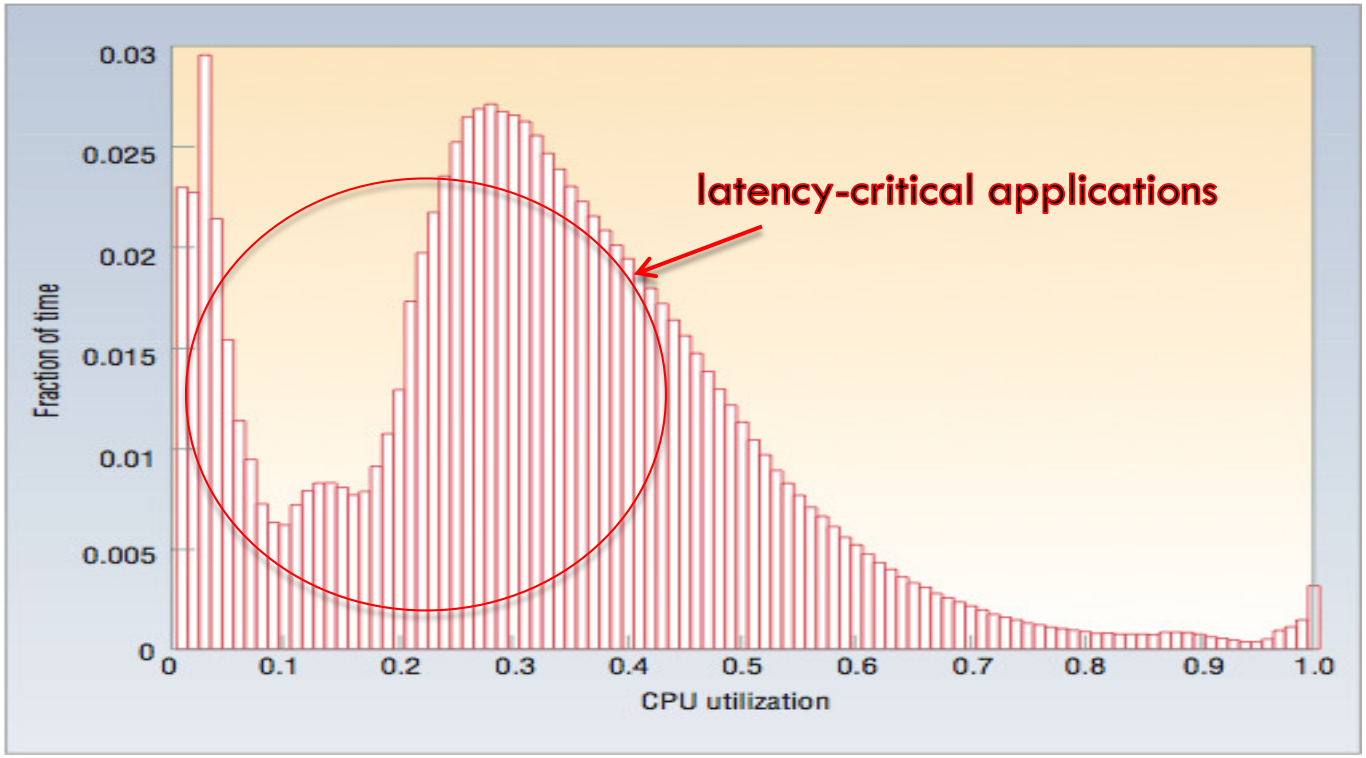
UBIK: EFFICIENT CACHE SHARING WITH STRICT QOS FOR LATENCY CRITICAL WORKLOADS

HARSHAD KASTURE, DANIEL SANCHEZ

ASPLOS 2014

Massachusetts Institute of Technology

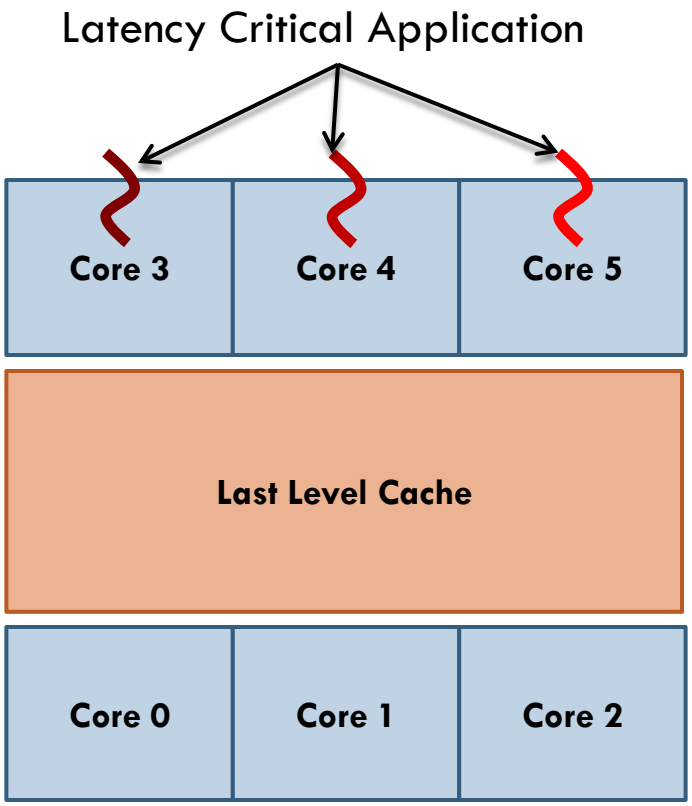
Motivation



L. Barroso and U. Hölzle, The Case for Energy-Proportional Computing

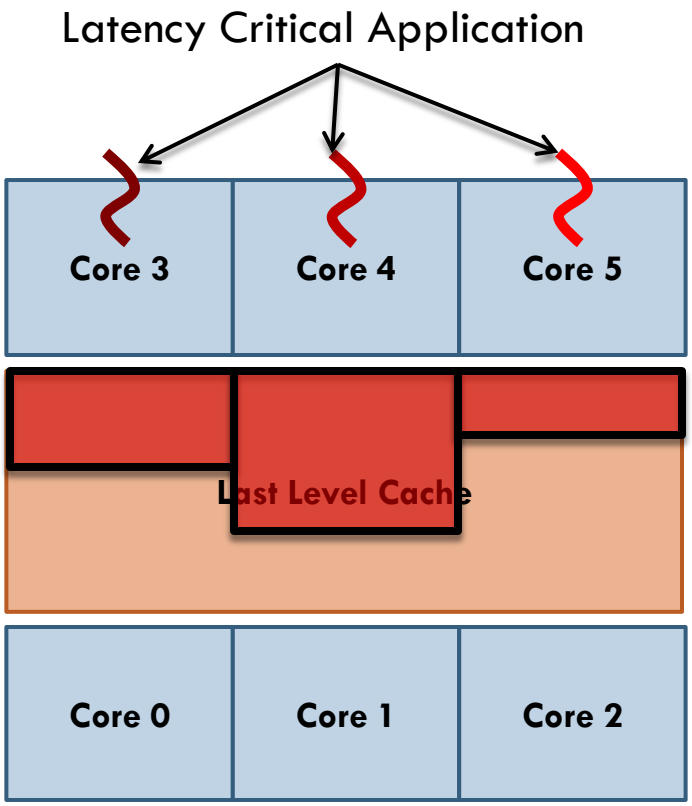
- Low server utilization in datacenters is a major source of inefficiency

Common Industry Practice



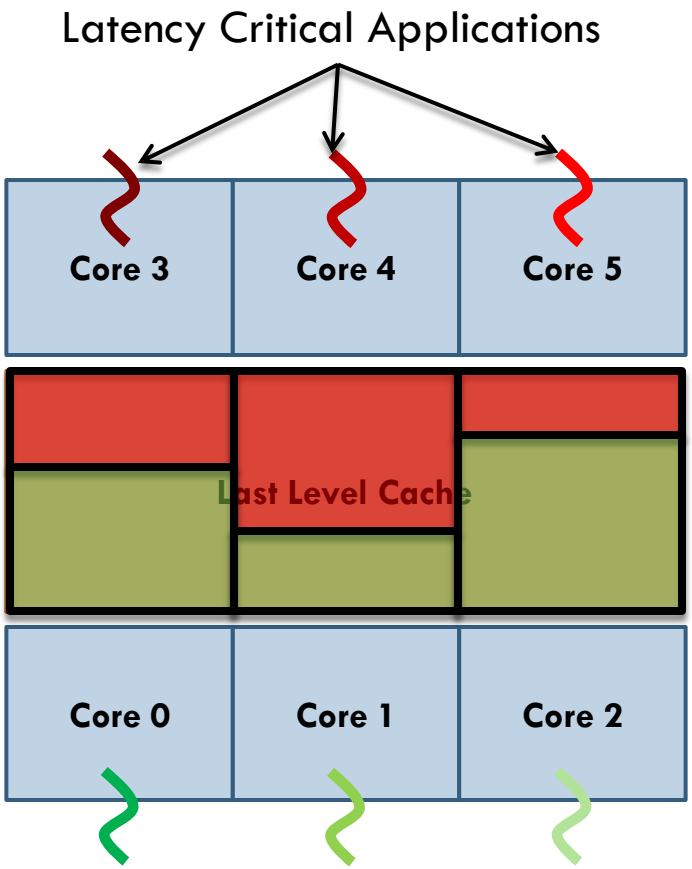
- Dedicated machines for latency-critical applications guarantees QoS

Common Industry Practice



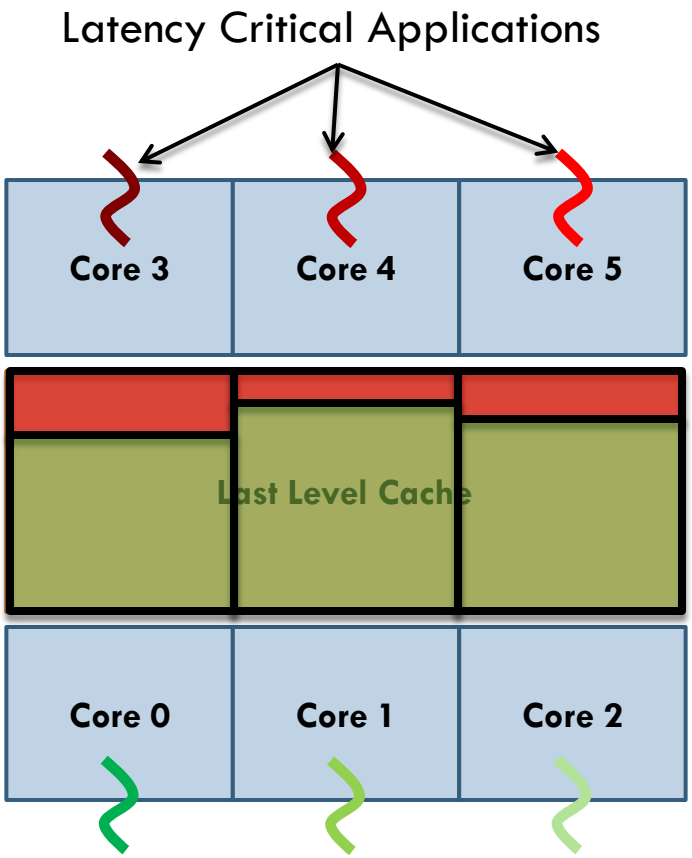
- Dedicated machines for latency-critical applications guarantees QoS
  - Under utilization of machine resources

Colocation to Improve Utilization



- Can utilize spare resources by colocating batch apps

Sharing Causes Interference!

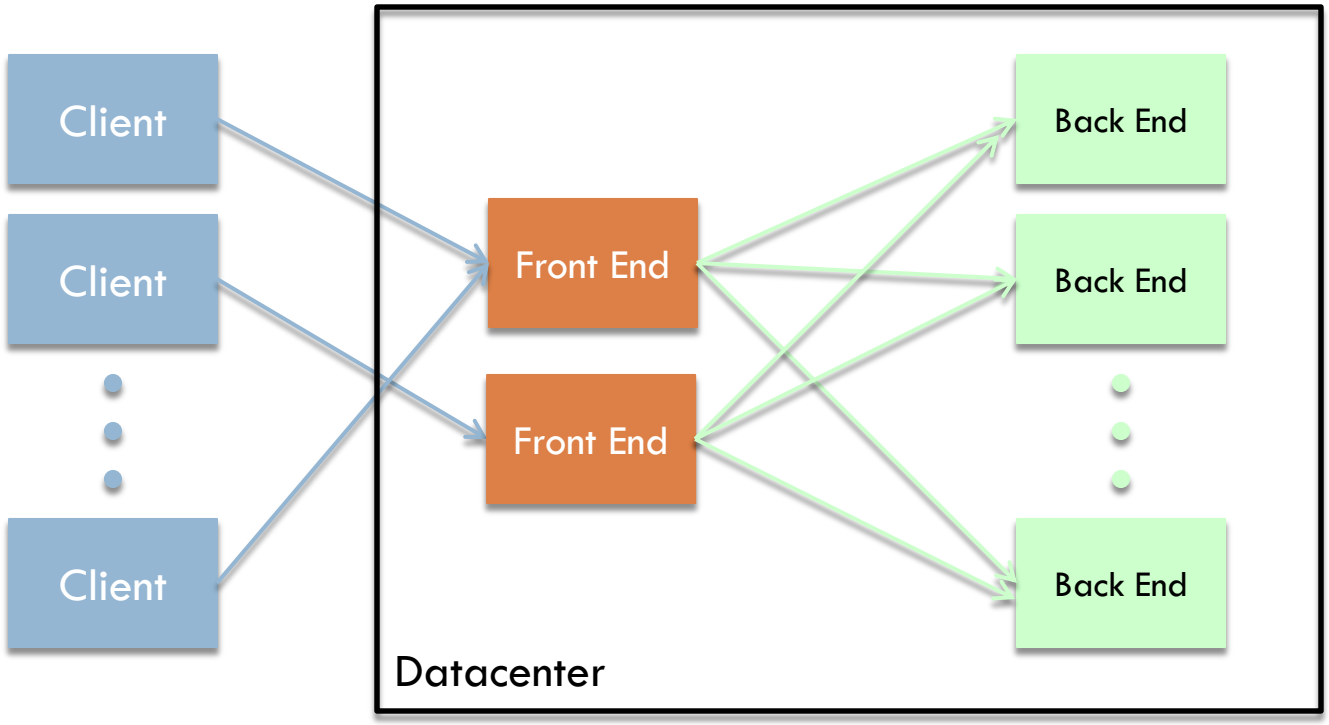


- Can utilize spare resources by colocating batch apps
  - Contention in shared resources degrades QoS

