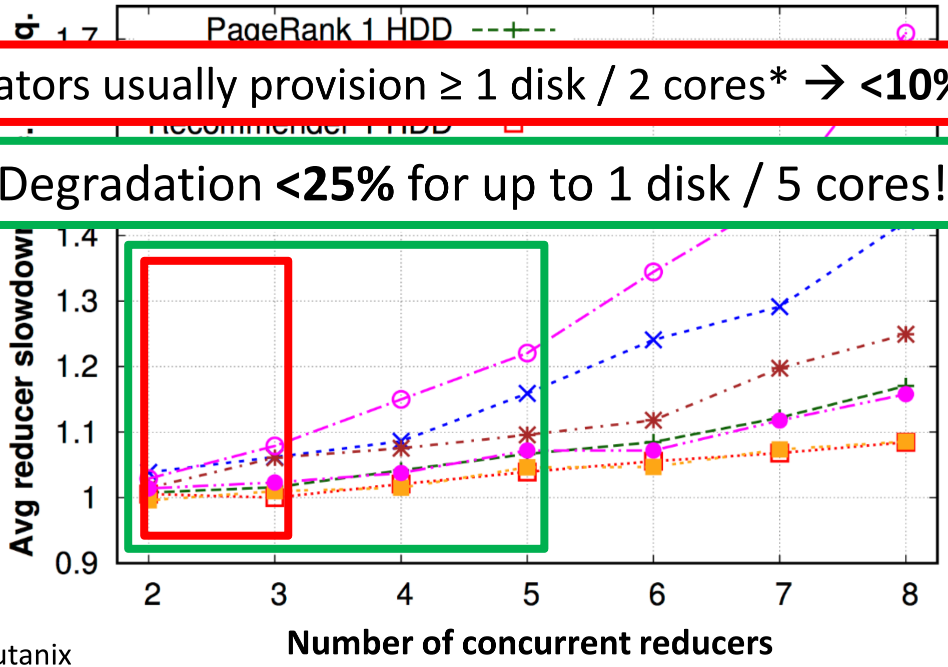
* Facebook (2010) and Nutanix

Does the additional I/O cause disk contention?

- Measure slowdown of elastic tasks on same machine spilling to the same disk

Cluster operators usually provision ≥ 1 disk / 2 cores* → **<10%** slowdown!

Degradation **<25%** for up to 1 disk / 5 cores!



* Facebook (2010) and Nutanix

Summary: Memory Elasticity of real workloads

✓ **Modest performance penalties (<1.6x median)**

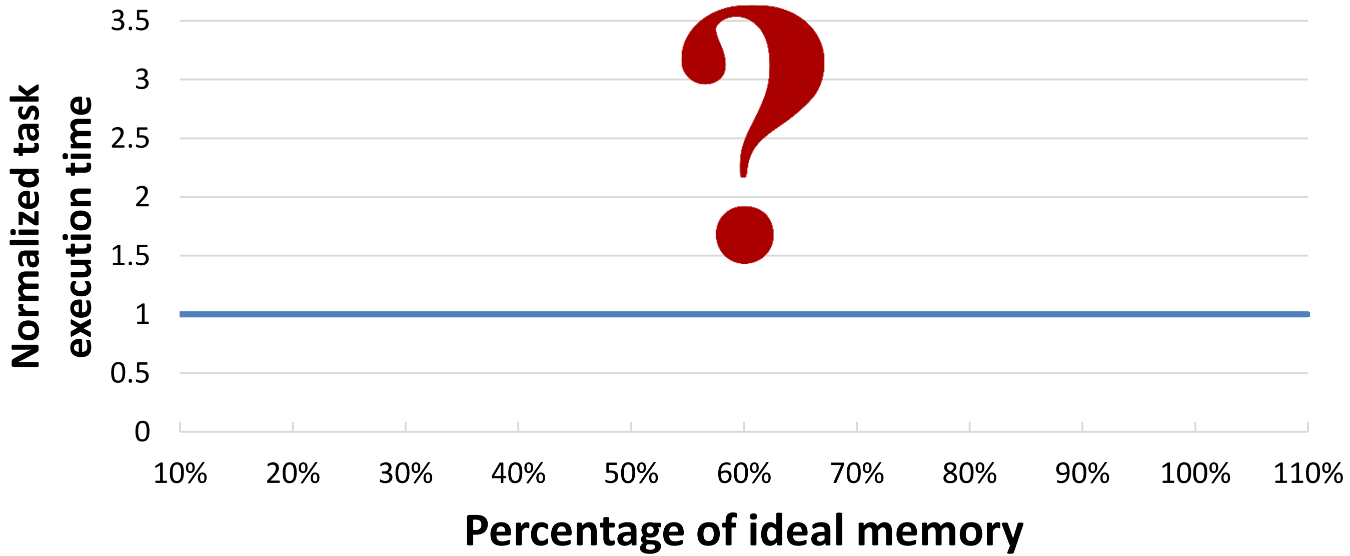
✓ **Similar penalties for 10% and 90% of ideal memory**

✓ **Disk contention negligible for existing clusters' setup (<10%)**

MODELING MEMORY ELASTICITY

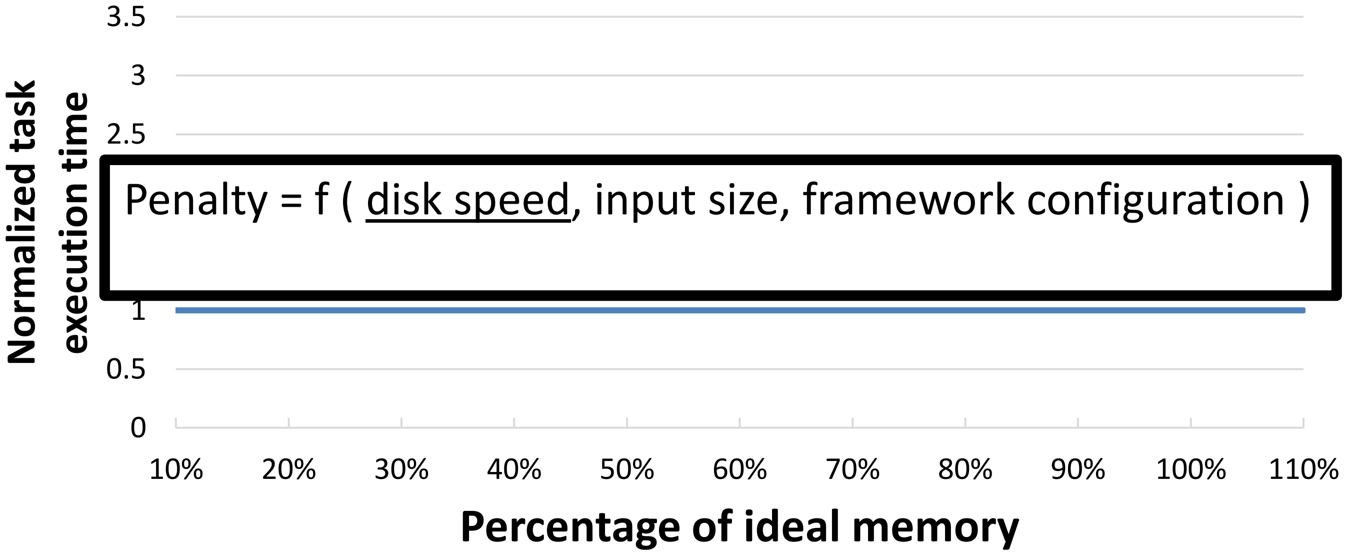
Modeling Memory Elasticity

- How does penalty vary for a task?



