leasuring and Optimizing Tail Latenc

thryn S McKinley, Google

Yang, Stephen M Blackburn,

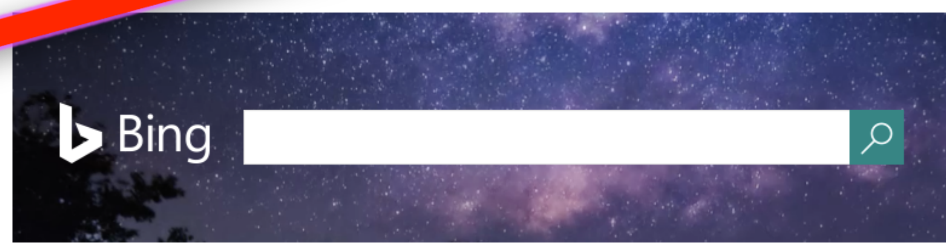
d Haque, Sameh Elnikety, Yuxiong He, Ricardo Bianchini

Google Clou

Tail Latency Matters

TOP PRIORITY

Google
Australia

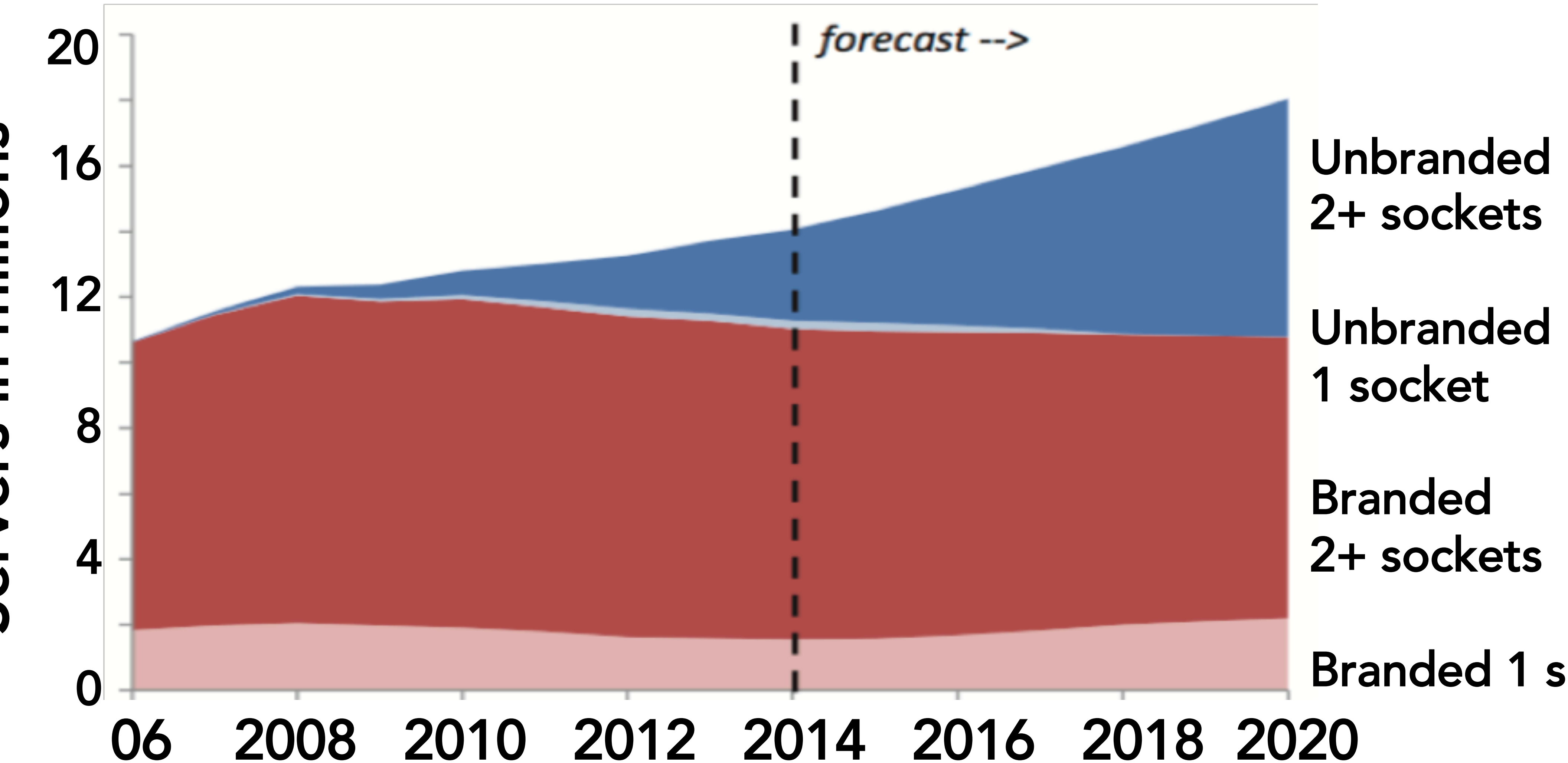


0 millisecond delay decreased
arches/user by 0.59%. [Jack Brutlag, Google]

Two second slowdown reduced
revenue/user by 4.3%. [Eric Schurman, B

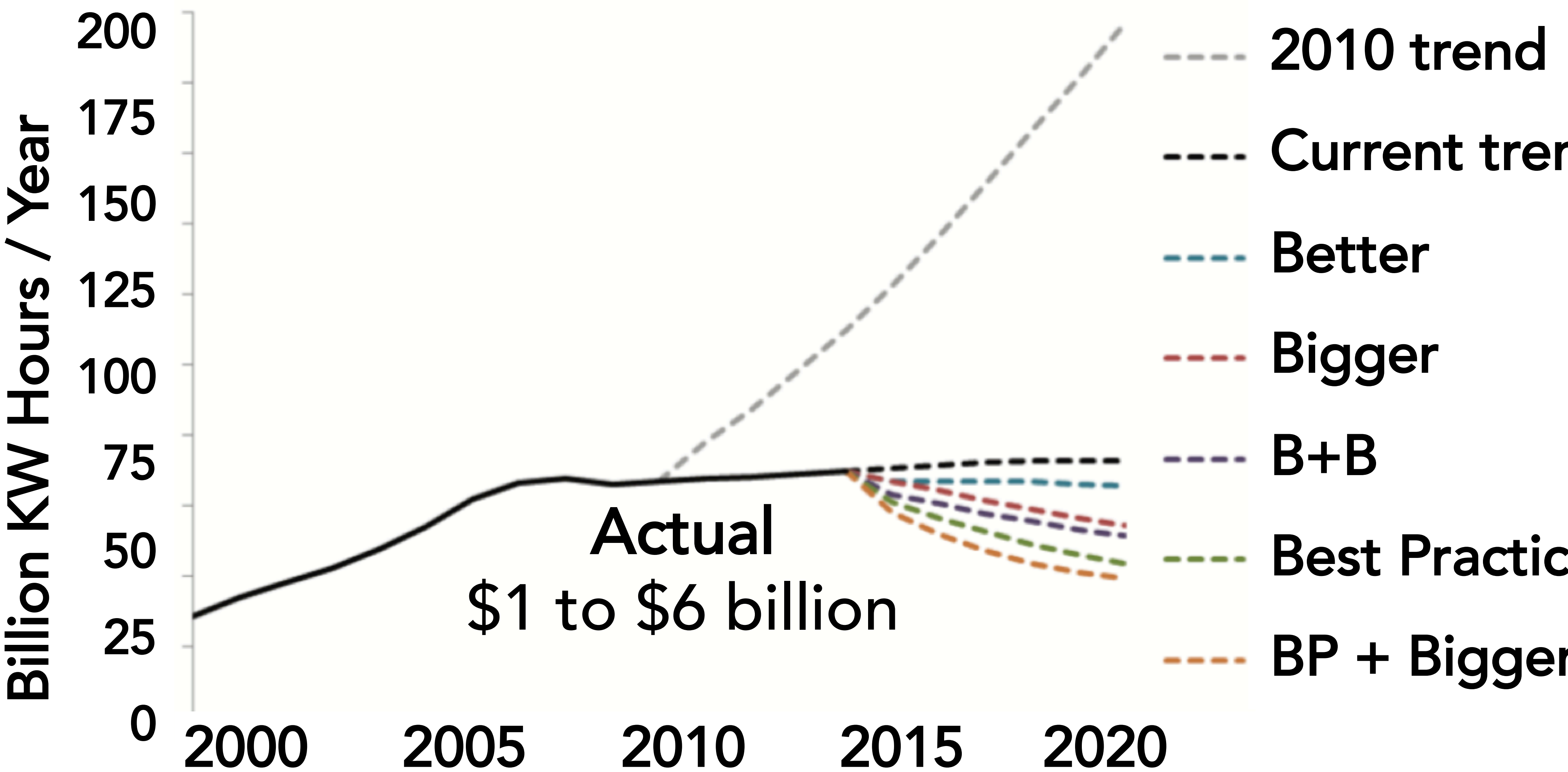

ogle/Connie Zhou

Servers in US datacenters



*Shehabi et al., United States Data Cen
Usage Report, Lawrence Berkeley, 201

Electricity in US datacenters



*Shehabi et al., United States Data Cen
Usage Report, Lawrence Berkeley, 201

Datacenter economics quick facts*

~ $500,000	Cost of small datacenter
~3,000,000	US datacenters in 2016
~ $1.5 trillion	US Capital investment to date
~ $3,000,000,000	KW dollars / year
~ $30,000,000	Savings from 1% less work
Lots more	by not building a datacenter

*Shehabi et al., United States Data Cen
Usage Report, Lawrence Berkeley, 201

TOP PRIORITY

Efficiency

Google Clou

il Latency

TOP PRIORITY

Efficiency

Google Clou

il Latency

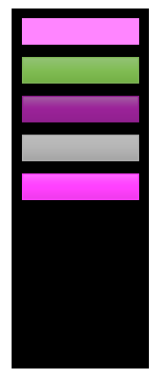
BOTH ?!

Efficiency

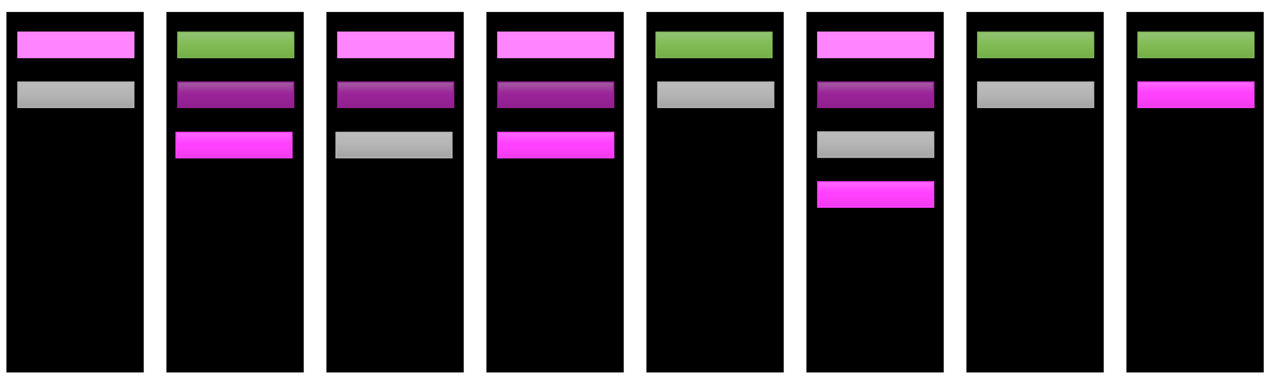
Google Clou

Server architecture

client

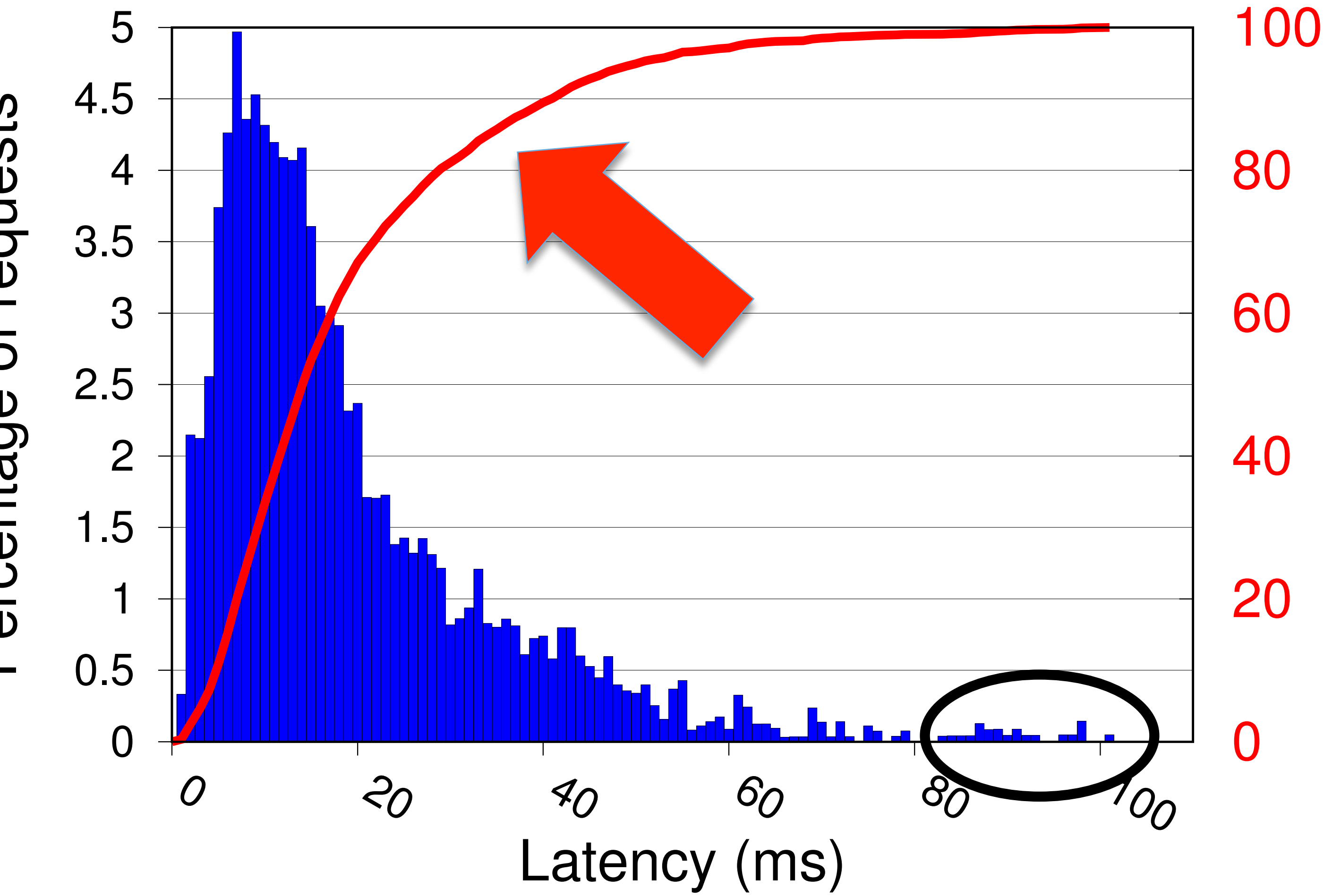


aggregator



workers

Characteristics of interactive services



LC

Bursty, diurnal

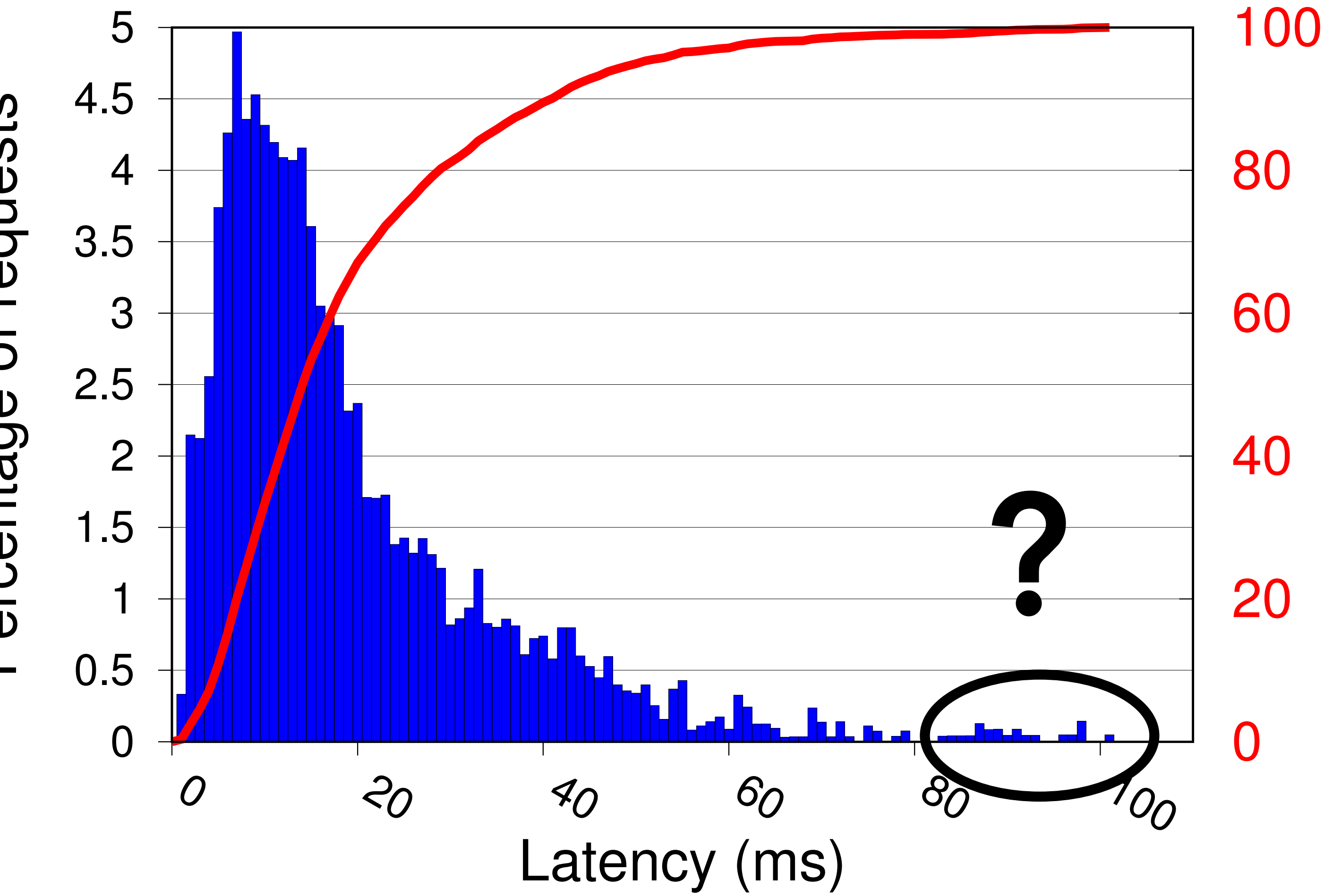
CDF changes slowly

Slowest server dictates

Orders of magnitude di

average & tail - 99th 9

What is in the tail?