Outline

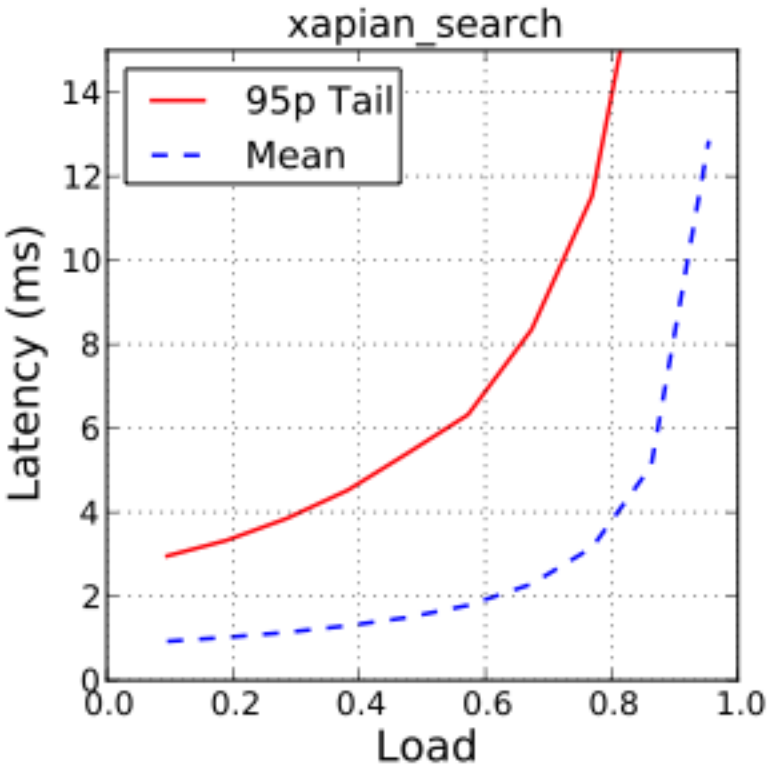
- Introduction
- Analysis of latency-critical apps
- Inertia-oblivious cache management schemes
- Ubik: Inertia-aware cache management
- Evaluation

Understanding Latency-Critical Applications



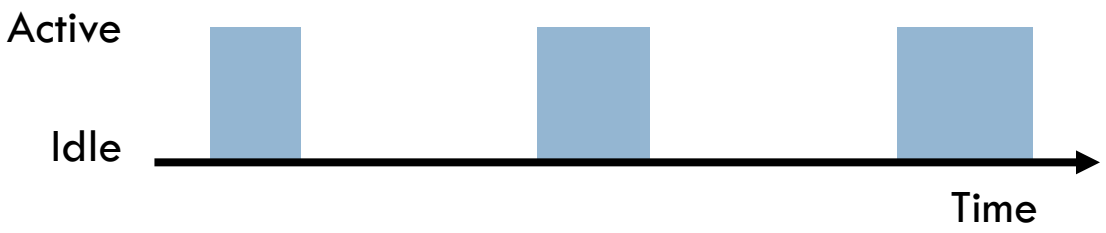
- Large number of backend servers participate in handling every user request
  - Total service time determined by tail latency behavior of backend

Understanding Latency-Critical Applications



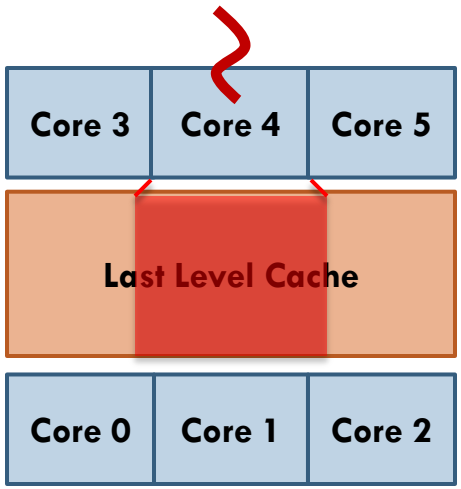
- Service latency highly sensitive to changes in load

Understanding Latency-Critical Applications

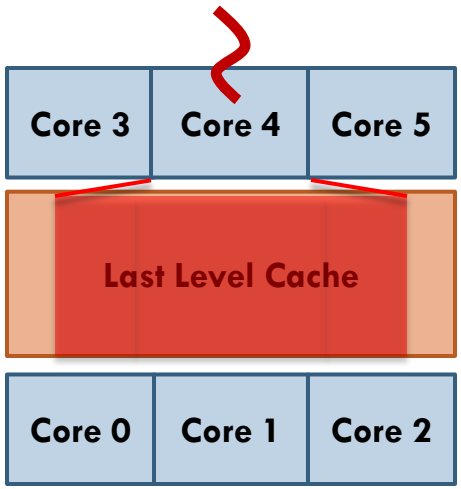


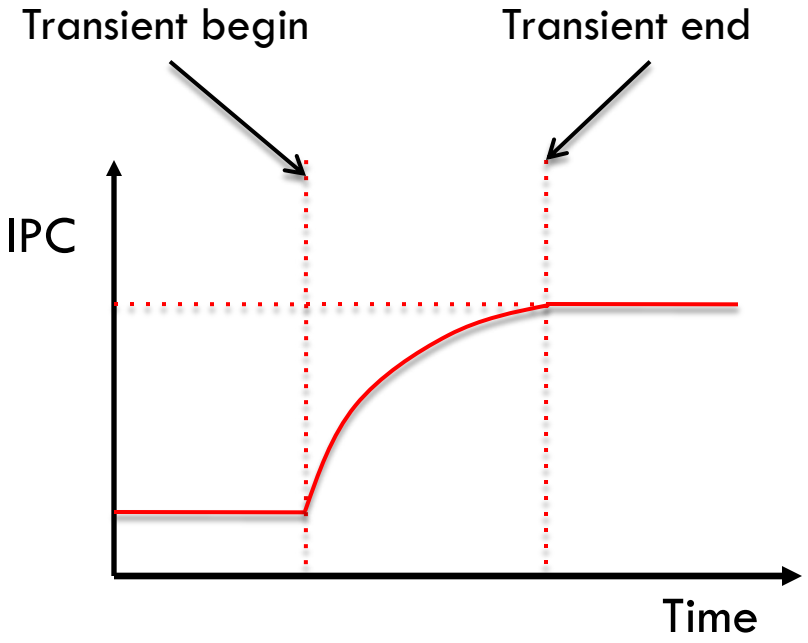
- Short bursts of activity interspersed with idle periods
  - Need guaranteed high performance during active periods

Inertia and Transient Behavior



Inertia and Transient Behavior



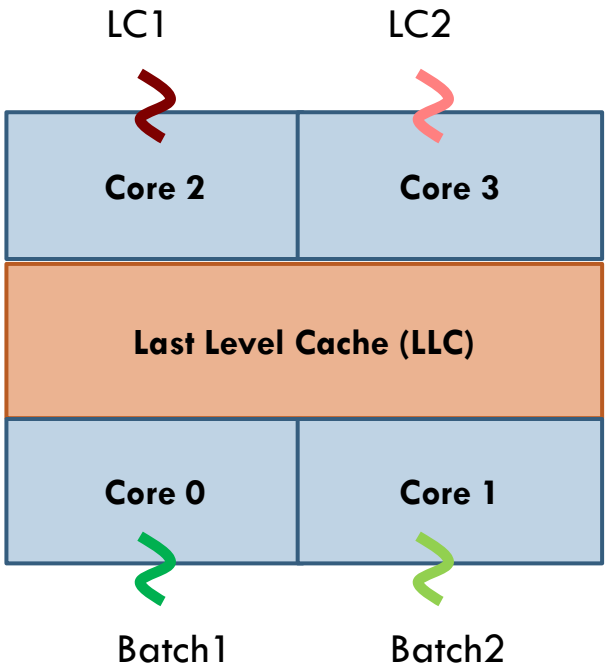


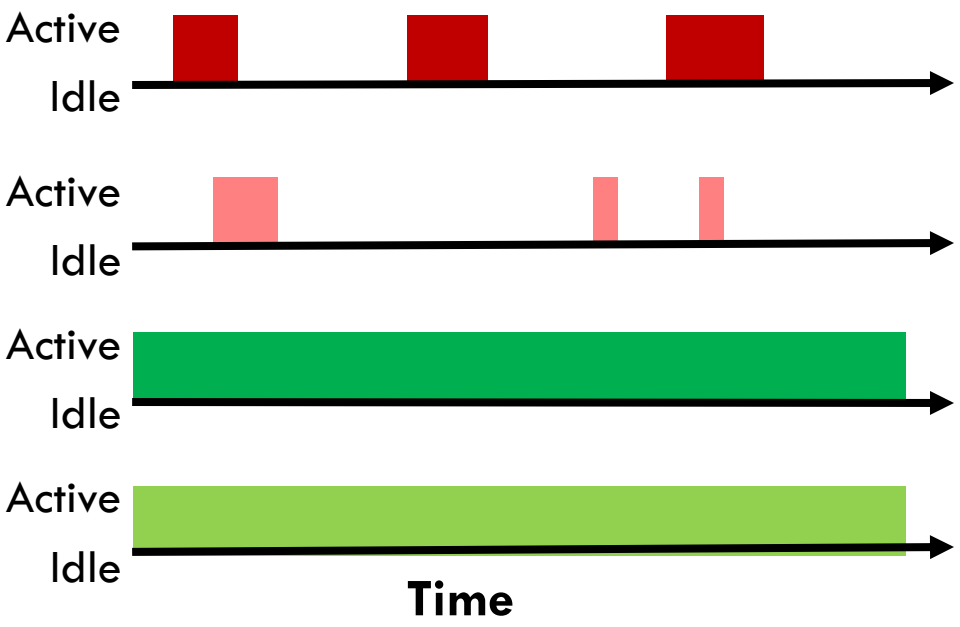
- Transient lengths can dominate tail latency!
  - Any dynamic reconfiguration scheme has to be inertia-aware
- Many hardware resources exhibit inertia
  - branch predictors, prefetchers, memory bandwidth…
  - LLCs are one of the biggest sources of inertia

Outline

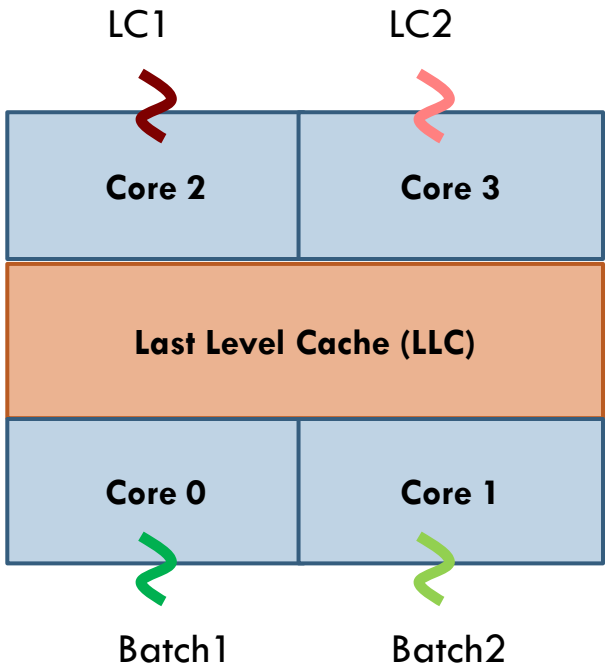
- Introduction
- Analysis of latency-critical apps
- Inertia-oblivious cache management schemes
- Ubik: Inertia-aware cache management
- Evaluation

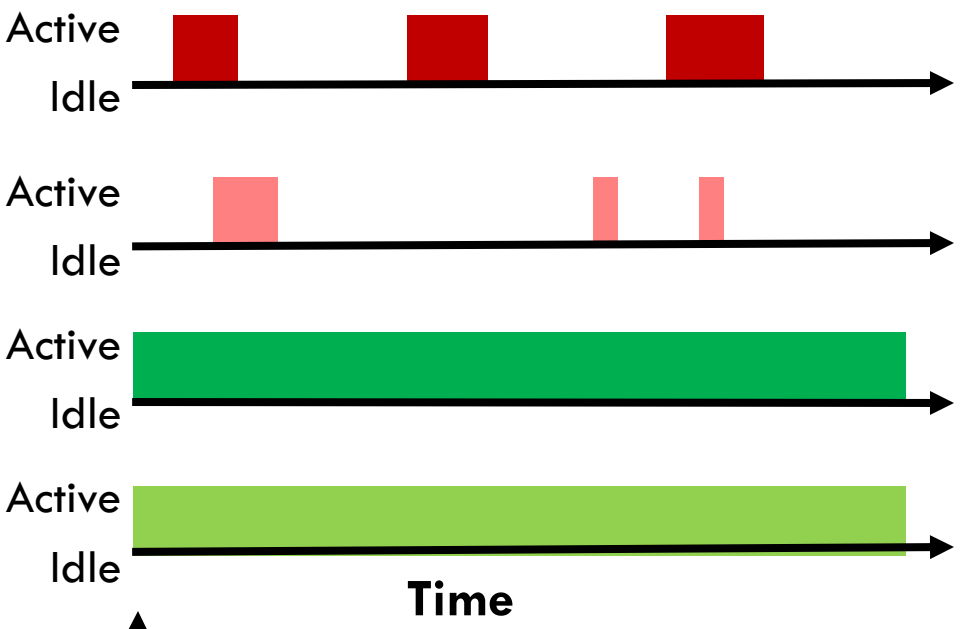
Inertia-Oblivious Cache Management

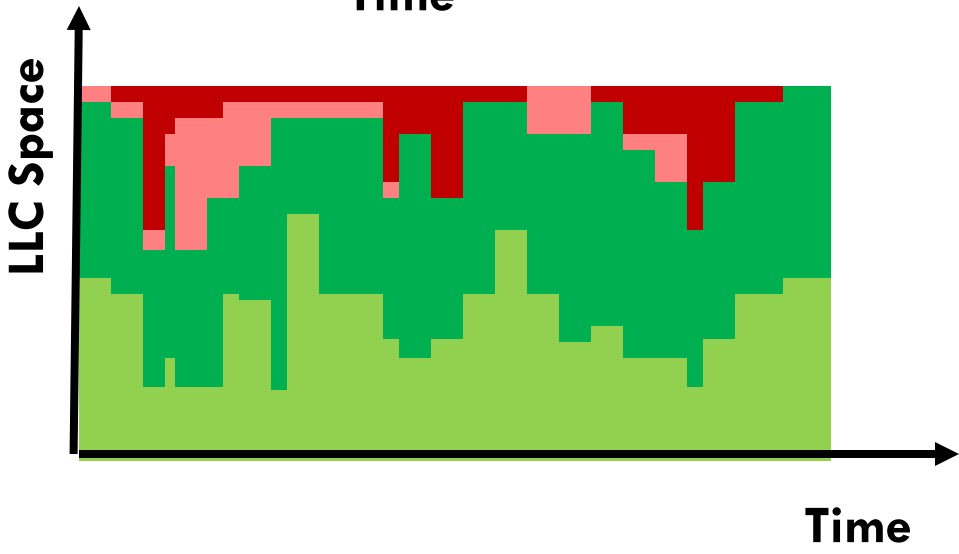




Unmanaged LLC (LRU Replacement)