Modeling Memory Elasticity

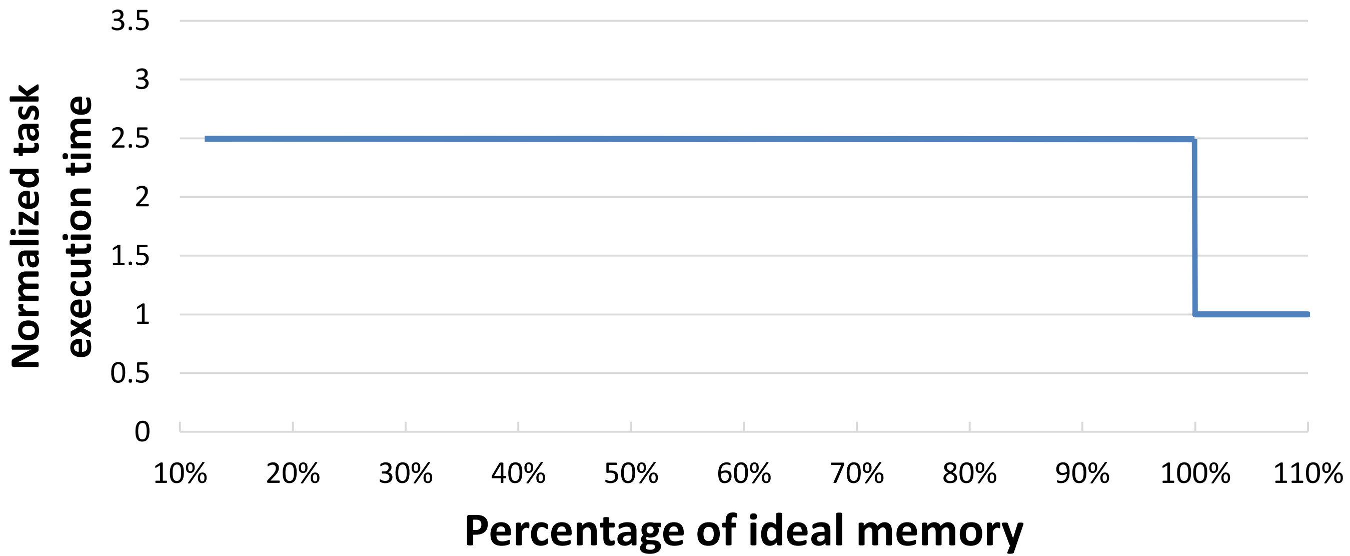
- How does penalty vary for a task?



Penalty = f (disk speed, input size, framework configuration)