Roadmap

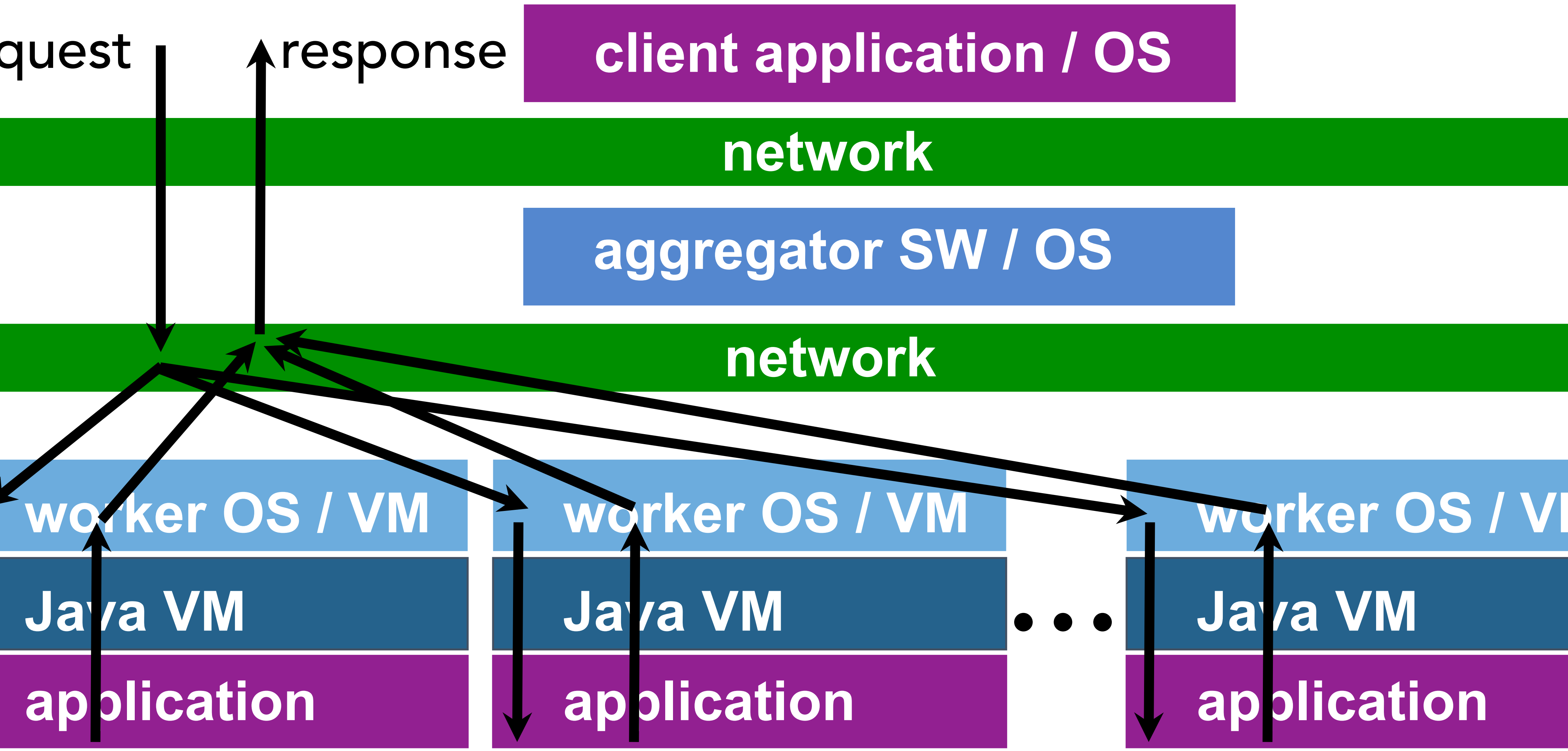
Diagnosing the tail with continuous profiling

Noise	systems are not perfect
Queuing	too much load is bad, but so is over provisioni
Work	many requests are long

nsights Use the CDF off line

Long requests reveal themselves, treat them specially

Simplified life of a request

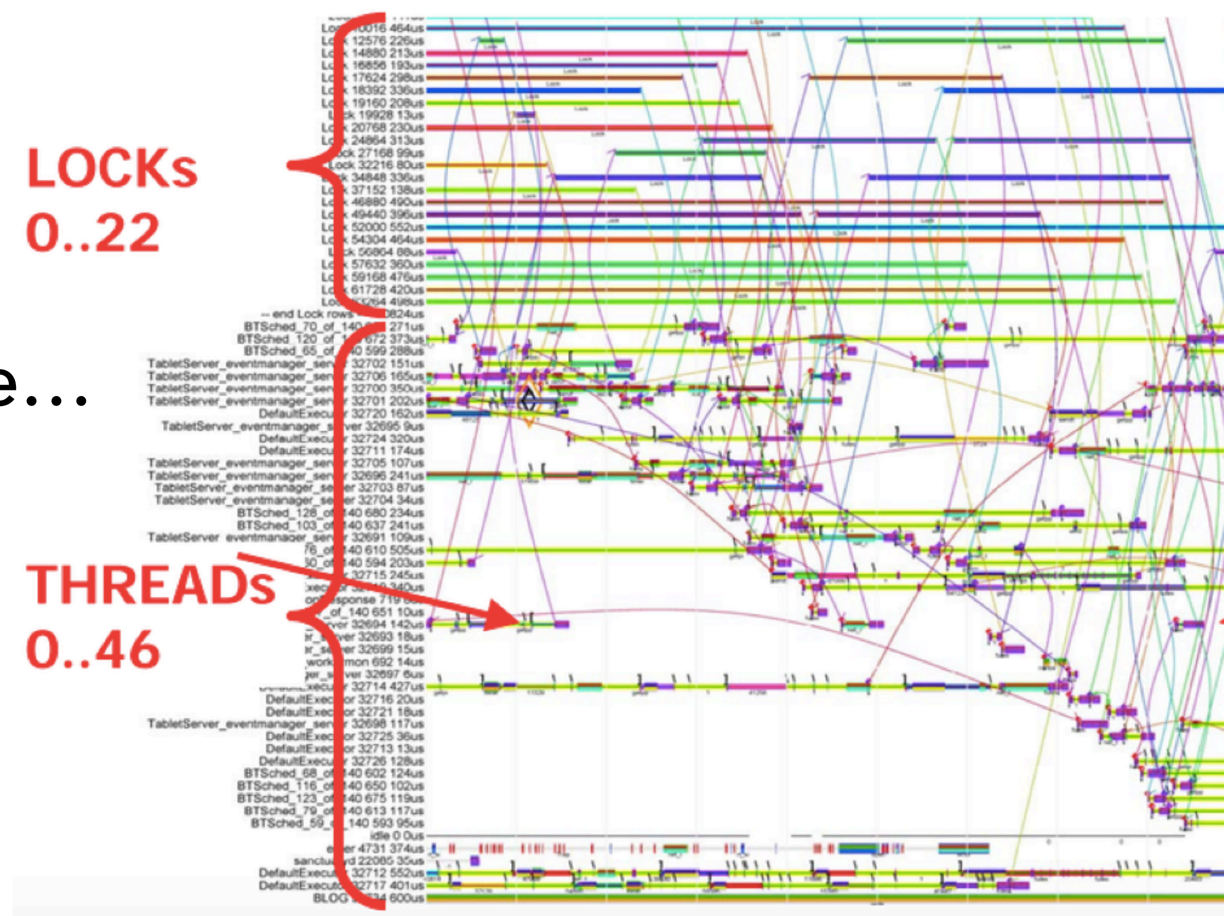


Prior state of the art

Dick Site, Google https://www.youtube.com/watch?v=QBu2Ae8-8LM

@ Google

- Hand instrument system
- % on-line budget
- sample – but tails are rare…
- Off-line schematics
- Have insight
- mprove the system



Request profiling

- Hand instrument system
  - % on-line budget
  - sample – but tails are rare…
- Off-line schematics
- Have insight
- mprove the system

Request profiling

Hand instrument system	Automated instrumentation
% on-line budget	1% on-line budget
sample – but tails are rare…	continuous on-line profiling
Off-line schematics	Off-line schematics
Have insight	Have insight
mprove the system	Improve the system
	+ On-line optimization

Automated cycle-level on-line profiling

[ISCA’15 (Top Picks HM), ATC’1

nsight Hardware & software generate signals

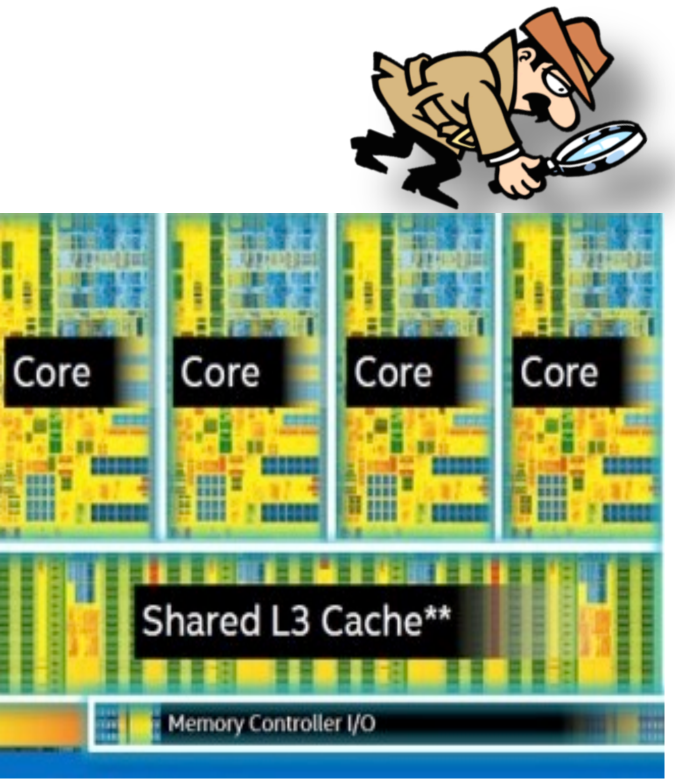
	hardware signals performance counters	software signals memory locations
counters	✓	✓
tags	✓	✓

SHIM Design

SCA’15 (Top Picks HM), ATC’16

Observe global state from other core

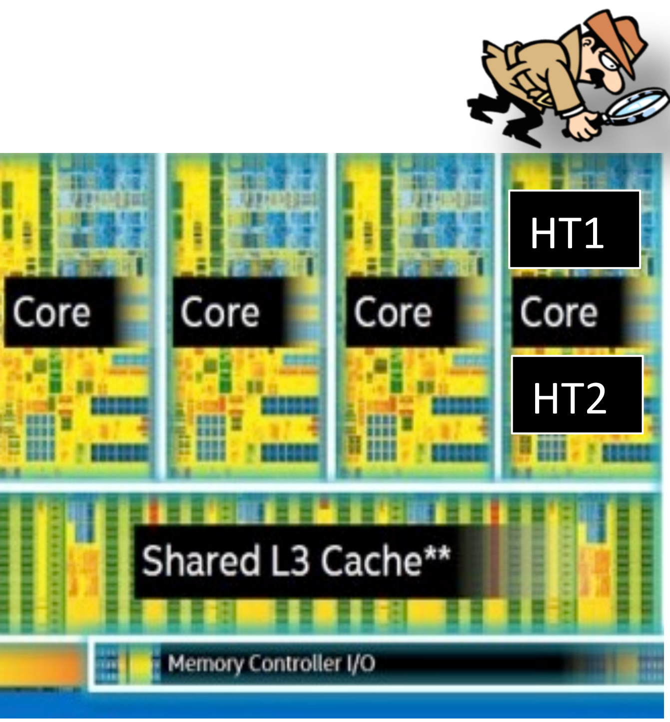
```
while (true):
   for counter in LLC misses, cycles:
       buf[i++] = readCounter(counter)
```

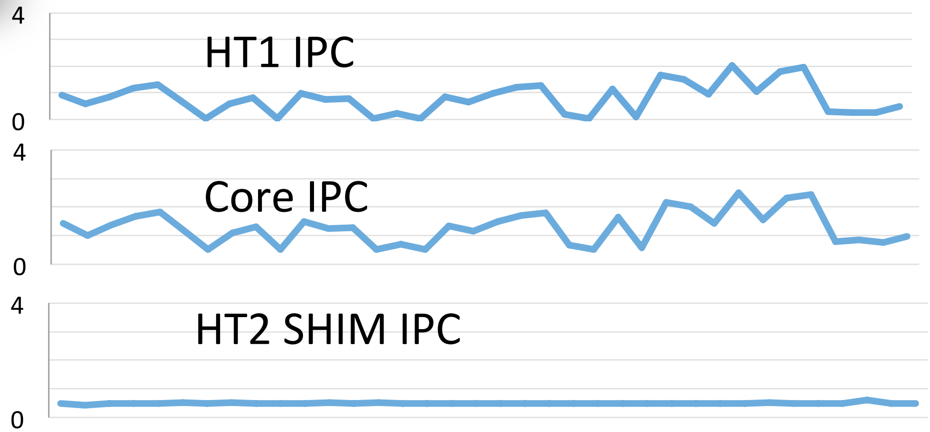


LLC misses per cycle

Observe local state with SMT hardware

```
while (true):
   for counter in HT2 SHIM, Core, Cycles:
       buf[i++] = readCounter(counter);
```