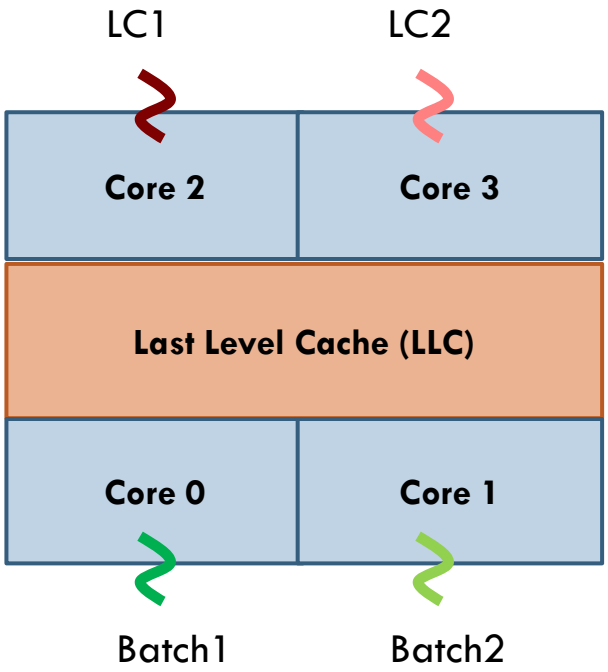


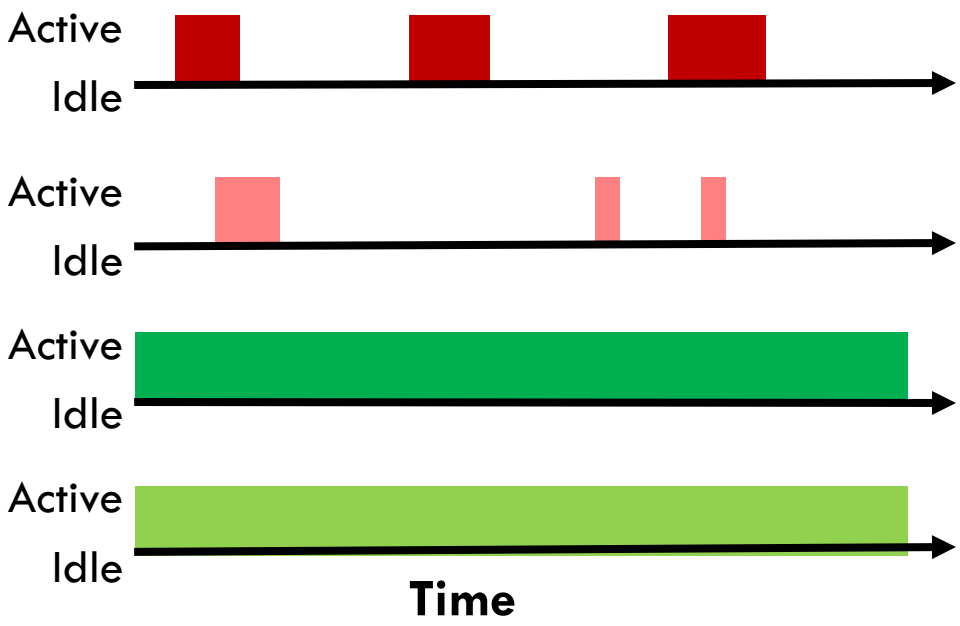


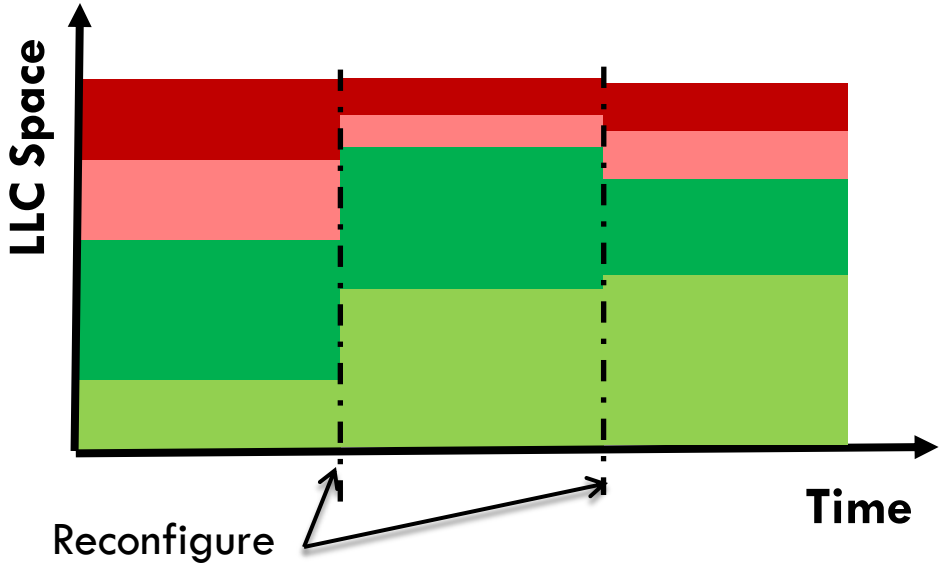


✖ Unconstrained interference results in poor tail-latency behavior

Utility Based Cache Partitioning (UCP)



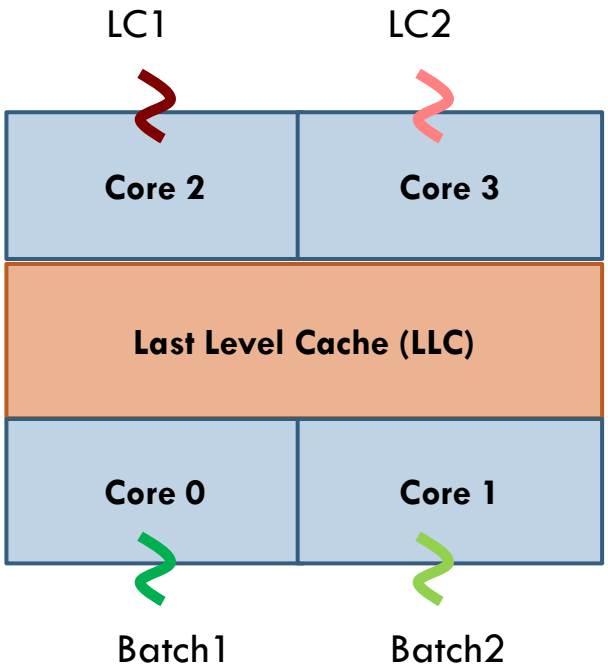




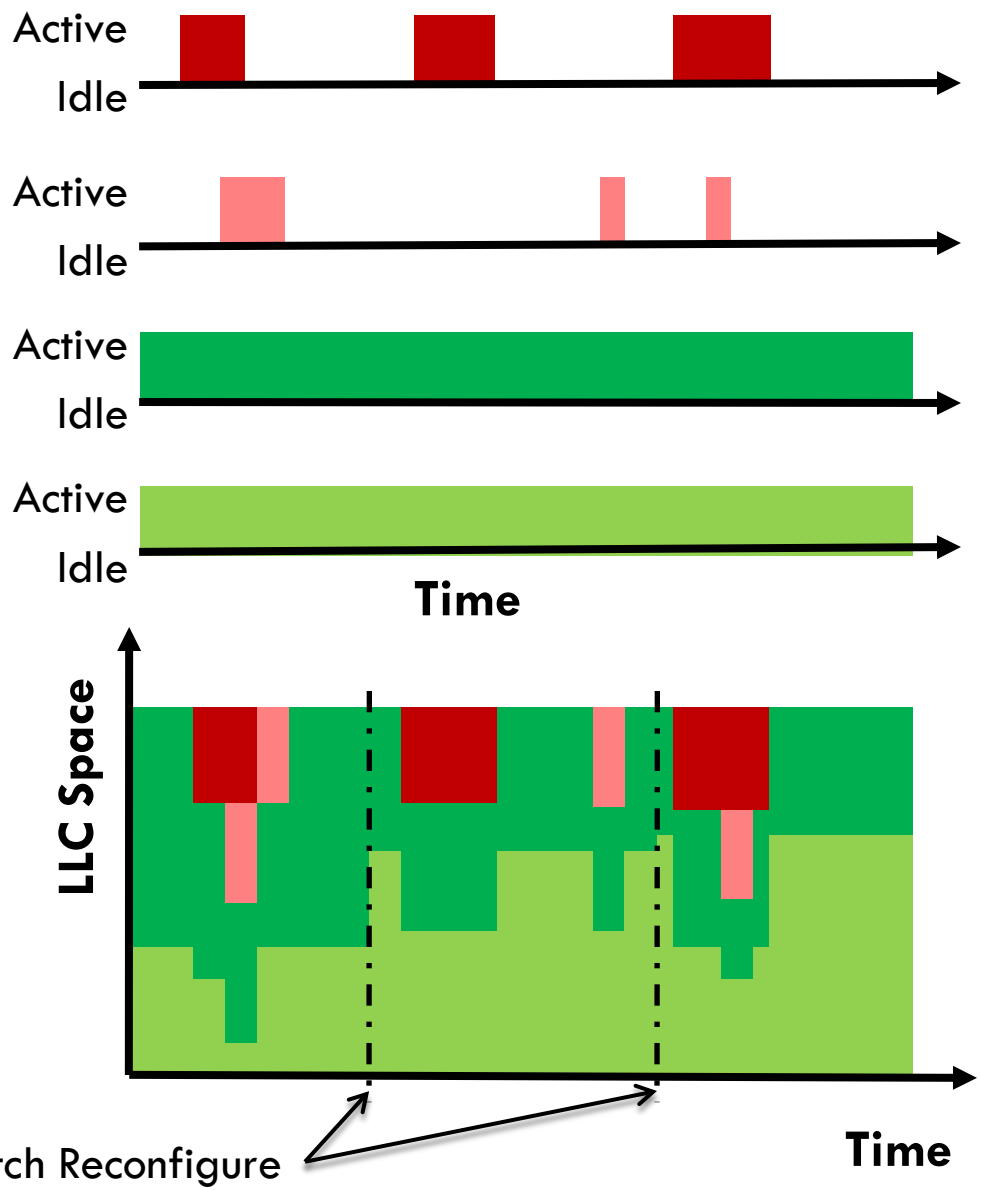
✔ High batch throughput

✖ Poor tail latency (low allocation)

