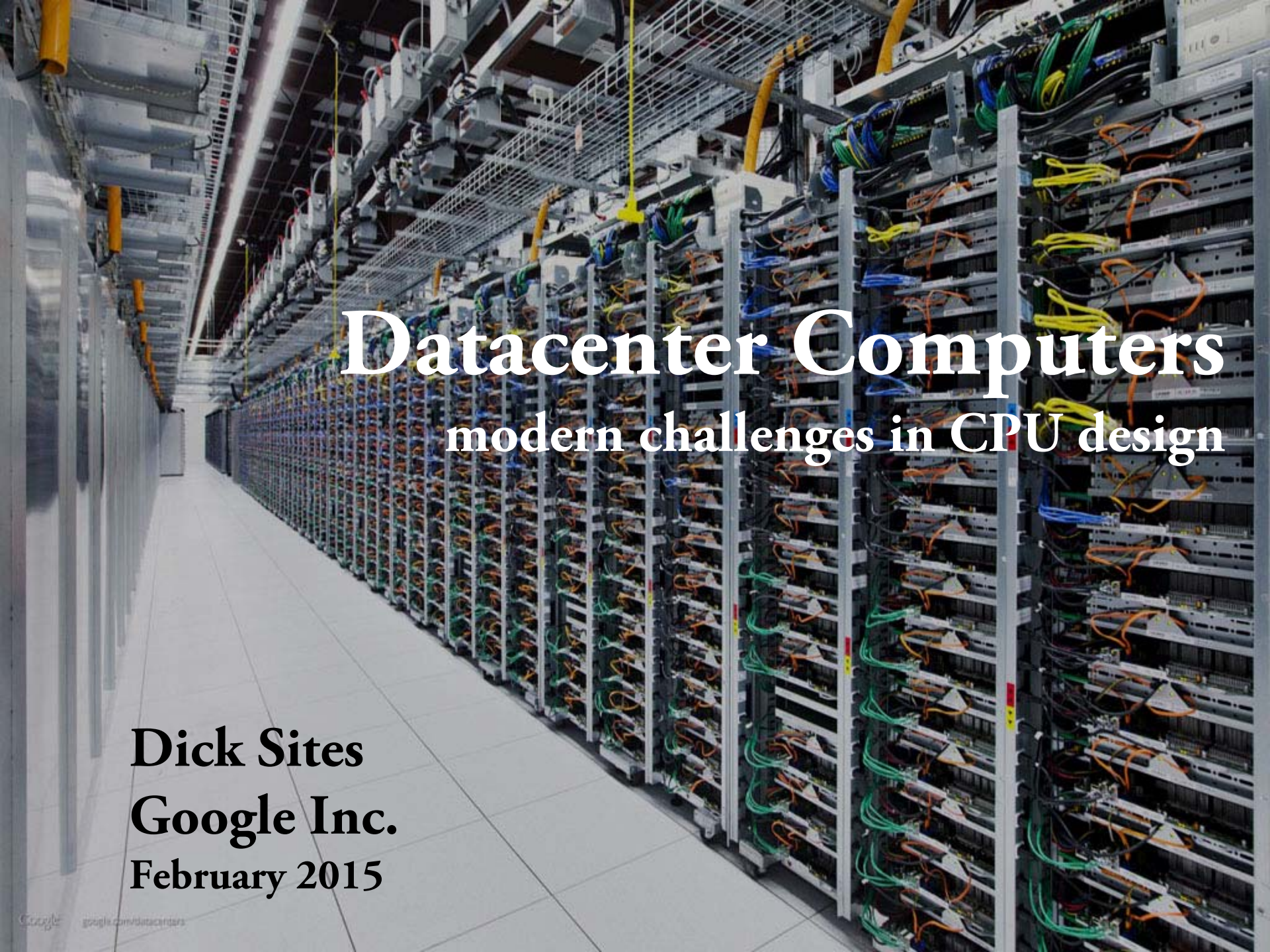


Datacenter Computers

modern challenges in CPU design

Dick Sites
Google Inc.

February 2015

Thesis: Servers and desktops require different design emphasis

Goals

- Draw a vivid picture of a computing environment that may be foreign to your experience so far

- Expose some ongoing research problems

- Perhaps inspire contributions in this area

Analogy (S.A.T. pre-2005)

:

as

:

Analogy (S.A.T. pre-2005)

:

as

:

Datacenter Servers are Different

① Move data: big and small

② Real-time transactions: 1000s per second

③ Isolation between programs

④ Measurement underpinnings

① Move data: big and small

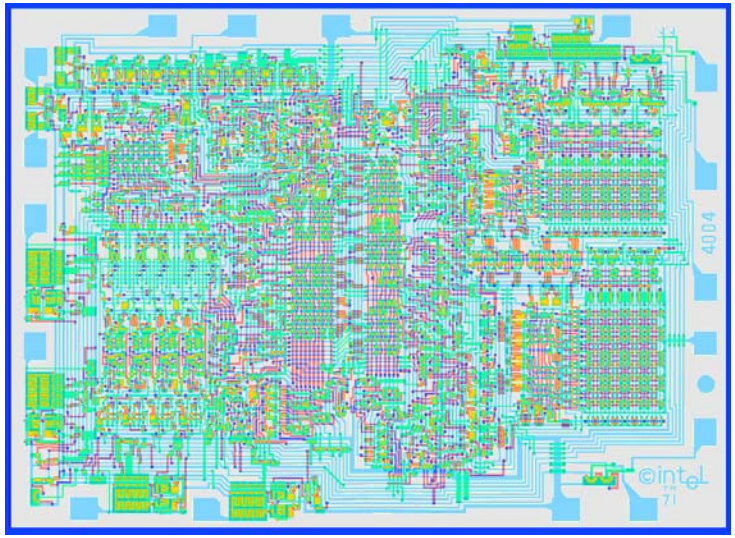
Move data: big and small

- Move *lots* of data
  - Disk to/from RAM
  - Network to/from RAM
  - SSD to/from RAM
  - Within RAM

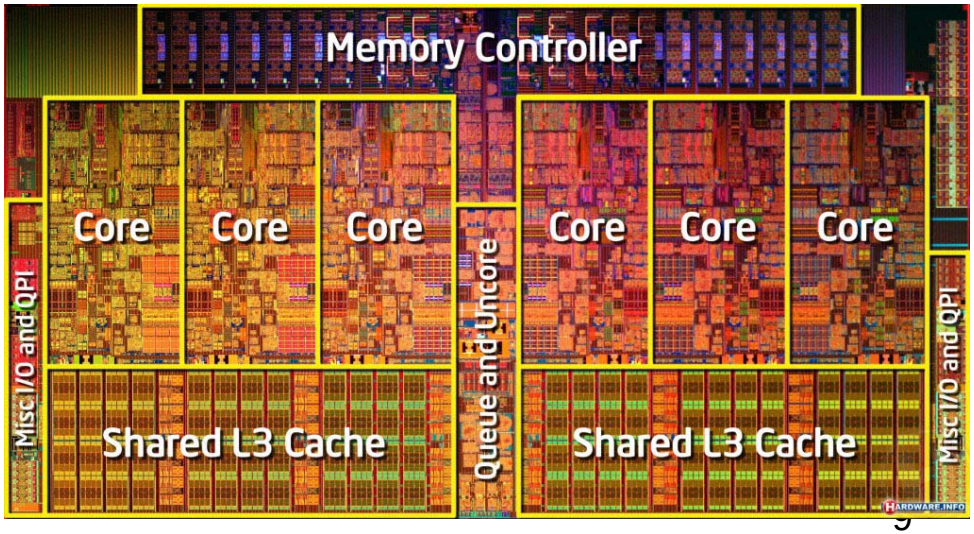
- Bulk data
- *Short* data: variable-length items
- Compress, encrypt, checksum, hash, sort

Lots of memory

- 4004: no memory

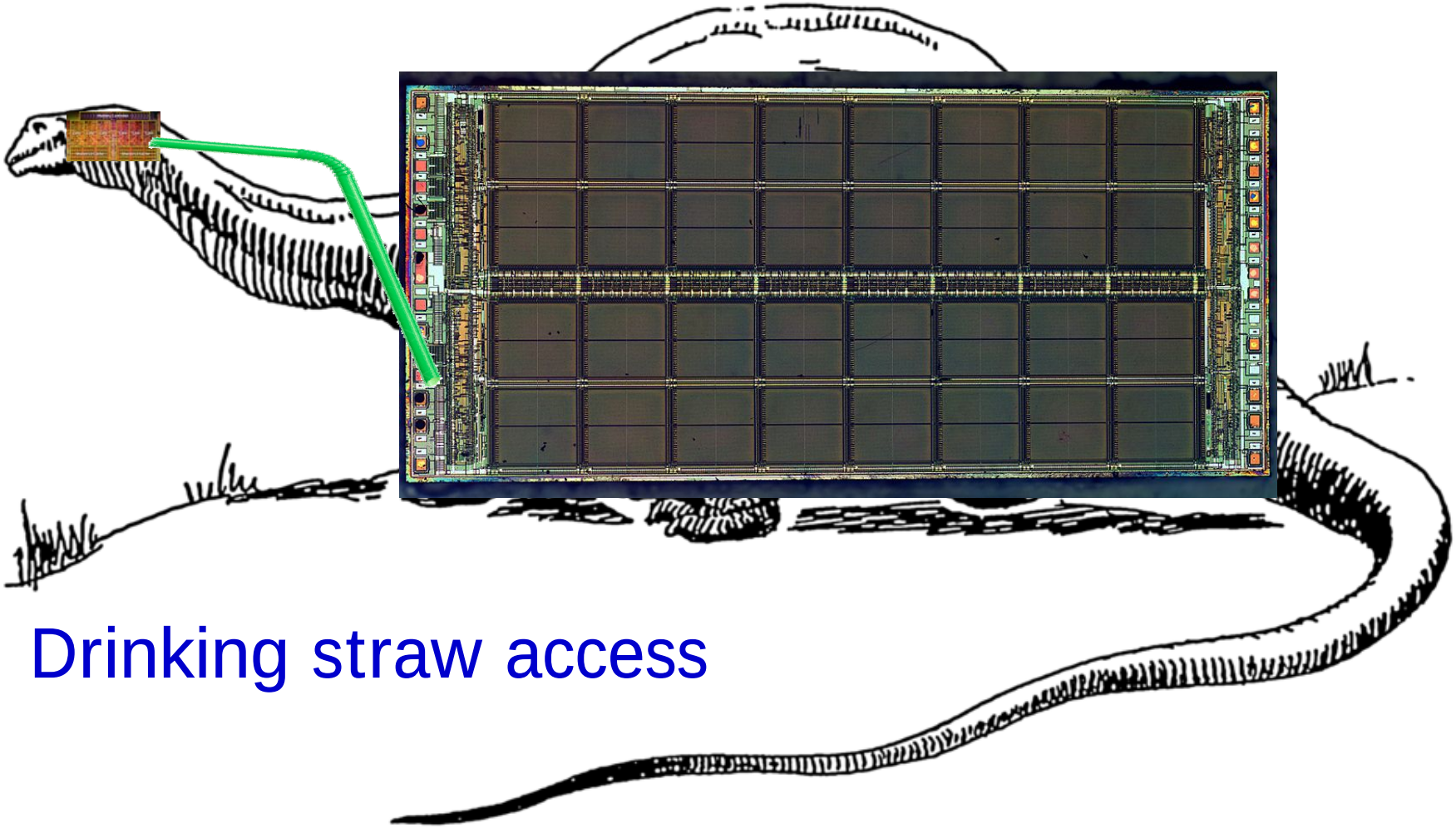


- i7: 12MB L3 cache



Little brain, **LOTS** of memory

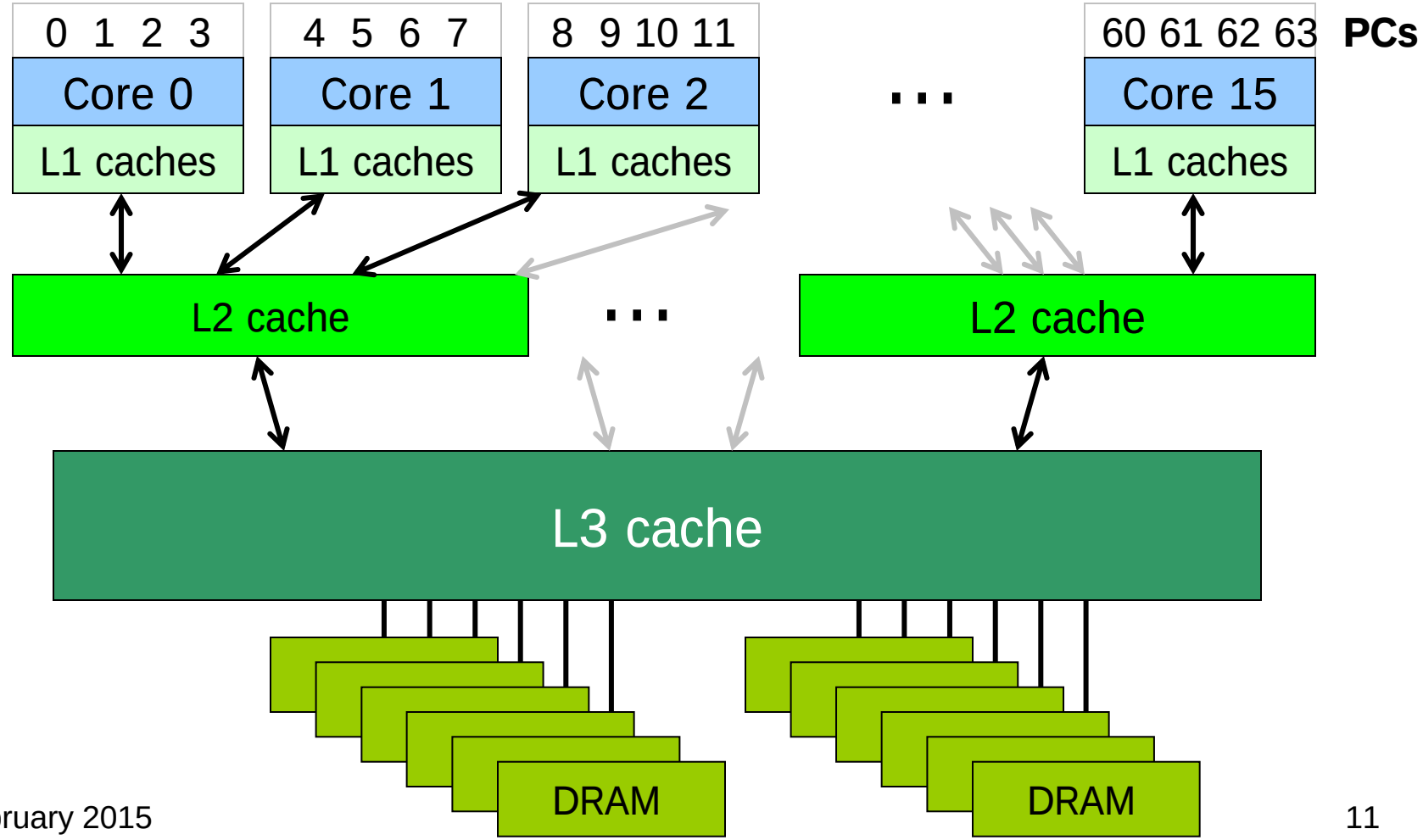
- Server 64GB-1TB



Drinking straw access

High-Bandwidth Cache Structure

- Hypothetical 64-CPU-thread server



Forcing Function

Move 16 bytes every CPU cycle

What are the direct consequences of this goal?