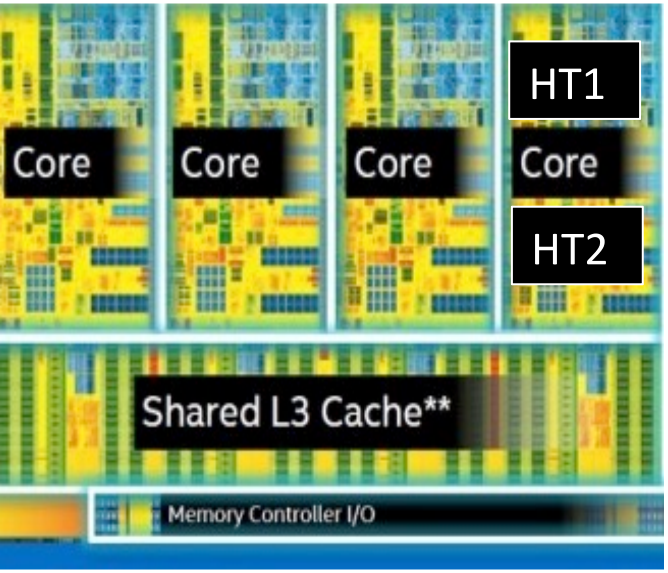


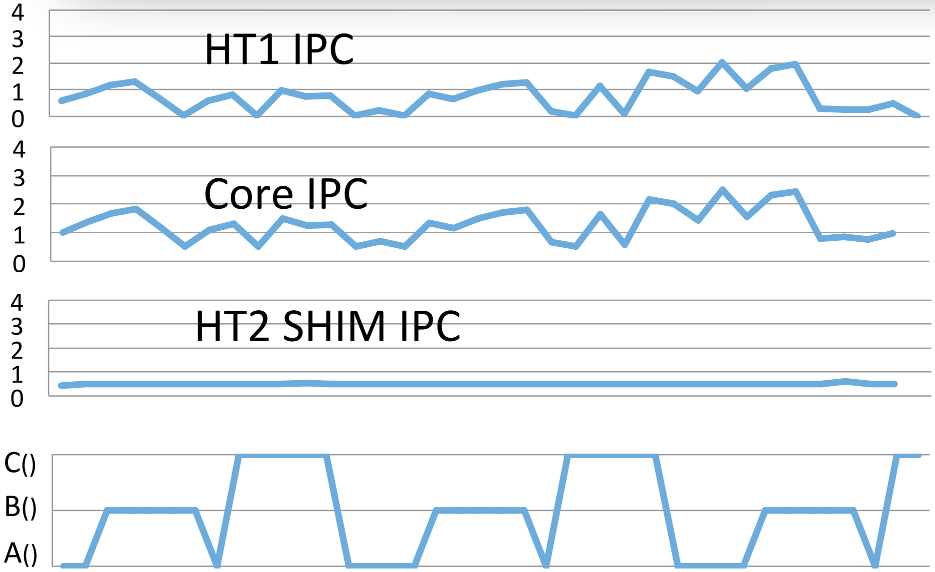


HT1 IPC = Core IPC – HT2 SHIM IPC

Correlate hardware & software events

```
while (true):
   for counter in HT2 SHIM, Core, cycles:
       buf[i++] = readCounter(counter);
   tid = thread on HT1
   buf[i++] = tid.method;
```

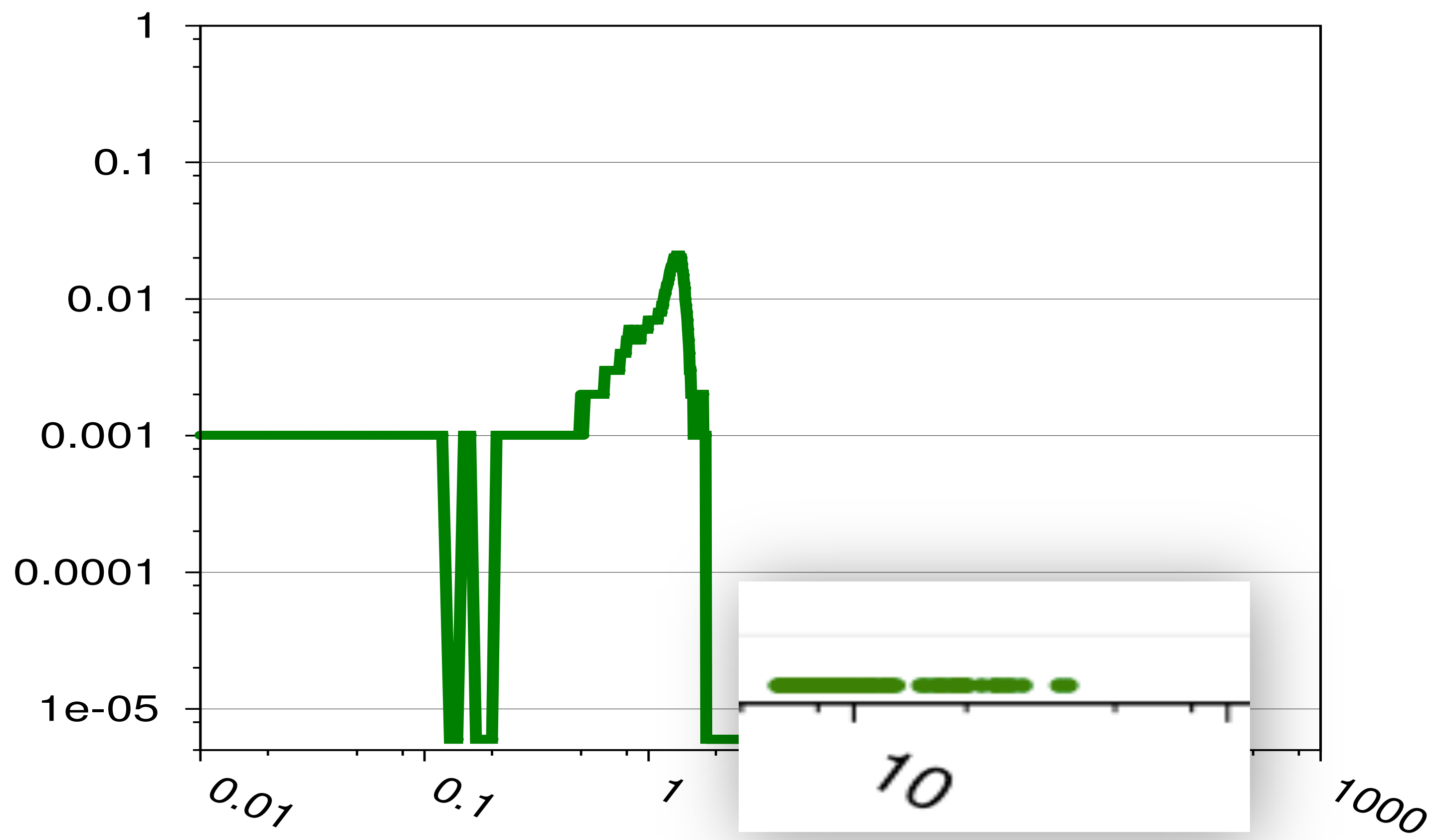




Fidelity

Raw samples

samples
scale)

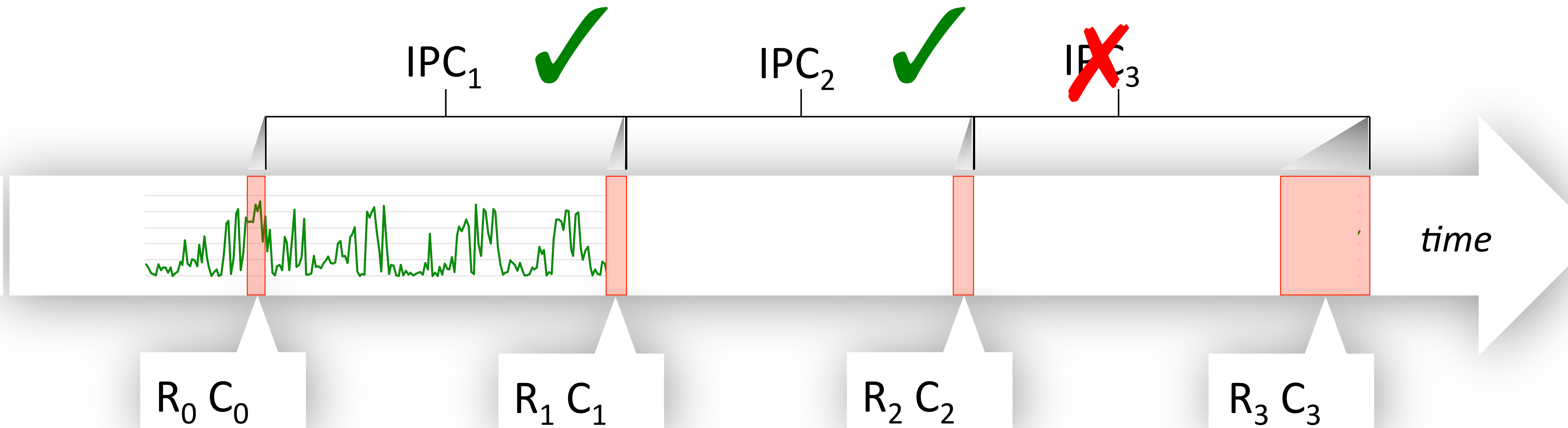


IPC (log scale)

Problem: samples are not atomic

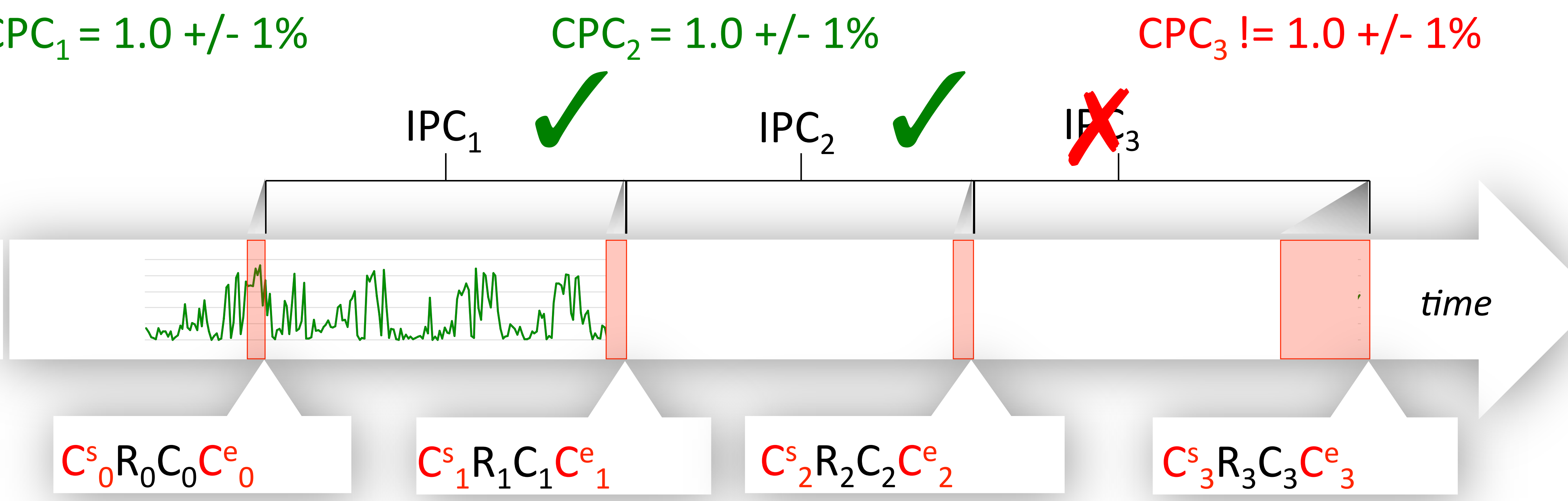
Counters C: cycles R: retired instructions