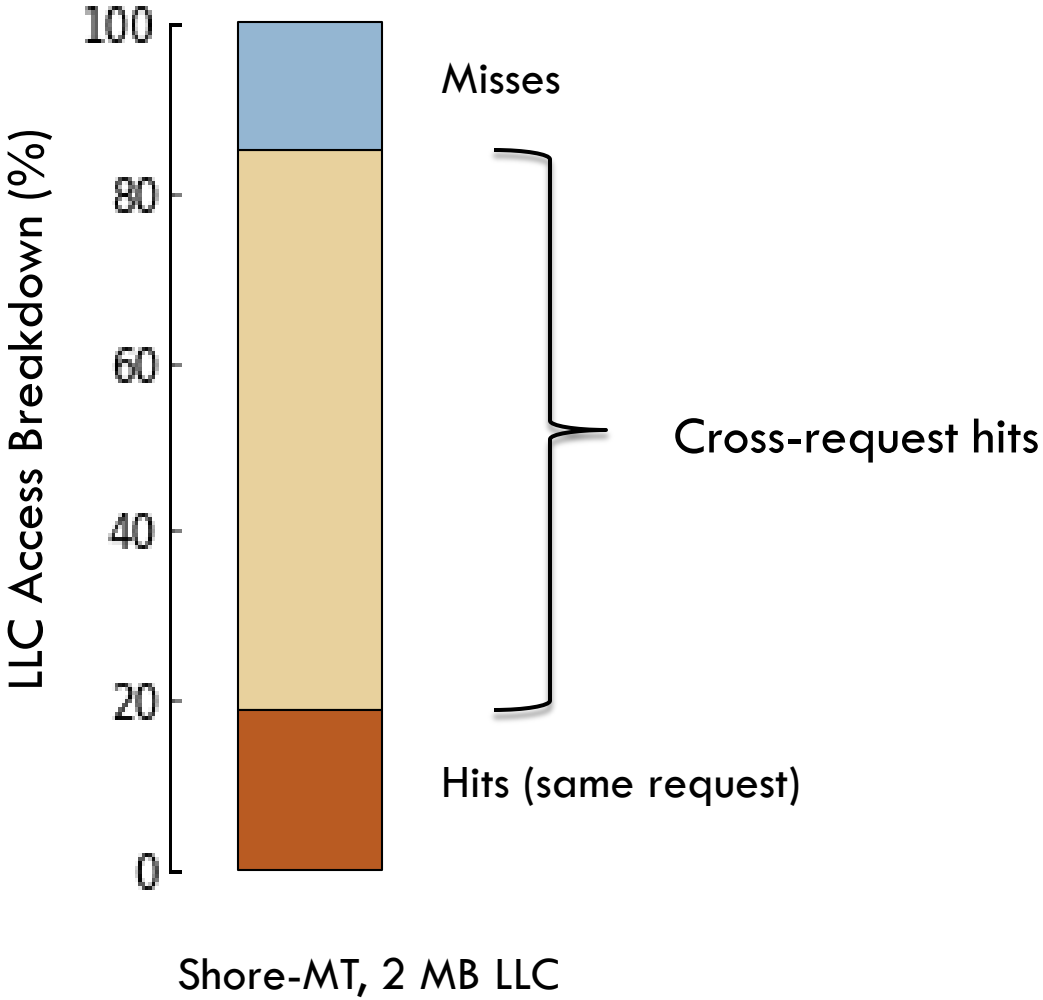
OnOff: Efficient but Unsafe



✔ High batch throughput

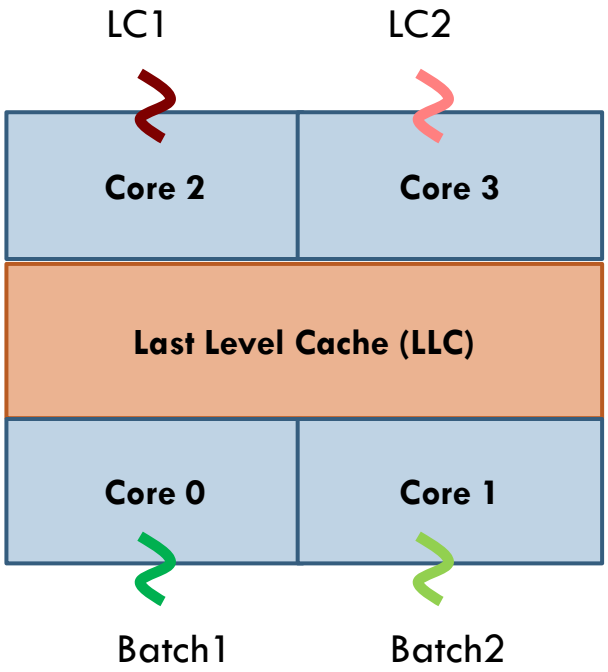


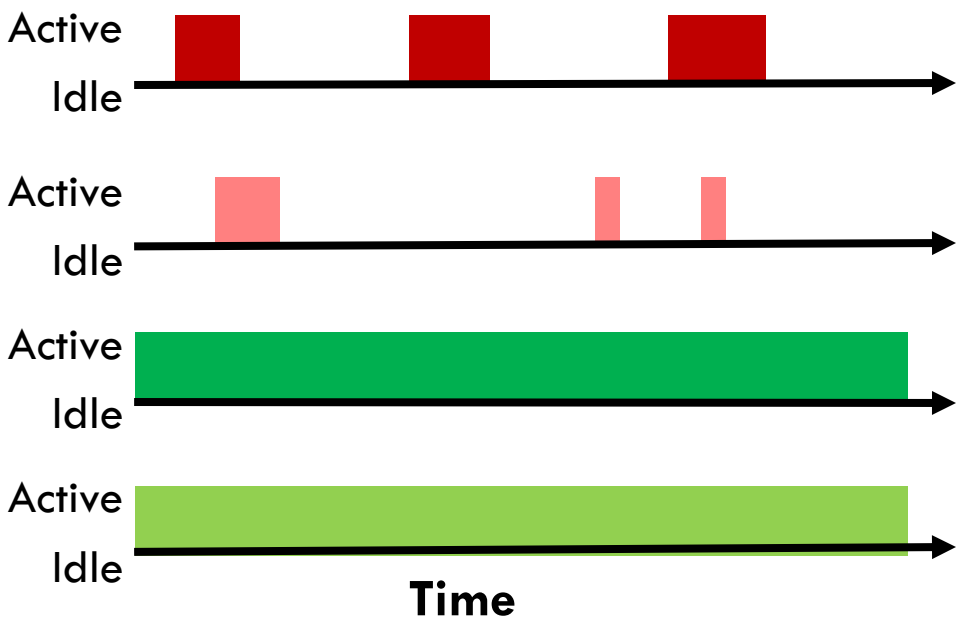
Cross-Request LLC Inertia

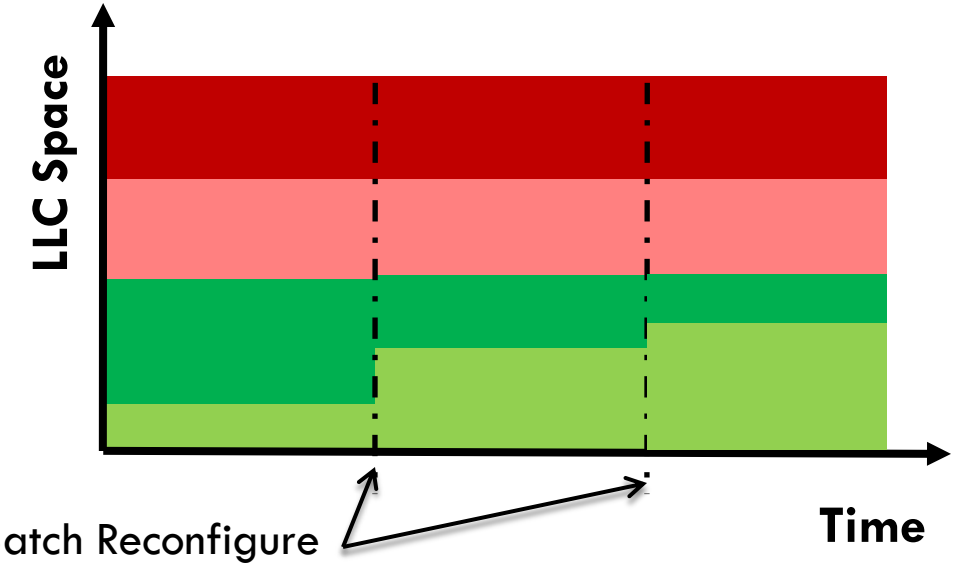


- Other applications qualitatively similar (see paper for details)

StaticLC: Safe but Inefficient







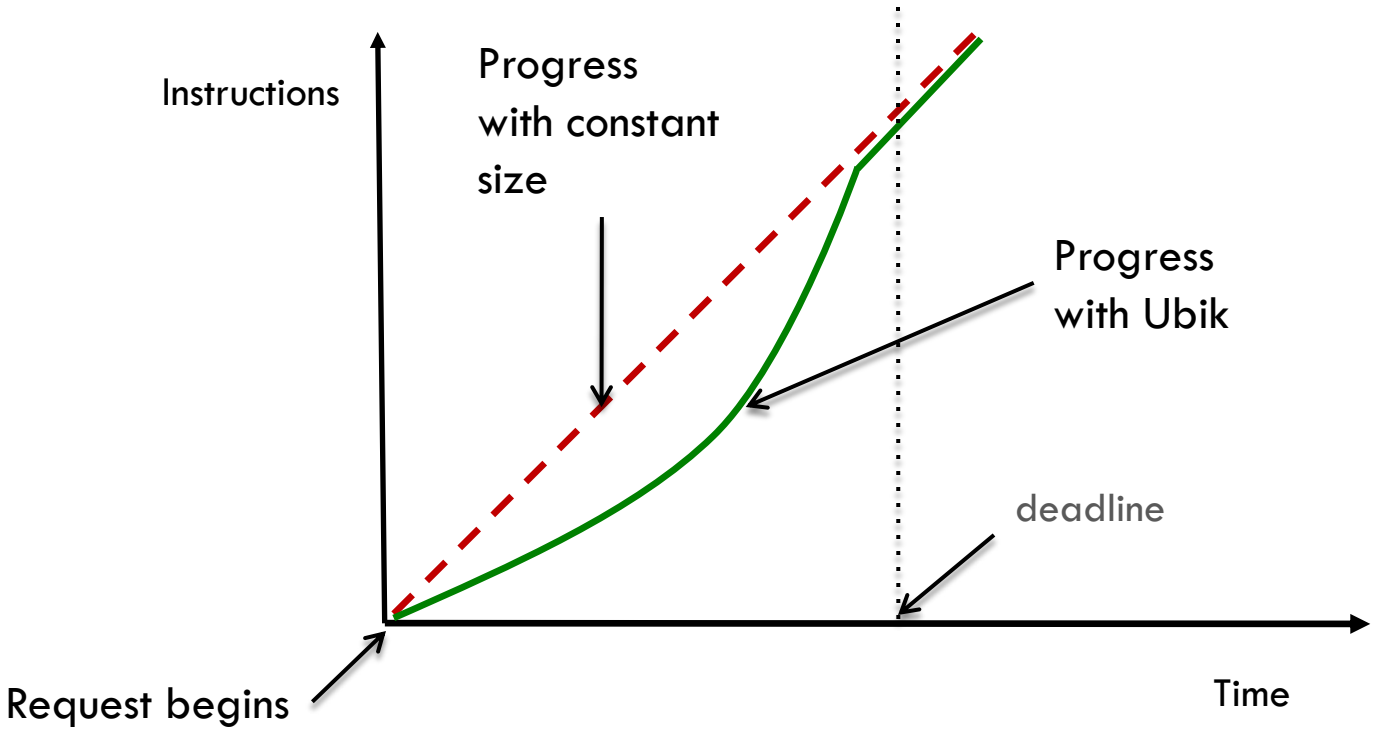
✔ Low tail latency (preserve LLC state)

✖ Low batch throughput (poor space utilization)

Outline

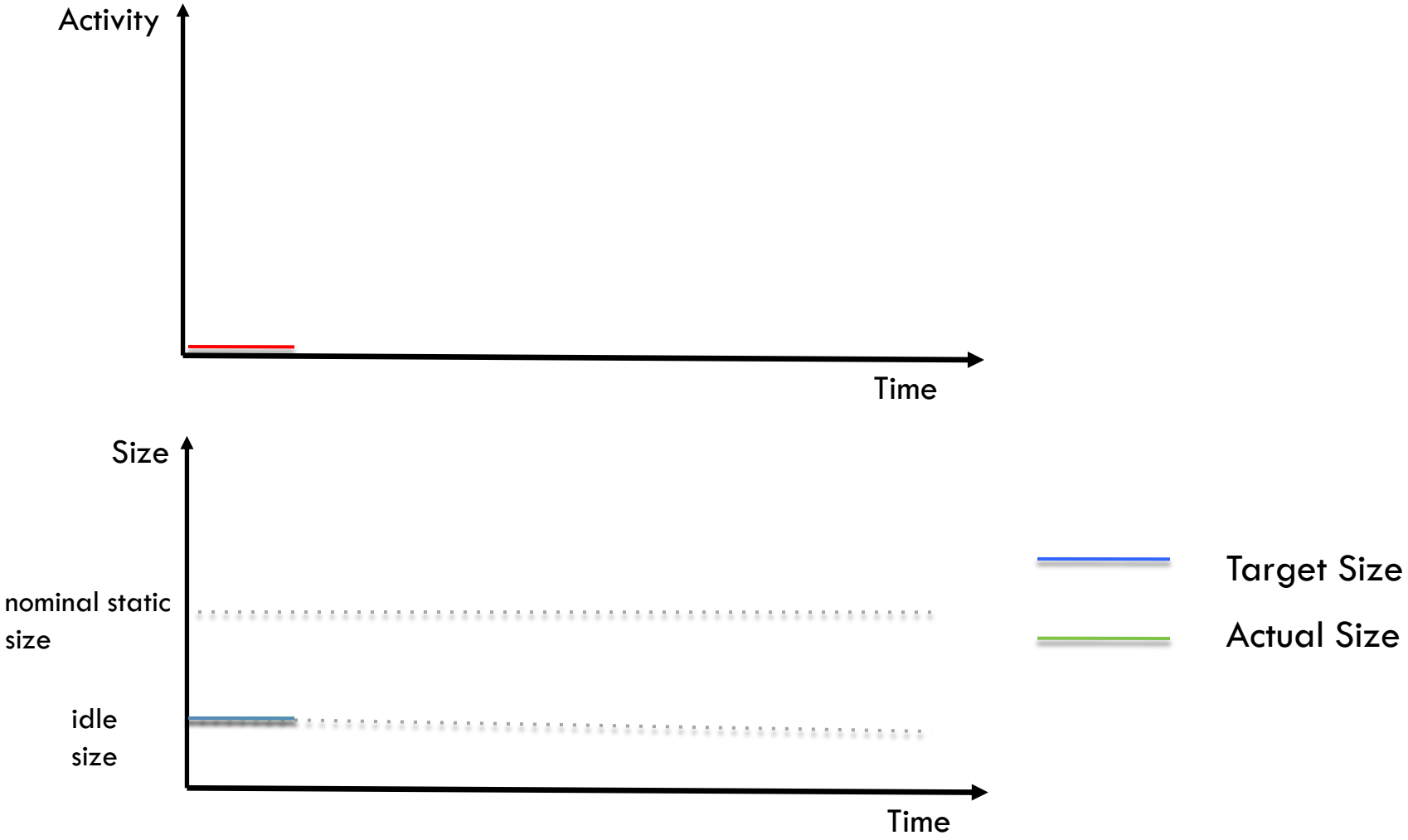
- Introduction
- Analysis of latency-critical apps
- Inertia-oblivious cache management schemes
- Ubik: Inertia-aware cache management
- Evaluation