Move 16 bytes every CPU cycle

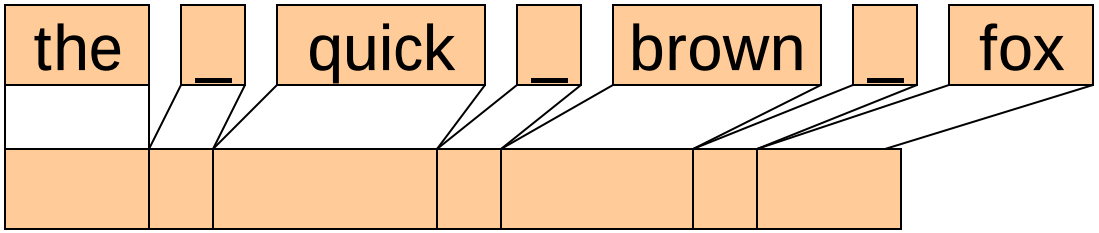
- Load 16B, Store 16B, test, branch
  - All in one cycle = 4-way issue minimum
  - Need some 16-byte registers

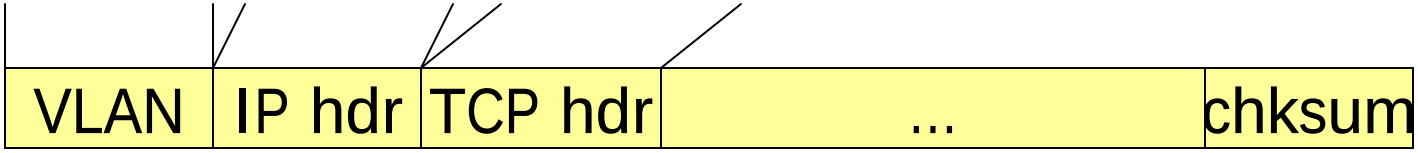
- At 3.2GHz, 50 GB/s read + 50 GB/s write,
  - 100 GB/sec, one L1 D-cache

- Must avoid *reading* cache line before write
  - if it is to be fully written (else 150 GB/s)

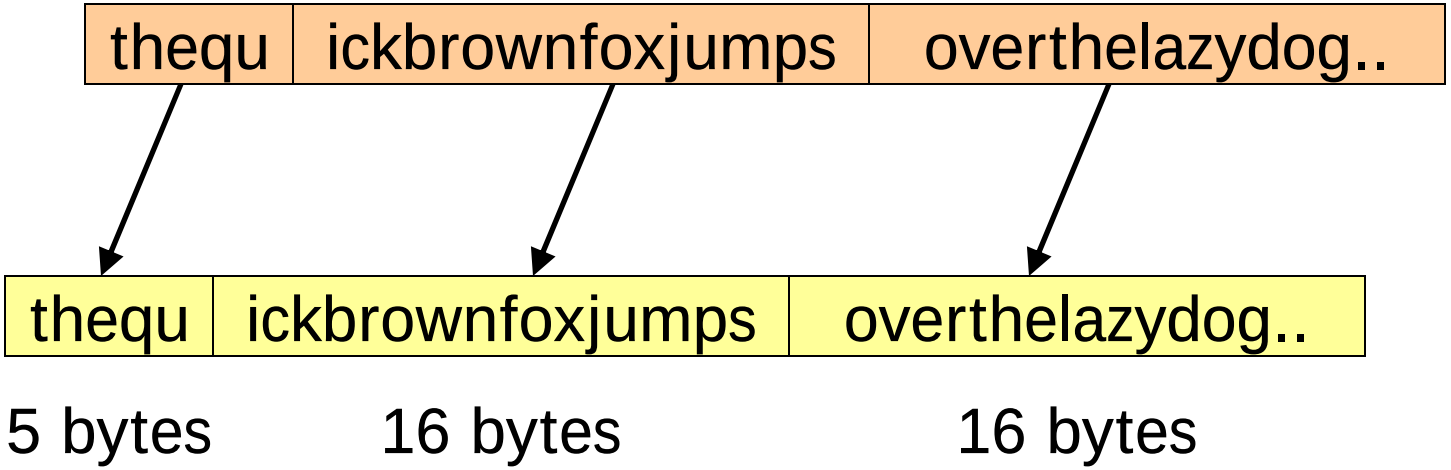
Short strings

- Parsing, words, packet headers, scatter-gather, checksums



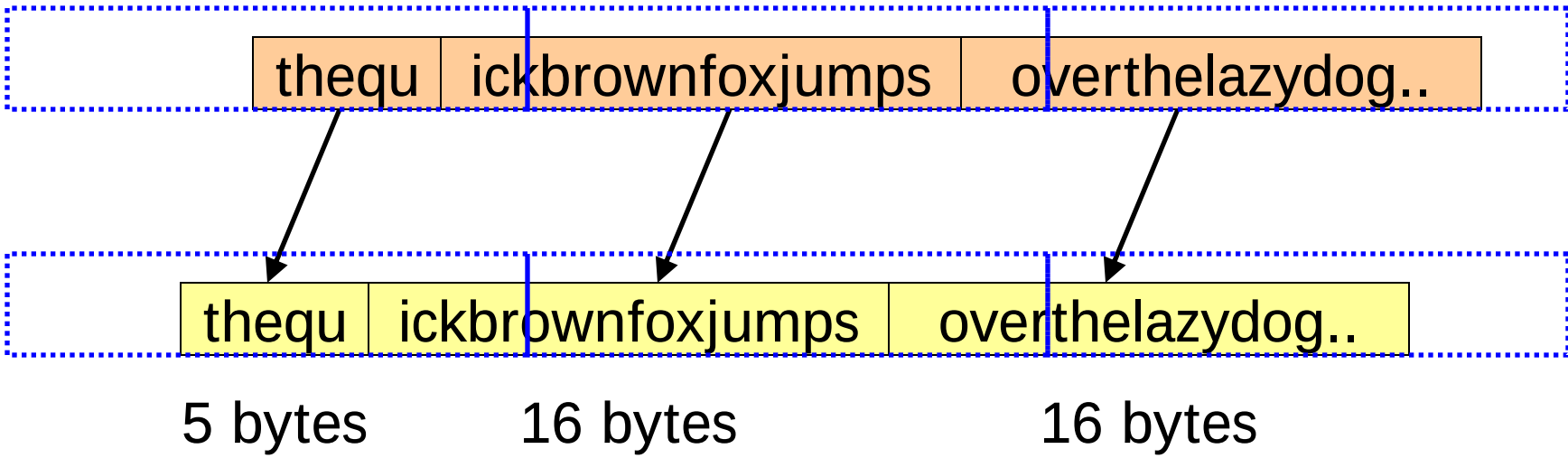


Generic Move, 37 bytes



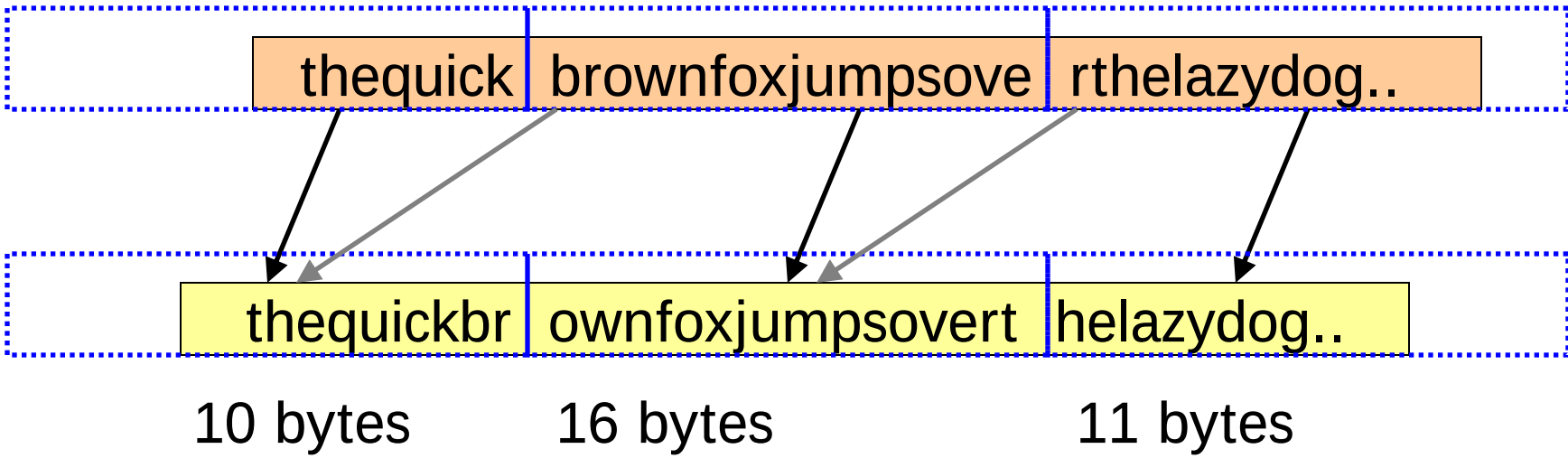
Generic Move, 37 bytes

16B aligned
cache accesses

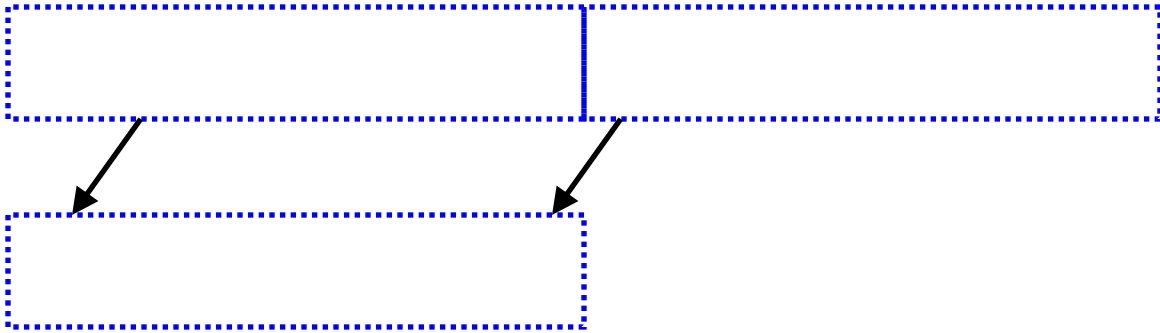


Generic Move, 37 bytes, aligned target

16B aligned
cache accesses



32B shift helpful:
LD, >>, ST

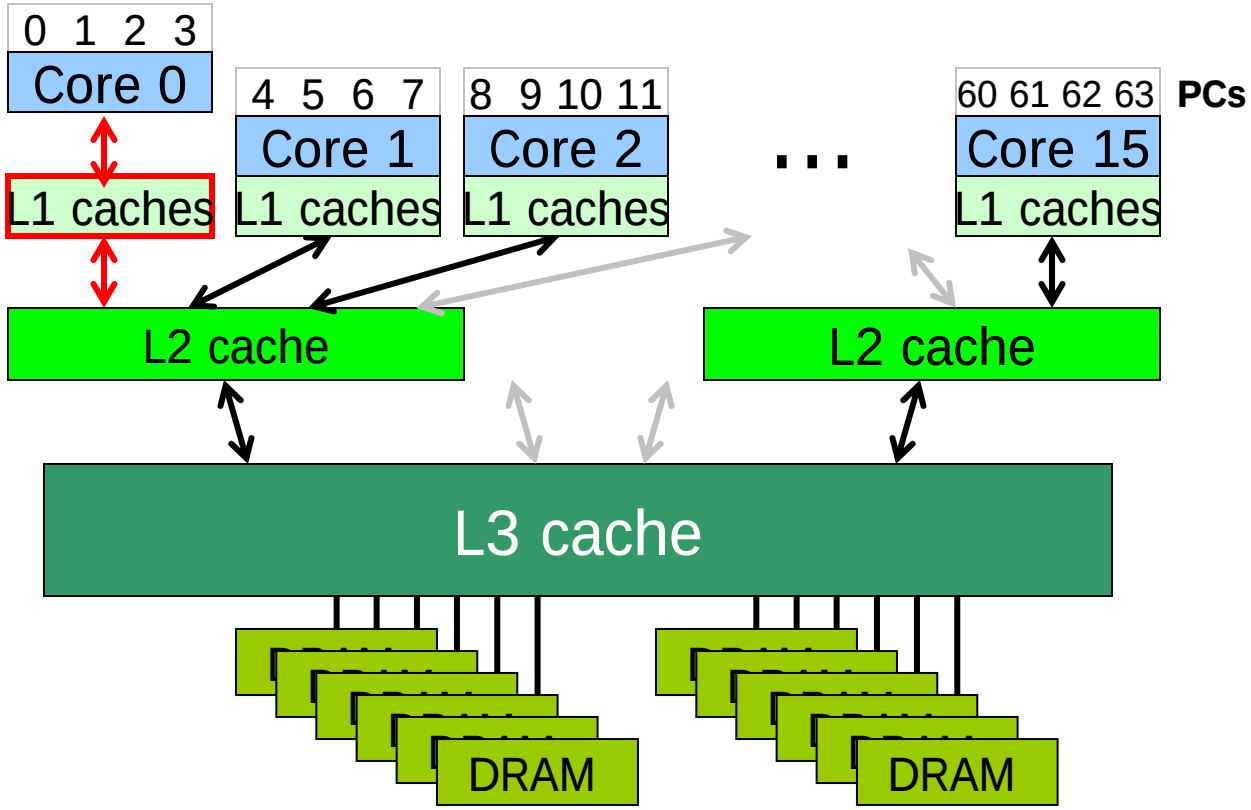


Useful Instructions

- **Load Partial** R1, R2, R3
- **Store Partial** R1, R2, R3
  - Load/store R1 low bytes with 0(R2), length 0..15 from R3 low 4 bits, 0 pad high bytes, R1 = 16 byte register
  - 0(R2) can be unaligned
  - Length zero *never* segfaults
  - Similar to Power architecture Move Assist instructions
- Handles all short moves
- Remaining length is always multiple of 16