$IPC = (R_t - R_{t-1}) / (C_t - C_{t-1})$

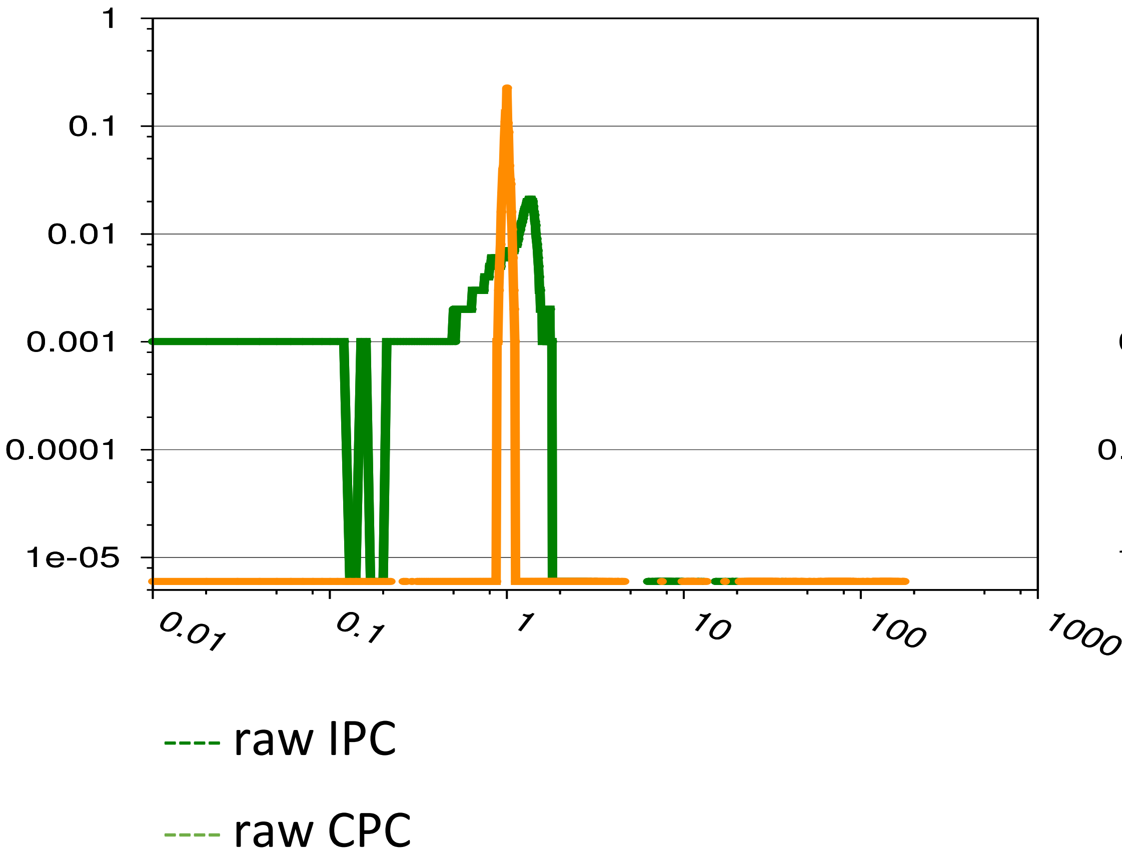


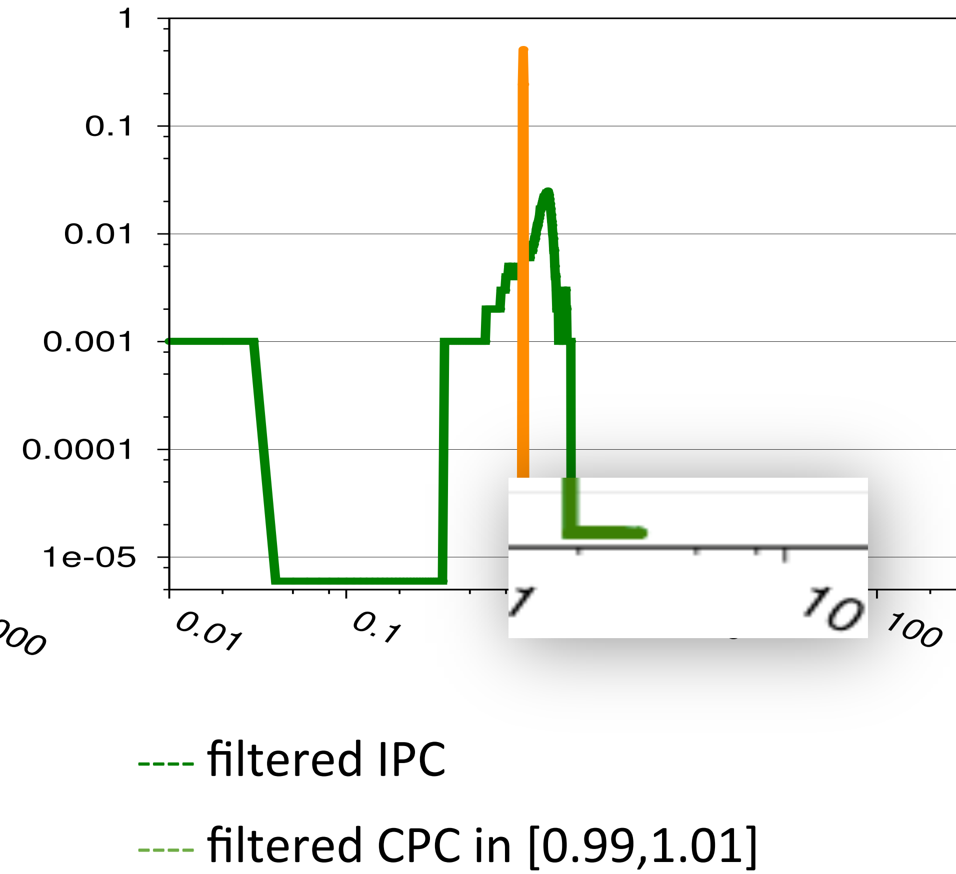
Solution: use clock as ground truth

$CPC = (C^e_t - C^e_{t-1}) / (C^s_t - C^s_{t-1})$ *this should be 1!*

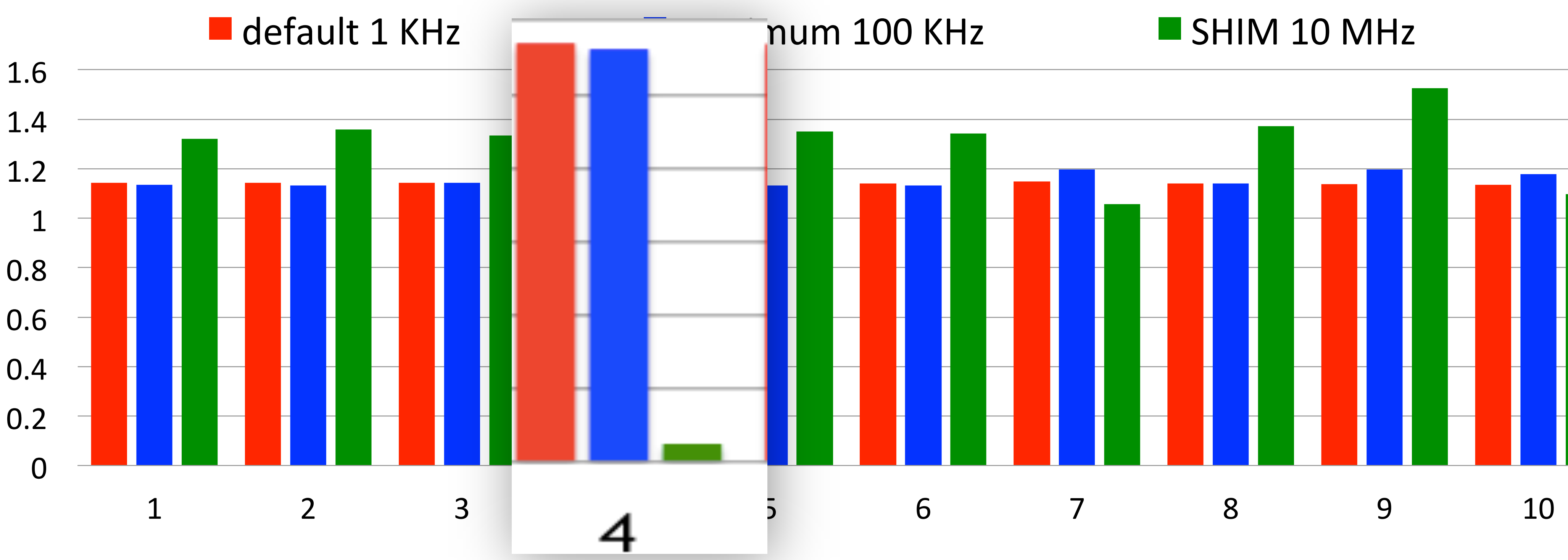


Filtering Lusearch IPC samples





PC of individual methods in Lucene

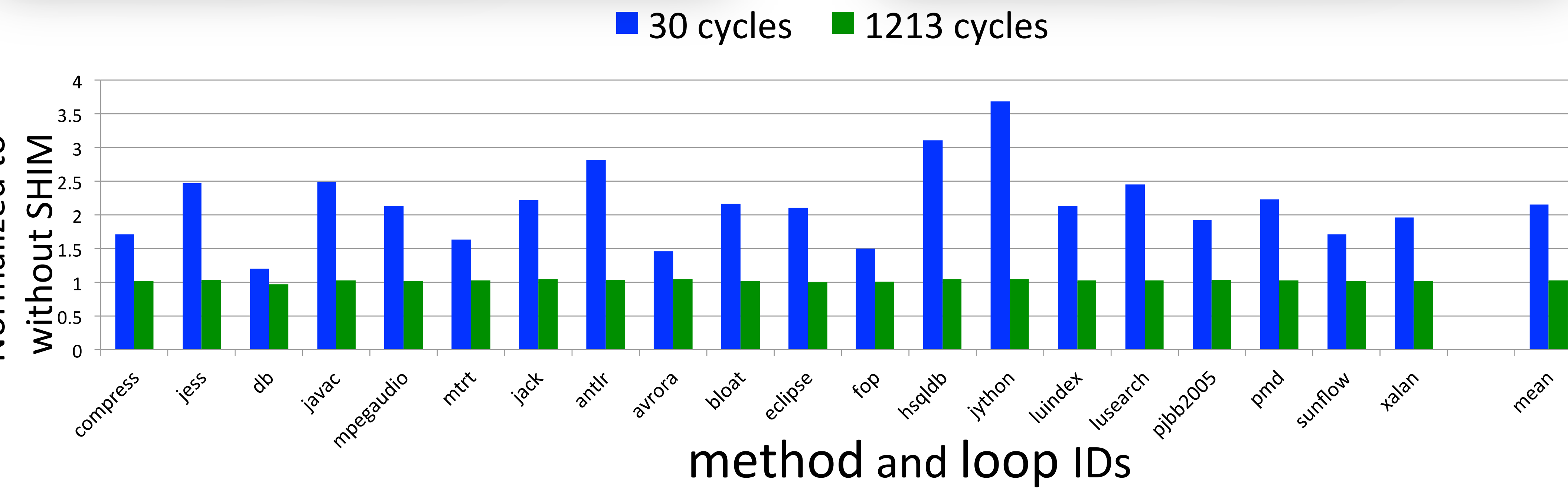


top 10 methods (74% total execution time)

Overheads from other core

113MHz: 3+ orders of magnitude over interrupt 'maximum'

3MHz: 1+ order of magnitude over interrupt 'maximum'



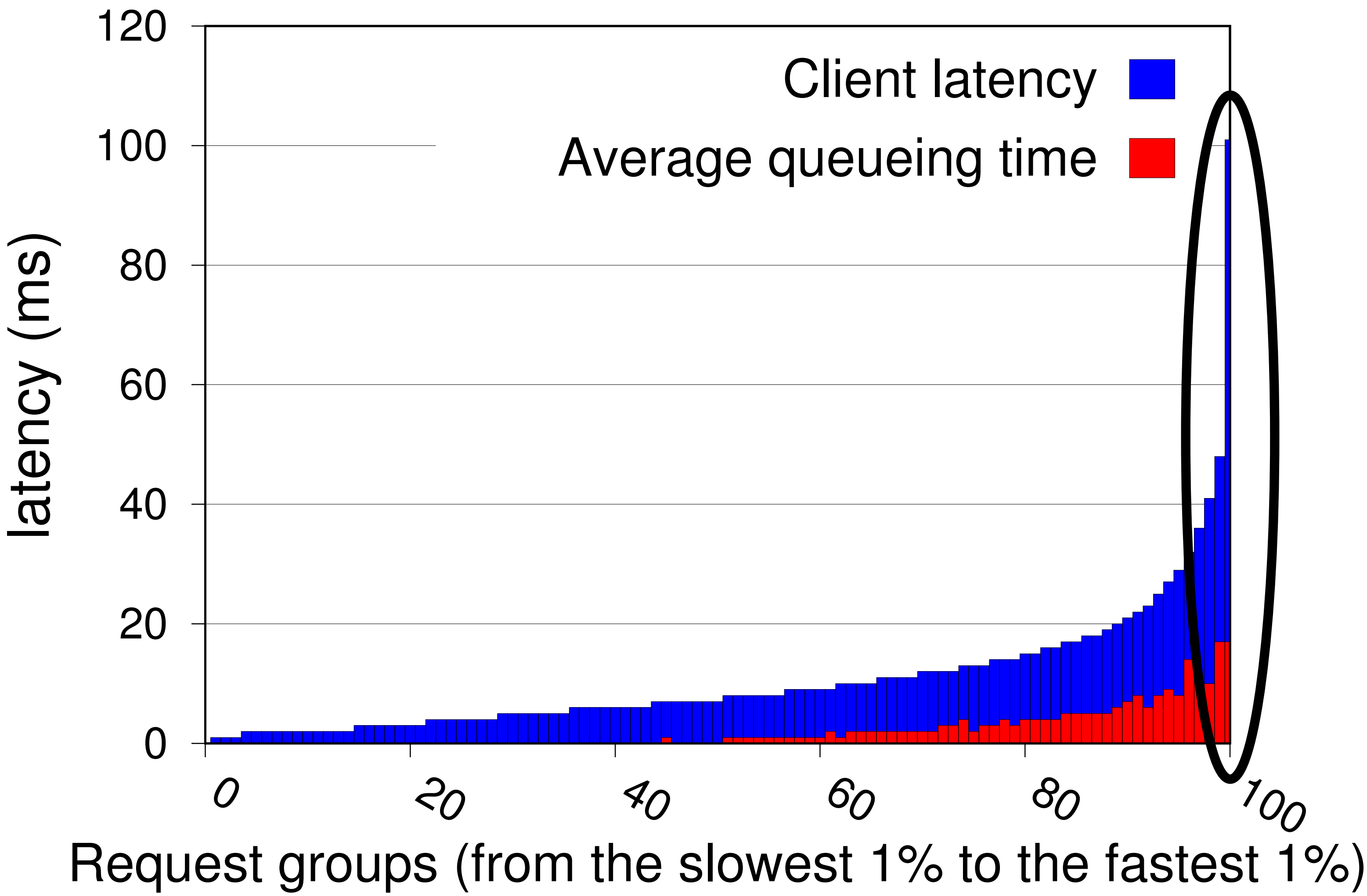
Overheads from write invalidations

Understanding Tail Latency

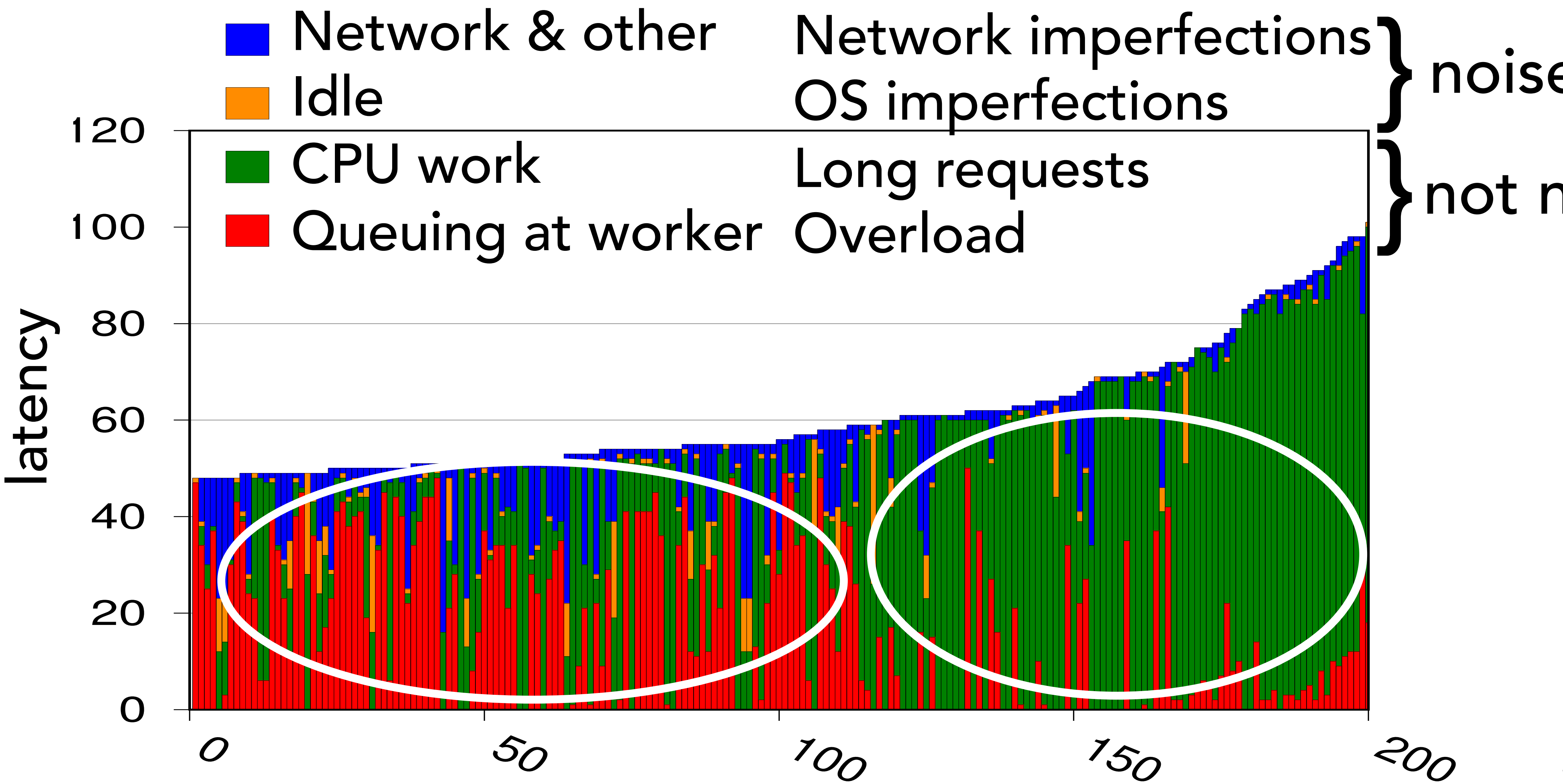
SHIM signals

- Requests
  - thread ids
  - request id – configure
  - time stamps, PC
- System threads
  - thread ids
  - time stamp, PC