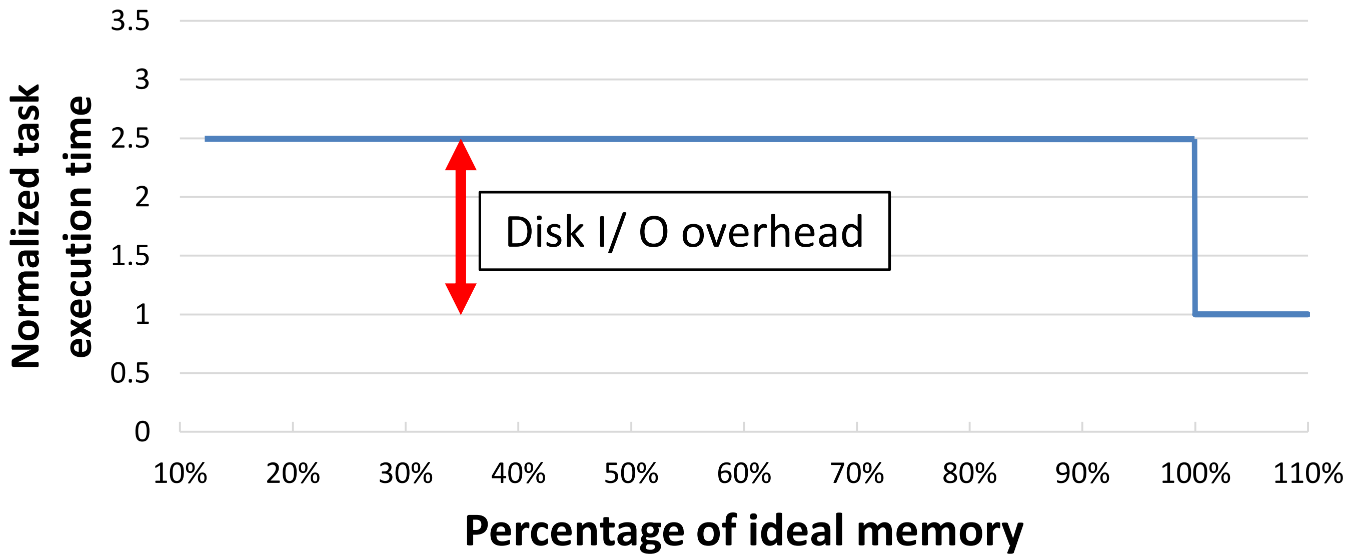
Modeling Memory Elasticity

- Penalties vary little between percentages → <u>step</u> model



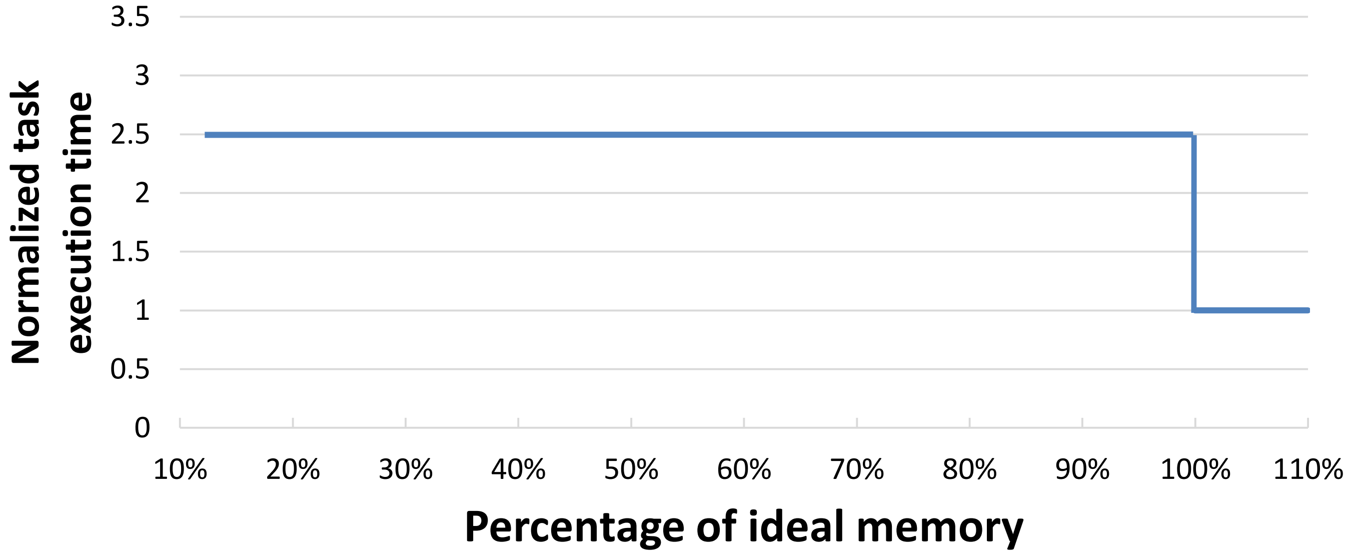
Modeling Memory Elasticity

- Penalties vary little between percentages → <u>step</u> model



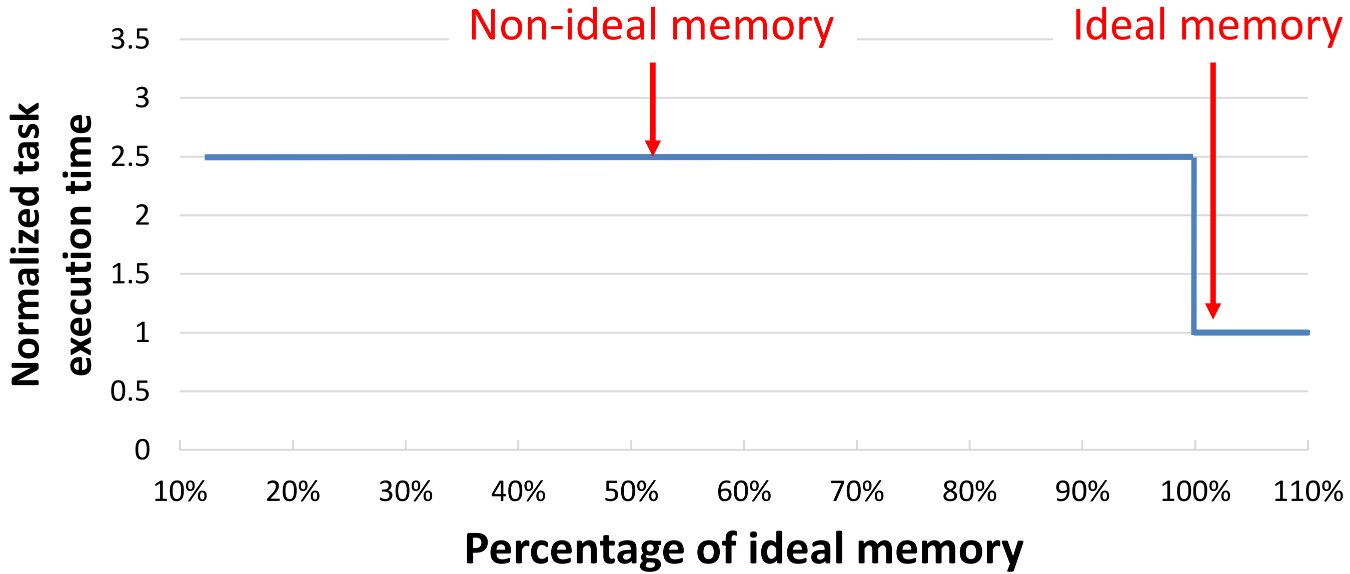
Modeling Memory Elasticity

