CPU Cache Partitioning for Networked Systems

Thomas Kim[†], Sol Boucher[†], Hyeontaek Lim[†], David G. Andersen[†], Michael Kaminsky[‡]

[†]Carnegie Mellon University, [‡]Intel Labs

PROBLEMS

- Datacenter workloads have strict latency SLOs
- High utilization crucial for cost efficiency
- Collocated tasks suffer cache contention
- Overprovisioning required to meet SLOs

GOALS

- Guarantee SLOs for networked systems
- Avoid excessive overprovisioning

OUR WORK

- Minimize tail latency using cache partitioning
- Avoid resource deprivation of contending processes

Hardware cache partitioning

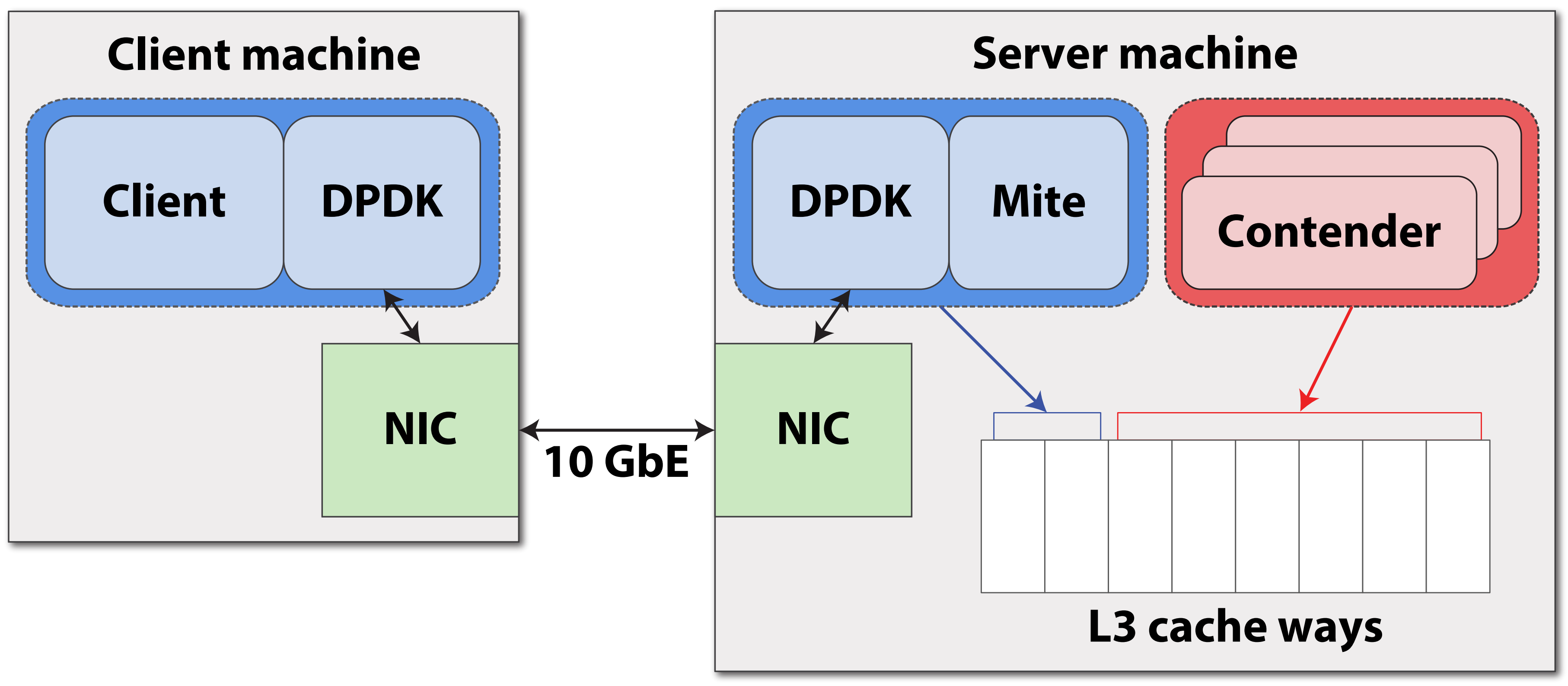
- Intel® Cache Allocation Technology (CAT)
  - Commercial deployment of cache partitioning
  - Implemented as way partitioning of L3 cache