Ubik: Performance Guarantee

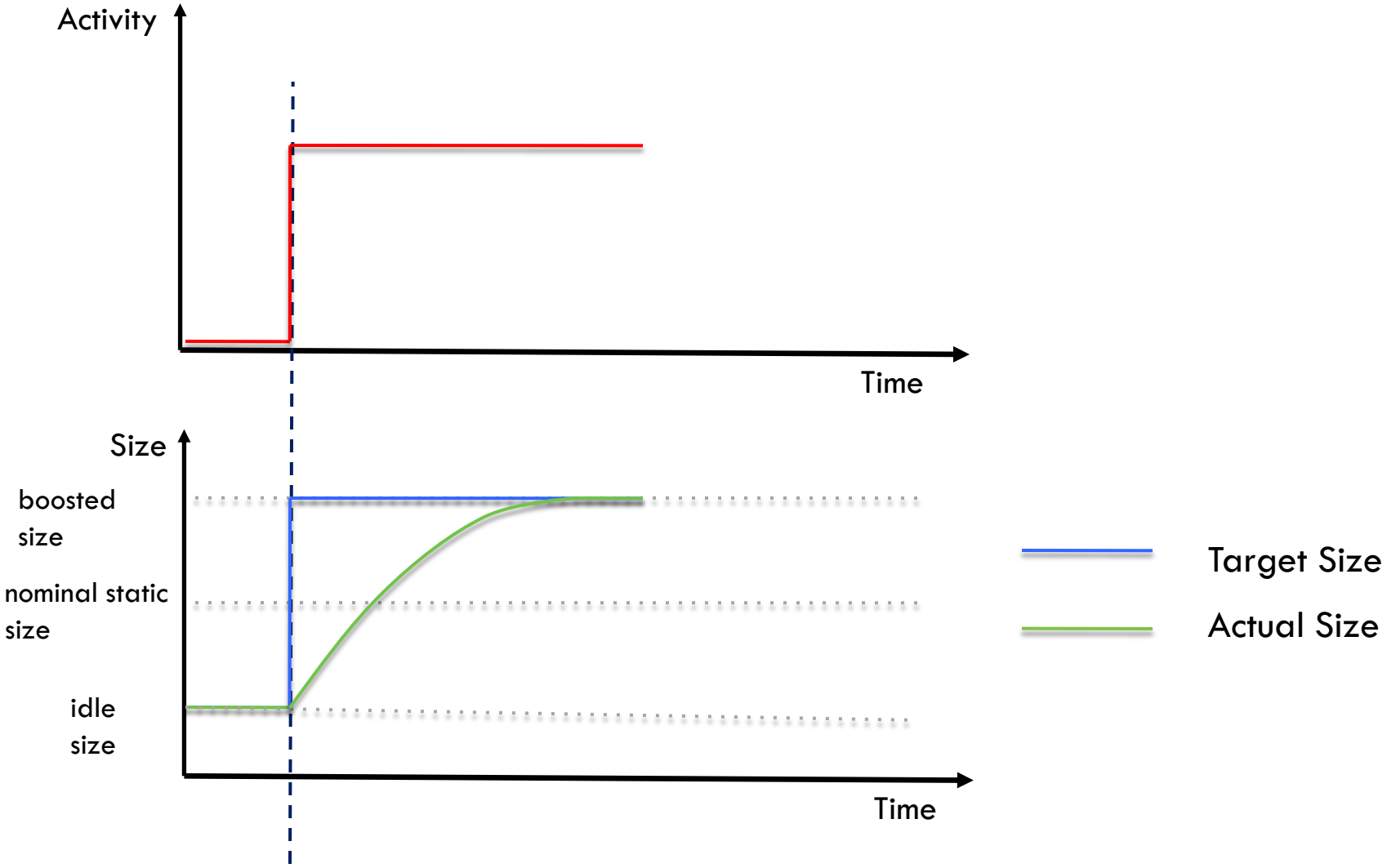


- Performance as well as overall progress under Ubik after the deadline is identical to static partitioning

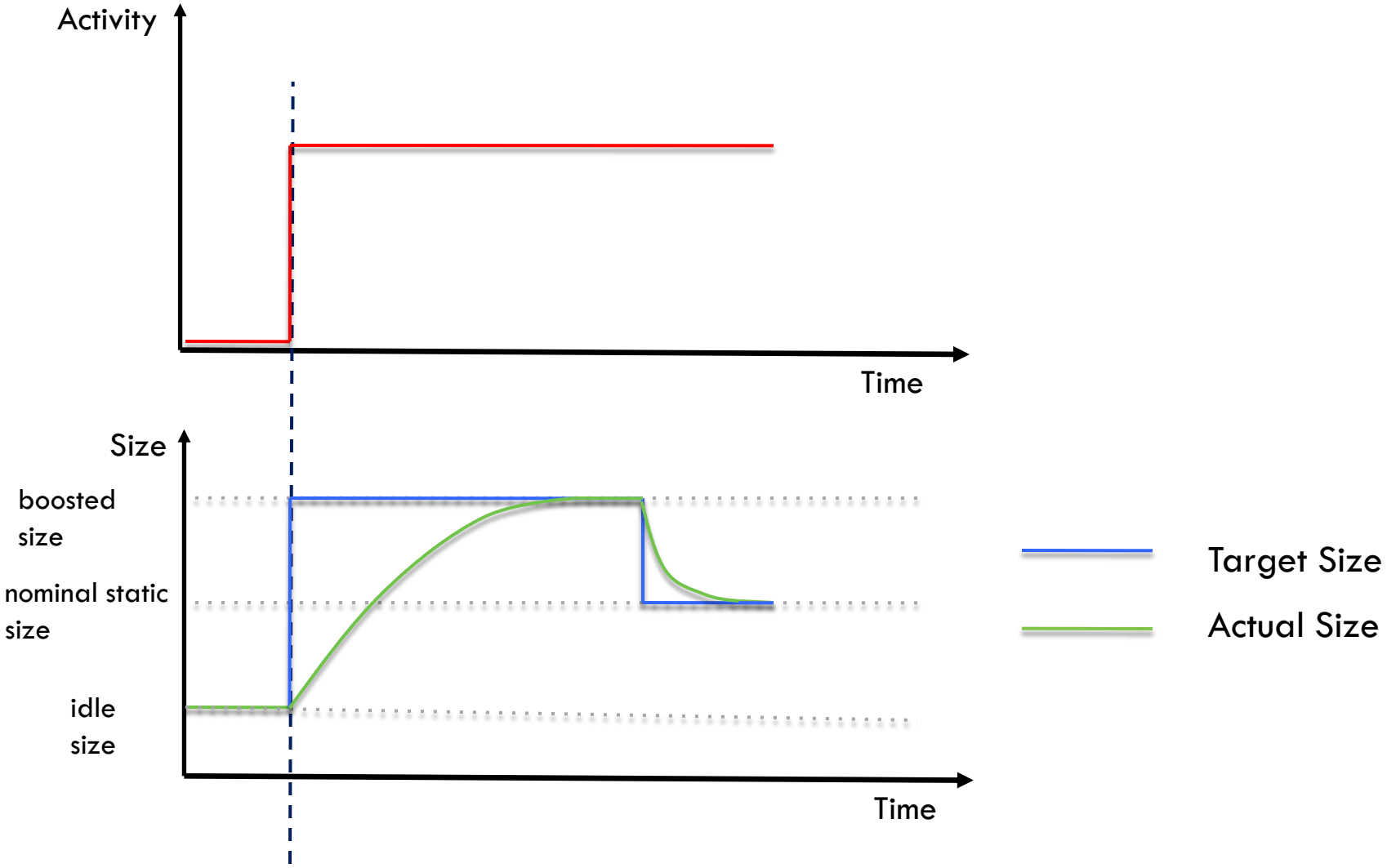
Ubik: Overview



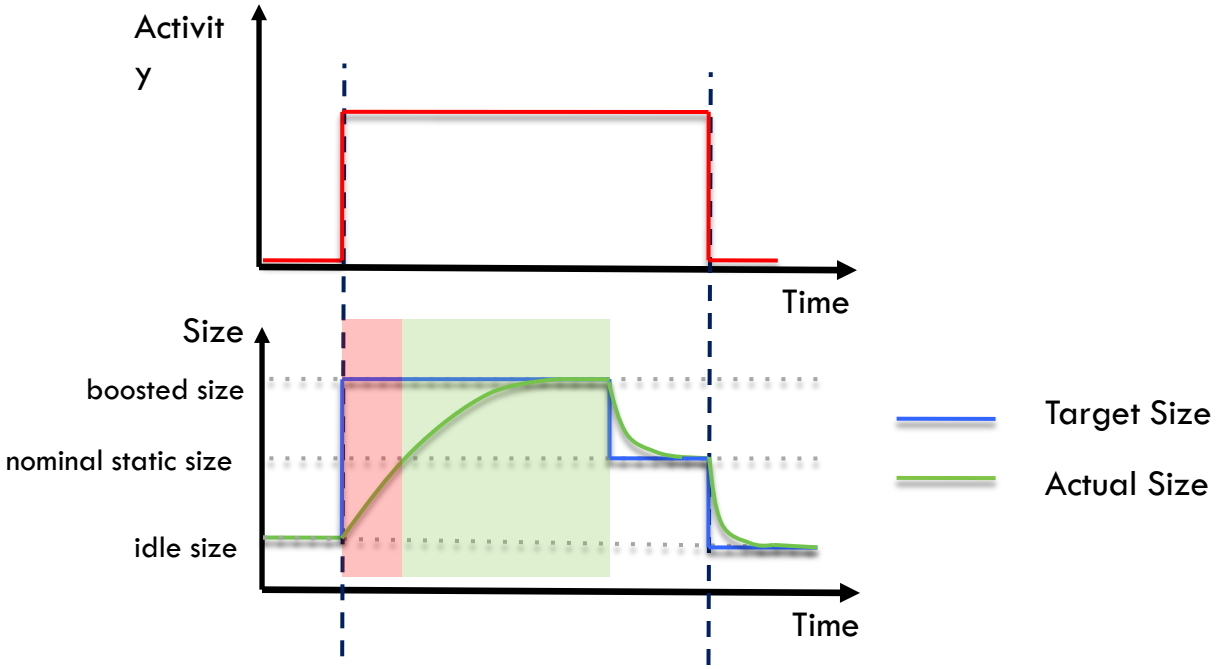
Ubik: Overview



Ubik: Overview



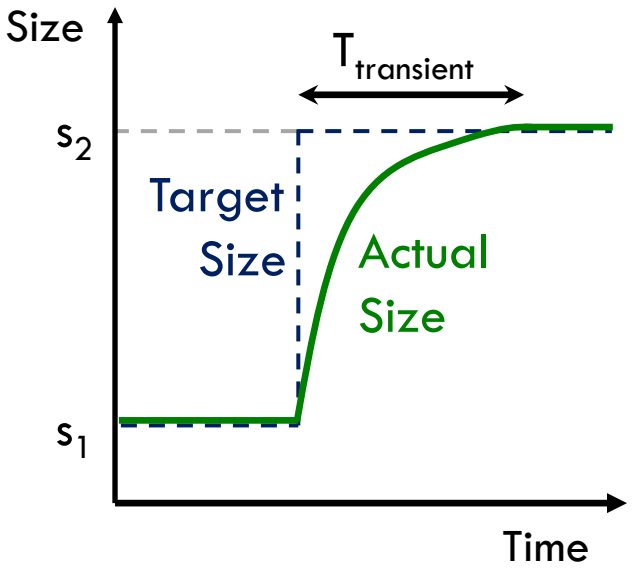
Ubik: Overview

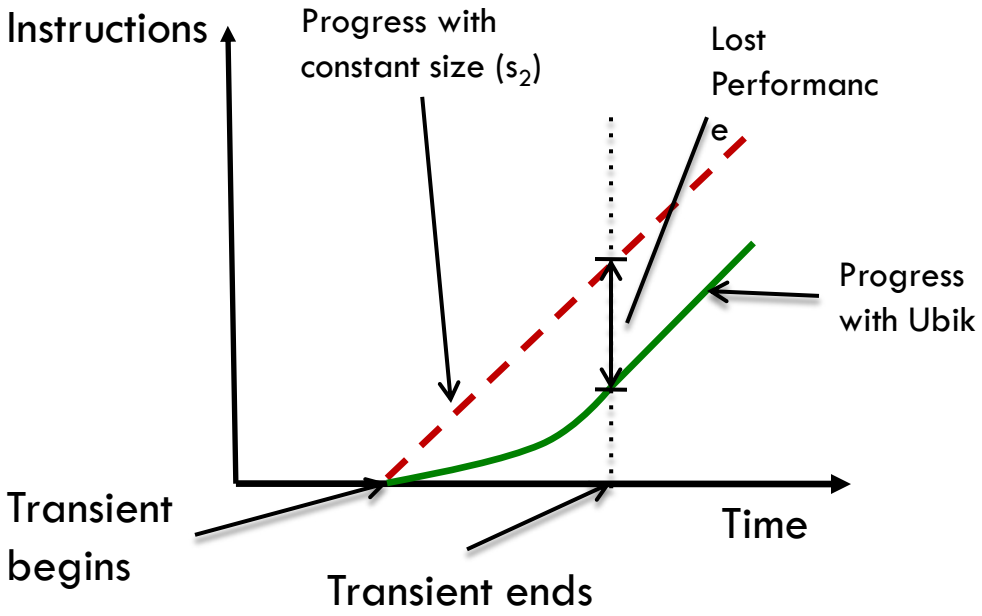


- Constraint: Cycles lost during ▇ should be compensated for by the cycles gained during ▇ before the deadline

Analyzing Transients

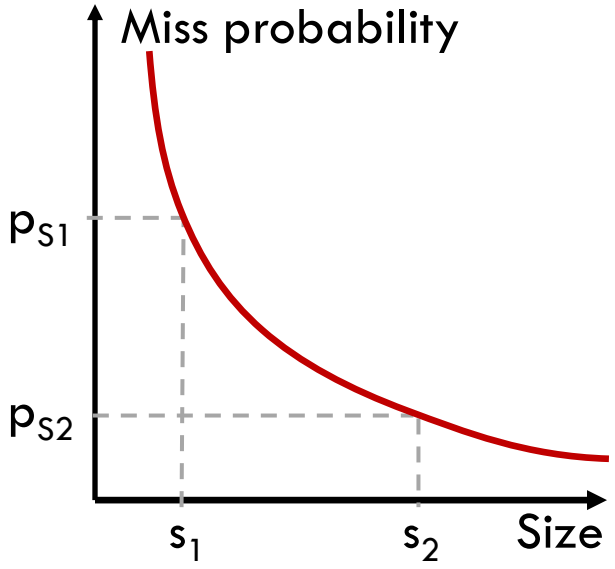
- Need **accurate** predictions for
  - The length of the transient from s_1 to s_2
  - Cycles lost during the transient from s_1 to s_2