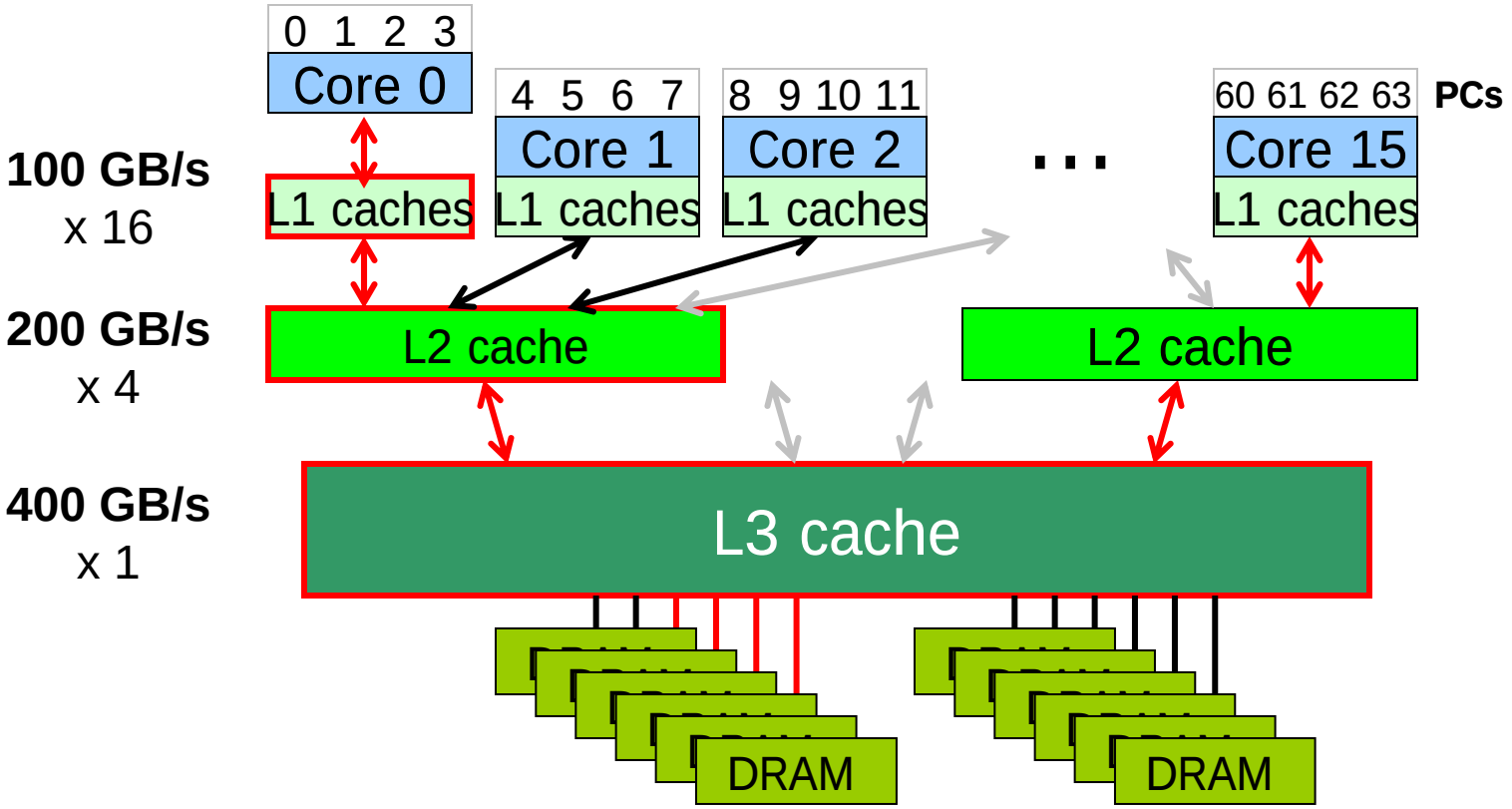
Move 16 bytes every CPU cycle

- L1 data cache: 2 aligned LD/ST accesses per cycle: 100 GB/s, plus 100 GB/s fill



Move 16 bytes every CPU cycle

- **Sustained** throughout L2, L3, and RAM
- Perhaps four CPUs simultaneously



Top 20 Stream Copy Bandwidth (April 2014)

	Date	Machine ID	ncpus	COPY
1.	2012.08.14	SGI_Altix_UV_2000	2048	6591 GB/s
2.	2011.04.05	SGI_Altix_UV_1000	2048	5321
3.	2006.07.10	SGI_Altix_4700	1024	3661
4.	2013.03.26	Fujitsu_SPARC_M10-4S	1024	3474
5.	2011.06.06	ScaleMP_Xeon_X6560_64B	768	1493
6.	2004.12.22	SGI_Altix_3700_Bx2	512	906
7.	2003.11.13	SGI_Altix_3000	512	854
8.	2003.10.02	NEC_SX-7	32	876
9.	2008.04.07	IBM_Power_595	64	679
10.	2013.09.12	Oracle_SPARC_T5-8	128	604
11.	1999.12.07	NEC_SX-5-16A	16	607
12.	2009.08.10	ScaleMP_XeonX5570_vSMP_16B	128	437
13.	1997.06.10	NEC_SX-4	32	434
14.	2004.08.11	HP_AlphaServer_GS1280-1300	64	407
	Our Forcing Function		**1**	**400 GB/s**
15.	1996.11.21	Cray_T932_321024-3E	32	310
16.	2014.04.24	Oracle_Sun_Server_X4-4	60	221
17.	2007.04.17	Fujitsu/Sun_Enterprise_M9000	128	224
18.	2002.10.16	NEC_SX-6	8	202
19.	2006.07.23	IBM_System_p5_595	64	186
20.	2013.09.17	Intel_XeonPhi_SE10P	61	169

https://www.cs.virginia.edu/stream/top20/Bandwidth.html

Latency

- Buy 256GB of RAM only if you use it
  - Higher cache miss rates
  - And main memory is ~200 cycles away

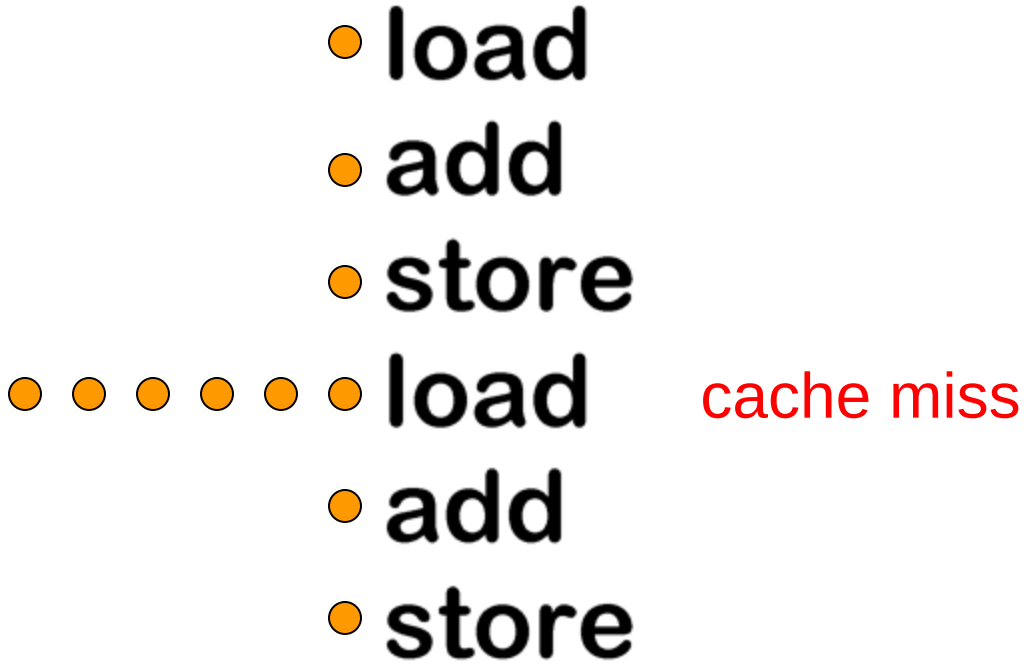
Cache Hits

1-cycle hit @4 Hz

- load
- add
- store
- load
- add
- store

Cache Miss to Main Memory

200-cycle miss @4 Hz



What constructive work could you do during this time?

Latency

- Buy 256GB of RAM only if you use it
  - Higher cache miss rates
  - And main memory is ~200 cycles away

- For starters, we need to prefetch about 16B * 200cy = 3.2KB to meet our forcing function; call it 4KB

Additional Considerations

- L1 cache size = associativity * page size
  - Need bigger than 4KB pages
- Translation buffer at 256 x 4KB covers only 1MB of memory
  - Covers much less than on-chip caches
  - TB miss can consume **15%** of total CPU time
  - Need bigger than 4KB pages
- With 256GB of RAM @4KB: 64M pages
  - Need bigger than 4KB pages

Move data: big and small

- It’s still the memory
- Need a coordinated design of instruction architecture and memory implementation to achieve high bandwidth with low delay

Modern challenges in CPU design

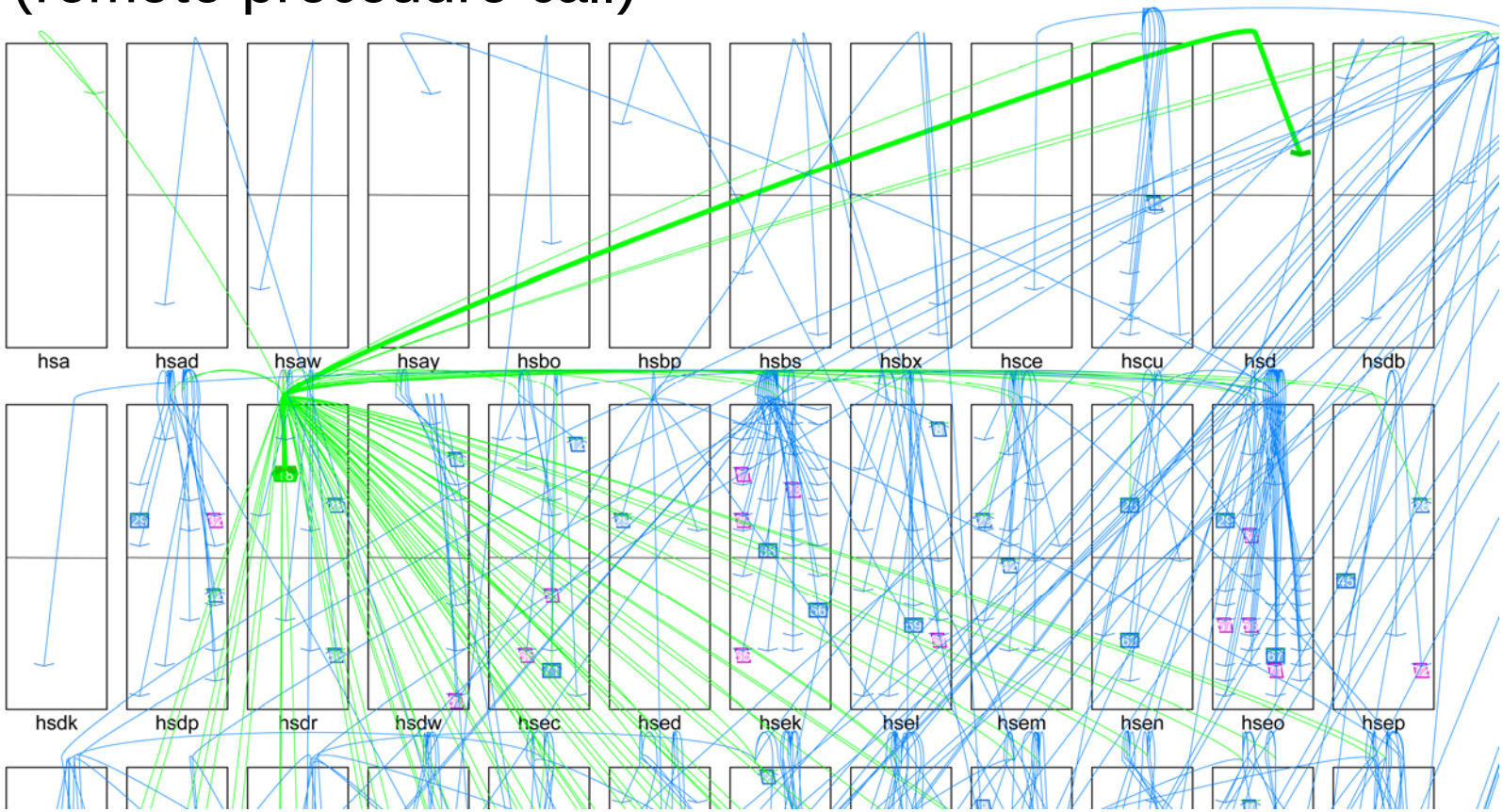
- **Lots** of memory
- Multi-issue CPU instructions every cycle that drive full bandwidth
- Full bandwidth all the way to RAM, not just to L1 cache
- More prefetching in software
- Bigger page size(s)