Contributions

- Achieve microsecond-scale 99.9th percentile tail latency SLOs
- Minimal degradation of background task throughput

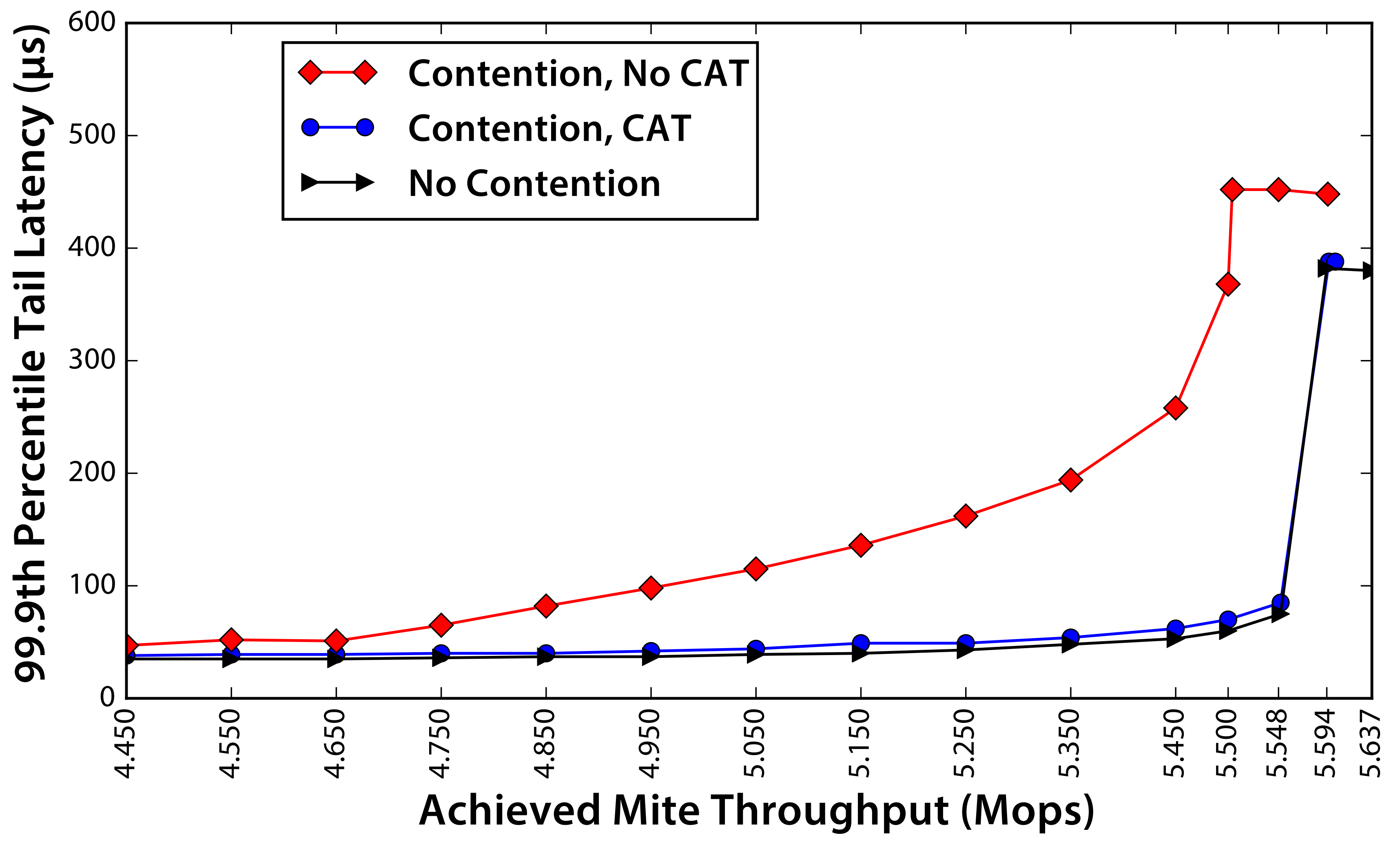
Experimental design

- Mite: latency-critical task
  - Single-threaded MICA key-value store
- Contender: throughput-intensive batch workload
  - Fifteen TensorFlow threads training on MNIST
- Measure end-to-end tail latency of client↔mite queries
- *No CAT* trials use unpartitioned L3 cache
- *No Contention* trials omit contender threads