Hardware Support

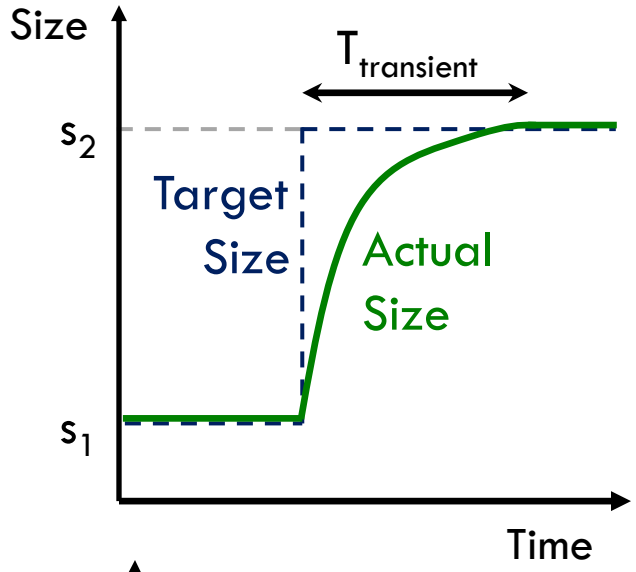
- Utility monitors to measure per-application miss curves



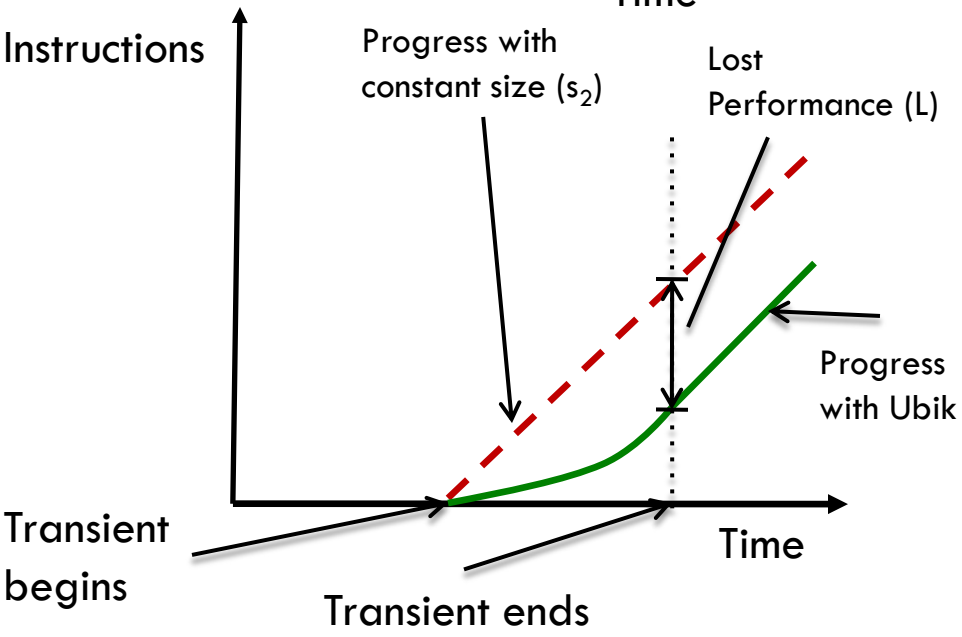
- Fine grained cache partitioning

- Memory Level Parallelism (MLP) profiler

Bounds on Transient Behavior



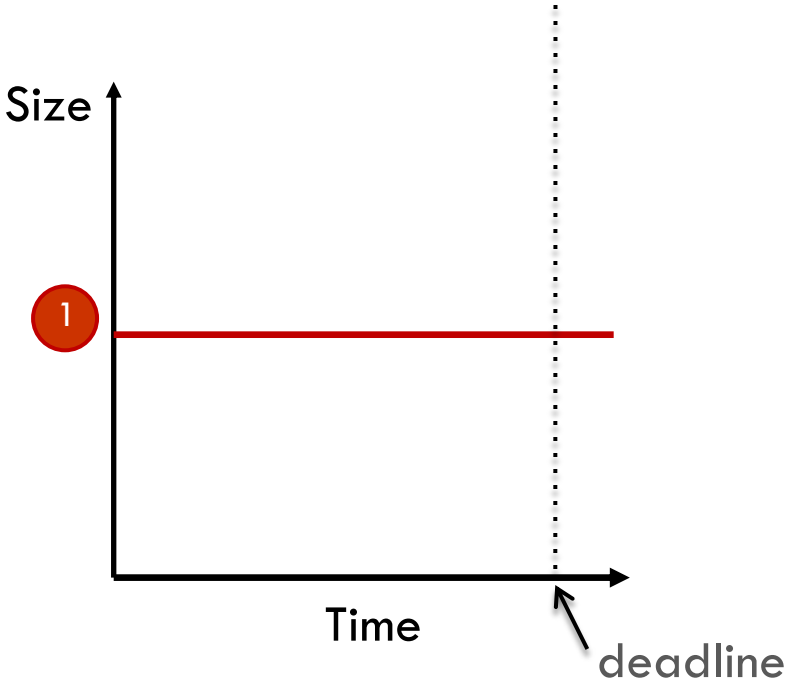
$$T_{transient} = \sum_{s=s_1}^{s_2-1} \frac{c}{p_s} + M \leq (s_2 - s_1)\left(\frac{c}{p_{s_2}} + M\right)$$



$$L = M \sum_{s=s_1}^{s_2-1} 1 - \frac{p_{s_2}}{p_s} \leq M\,(s_2 - s_1)\left(1 - \frac{p_{s_2}}{p_{s_1}}\right)$$

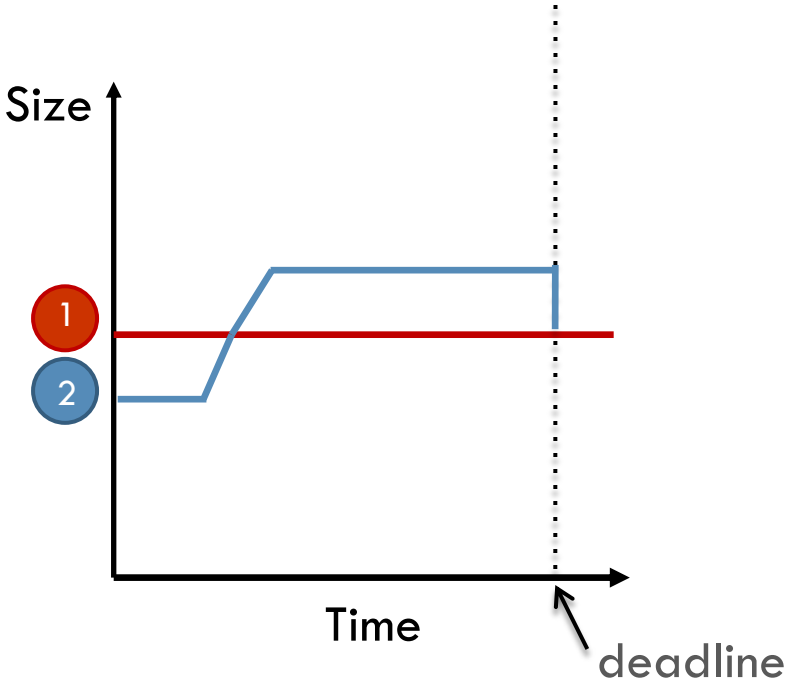
Ubik: Partition Sizing

- Use transient analysis to identify feasible (*idle size, boosted size*) pairs



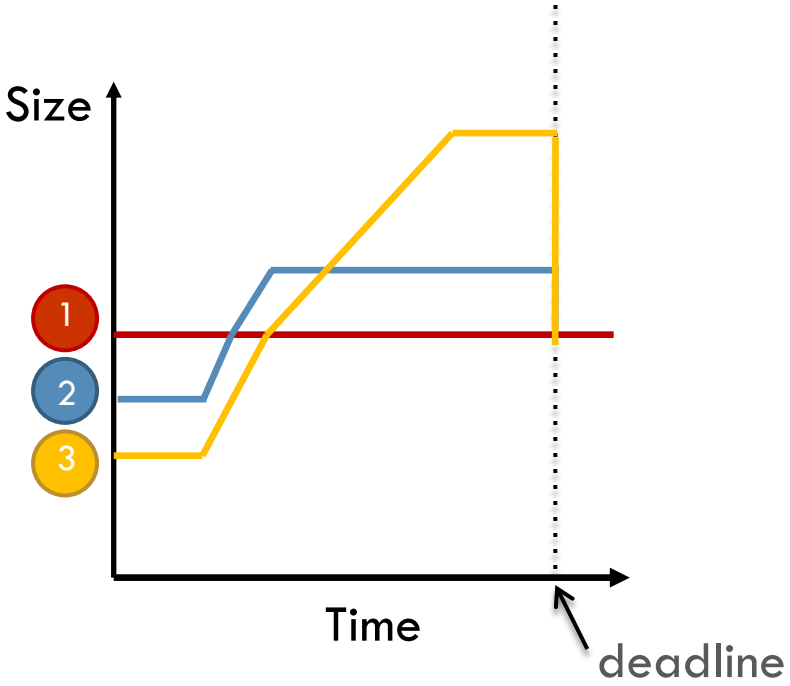
Ubik: Partition Sizing

- Use transient analysis to identify feasible (*idle size*, *boosted size*) pairs



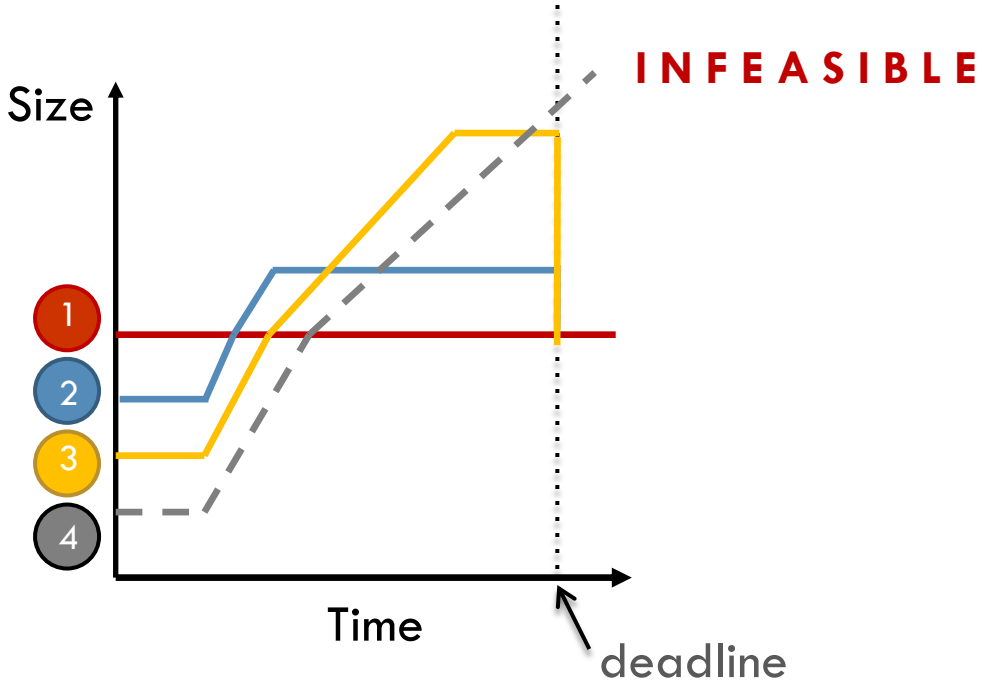
Ubik: Partition Sizing

- Use transient analysis to identify feasible (*idle size*, *boosted size*) pairs



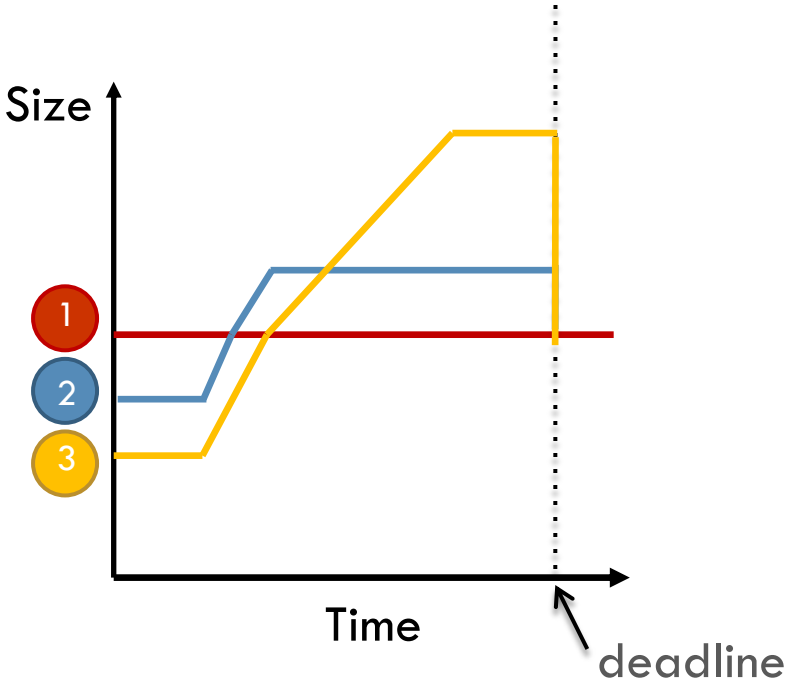
Ubik: Partition Sizing

- Use transient analysis to identify feasible (*idle size*, *boosted size*) pairs