Topic: do better

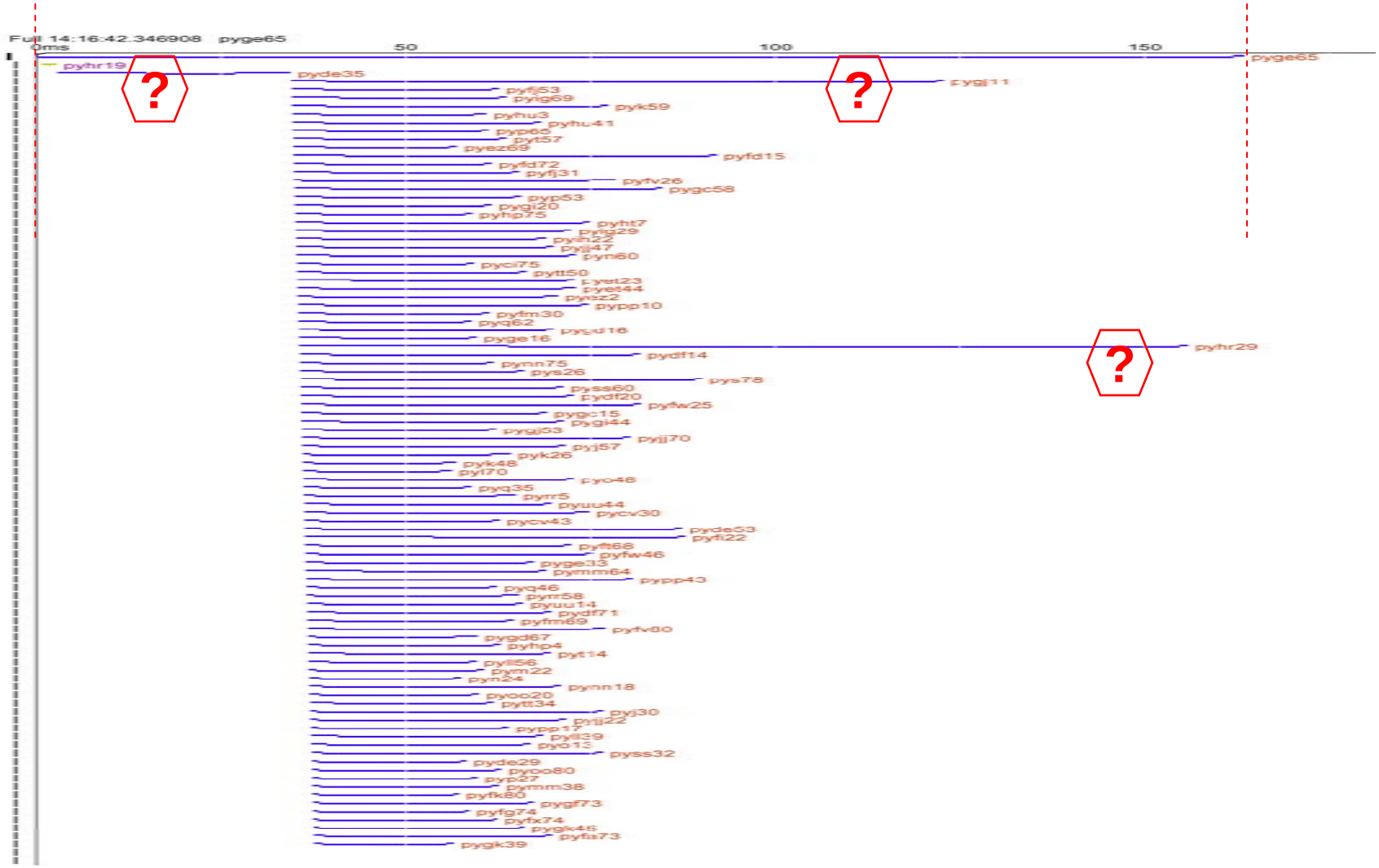
② Real-time transactions: 1000s per second

A **Single** Transaction Across ~40 Racks of ~60 Servers Each

- Each arc is a related client-server RPC (remote procedure call)

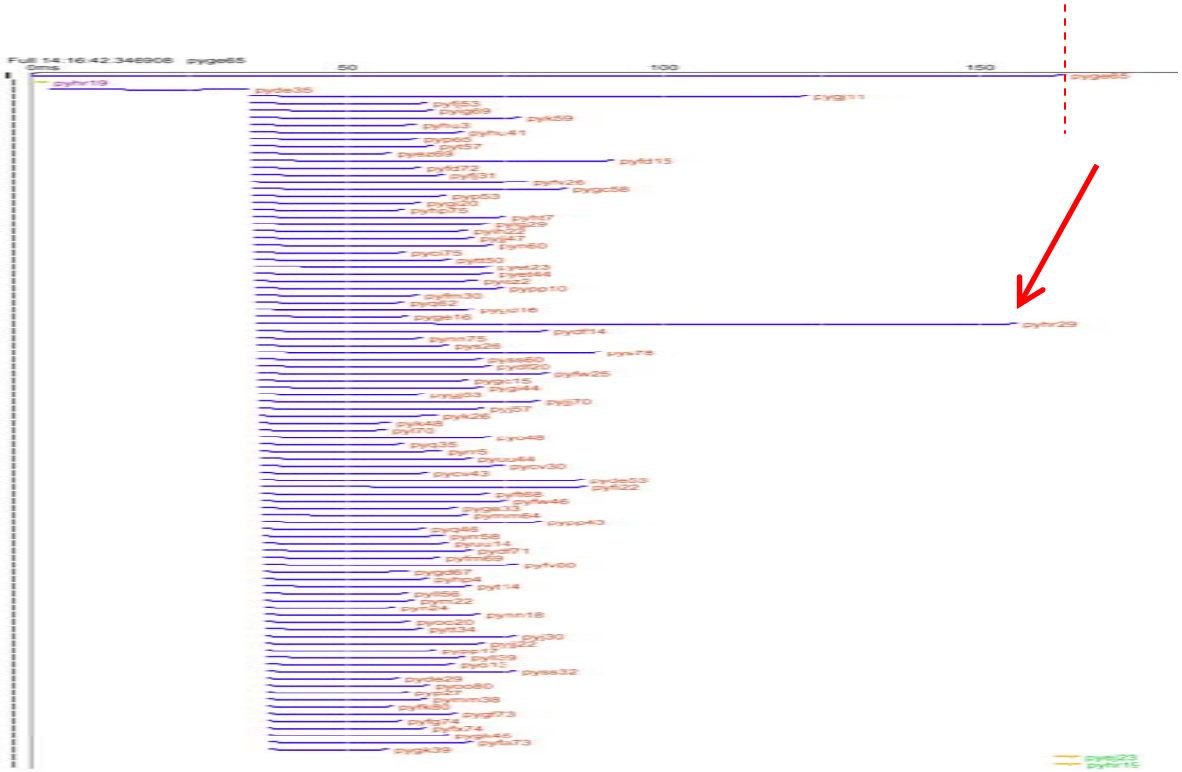


A **Single** Transaction RPC Tree vs Time: Client & 93 Servers



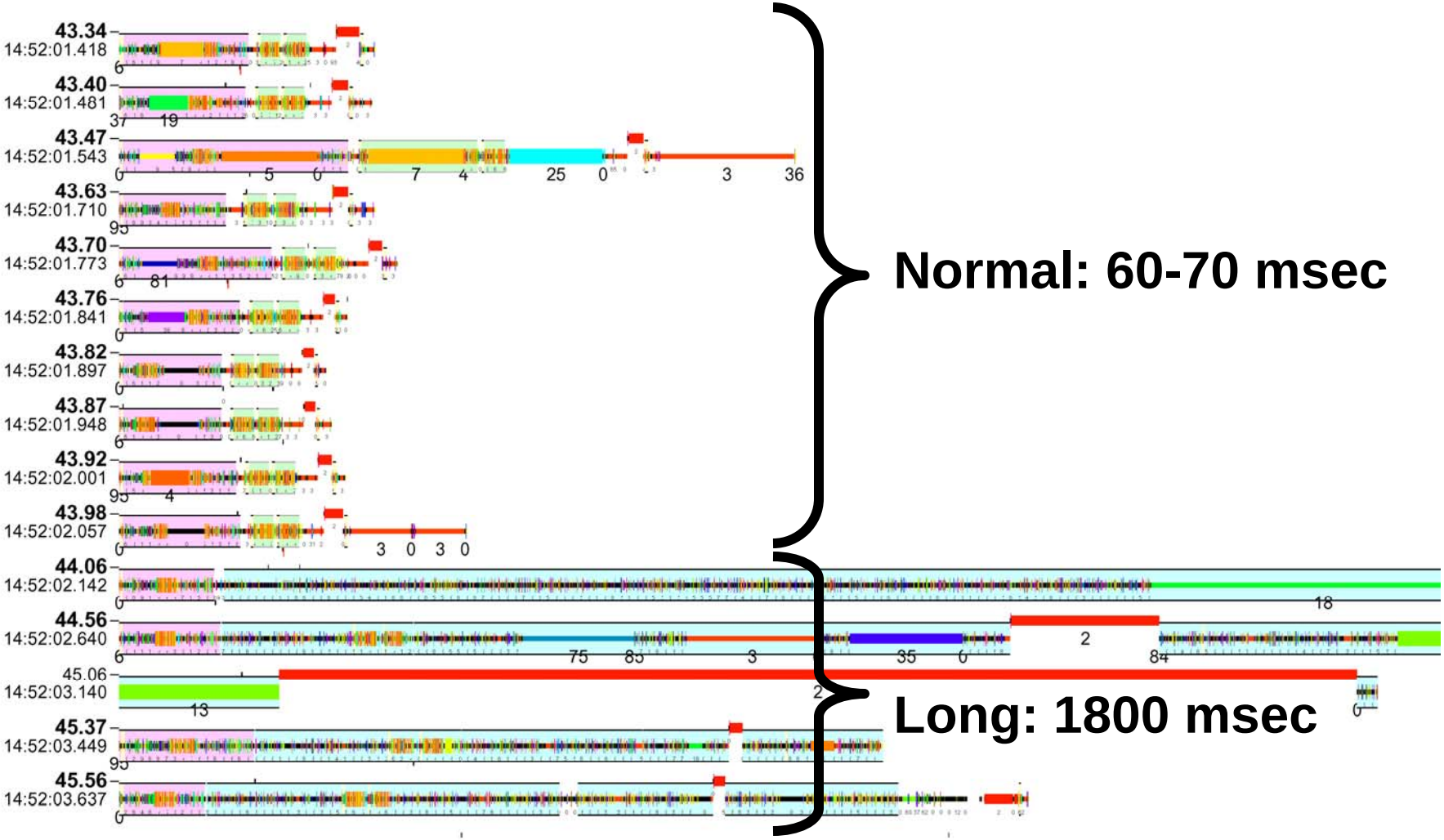
Single Transaction Tail Latency

- One slow response out of 93 parallel RPCs slows the *entire* transaction



One Server, One User-Mode Thread vs. Time

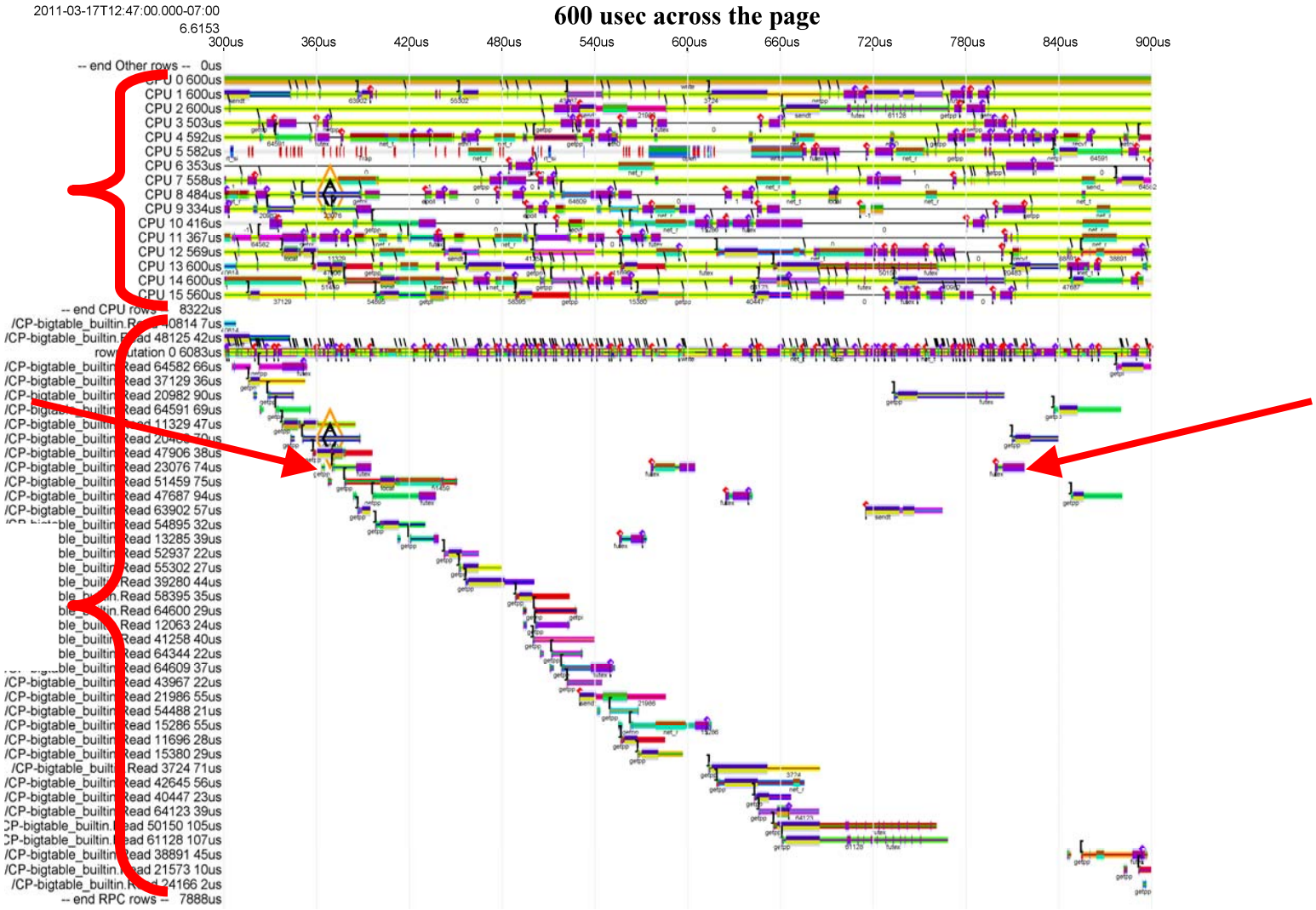
Eleven Sequential Transactions (circa 2004)