- Requires 2 profiling runs → infers all other points



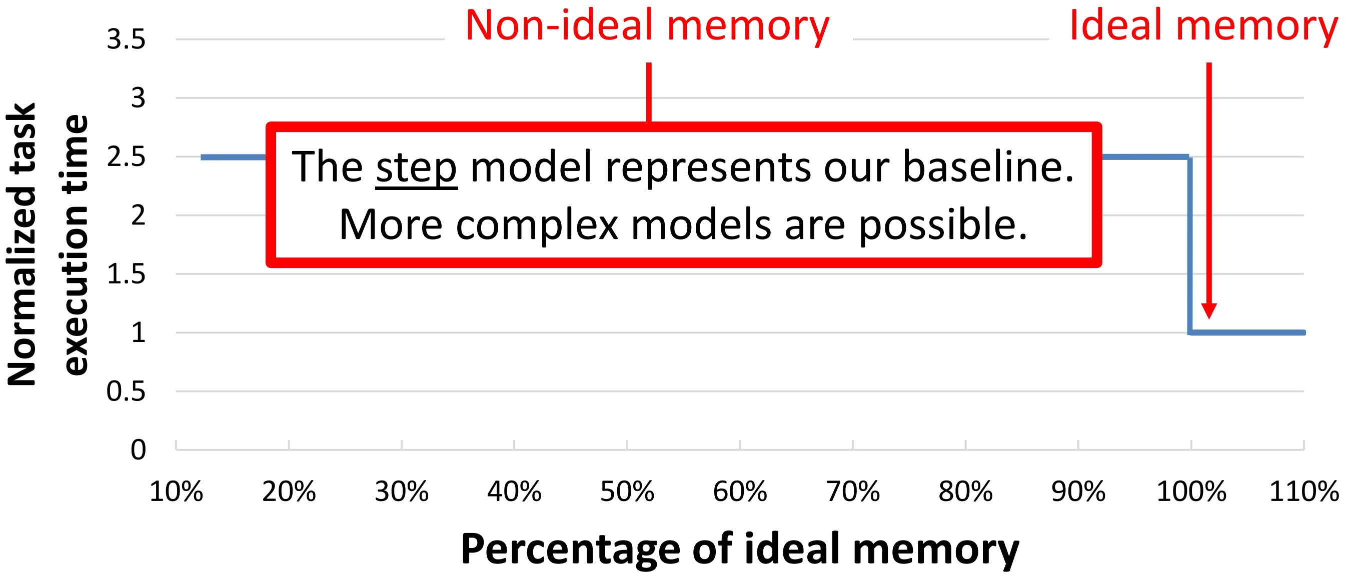
Modeling Memory Elasticity

- Requires 2 profiling runs → infers all other points

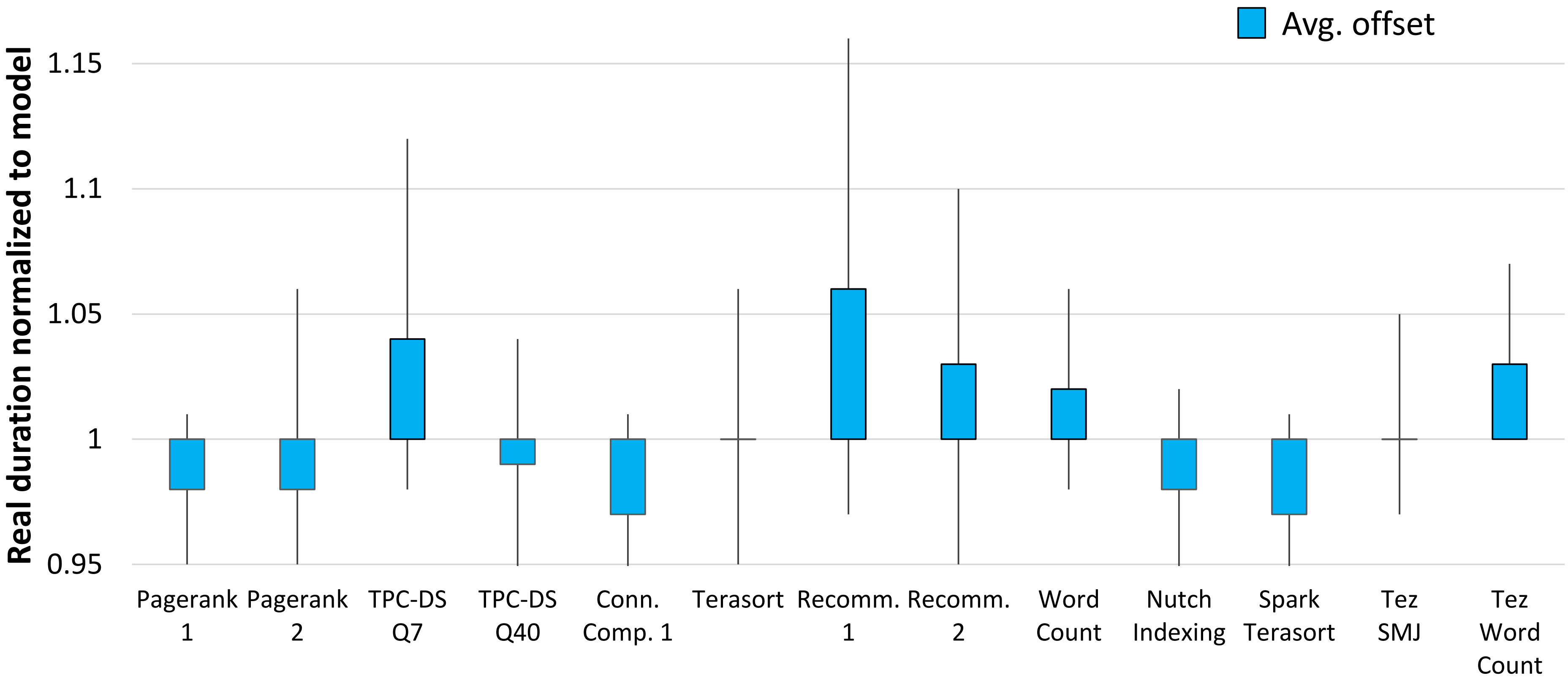


Modeling Memory Elasticity

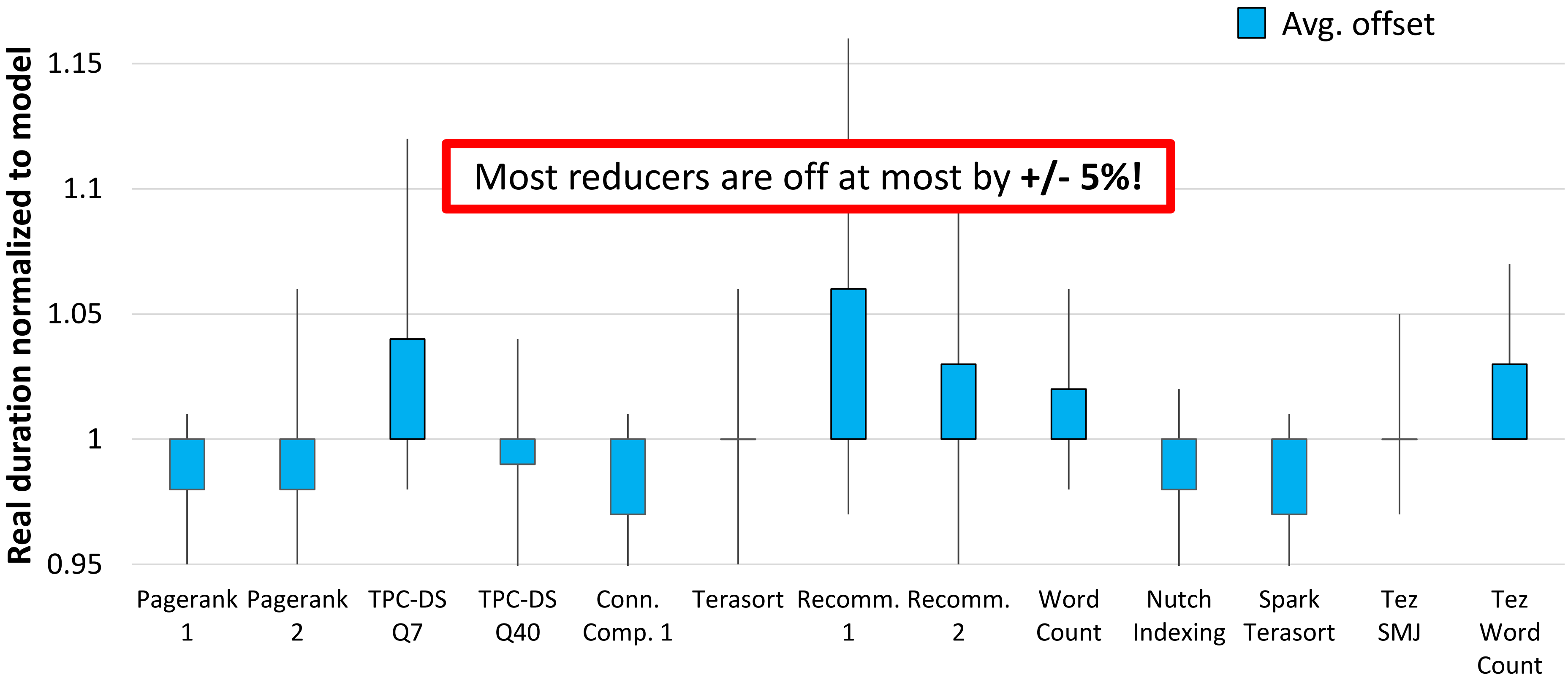
- Requires 2 profiling runs → infers all other points