Ubik: Partition Sizing

- Use transient analysis to identify feasible (*idle size, boosted size*) pairs
  - Choose the pair that yields the maximum batch throughput



- See paper for details

Outline

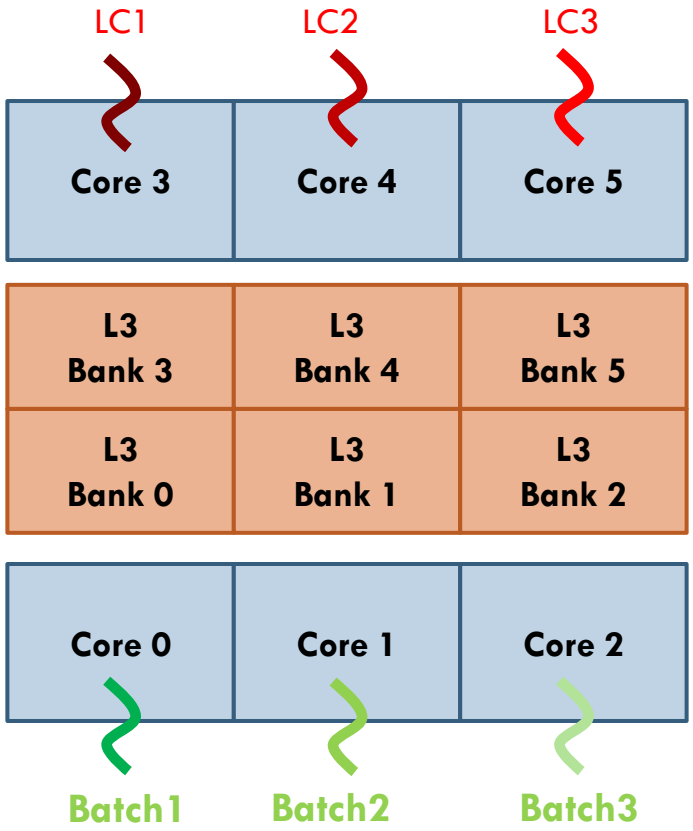
- Introduction
- Analysis of latency-critical apps
- Inertia-oblivious cache management schemes
- Ubik: Inertia-aware cache management
- Evaluation

Workloads

- Five diverse latency-critical apps
  - xapian (search engine)
  - masstree (in-memory key-value store)
  - moses (statistical machine translation)
  - shore-mt (multi-threaded DBMS)
  - specjbb (java middleware)

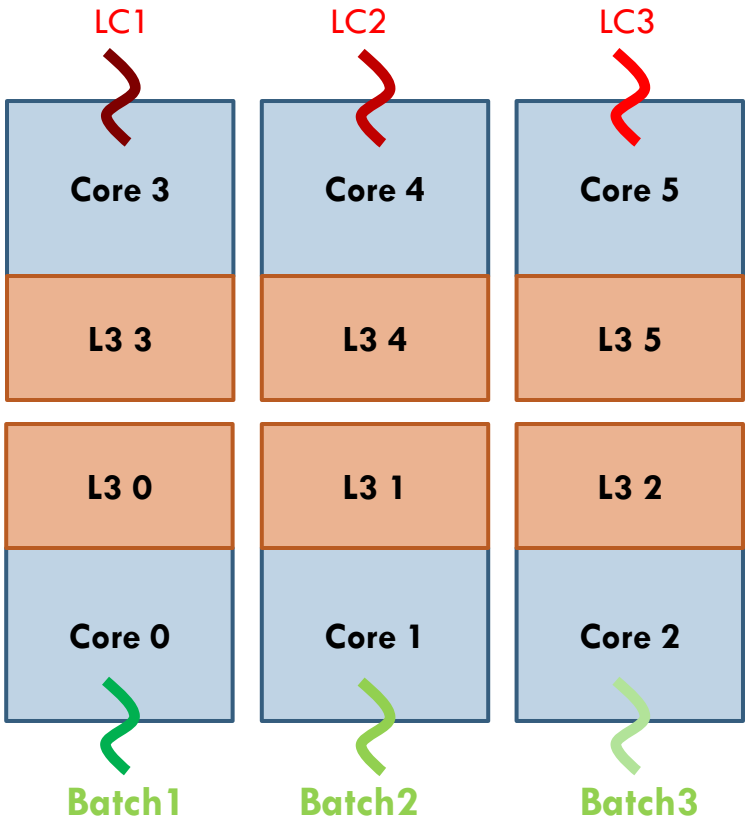
- Batch applications: random mixes of SPECCPU 2006 benchmarks

Target System



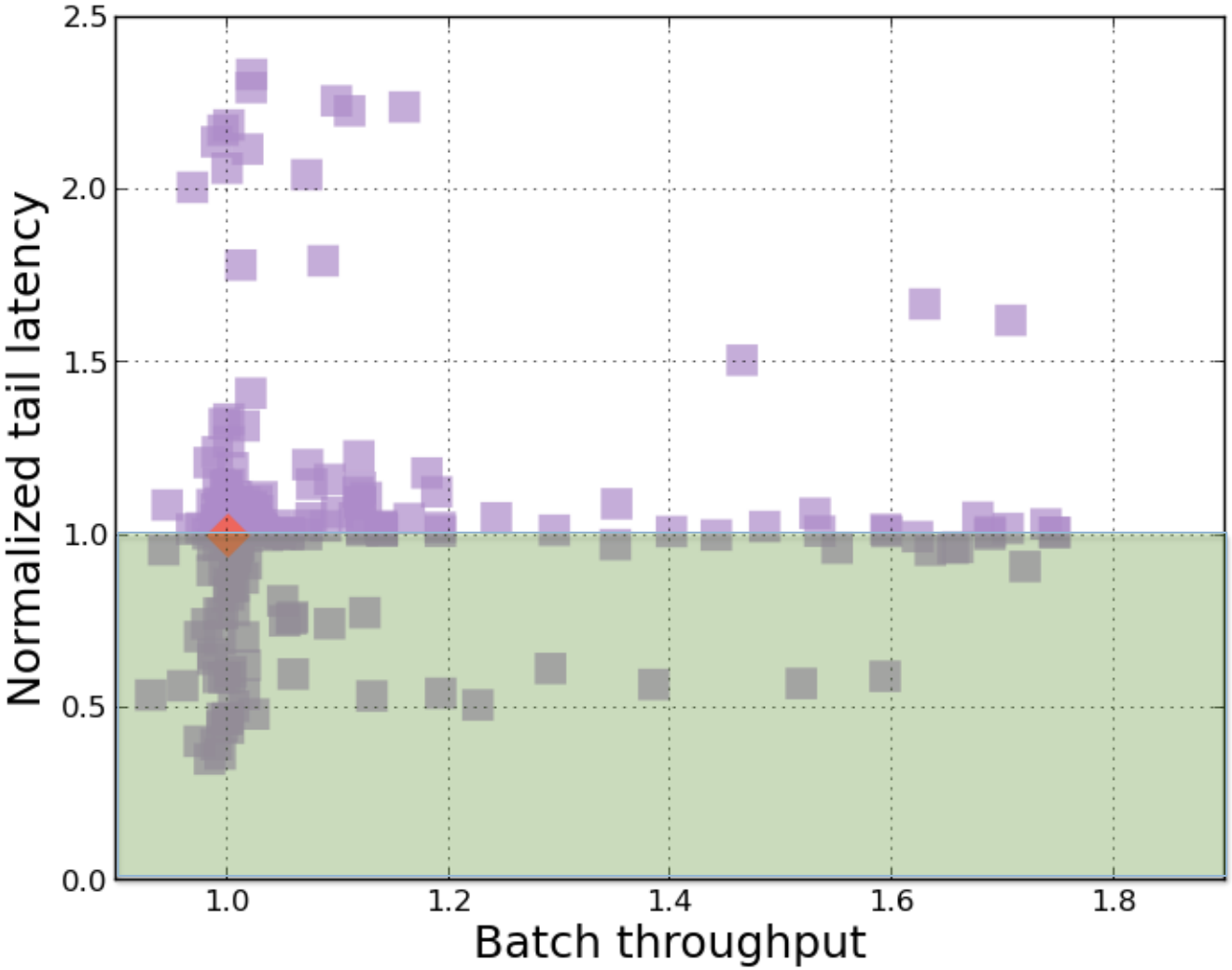
- 6 OOO cores
  - Private L1I, L1D and L2 caches
  - 12MB shared LLC
- 400 6-app mixes: 3 latency-critical + 3 batch apps
  - Apps pinned to cores

Metrics

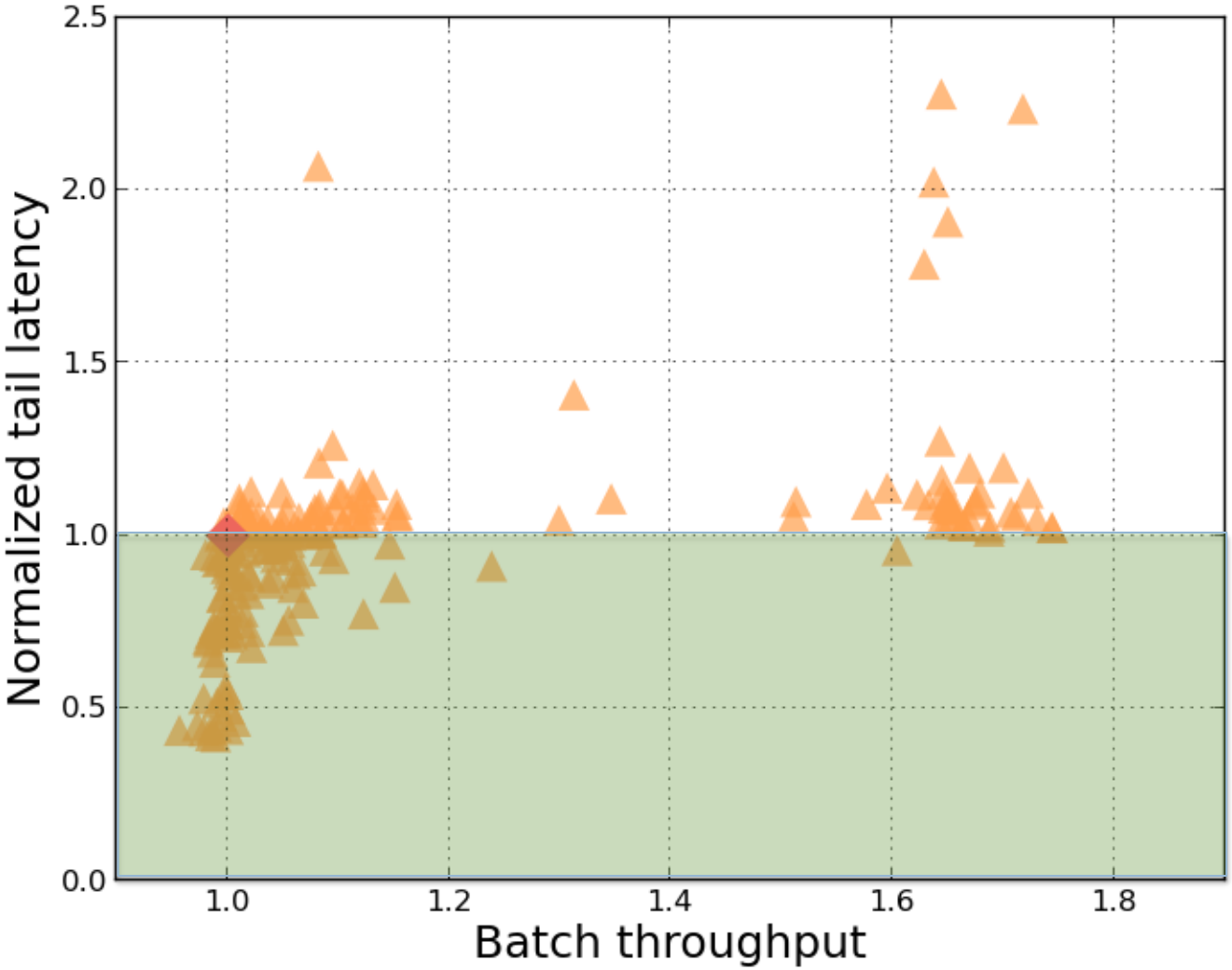


- Baseline system has private LLCs
- We report
  - Normalized tail latency
  - Throughput improvement for batch applications