All requests



The Tail Longest 200 requests



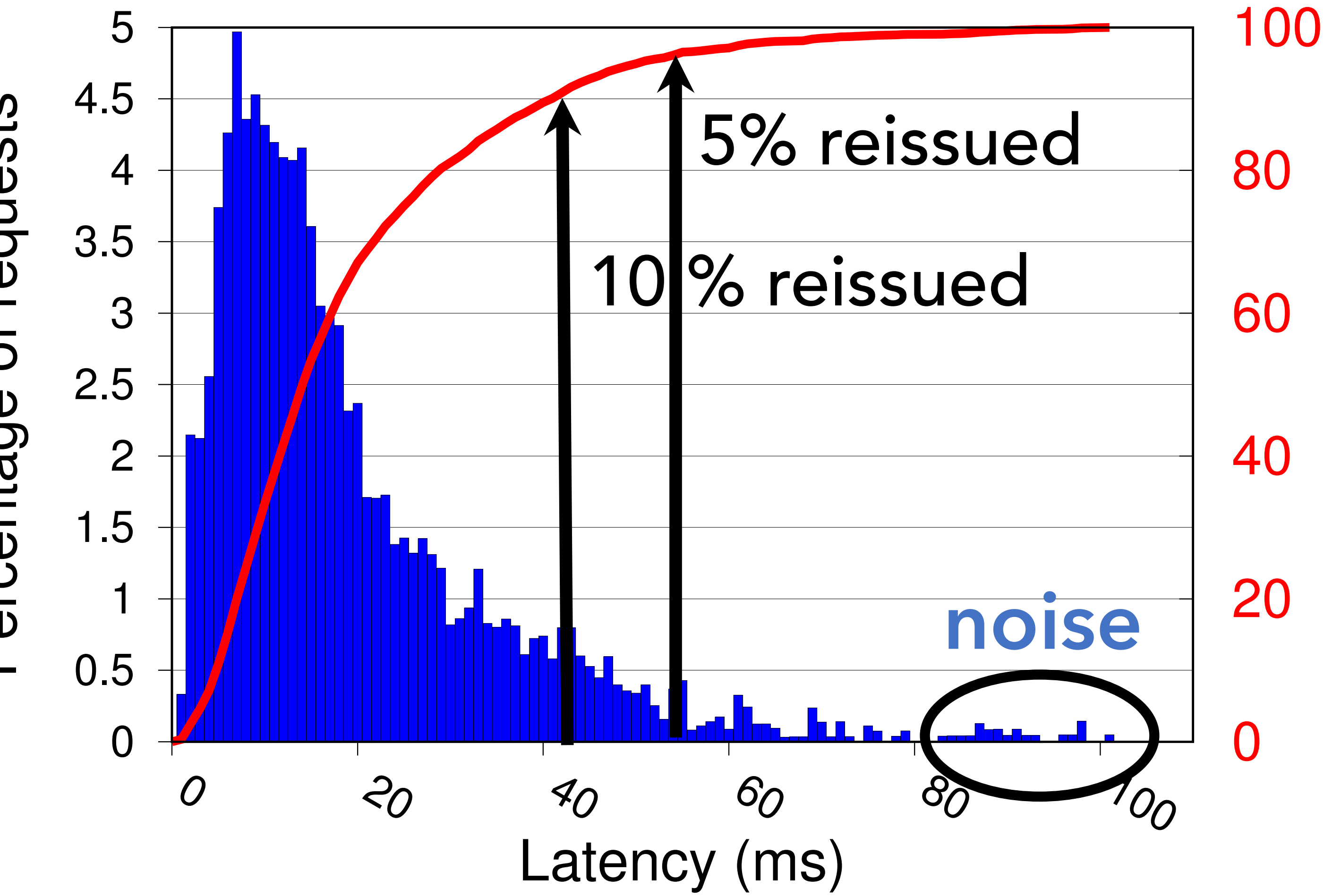
Insight

Long requests reveal themselves

Regardless of the cause

Noise   Replicate & reissue

The Tail at Scale, Dean & Barroso, CACM'13



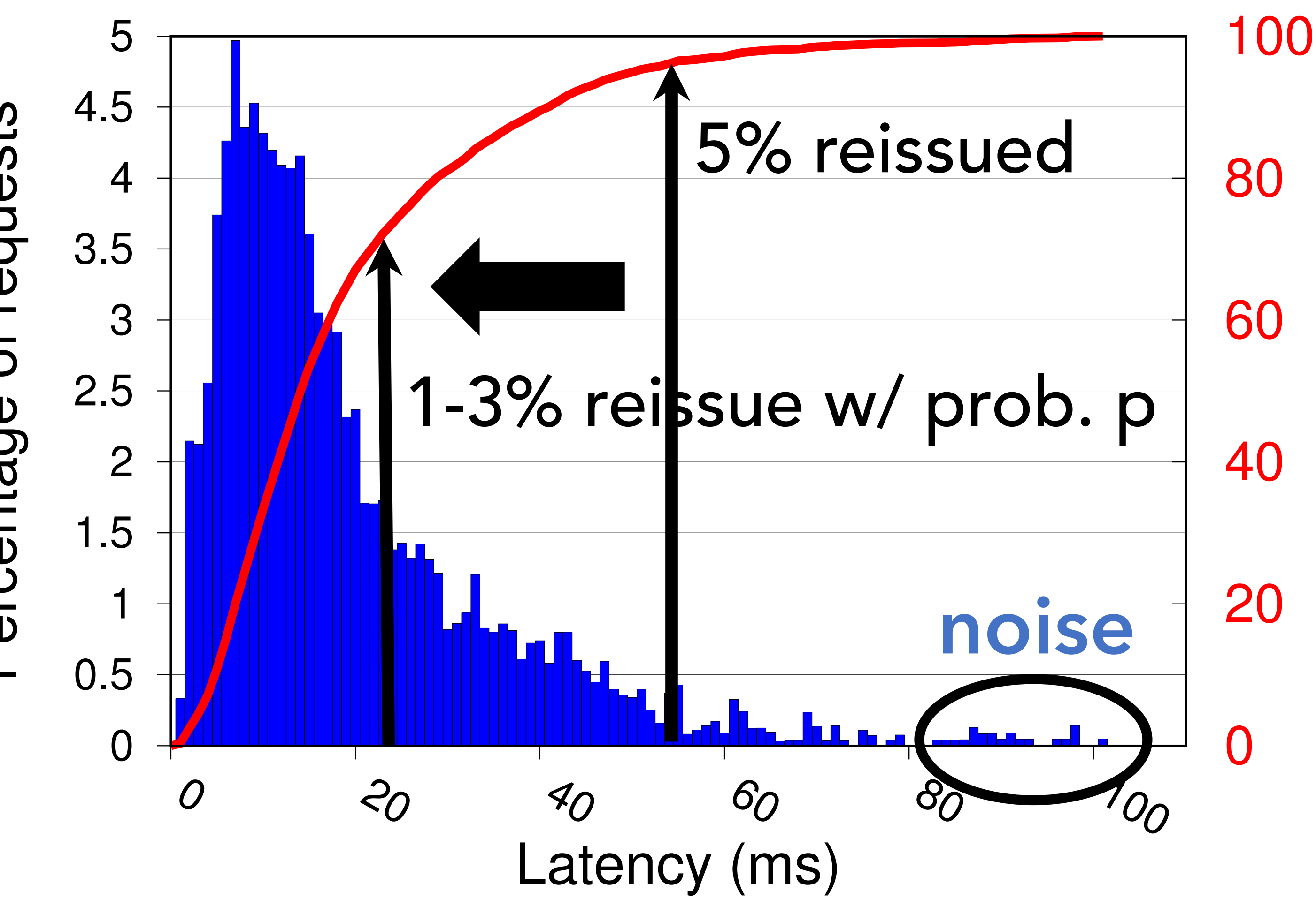
All requests?

CFD for cost & pote

Fixed issue time

Probabilistic reissue

Optimal Reissue Policies for Reducing Tail Latencies, Kaler, He, & Elnickety , SPAA'17

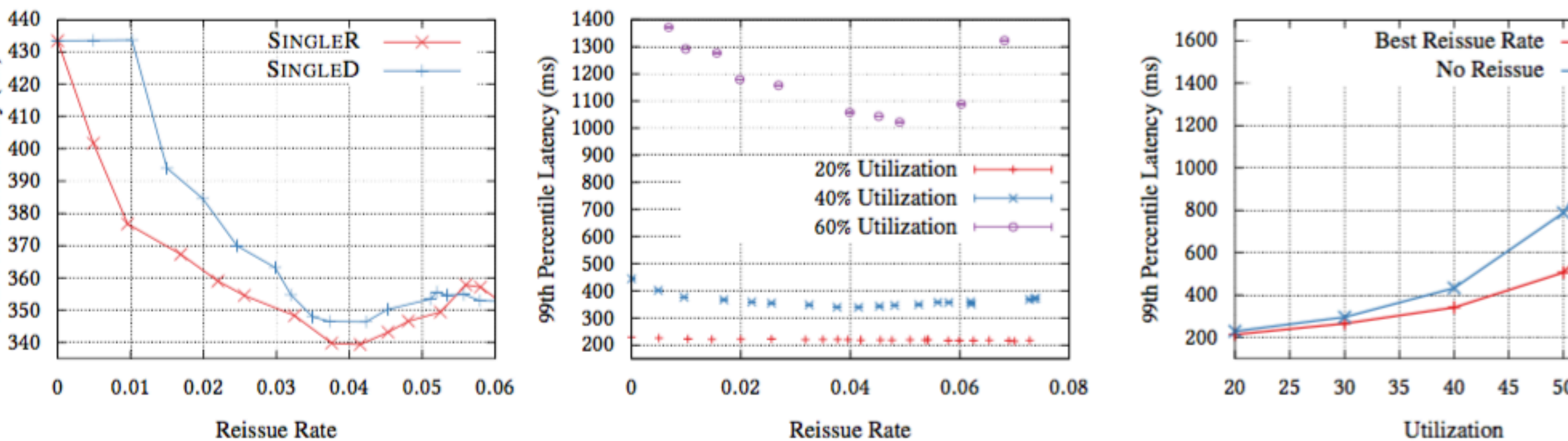


Adding randomness to reissue makes *one* earli reissue time d (vs n) opt

Probability is proportio reissue budget & noise

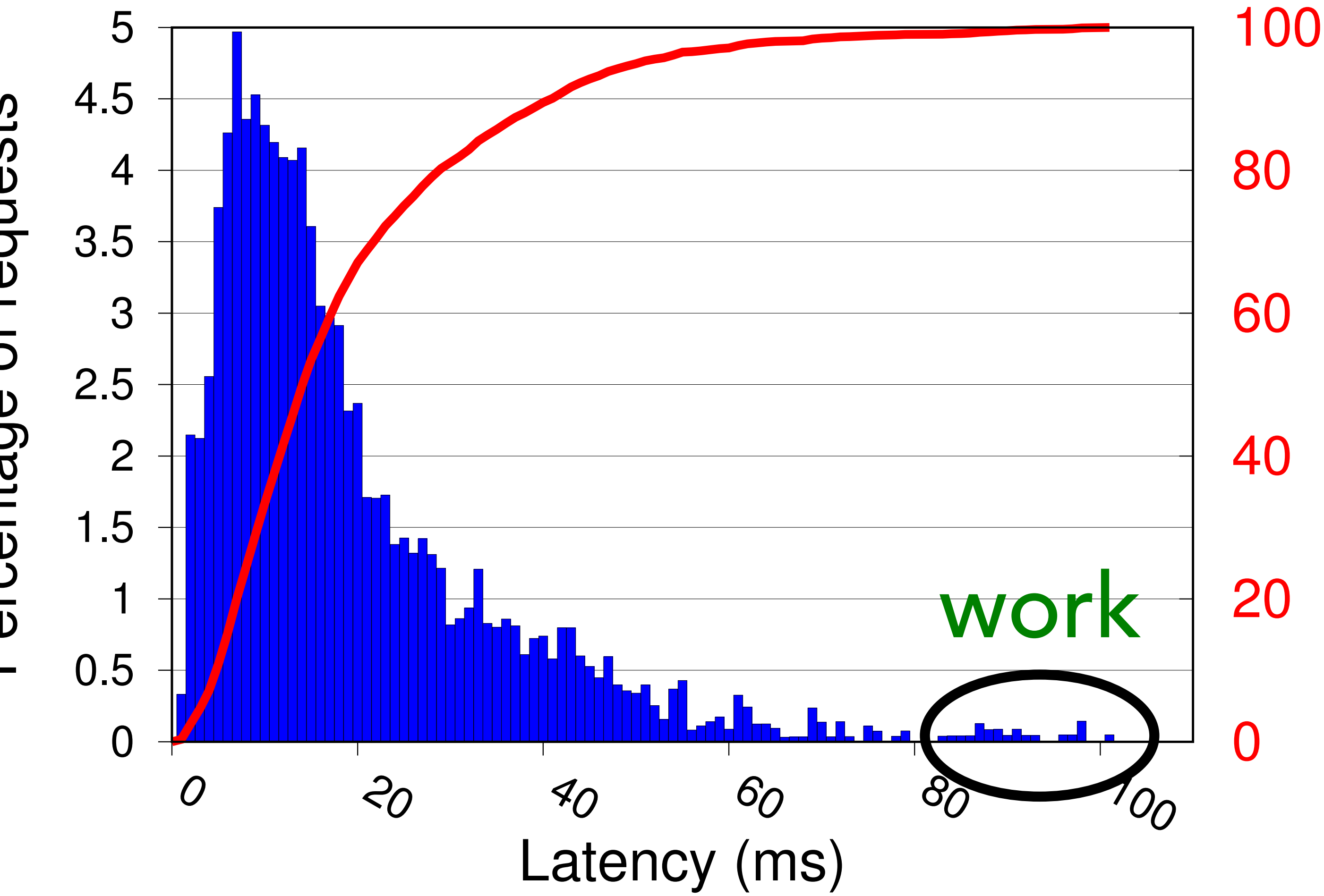
Single R Probabilistic reissue

Optimal Reissue Policies for Reducing Tail Latencies, Kaler, He, & Elnickety , SPAA'17



(a) SINGLER vs SINGLED (b) Latency vs Reissue Rate (c) Best Latency vs Utilizati

Work   Speed up the tail *efficiently*



Judicious parallelism
[ASPLOS'15]

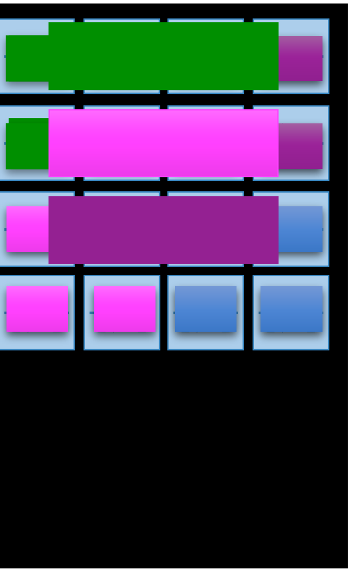
DVFS faster on the t
[DISC'14, MICRO'17]

Asymmetric multicor
[DISC'14, MICRO'17]

Work Parallelism

Parallelism historically for **throughput**

Idea Parallelism for **tail latency**



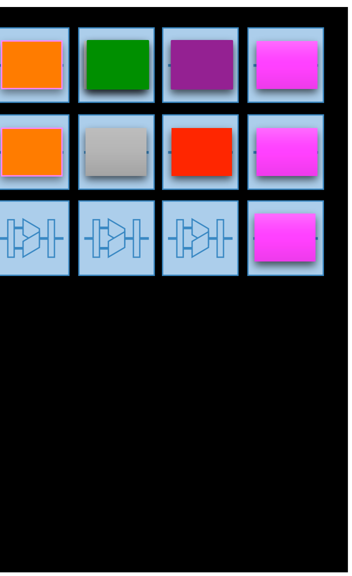
Queuing theory

Optimizing average latency maximizes throughpu

But not the tail!

Shortening the tail reduces queuing latency

Few-to-Many Dynamic Parallelism [ASPLOS'15]



Parallelism historically for **throughput**

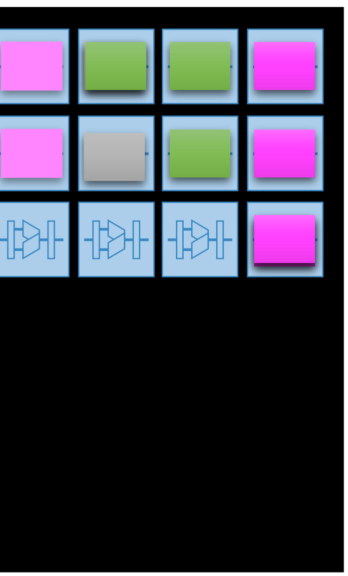
Idea Parallelism for **tail latency**

Insight Long requests reveal themselves

Approach Incrementally add parallelism to long requests – the tail – based on request progress & load

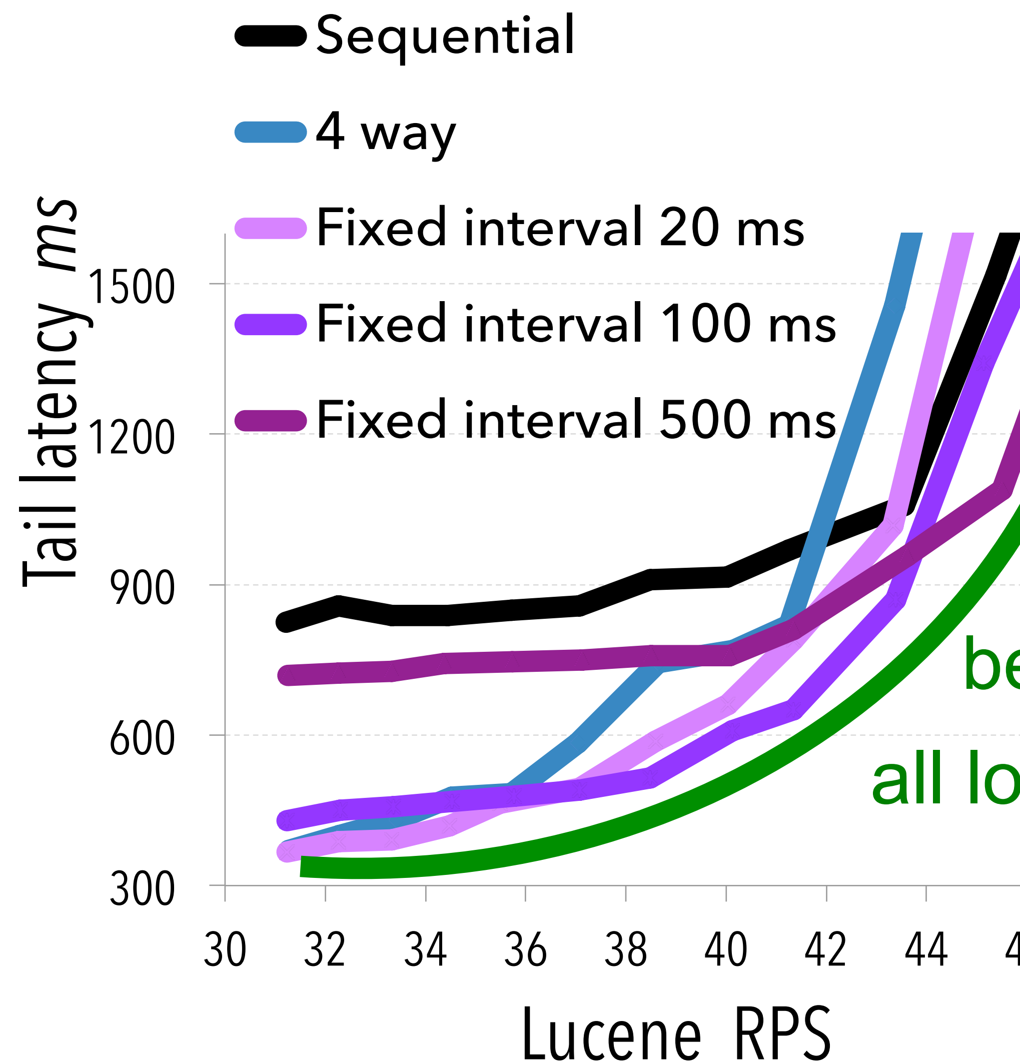
Few to Many at fixed delay d

Add thread every d *ms*



Long delay good at high load

Short delay good at low load



Offline

Profiles

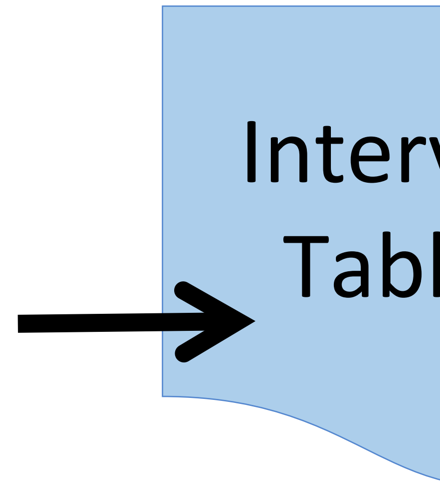
Sequential & parallel demand distribution