Accuracy of our reducer model



Accuracy of our reducer model



Summary: Modeling memory elasticity

✓ **Step model is adequate (more complex models available)**

✓ **Only 2 profiling points → full model**

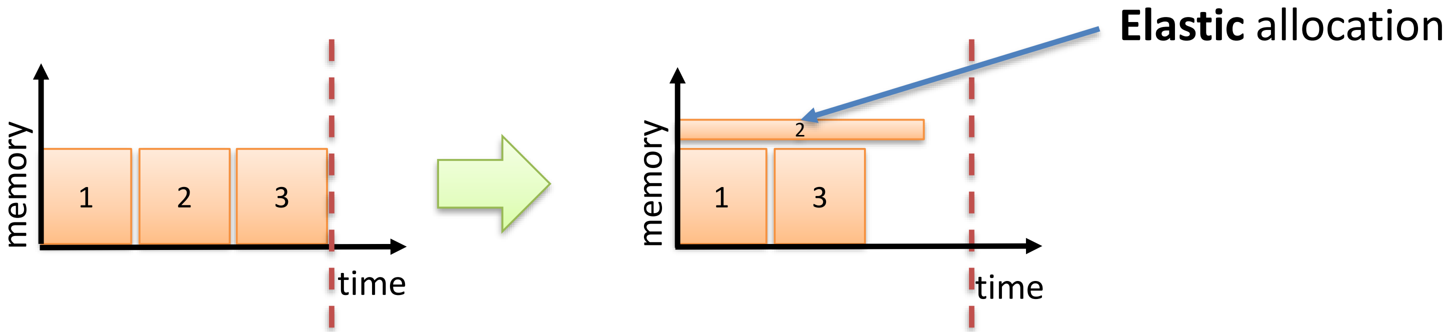
✓ **Models are very robust (+/- 5% error for most reducers)**

LEVERAGING MEMORY ELASTICITY IN CLUSTER SCHEDULING

How can a scheduler reason about Memory Elasticity?

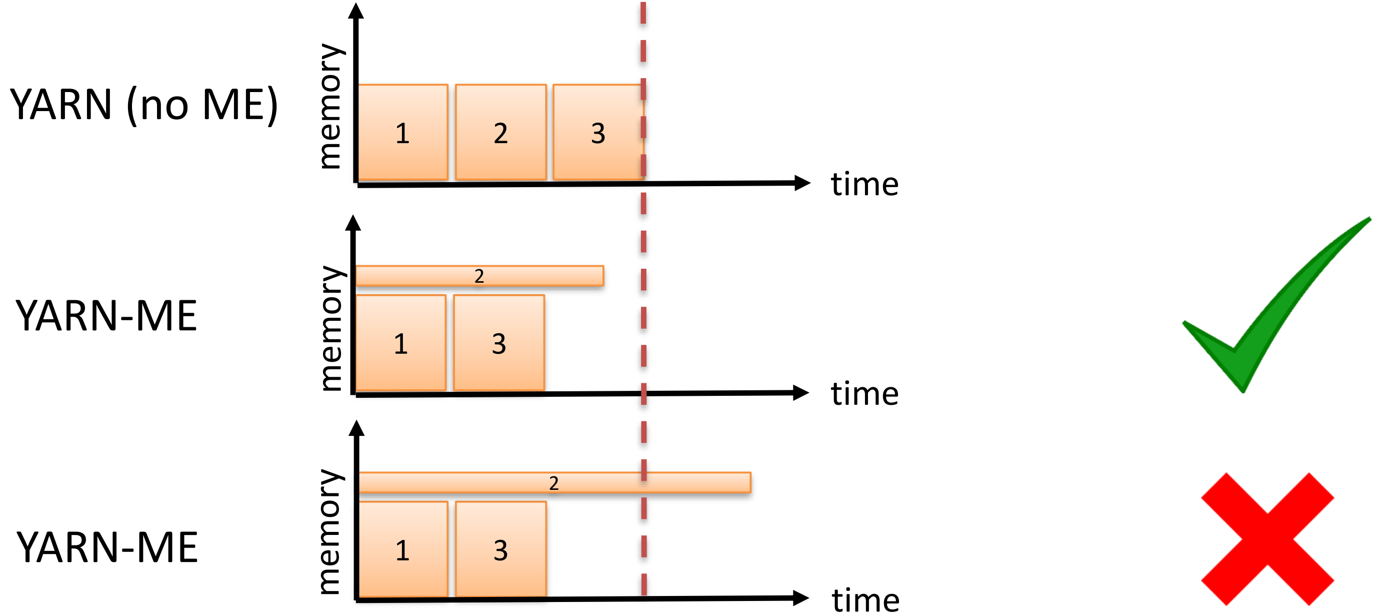
Trade-off between

↓ task **queueing time** ↑ task **execution time**

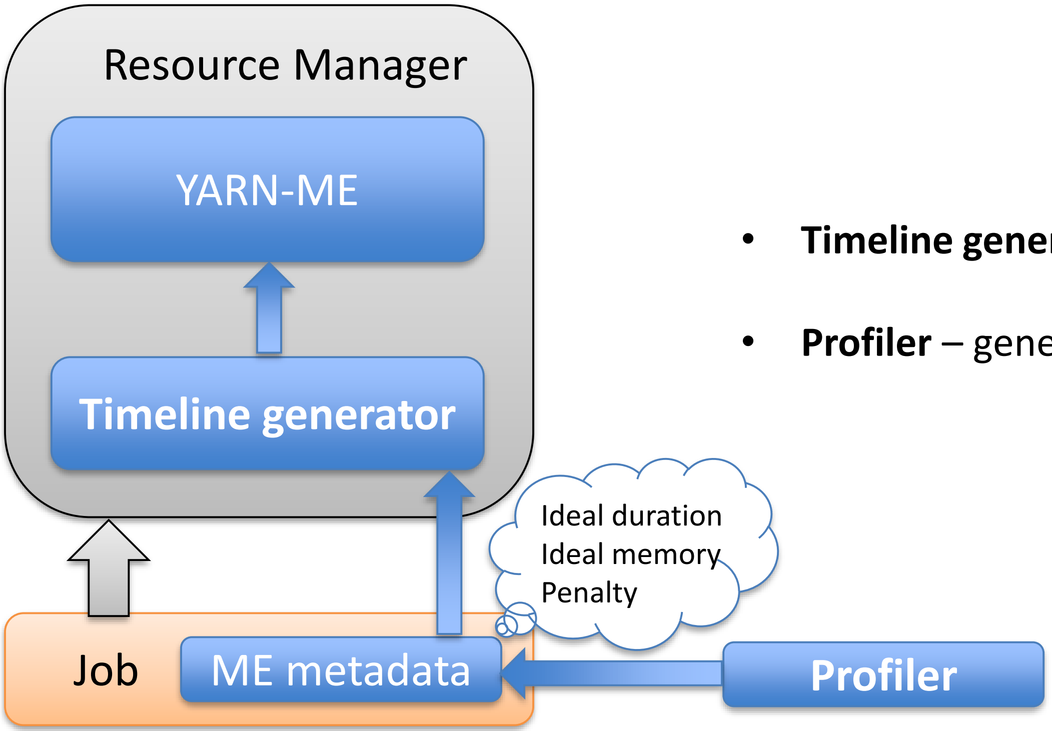


YARN-ME: Decision process

- Make an elastic allocation *iff* it does not exceed the expected job completion time