One Server, Four CPUs: User/kernel transitions

every CPU every nanosecond (Ktrace)

16 CPUs, 600us, Many RPCs



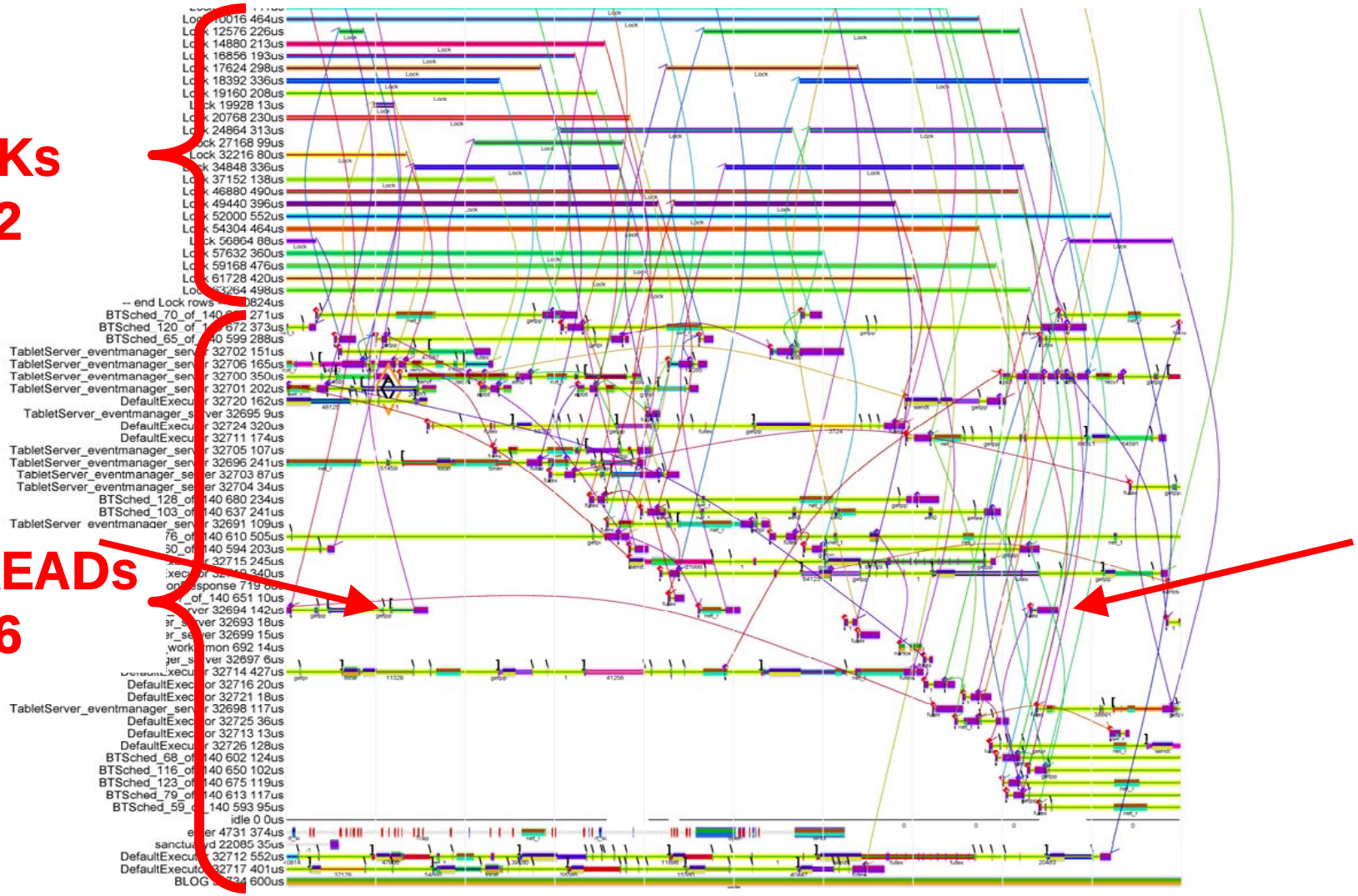
CPUs 0..15

RPCs 0..39

16 CPUs, 600us, Many RPCs

LOCKs 0..22

THREADs 0..46



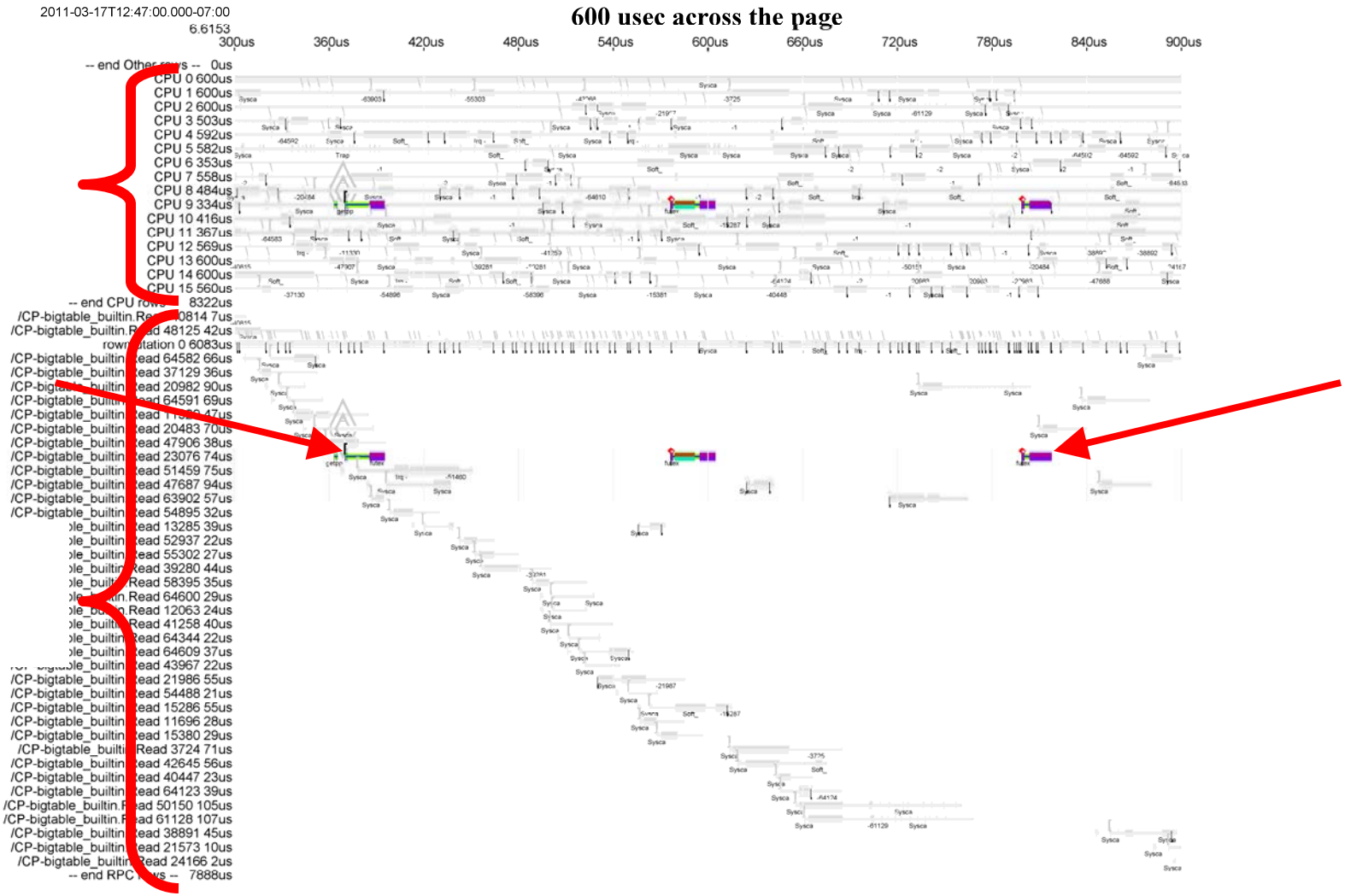
That is A LOT going on at once

- Let's look at just *one* long-tail RPC in context

16 CPUs, 600us, one RPC

CPUs 0..15

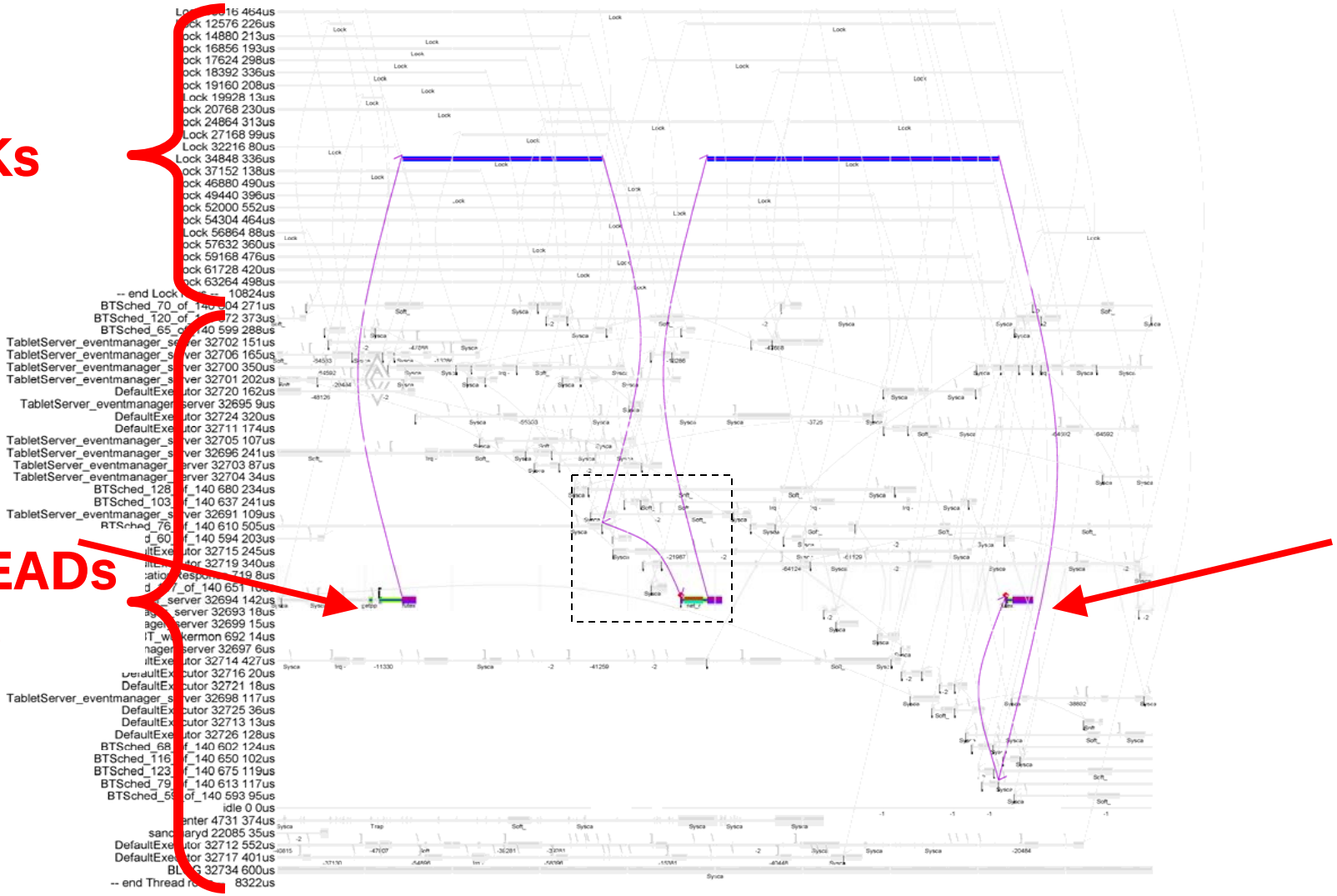
RPCs 0..39



16 CPUs, 600us, one RPC

LOCKs 0..22

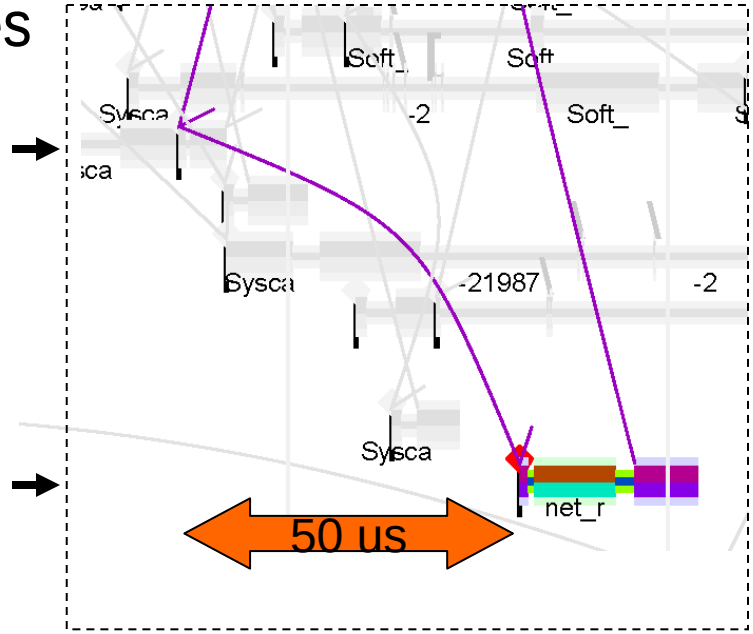
THREADs 0..46



Wakeup Detail

Thread 19 frees lock, sends wakeup to waiting thread 25

Thread 25 actually runs

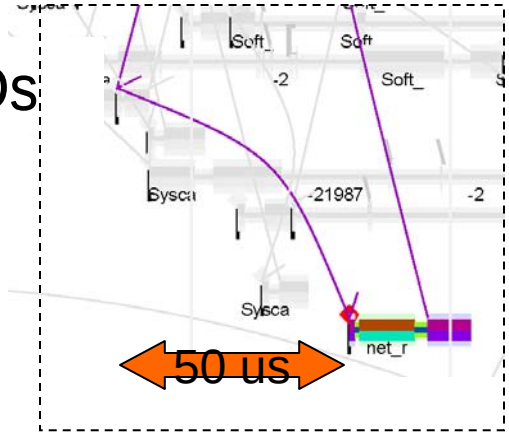


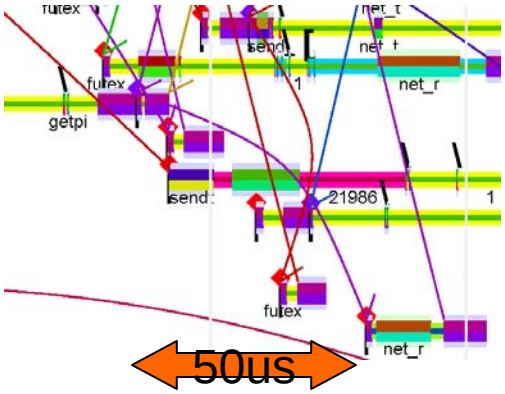
50us wakeup delay ??