Efficiency of parallelism

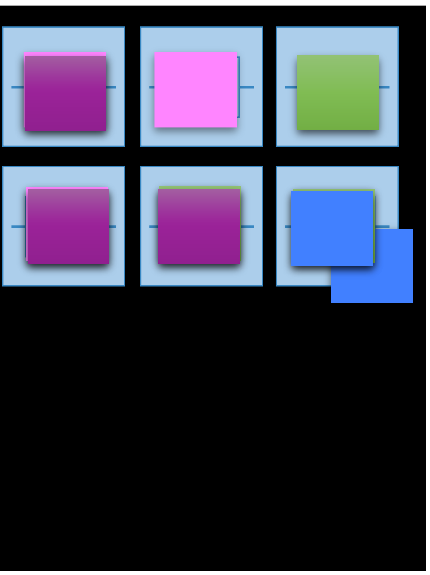
Choose maximum target parallelism

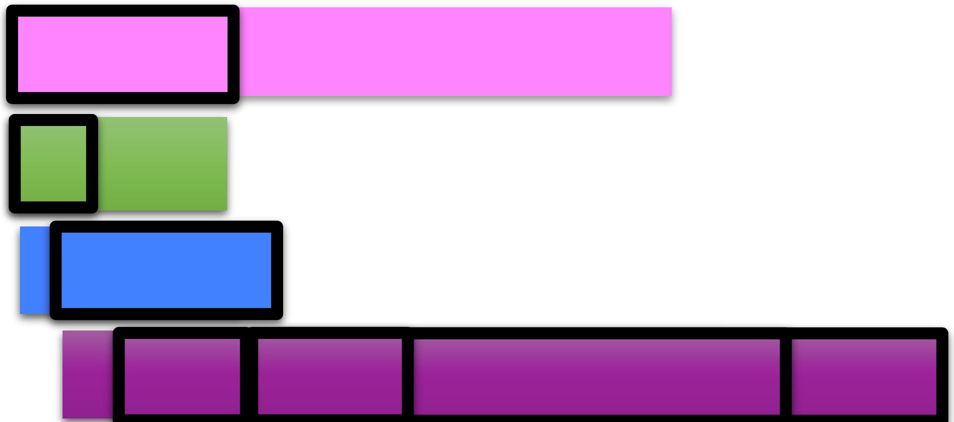
Utilize available hardware resources

Exhaustively explore parallelism given set of time intervals *t* & load find best tail latency & parallelis



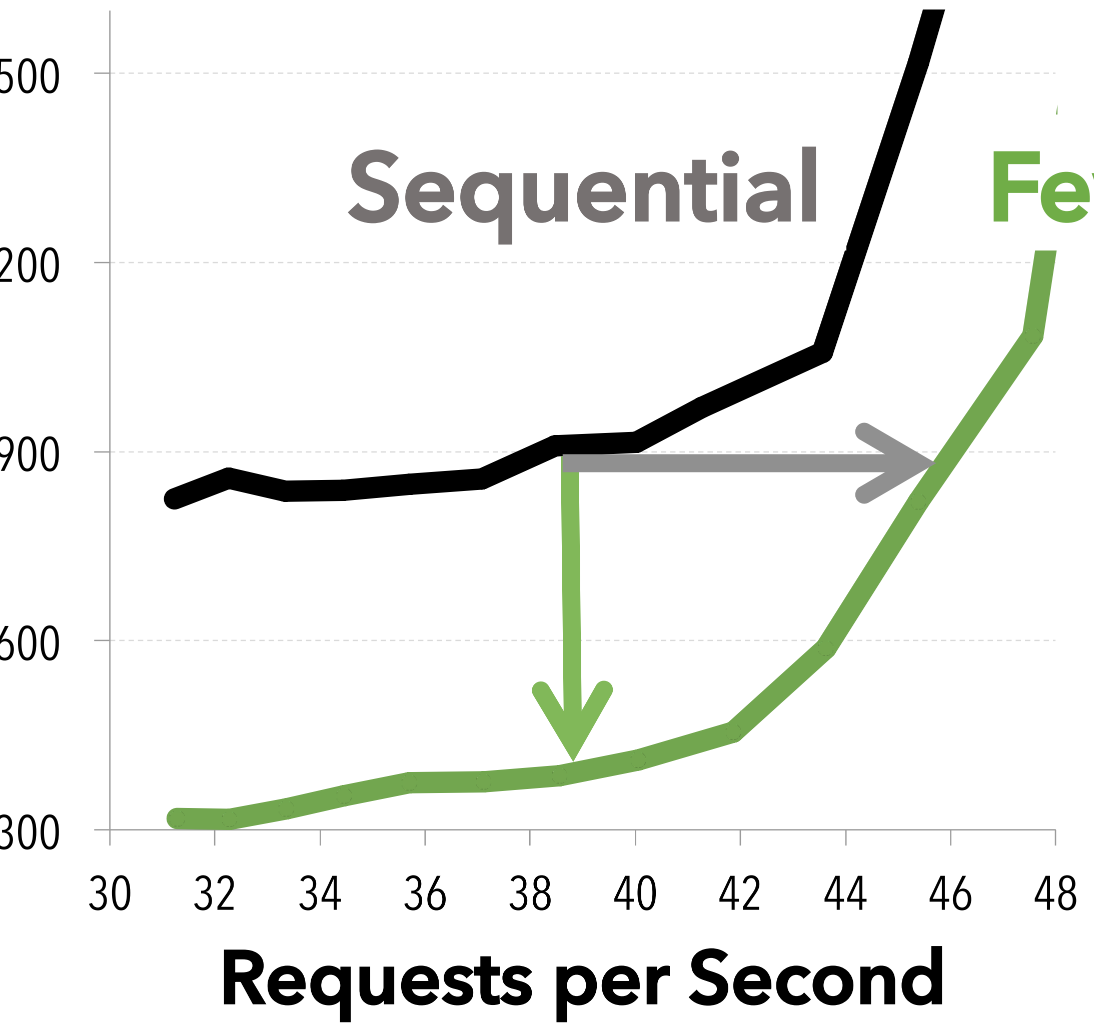
Online self scheduling





|requests|	$Interval_0$ = 0	$Interval_{1,2}$ = 50, 10
≤ 2	@ 0 parallelism = 3	
3	@ 0 parallelism = 1	@ 50, parallelism =
4 - 6	@ 50 parallelism = 1	@ 100, parallelism =
≥ 7	@ exit parallelism = 1	@ 100, parallelism =

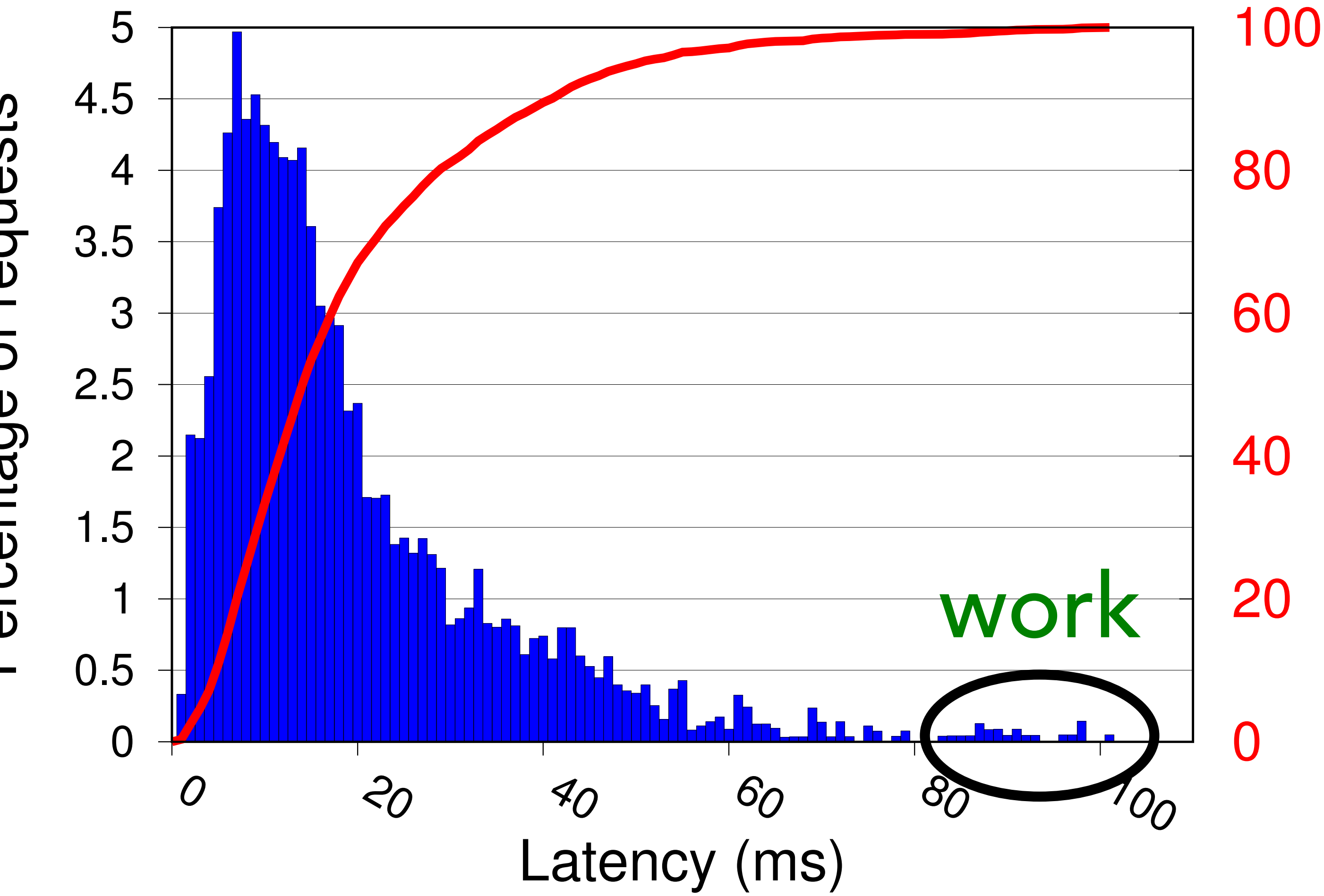
Evaluation 2x8 64 bit 2.3 GHz Xeon, 64 GB



21% fewer servers

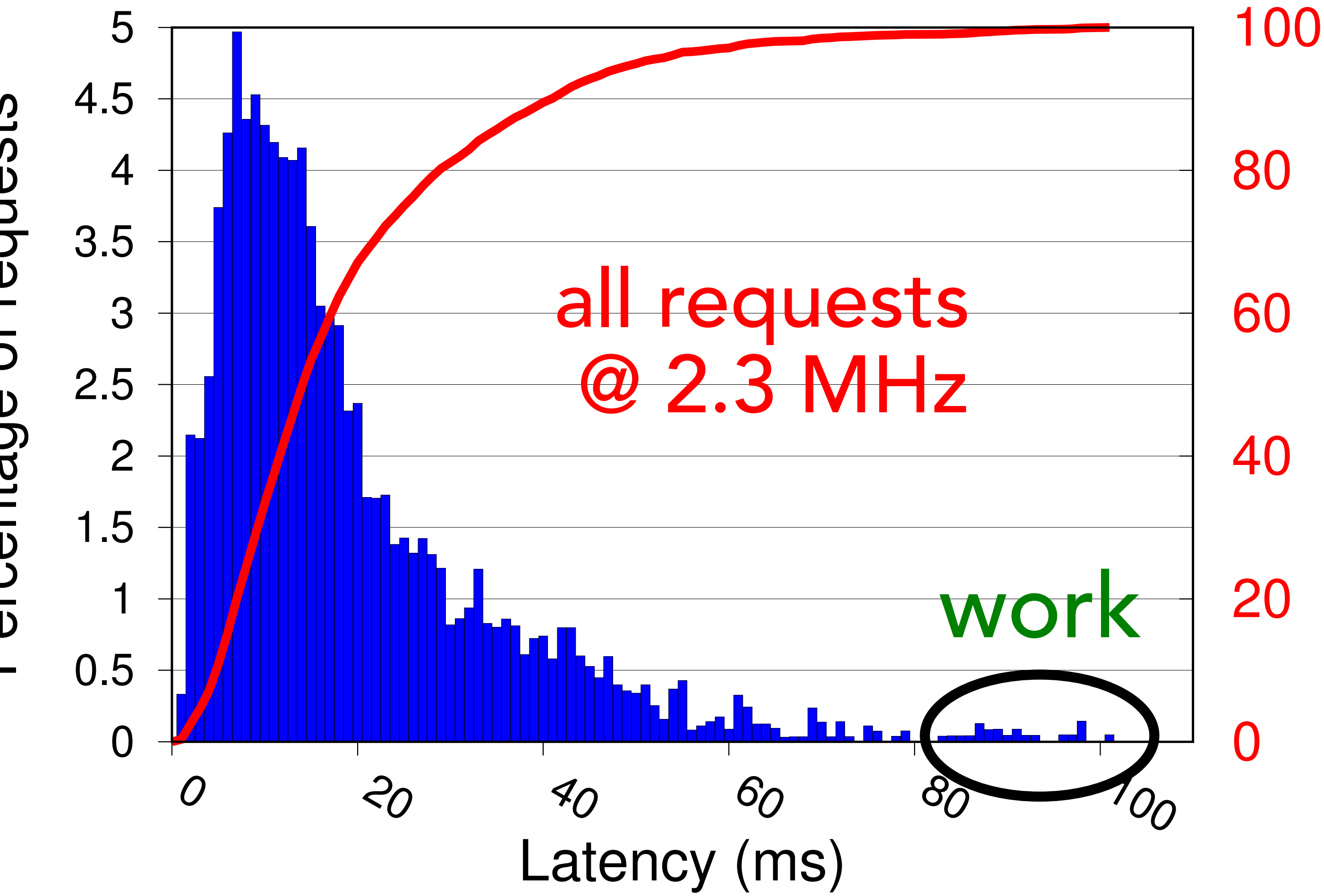
or reduce tail by 28

Work speed up the tail *efficiently*



Judicious parallelis
[ASPLOS'15] ✔

Work speed up the tail *efficiently*



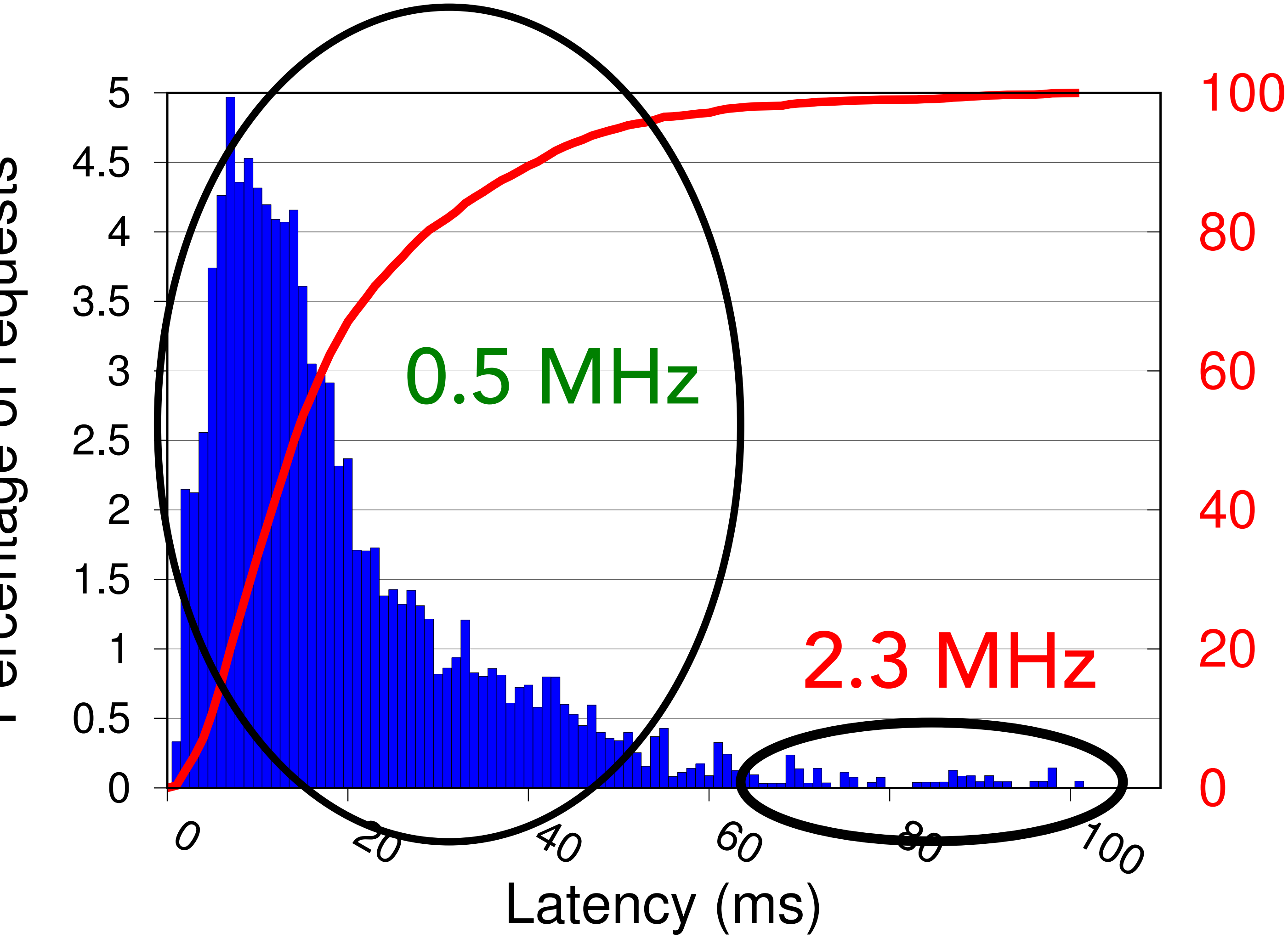
DVFS faster on the

[DISC'14, MICRO'17]