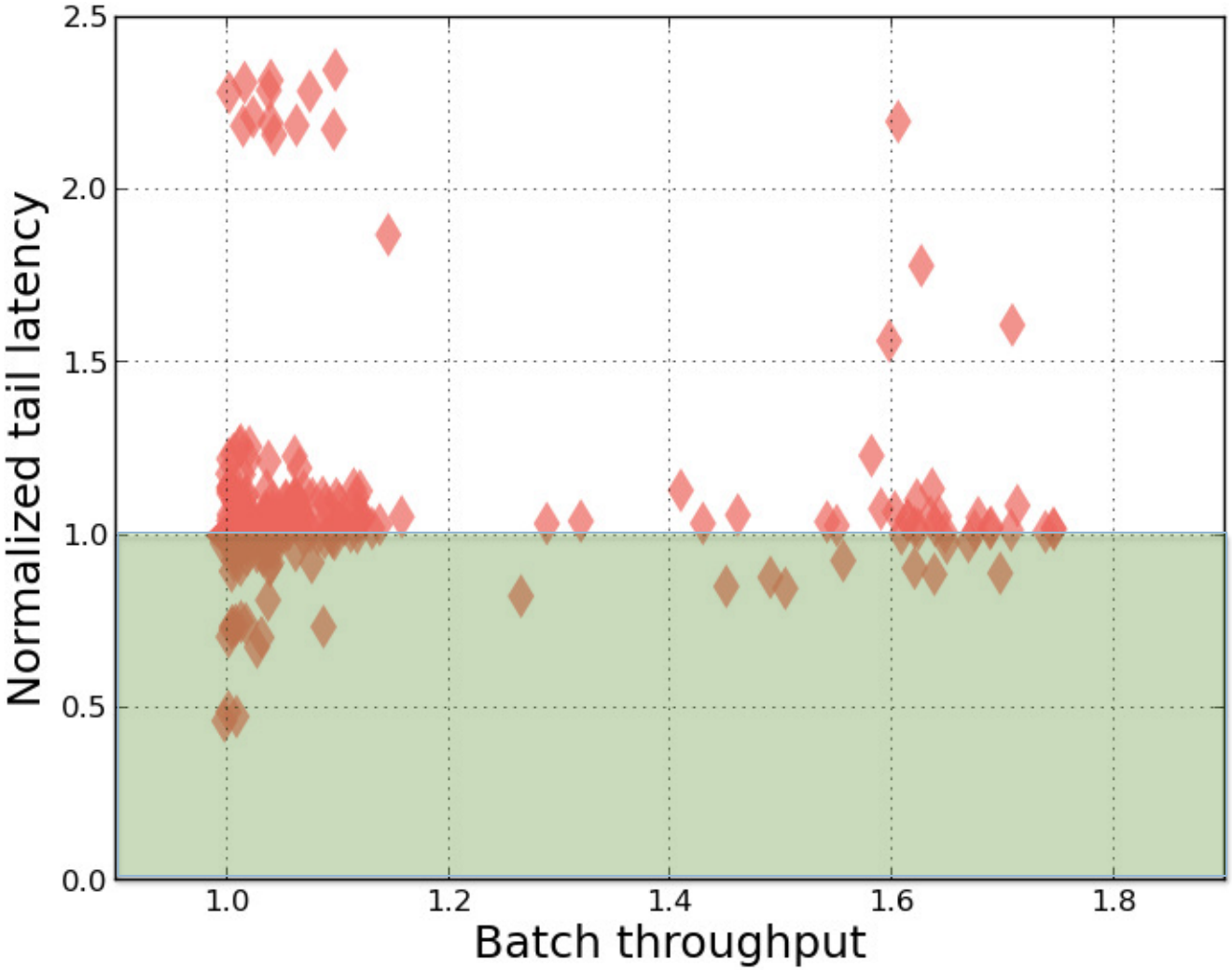
Results: Unmanaged LLC (LRU)



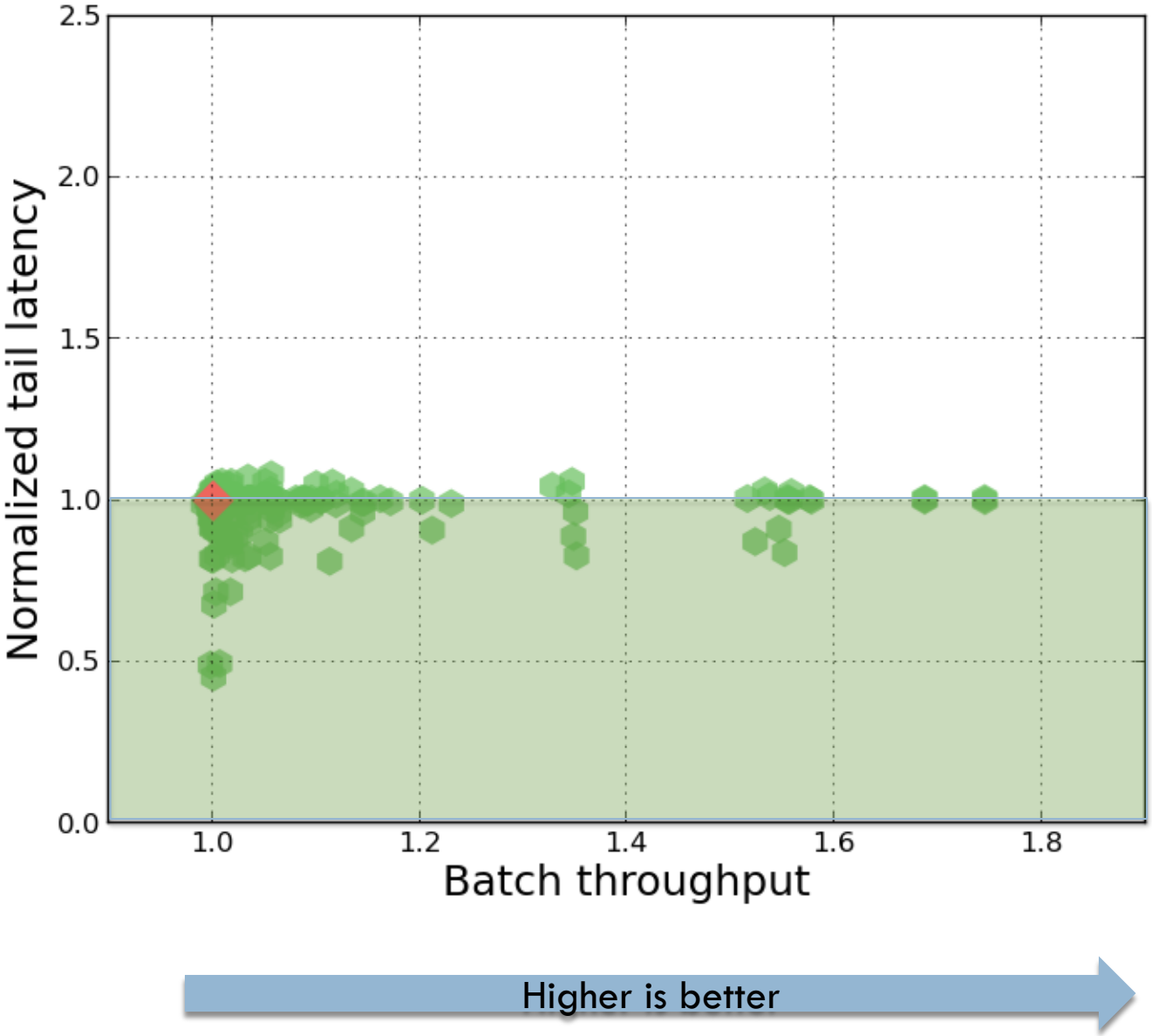
Results: UCP


2.5
2.0
1.5
1.0
0.5
0.0
Normalized tail latency
1.0
1.2
1.4
1.6
1.8
Batch throughput
Higher is better

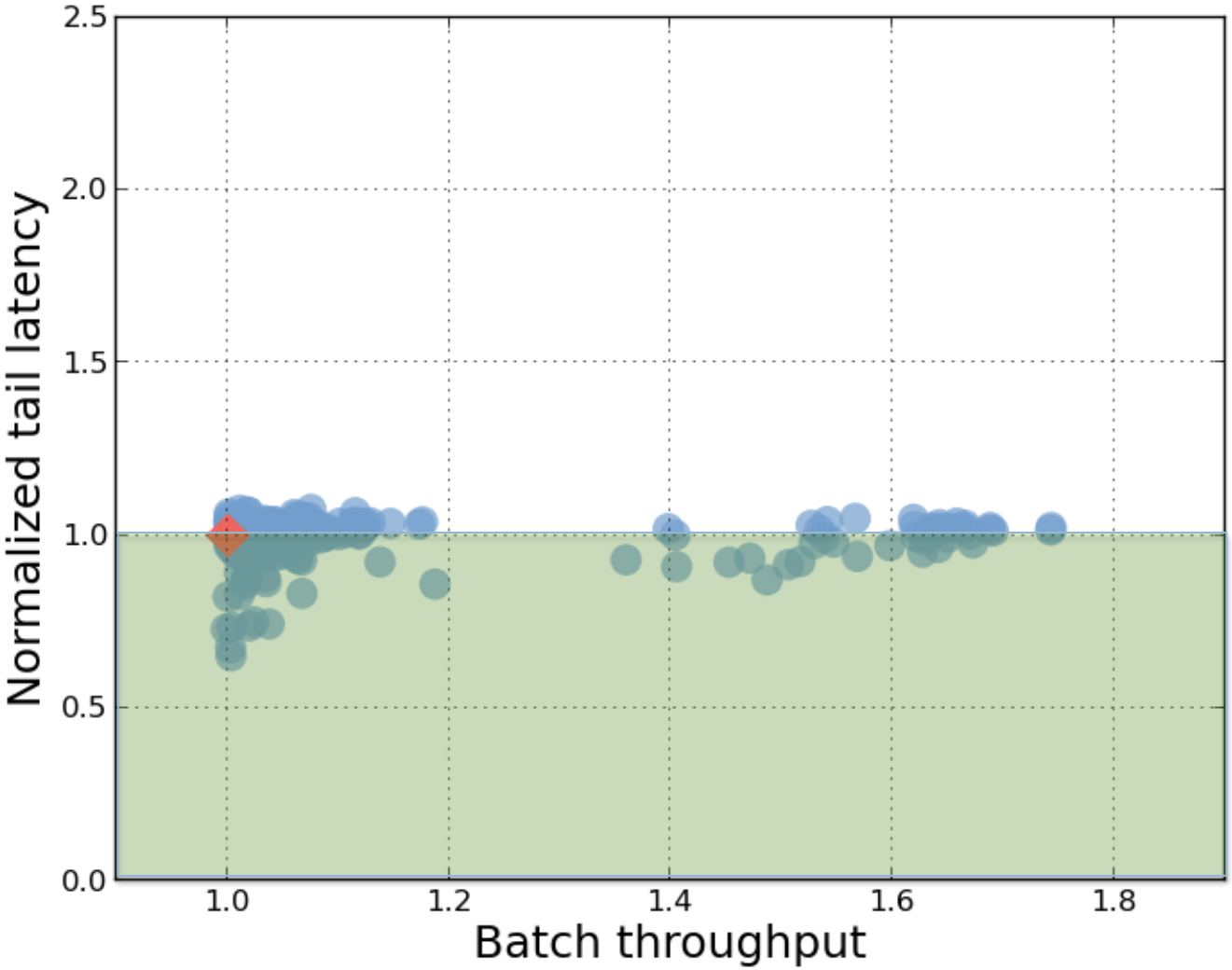
Results: OnOff



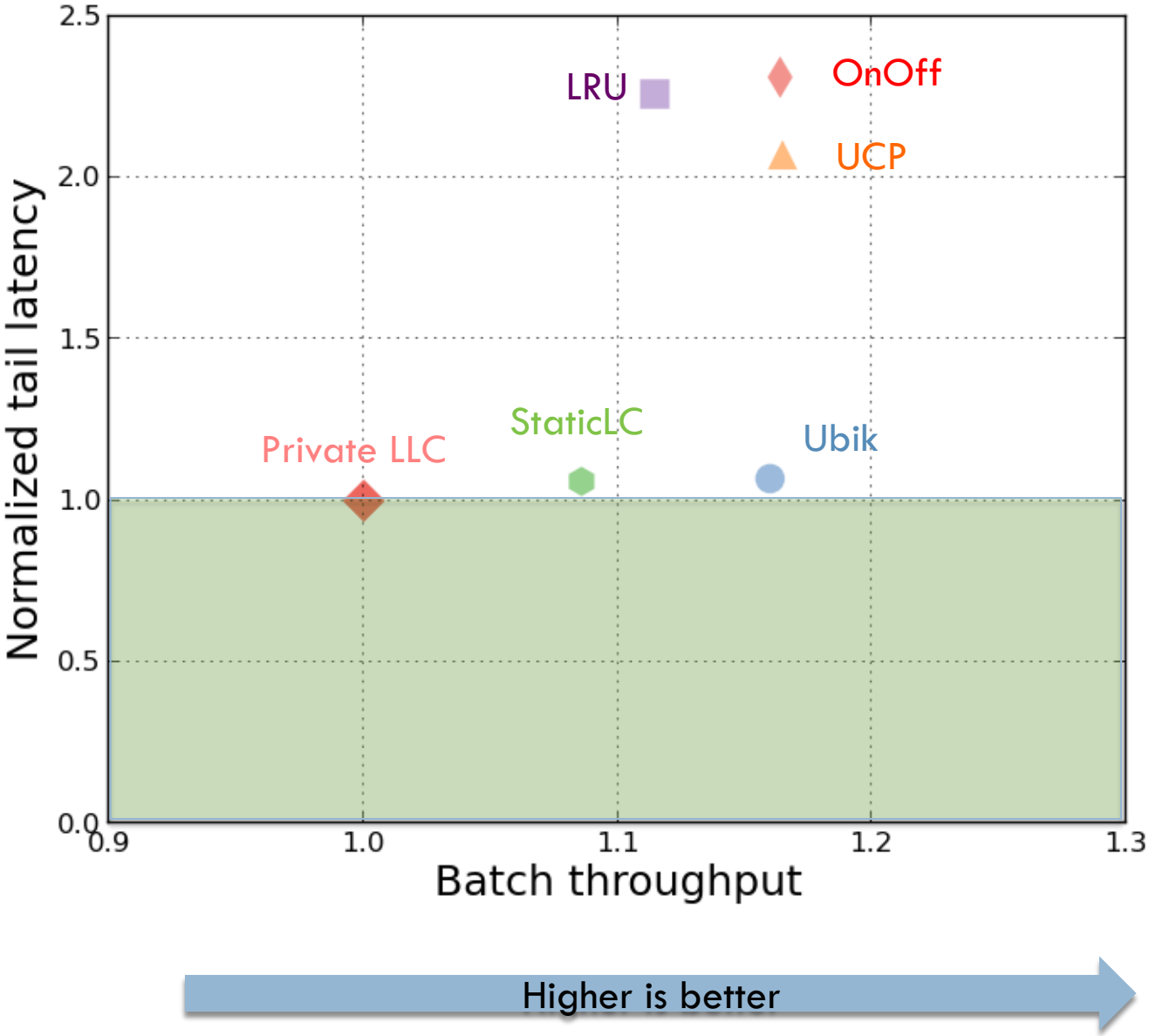
Results: StaticLC



Results: Ubik



Results: Summary



Conclusions

- To guarantee tail latency, dynamic resource management schemes must be inertia-aware

- Ubik: Inertia-aware cache capacity management
  - Preserves tail of latency-critical apps
  - Achieves high cache space utilization for batch apps
  - Requires minimal additional hardware

Thanks For Your Attention!

Questions?

Massachusetts
Institute of
Technology