Wakeup Detail

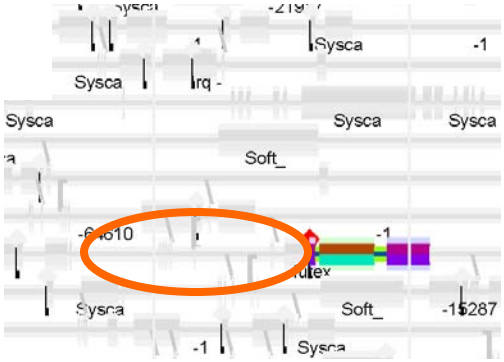
- Target CPU was busy; kernel waited

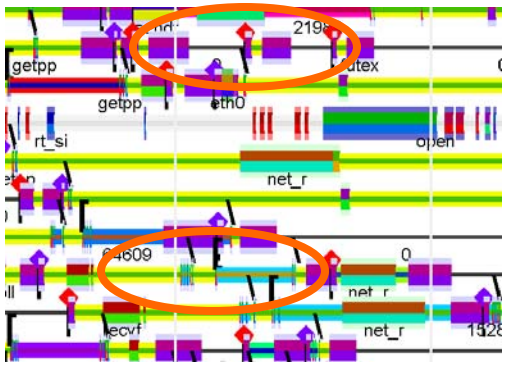
THREADs





CPUs





CPU 9 busy; But 3 was idle

CPU Scheduling, 2 Designs

- Re-dispatch on any idle CPU core
  - But if idle CPU core is in deep sleep, can take 75-100us to wake up
- Wait to re-dispatch on previous CPU core, to get cache hits
  - Saves squat if could use same L1 cache
  - Saves ~10us if could use same L2 cache
  - Saves ~100us if could use same L3 cache
  - Expensive if cross-socket cache refills
  - Don’t wait too long...

Real-time transactions: 1000s per second

- Not your father's SPECmarks
- To understand delays, need to track simultaneous transactions across servers, CPU cores, threads, queues, locks

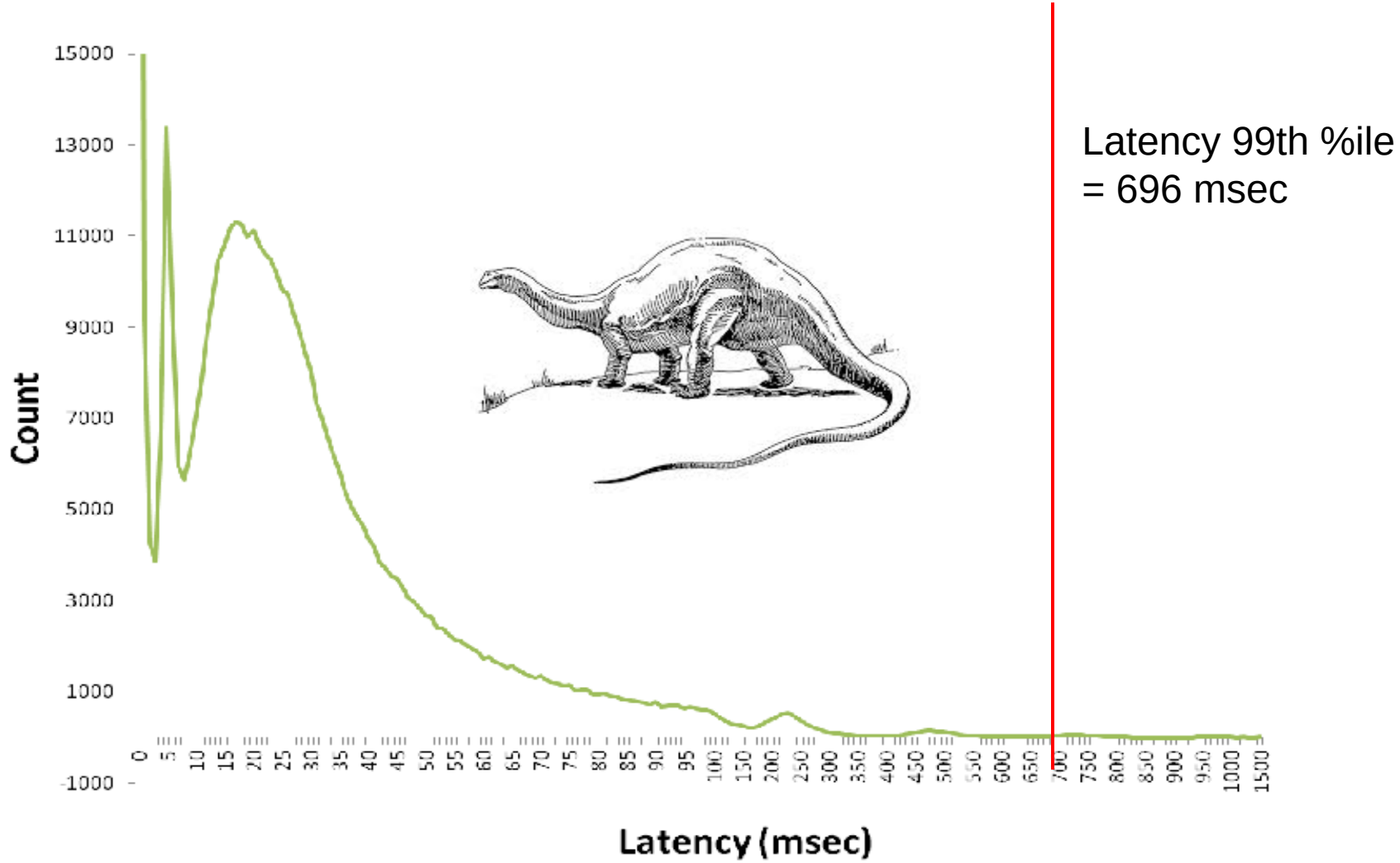
Modern challenges in CPU design

- A single transaction can touch thousands of servers in parallel
- The slowest parallel path dominates
- Tail latency is the enemy
  - Must control lock-holding times
  - Must control scheduler delays
  - Must control interference via shared resources

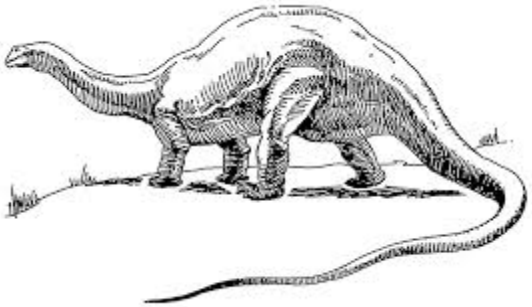
Topic: do better

③ Isolation between programs

Histogram: Disk Server Latency; Long Tail



Non-repeatable Tail Latency Comes from Unknown Interference



Isolation of programs reduces tail latency.

Reduced tail latency = higher utilization.

Higher utilization = $$$.

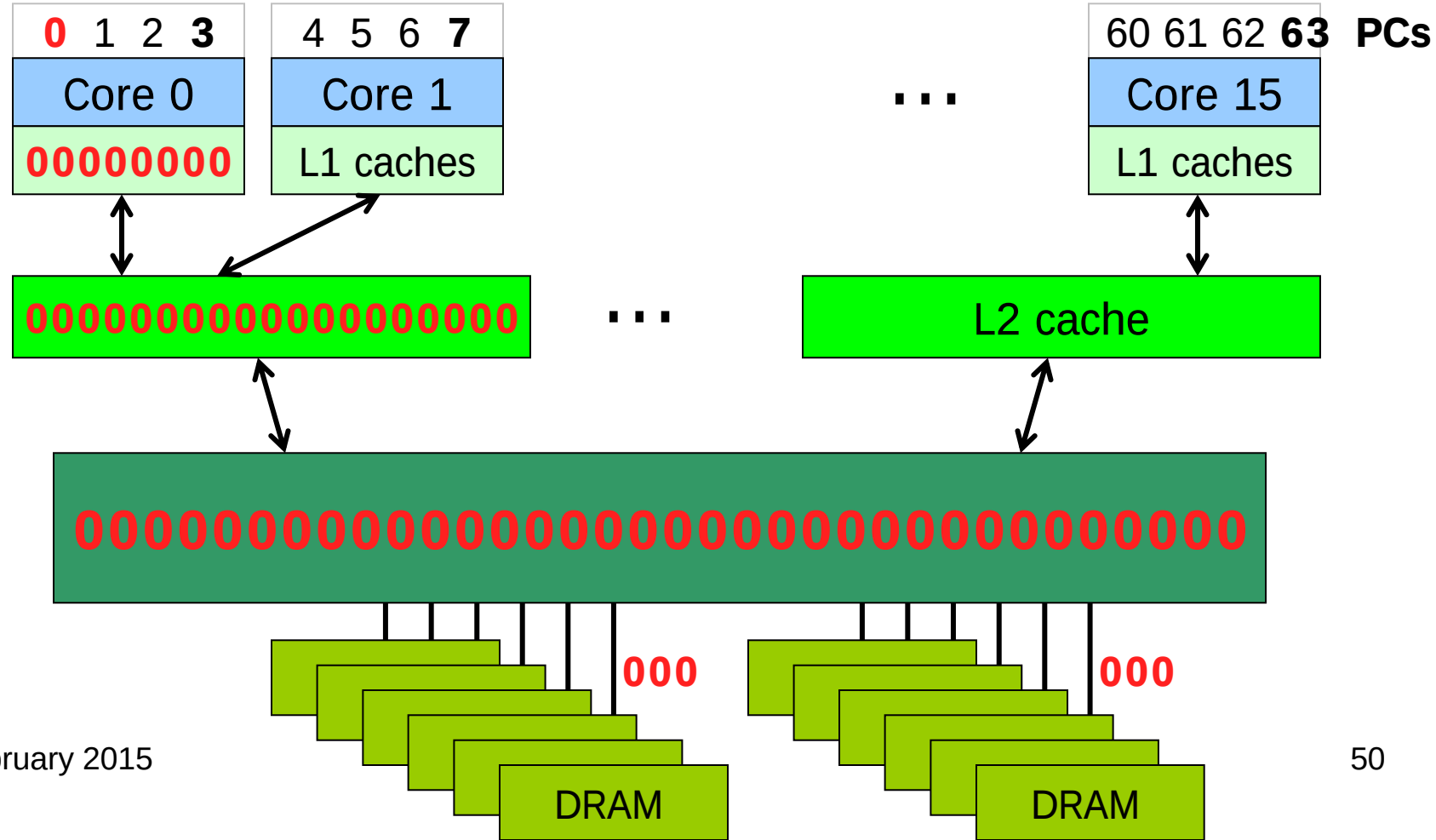
Many Sources of Interference

- Most interference comes from software
- But a bit from the hardware underpinnings

- In a shared apartment building, most interference comes from jerky neighbors
- But thin walls and bad kitchen venting can be the hardware underpinnings

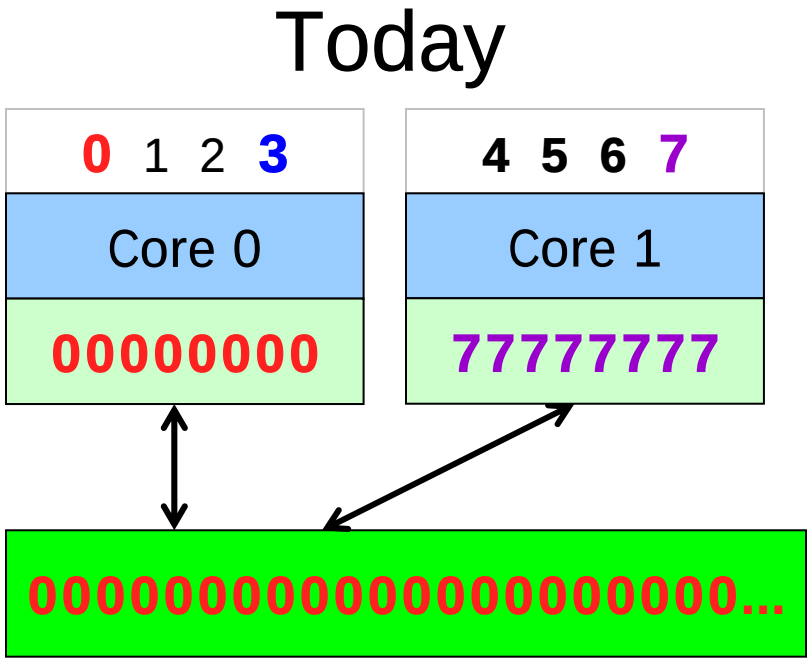
Isolation issue: Cache Interference

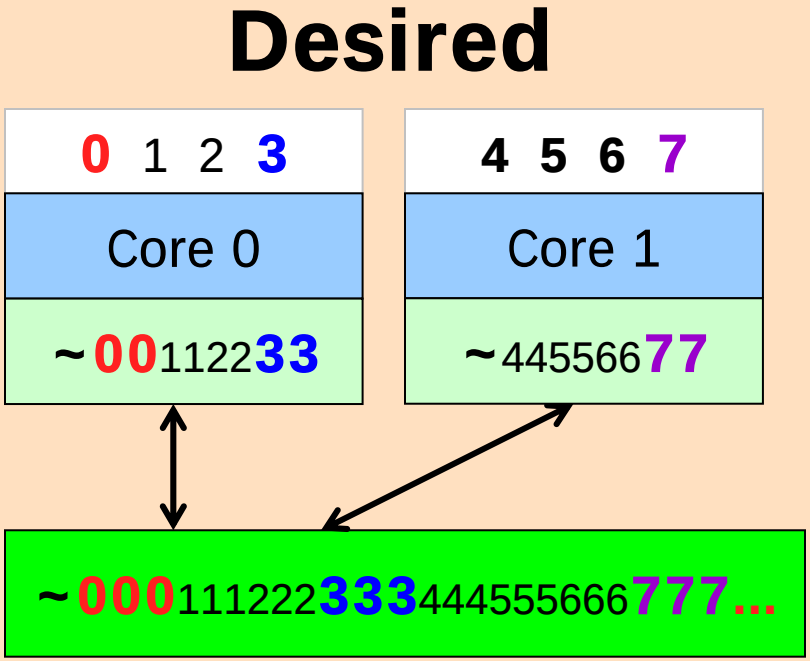
CPU thread 0 is moving 16B/cycle flat out, filling caches, hurting threads 3, 7, 63