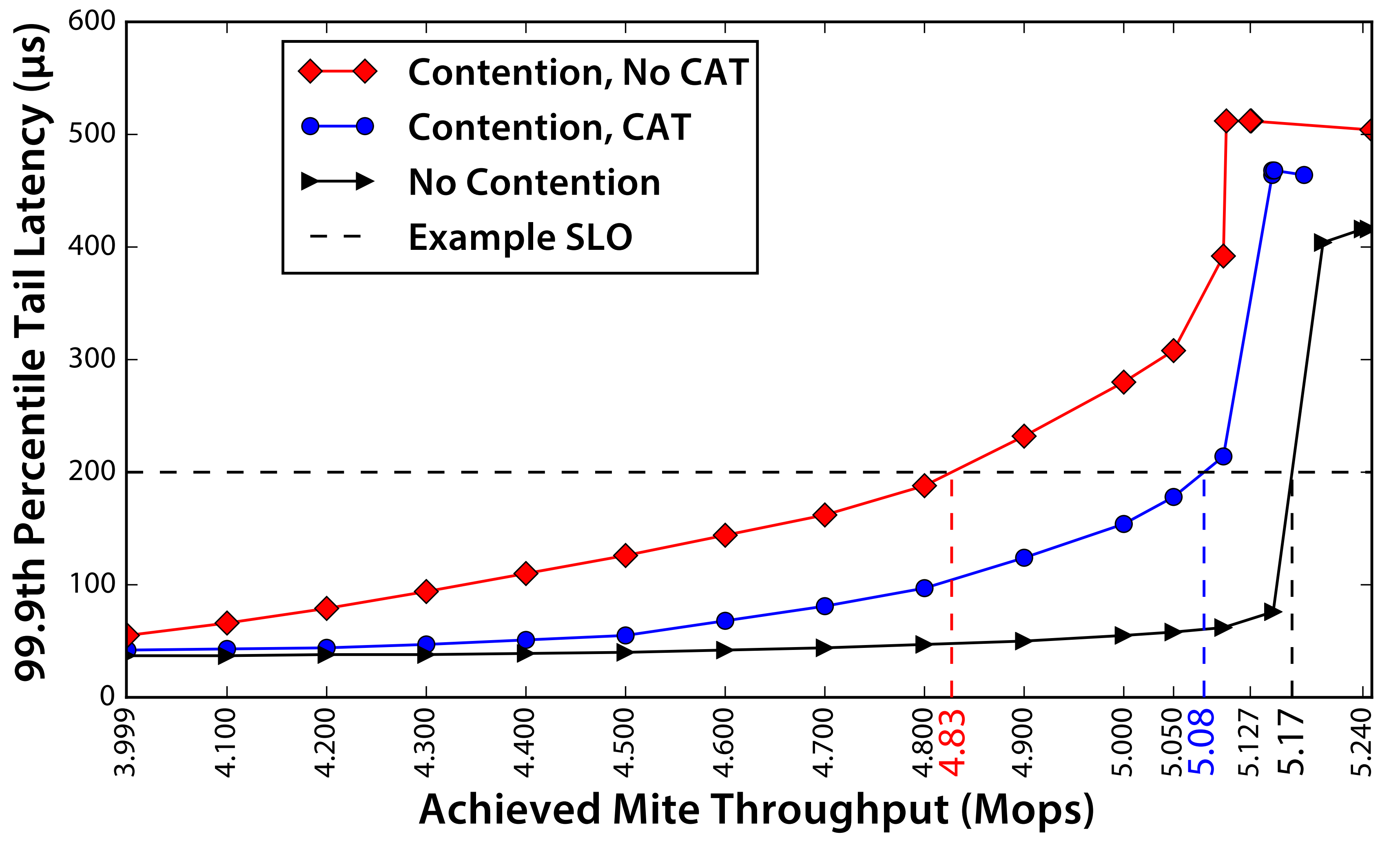


CAT reduces tail latency by up to 5.3x

Small (2 MB) working set: CAT negates the effects of contention

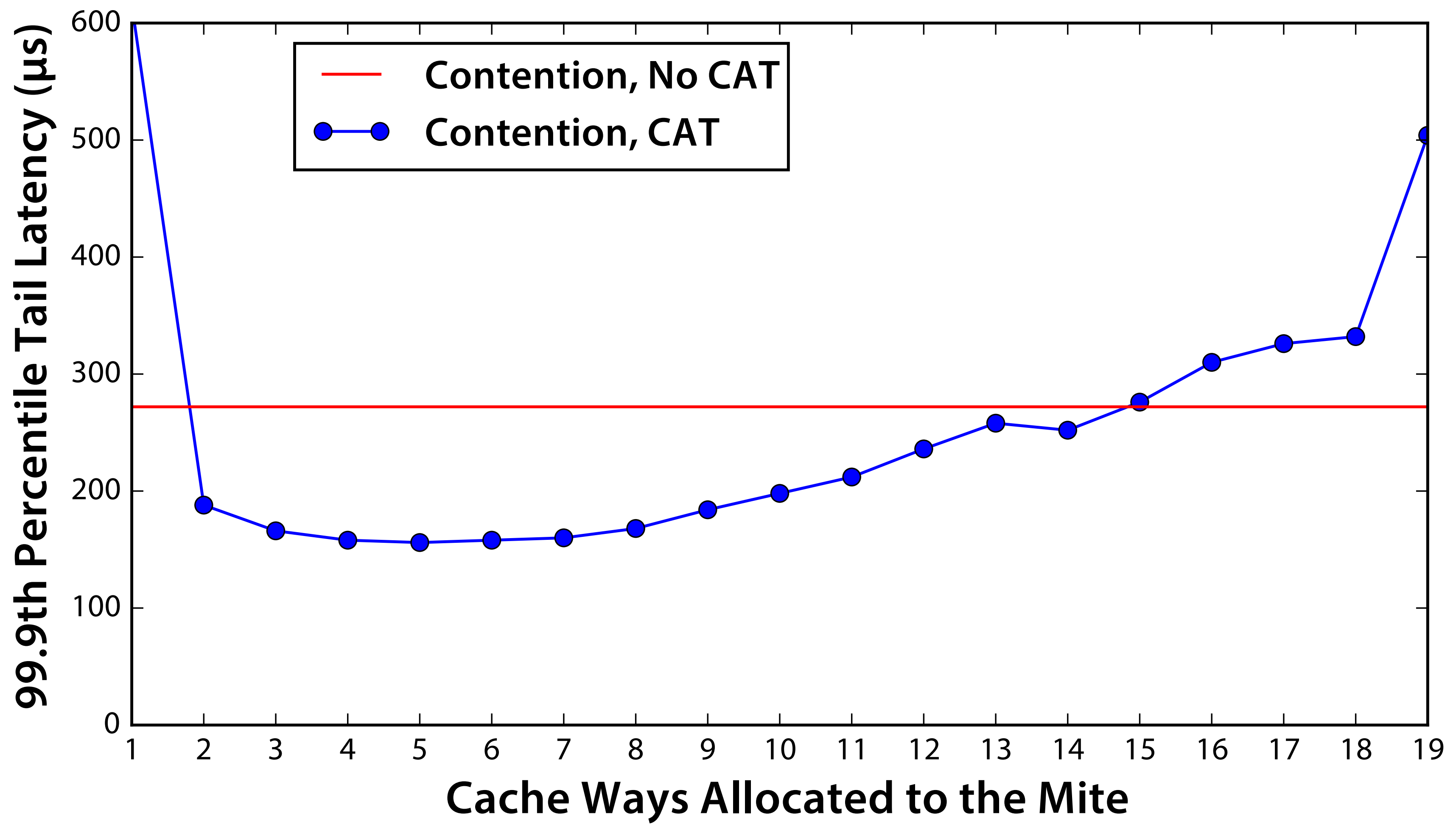


Large (1 GB) working set: CAT decreases latency by up to 1.8x



Large allocations raise latency

Optimizing for the mite alone can increase tail latency



Carnegie Mellon University