YARN-ME: Design and components

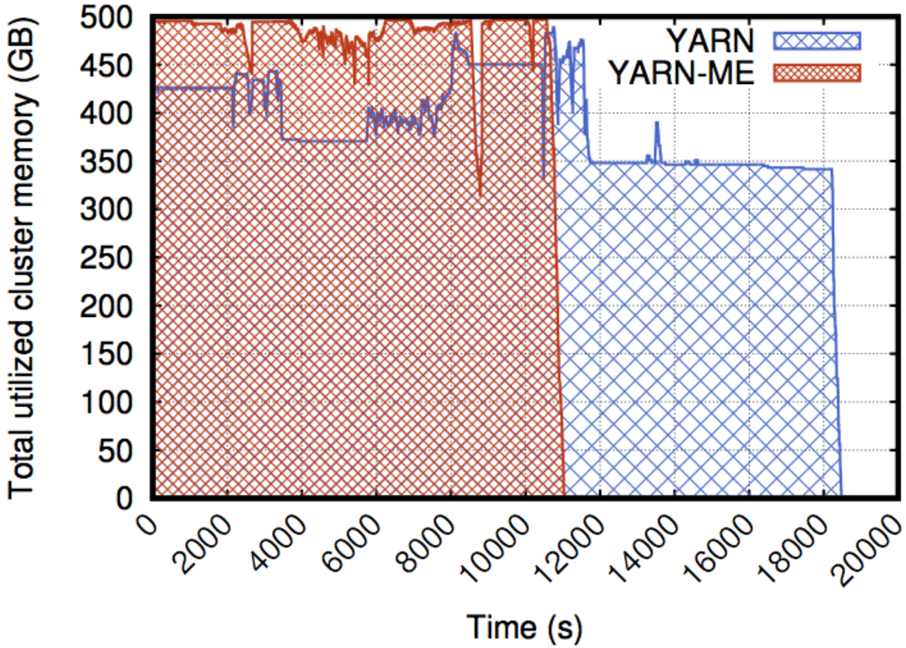


Components

- **Timeline generator** – computes expected JCTs
- **Profiler** – generates the model metadata