Asymmetric multico

(AMP) [DISC'14, MICRO'17]

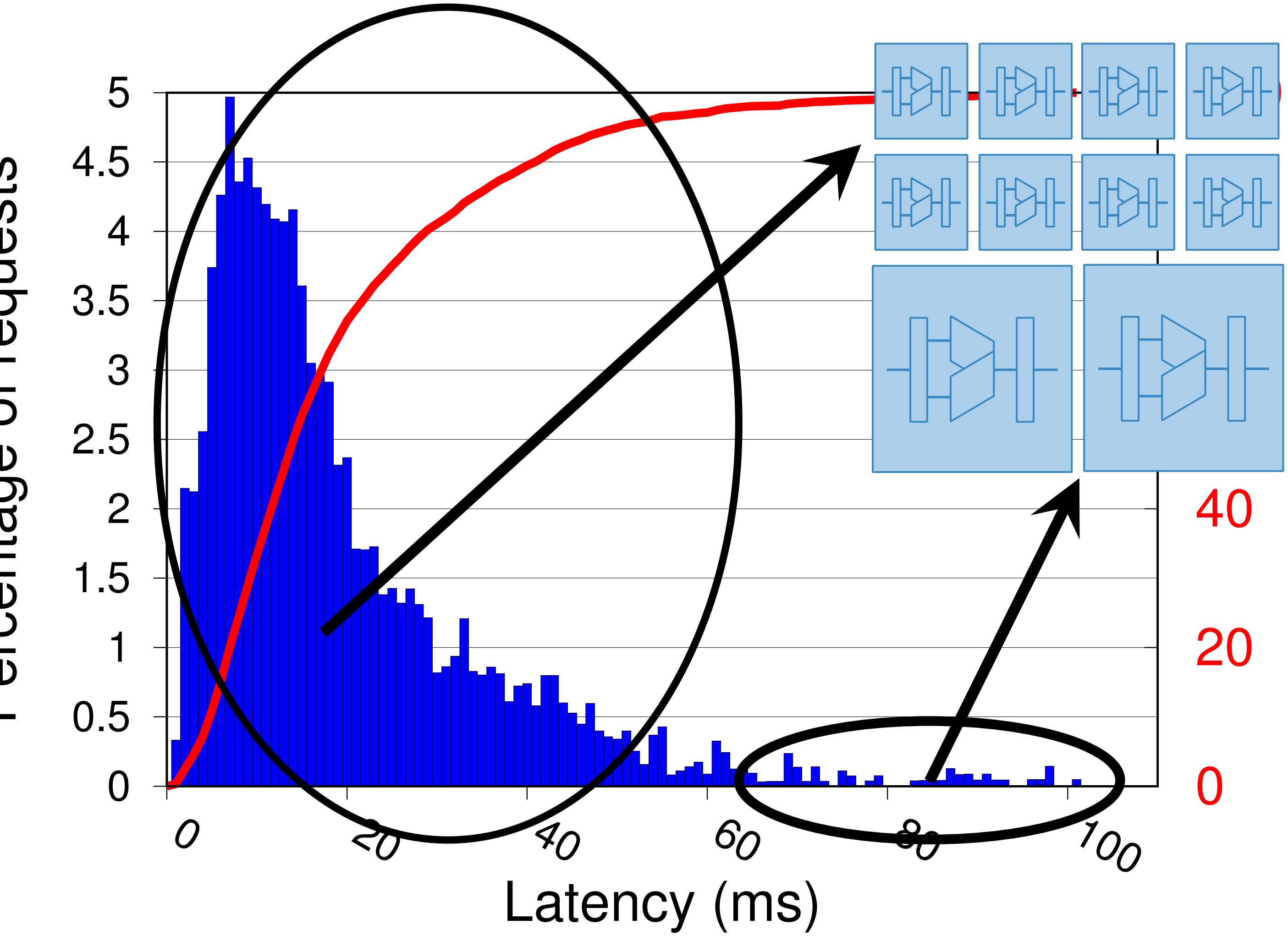
Speed up the tail *efficiently*



DVFS faster on the
[DISC'14, MICRO'17]

+ available in servers t

Speed up the tail *efficiently*



DVFS faster on the
[MICRO'17]

\+ available in servers t

Asymmetric multico
(AMP) [DISC'14, MICRO'17]

\+ much more energy e

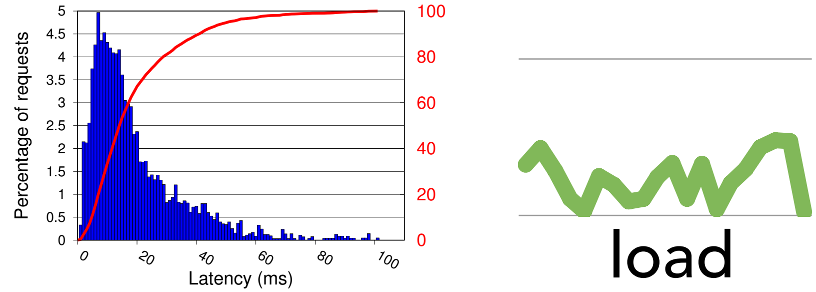
\+ hyper-threading is a

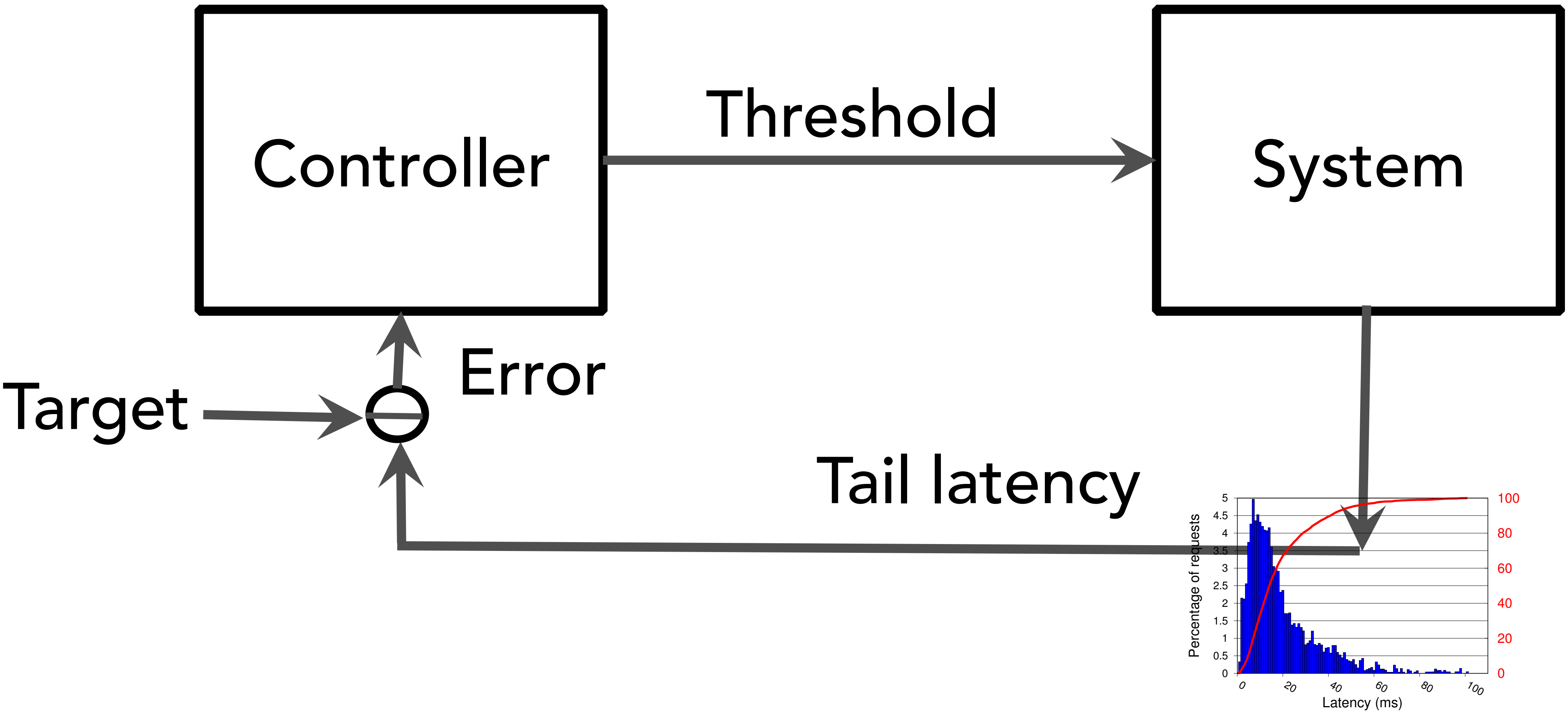
\- core competition

Adaptive Slow to Fast Framework

- Slow to fast migration is optimal [ICAC'13]
- Goal
  - Minimize energy consumption and satisfy a tail latency target
- Challenges
  - When to migrate?
  - What if the core speed is not available?
- Insight
  - Use **big core** just enough $th+(l_{99}-th)/sp \leq target$
  - Migrate oldest first and migrate early under load!

Controller design





Policies

All-cores

Pegasus adjust all-core frequencies for load

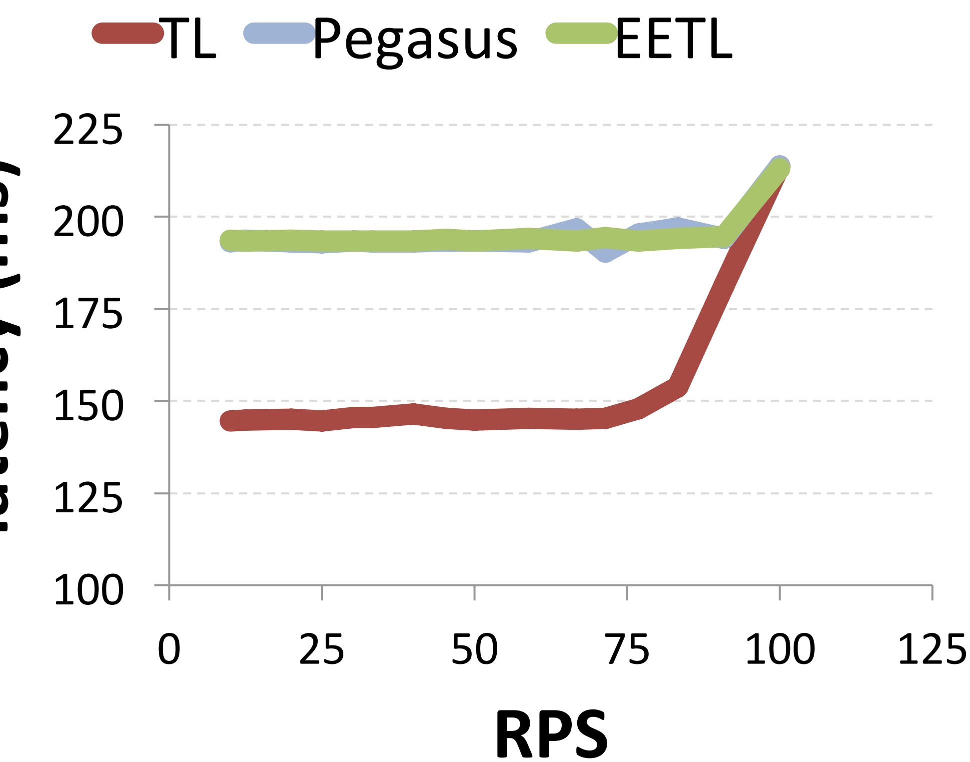
[Towards Energy Proportional…, Google & Stanford, ISCA'14]