Isolation issue: Cache Interference

- CPU thread 0 is moving 16B/cycle flat out, hurting threads 3, 7, 63



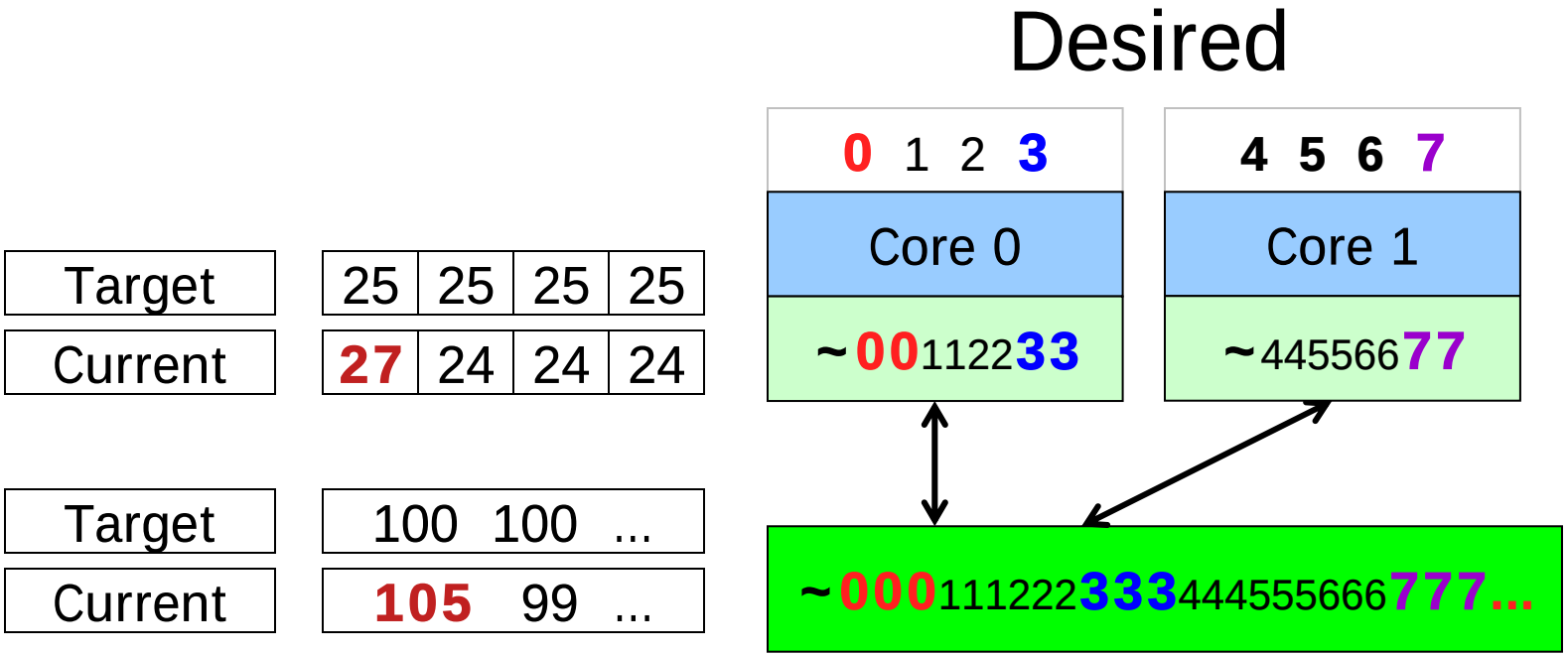


Cache Interference

- How to get there?
  - Partition by ways
    - no good if 16 threads and 8 ways
    - No good if result is direct-mapped
    - Underutilizes cache
  - Selective allocation
    - Give each thread a target cache size
    - Allocate lines freely if under target
    - Replace only *own* lines if over target
    - Allow over-budget slop to avoid underutilization

Cache Interference

- Each thread has target:current, replace only *own* lines if over target



Cache Interference

- Each thread has target:current, replace only *own* lines if over target
- Requires owner bits per cache line
  - expensive bits
- Requires 64 target/current at L3
- Fails if L3 not at least 64-way associative
  - Can rarely find *own* lines in a set

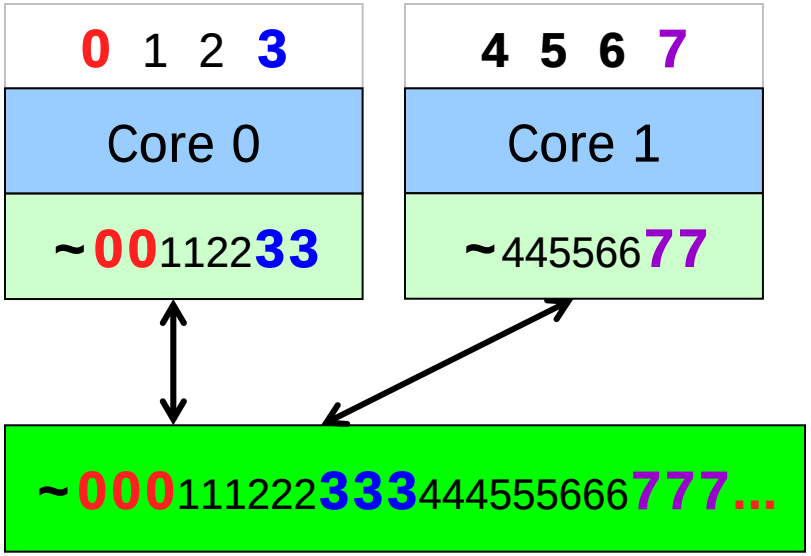
Design Improvement

- Track ownership *just* by incoming paths
  - Plus separate target for kernel accesses
  - Plus separate target for over-target accesses
- Fewer bits, 8-way assoc OK

Target	OV	K	0	1	2	3
Current	OV	K	0	1	2	3

Target	OV	K	0-3	4-7	8-...	...
Current	OV	K	0-3	4-7	8-...	...

Desired



Isolation between programs

- Good fences make good neighbors
- We need better hardware support for program isolation in shared memory systems

Modern challenges in CPU design

- Isolating programs from each other on a shared server is hard
- As an industry, we do it poorly
  - Shared CPU scheduling
  - Shared caches
  - Shared network links
  - Shared disks

- More hardware support needed
- More innovation needed

④ Measurement underpinnings

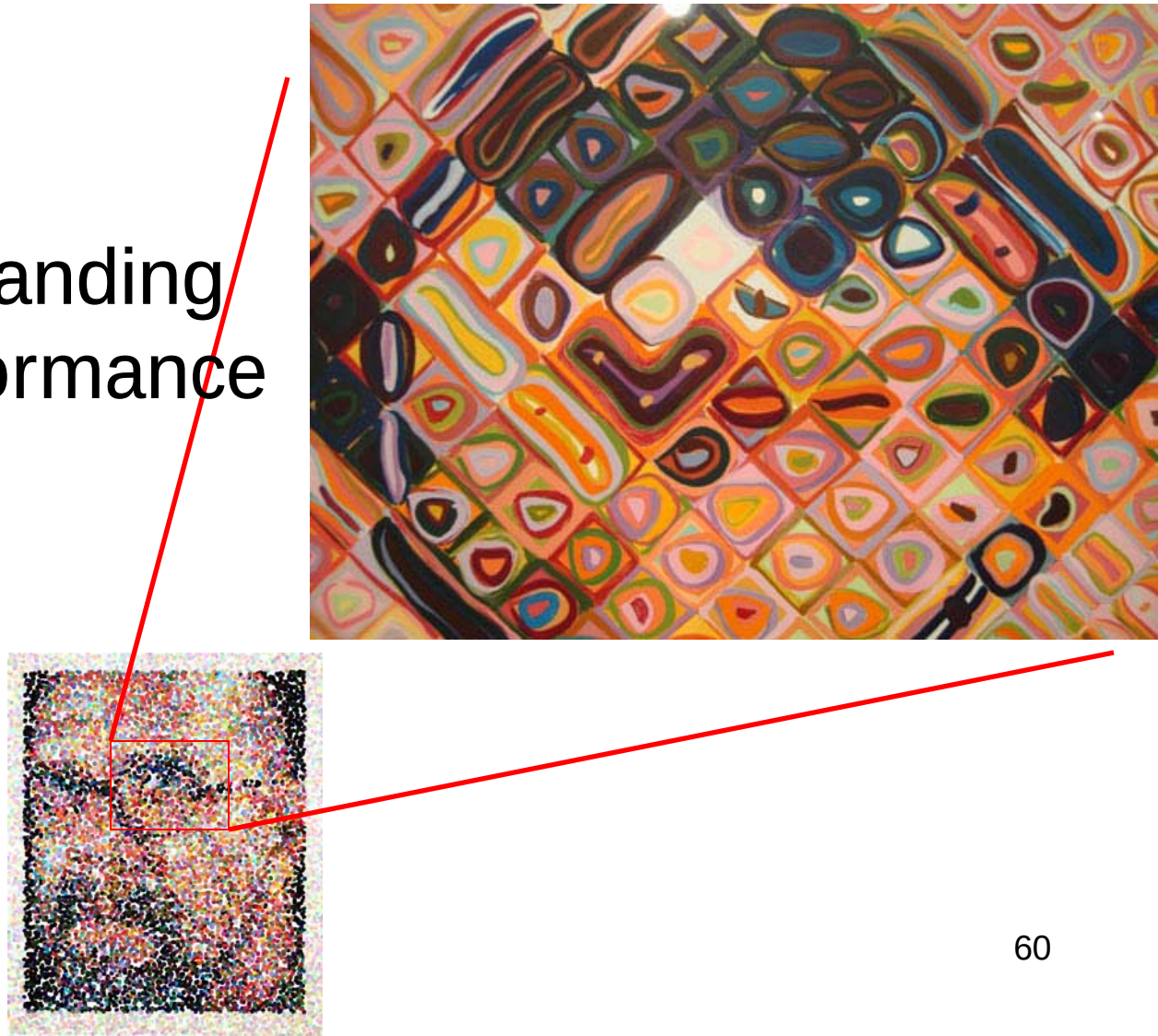
Profiles: What, not Why

- Samples of 1000s of transactions, merging results
- Pro: understanding *average* performance
- **Blind spots:** outliers, idle time

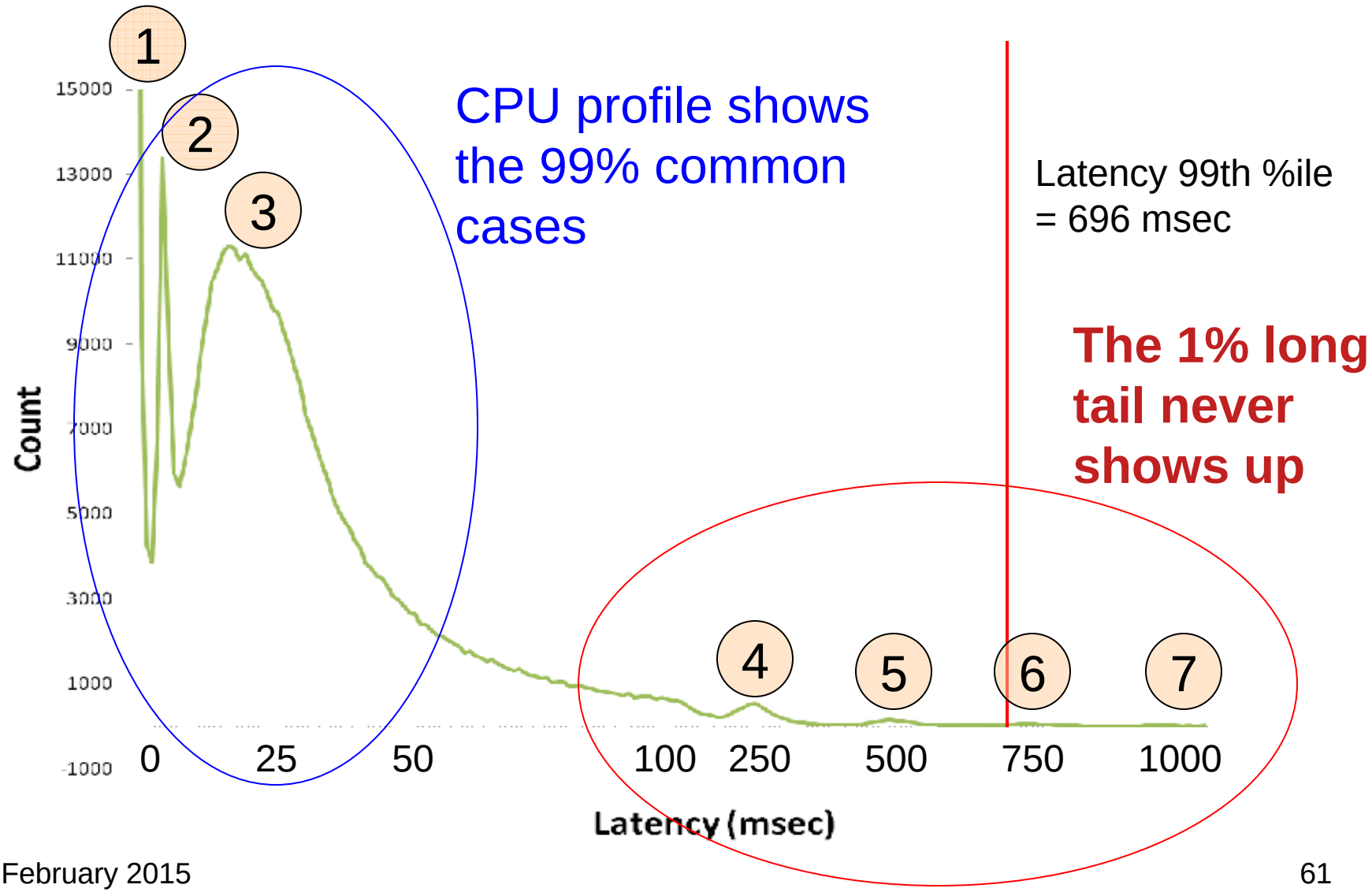
– Chuck Close

Traces: Why

- Full detail of individual transactions
- Pro: understanding *outlier* performance
- **Blind spot:** tracing overhead