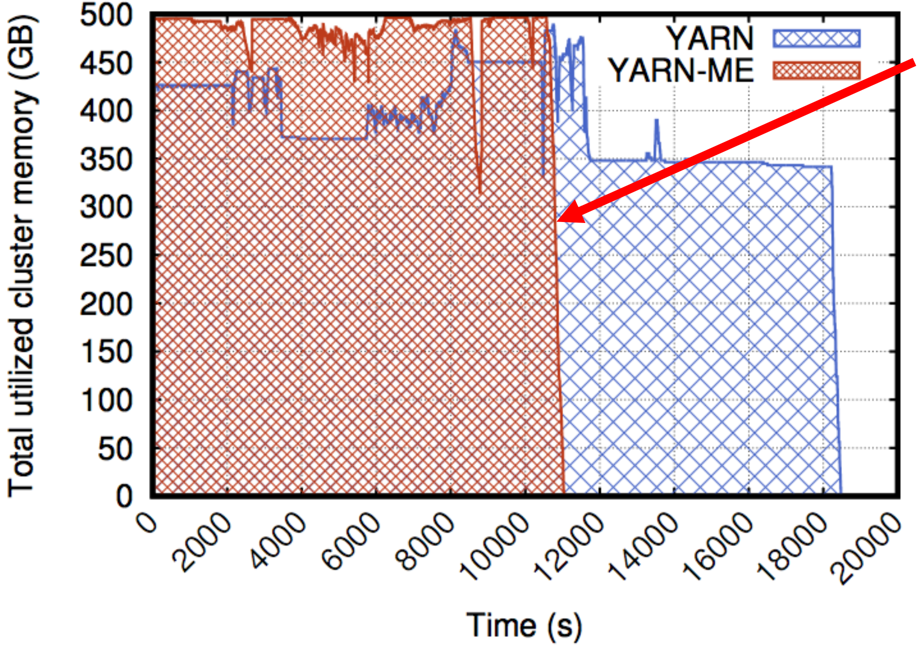
Memory utilization analysis for YARN-ME

- 50 node cluster
- <u>Homogeneous trace</u>: 5x Pagerank jobs



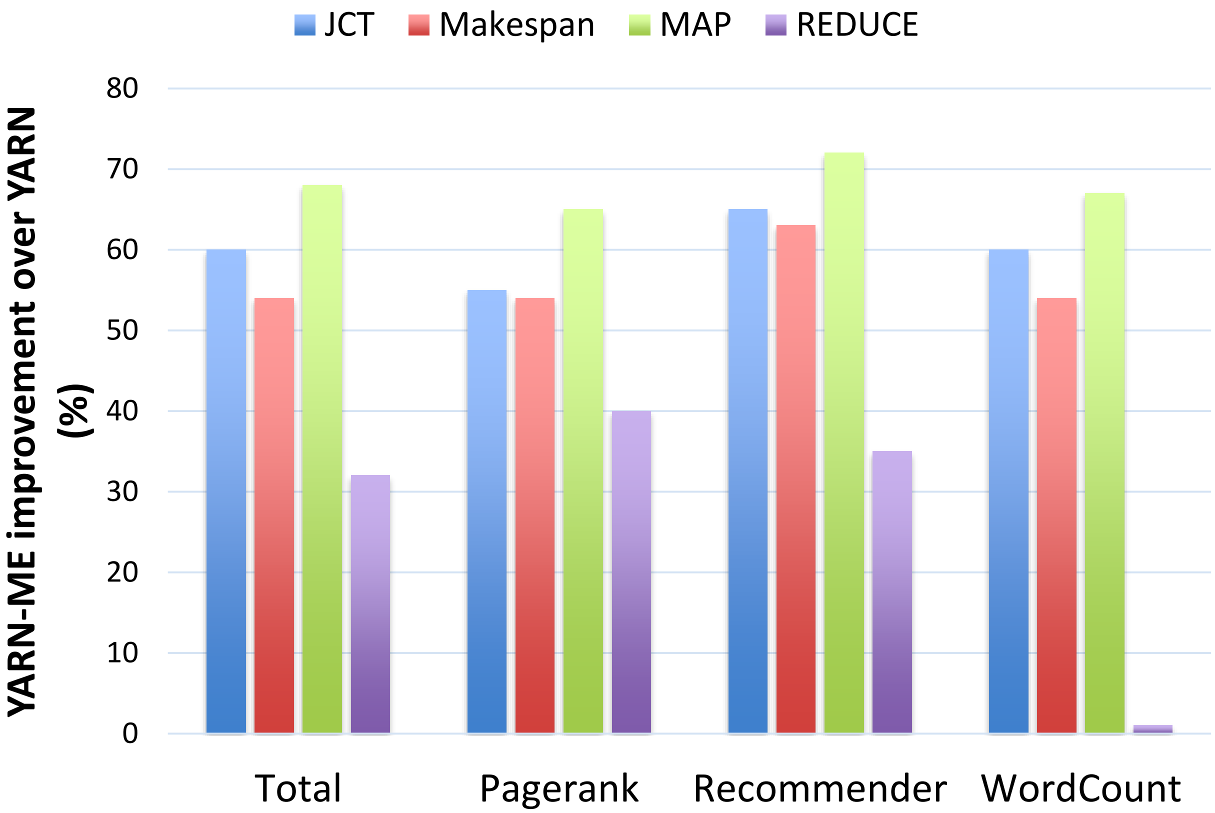
Memory utilization analysis for YARN-ME

- 50 node cluster
- Homogeneous trace: 5x Pagerank jobs



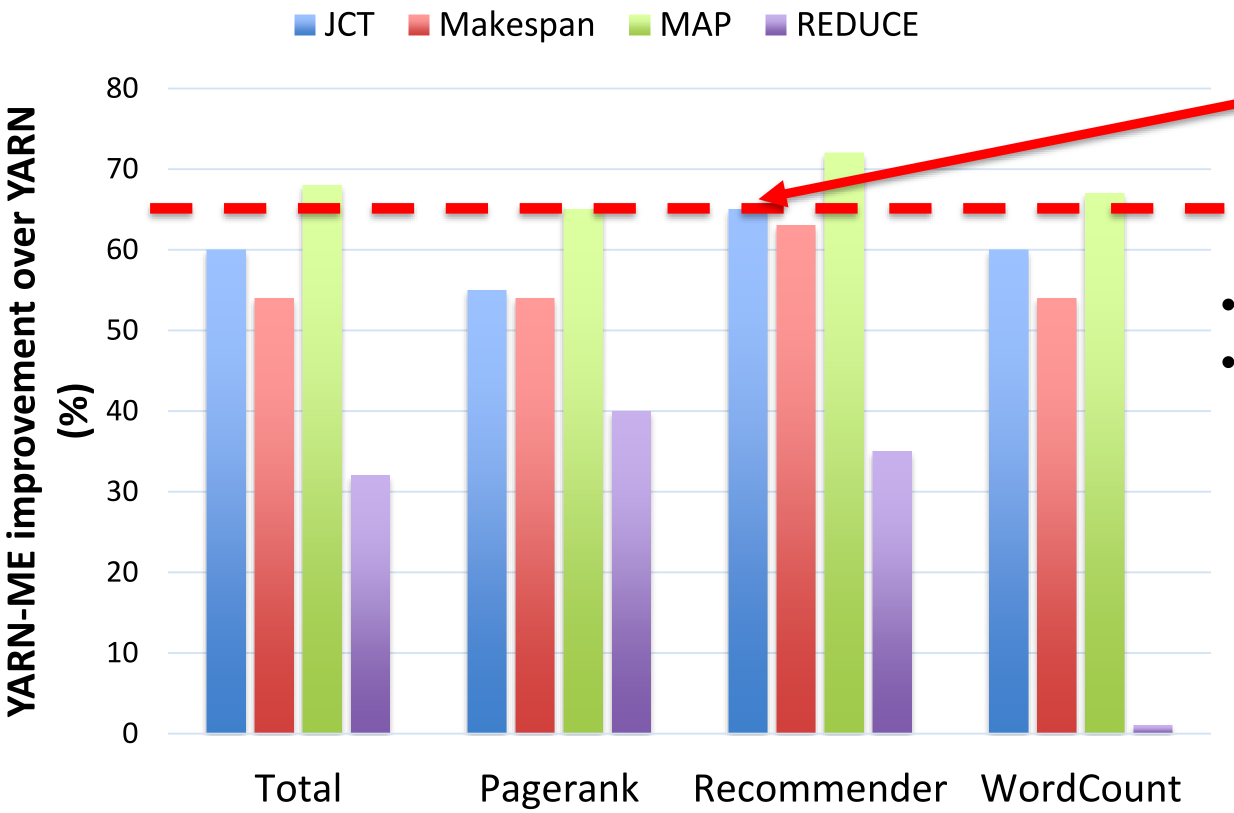
Memory utilization increased to **95%**

What gains can YARN-ME achieve for heterogeneous workloads?



- 50 node cluster
- Mixed trace – 14 jobs
    - 3x PageRank
    - 3x Recommender
    - 8x Wordcount

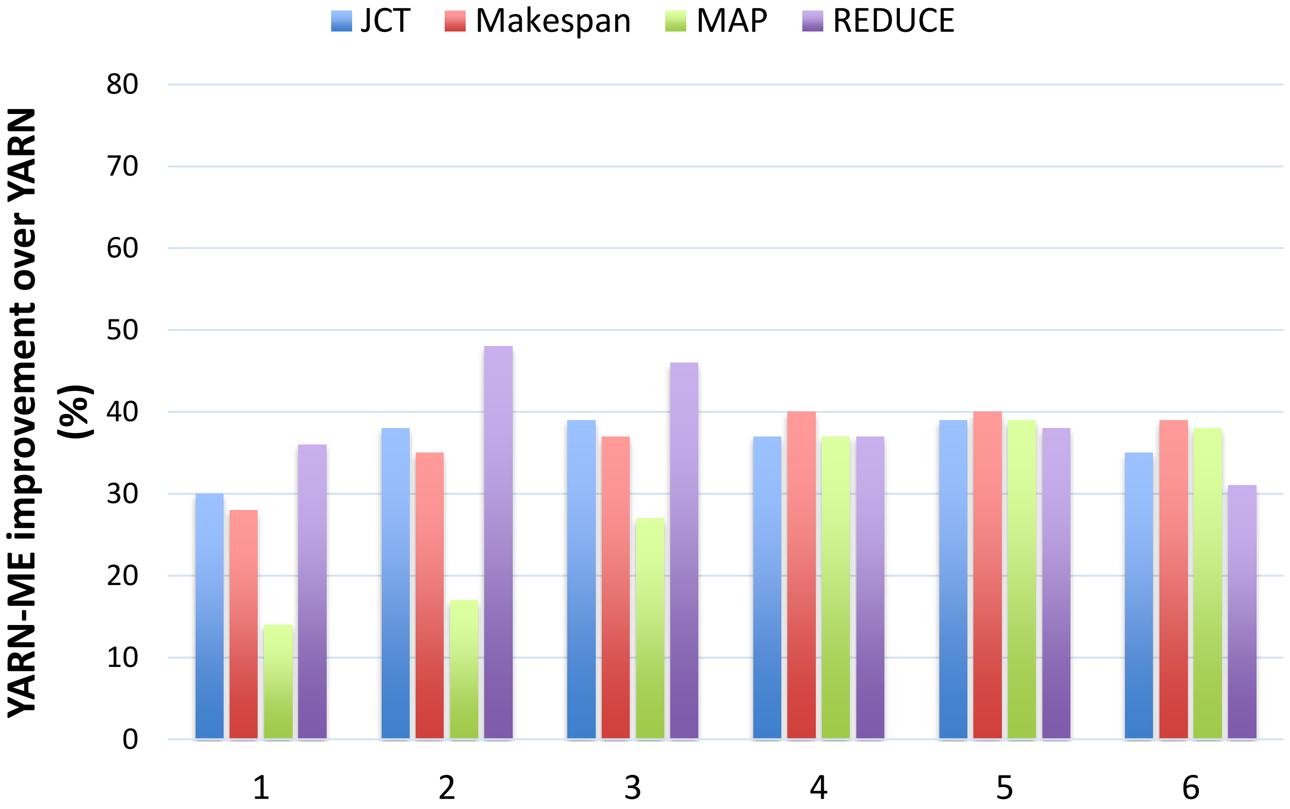
What gains can YARN-ME achieve for <u>heterogeneous</u> workloads?