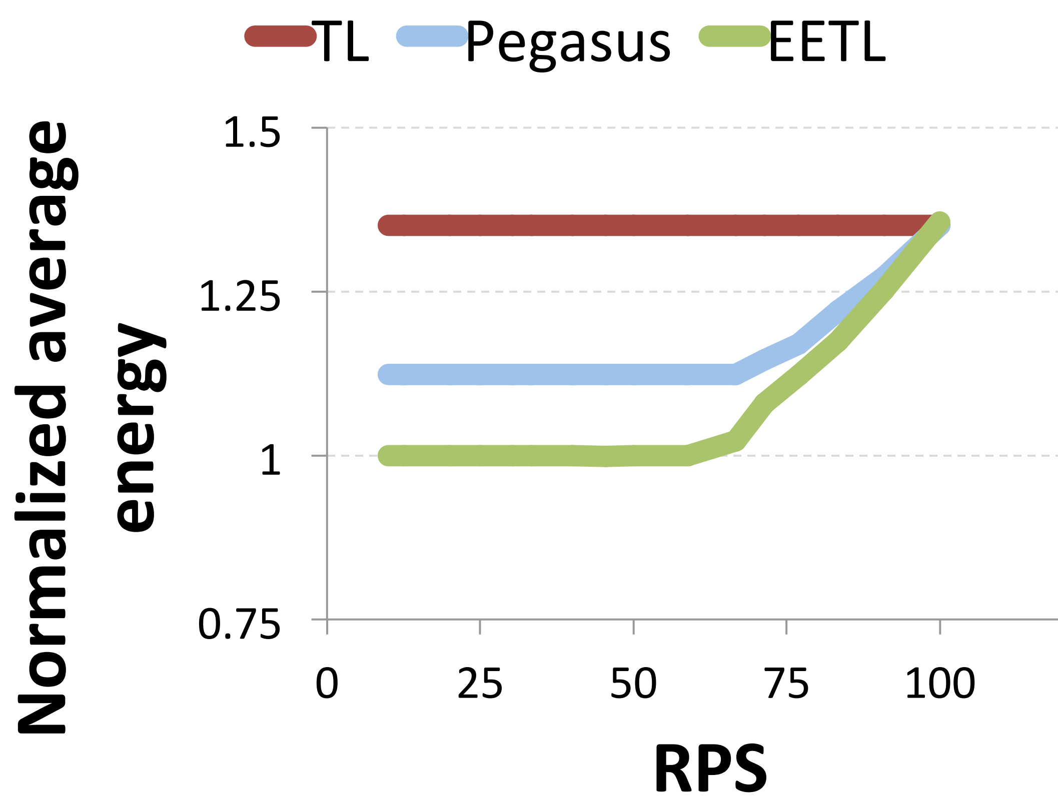
Per-core approaches

TL minimize tail latency using 0 threshold

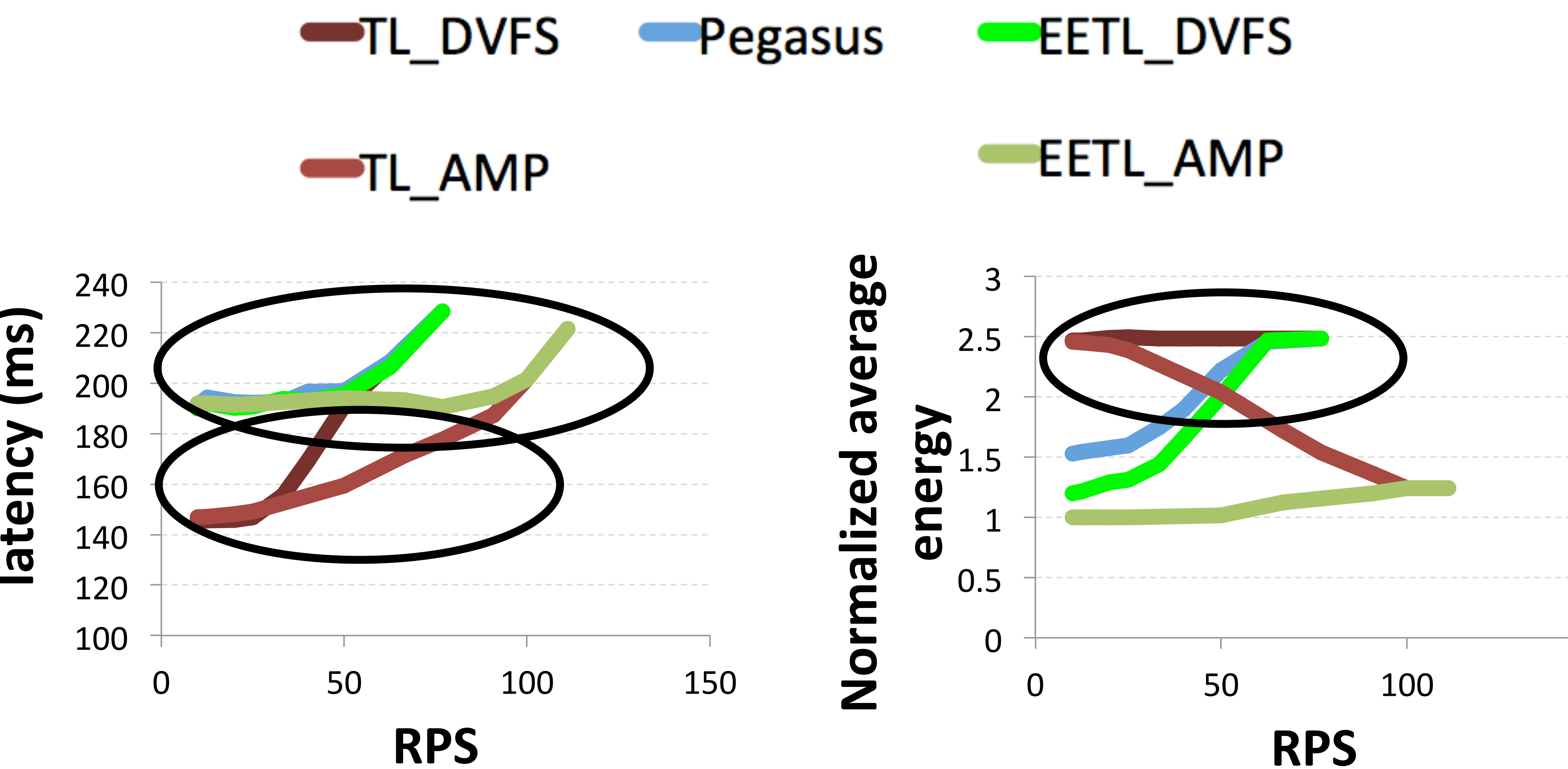
EETL energy efficient with target tail latency

Lucene with DVFS on Broadwell





Lucene on emulated AMP and DVFS



il Latency

BOTH !

Efficiency

Google Clou

Efficiency at scale for interactive workloads

- Diagnosing the tail with continuous profiling

Noise	replicate, systems are not perfect
Queuing	not today!
Work	judicious use of resources on long requests

- Request latency CDF is a powerful tool
- Tail efficiency ≠ average or throughput
- Hardware heterogeneity

Thank you

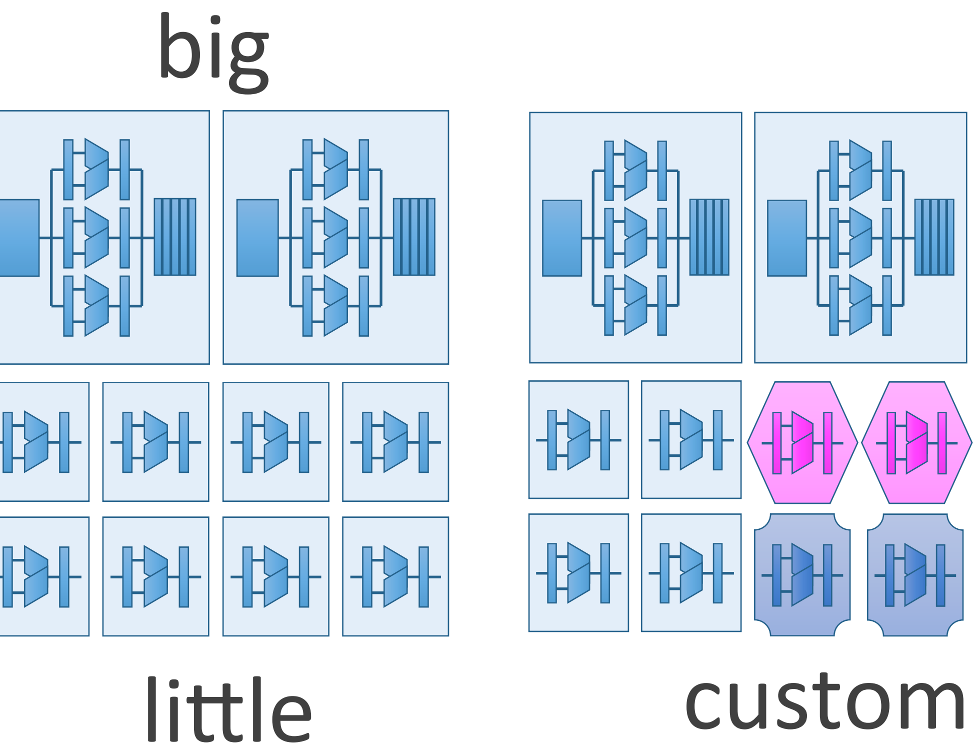
Requirements pull for heterogeneity!

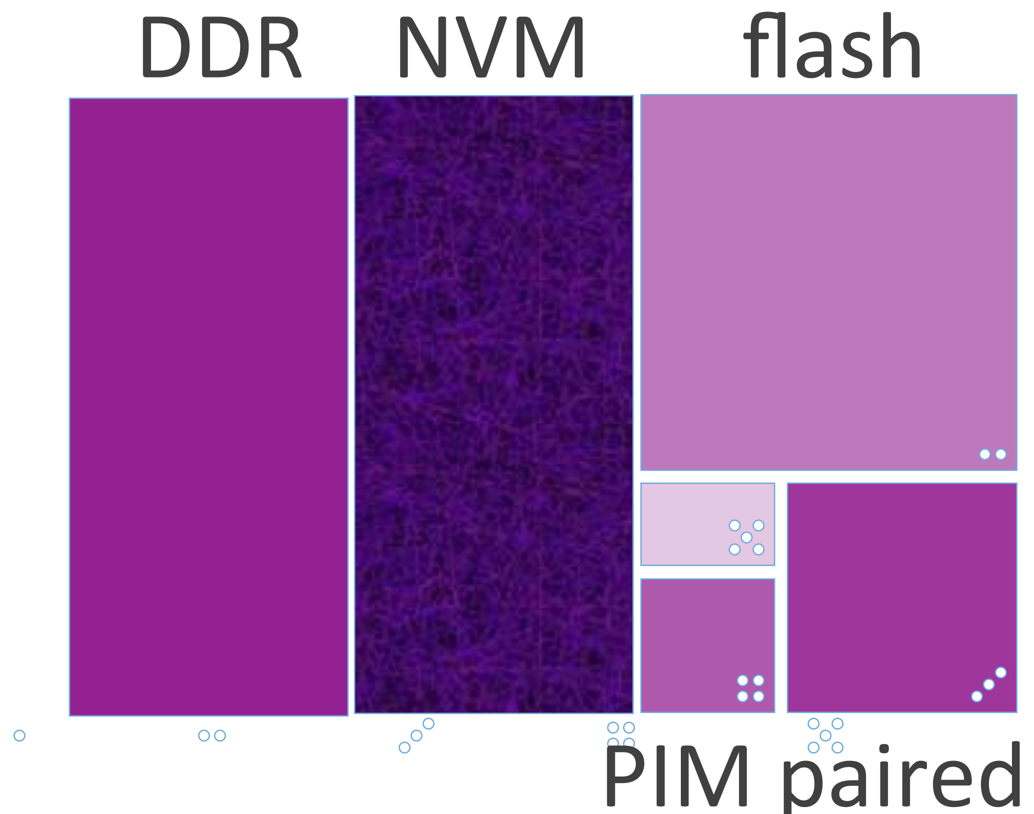
[DISC'14, ICAC'13, submission]

Heterogeneous hardware dominates homogeneous hardware for throughput, performance, and energy with a fixed power budget & variable request demand

Slow-to-Fast sacrifice average a bit to reduce energy & tail latency

Hardware heterogeneity – opportunity & challen





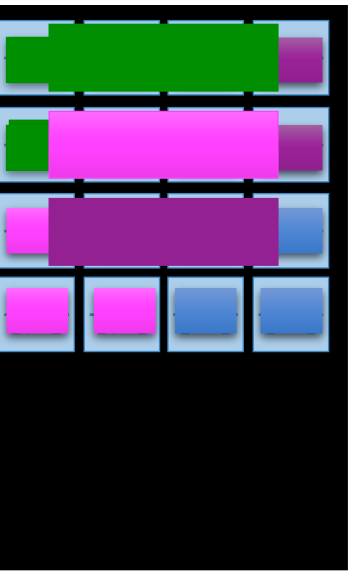
Processors

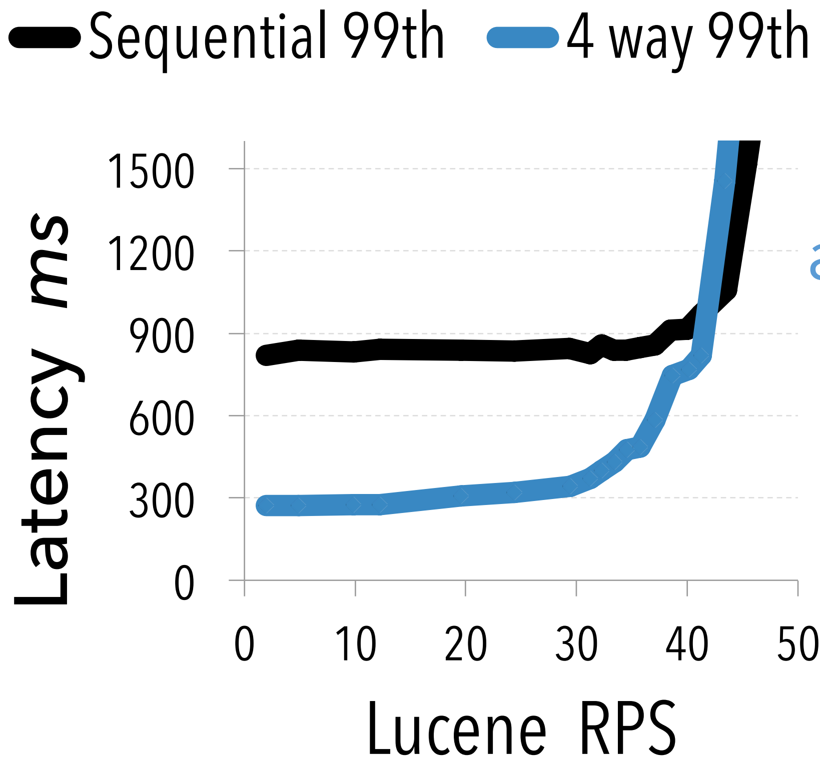
Memory

Parallelism

Parallelism historically for **throughput**

Idea Parallelism for **tail latency**





degrades at high load

improves at low load

Software & hardware

Lucene open source enterprise search Wikipedia English

- 10 GB index of 33 million pages
- 10k queries from Lucene nightly tests

Bing web search with one Index Serving Node (ISN)

- 160 GB web index in SSD, 17 GB cache
- 30k Bing user queries

Hardware 2x8 64 bit 2.3 GHz Xeon, 64 GB Windows

- 15 request servers, 1 core issues requests
- Target parallelism = 24 threads

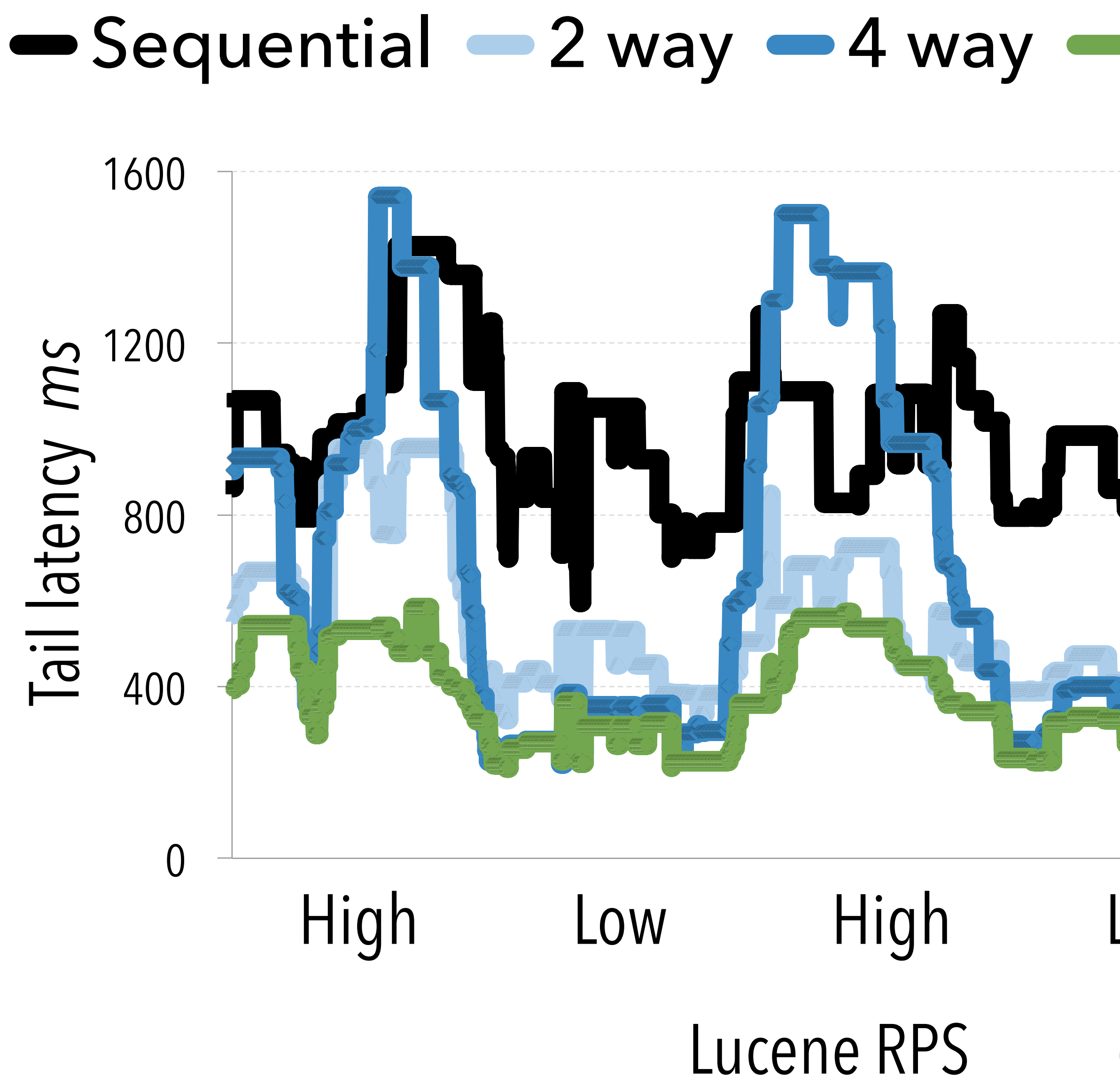
Policies

- **Sequential**
- **N way** single degree of parallelism for each request
- **Adaptive** Select parallelism degree when request starts using system load [EUROSYS'13]
- **Request Clairvoyant** parallelizes long requests by perfect prediction of tail
- **FM** Few to Many incrementally add parallelism

Load variation

Alternate between high & low load

FM adapts to bursts with low variance



Fewer servers: Total Cost of ownership