Histogram: Disk Server Latency; Long Tail

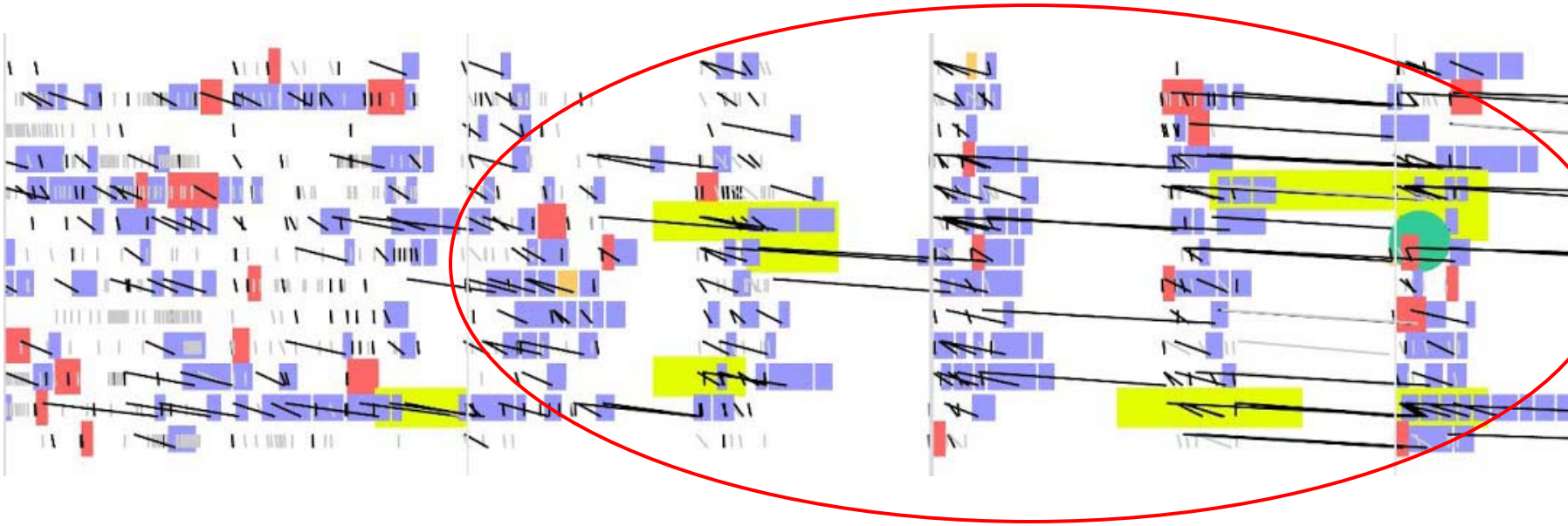


Trace: Disk Server, Event-by-Event

- Read RPC + disk time 
- Write hit, hit, miss 
- 700ms mixture 
- 13 disks, three normal seconds:

Trace: Disk Server, 13 disks, 1.5 sec

- Phase transition to 250ms boundaries *exactly*



Read

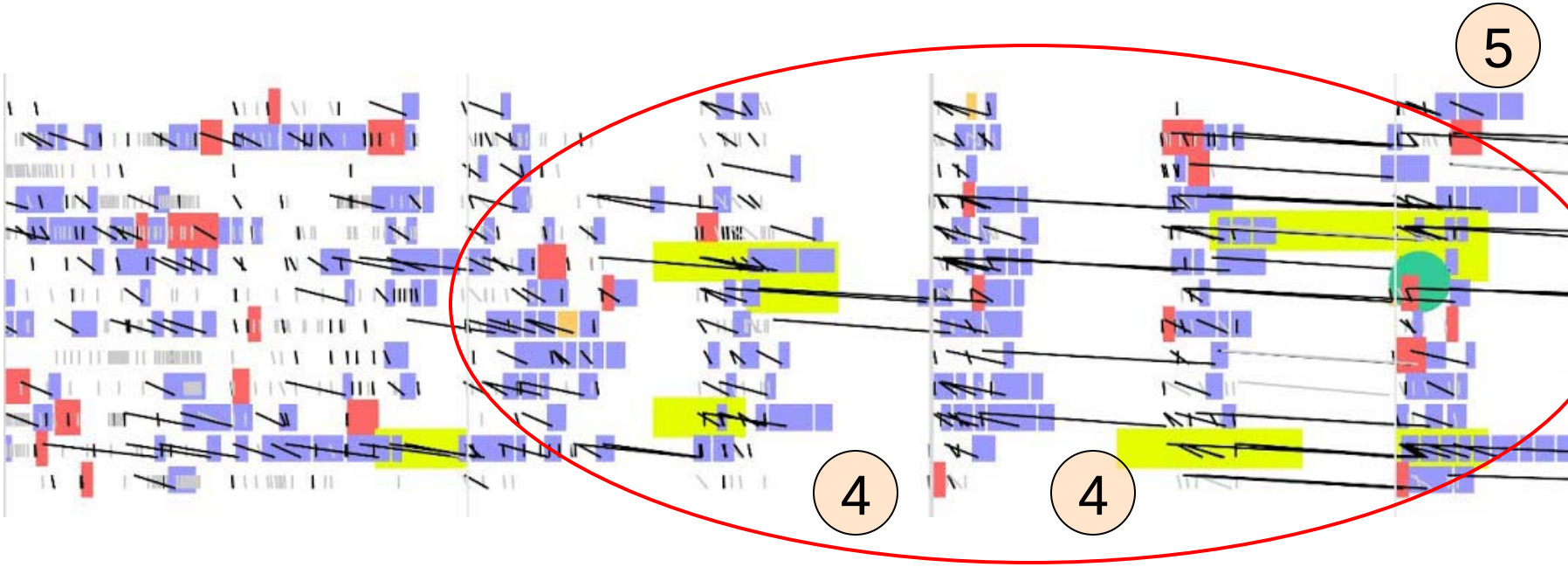
Write

Read hit

Write hit

Trace: Disk Server, 13 disks, 1.5 sec

- Latencies: 250ms, 500ms, ... *for nine minutes*



Read

Write

Read hit

Write hit

Why?

- Probably not on your guessing radar...
- Kernel throttling the CPU use of any process that is over purchased quota
- Only happened on old, slow servers

Disk Server, CPU Quota bug

- Understanding Why 4 sped up 25% of entire disk fleet worldwide!
  - Had been going on for three years
  - Savings paid my salary for 10 years
- Hanlon's razor: Never attribute to malice that which is adequately explained by stupidity.
- Sites' corollary: Never attribute to stupidity that which is adequately explained by software complexity.

Measurement Underpinnings

- All performance mysteries are simple once they are understood
- “Mystery” means that the picture in your head is wrong; software engineers are singularly inept at guessing how their view differs from reality

Modern challenges in CPU design

- Need low-overhead tools to observe the dynamics of performance anomalies
  - Transaction IDs
  - RPC trees
  - Timestamped transaction begin/end
- Traces
  - CPU kernel+user, RPC, lock, thread traces
  - Disk, network, power-consumption

Topic: do better

Summary: Datacenter Servers are Different

Datacenter Servers are Different

① Move data: big and small

② Real-time transactions: 1000s per second

③ Isolation between programs

④ Measurement underpinnings

References

Claude Shannon, *A Mathematical Theory of Communication*, The Bell System Technical Journal, Vol. 27, pp. 379–423, 623–656, July, October, 1948.
http://cs.ucf.edu/~dcm/Teaching/COP5611-Spring2013/Shannon48-MathTheoryComm.pdf

Richard Sites, *It's the Memory, Stupid!*, Microprocessor Report August 5, 1996.
http://cva.stanford.edu/classes/cs99s/papers/architects_look_to_future.pdf

Ravi Iyer, *CQoS: A Framework for Enabling QoS in Shared Caches of CMP Platforms*, ACM International Conference on Supercomputing, 2004.
http://cs.binghamton.edu/~apatel/cache_sharing/CQoS_iyer.pdf

M Bligh, M Desnoyers, R Schultz , *Linux Kernel Debugging on Google-sized clusters*, Linux Symposium, 2007
https://www.kernel.org/doc/mirror/ols2007v1.pdf#page=29

References

Daniel Sanchez and Christos Kozyrakis , *The ZCache: Decoupling Ways and Associativity*. IEEE/ACM Symp. on Microarchitecture (MICRO-43), 2010.

http://people.csail.mit.edu/sanchez/papers/2010.zcache.micro.pdf

Benjamin H. Sigelman, Luiz Andre Barroso, Mike Burrows, Pat Stephenson, Manoj Plakal, Donald Beaver, Saul Jaspan, Chandan Shanbhag, *Dapper, a Large-Scale Distributed Systems Tracing Infrastructure*, Google Technical Report dapper-2010-1, April 2010

http://scholar.google.com/scholar?q=Google+Technical+Report+dapper-2010-1&btnG=&hl=en&as_sdt=0%2C5&as_vis=1

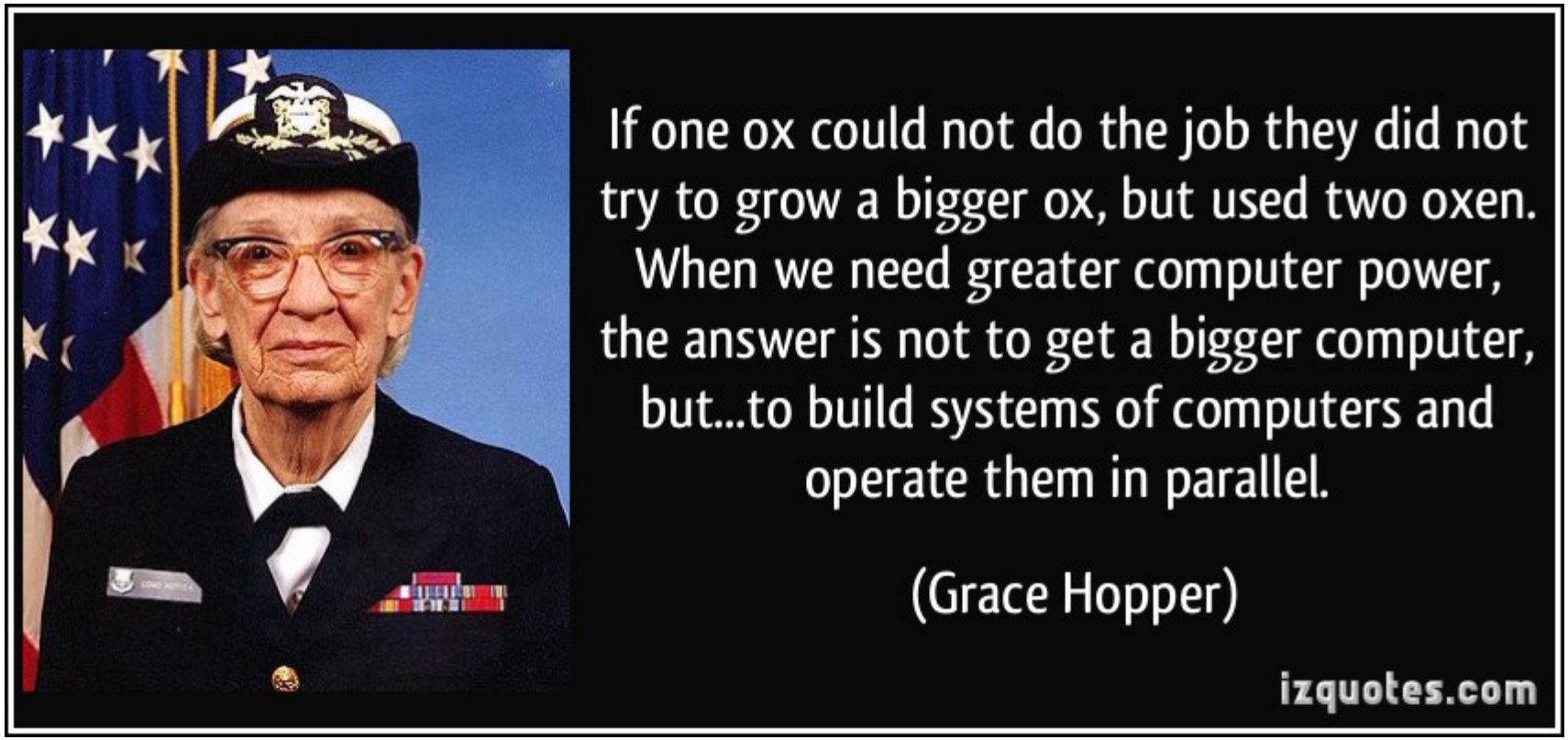
Daniel Sanchez, Christos Kozyrakis, *Vantage: Scalable and Efficient Fine-Grained Cache Partitioning*, Symp. on Computer Architecture ISCA 2011.

http://ppl.stanford.edu/papers/isca11-sanchez.pdf

Luiz André Barroso and Urs Hölzle , *The Datacenter as a Computer An Introduction to the Design of Warehouse-Scale Machines, 2nd Edition 2013*.

http://www.morganclaypool.com/doi/pdf/10.2200/S00516ED2V01Y201306CAC024

Thank You, Questions?



Thank You