Up to **65% improvement** for **JCT** and **makespan**

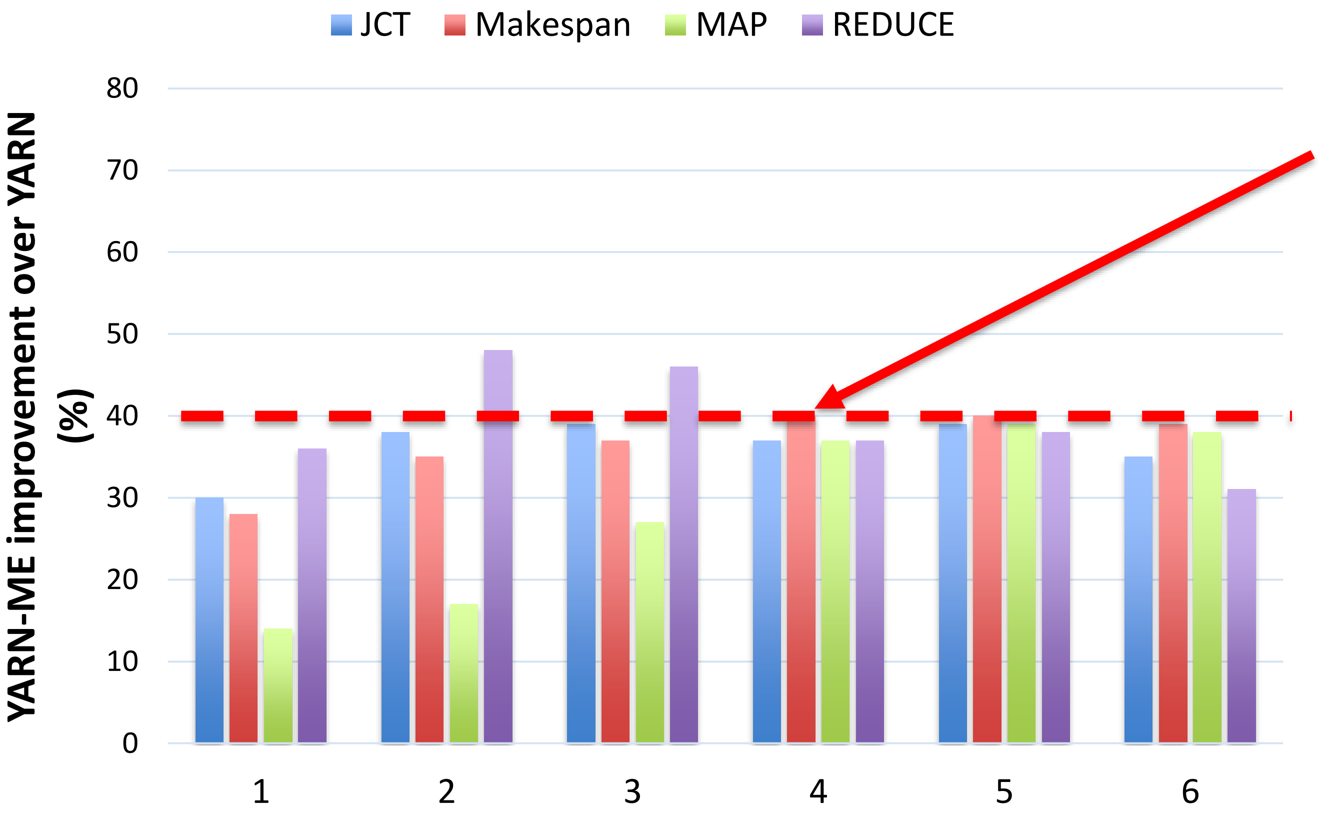
- 50 node cluster
- <u>Mixed trace – 14 jobs</u>
  - 3x PageRank
  - 3x Recommender
  - 8x Wordcount

What gains can YARN-ME achieve for homogeneous workloads?



- 50 node cluster
- Pagerank
  - *concurrent runs*

What gains can YARN-ME achieve for homogeneous workloads?



Up to **40% improvement** for **JCT** and **makespan**

- 50 node cluster
- Pagerank
  - *concurrent runs*

Trace-driven simulation of YARN-ME

- We built DSS (the Discrete Scheduler Simulator)
  - and it is open-source!

<u>Trace parameter sweep</u>

- > 8,000 traces
- up to 3,000 nodes
- results comparable to real workloads

<u>Robustness analysis</u>

- > 20,000 traces
- YARN-ME is robust to model mis-estimations

Related work

- Efficient packing → better resource utilization
  - Tetris [**SIGCOMM ‘14**], GRAPHENE [**OSDI ‘16**]
- Collocate batch-jobs with latency-critical services
  - Heracles [**ISCA ‘15**]
- Resource over-committing
  - Apollo [**OSDI ‘14**], Borg [**EuroSys ‘15**]
- Suspend tasks under memory pressure
  - ITask [**SOSP ‘15**]

Conclusion: Don't cry over spilled records!

Fork us on GitHub

- ✓ Memory Elasticity → highly **predictable**, low penalty
- ✓ Memory Elasticity in scheduling → trade task **queueing-time** for **running-time**
- ✓ YARN-ME → up to 60% improvement in average JCT
- ✓ DSS code available: https://github.com/epfl-labos/dss