Keeping up with the architects.

Andrew Warfield, UBC and Coho Data

About this keynote.
(And the things I'm not going to talk about.)

Not going to talk about any of this stuff right now (but happy to in the hallway track)

- Finished PhD at Cambridge in 2006
- Worked in industrial research (AT&T and Intel)
- Two startups (XenSource and Coho Data)
- Associate prof at UBC
- Three kids
- I went heli skiing last Friday.

Here's what I am going to do

- Make some pretty obvious observations about technology directions.
- Draw some dodgy and highly speculative conclusions from those observations.
- Try to influence your research.

- Disclaimer: this is not a conference talk, nor is it 5 stapled together conference talks.
- Another disclaimer: I'm going to give you more problems than solutions.

So let's go…

Section 5: Evaluation.

- (At the end of the day, all systems papers are about performance.)
- Probably because it's one of the only things we know how to measure.
- There are two types of performance results:

  1. Small improvements in a very large system.
  2. Speed ups that are so significant that they change functionality.

- Google and Facebook and Amazon and Microsoft are probably a lot better at solving meaningful problems with their systems than you are.

Here are the high-level trends/ideas behind this talk

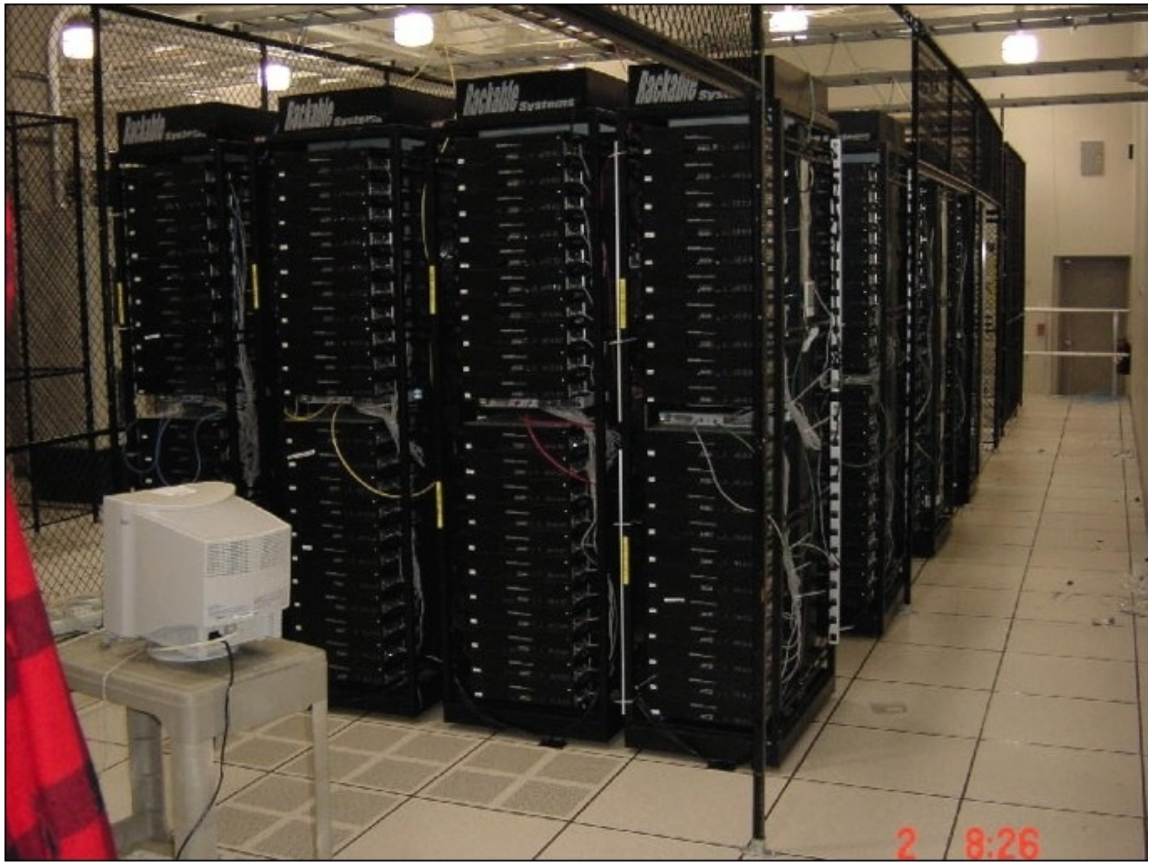
1. **Diminishing scarcity.**
2. Practical/sensible to own your own hardware again.
3. The software we have is turning out to be a bigger, slower, more onerous burden than the hardware it runs on.
    - It is a poor match for changing performance and failure characteristics of hardware.
    - It is a poor match for the operational needs of users.

Consequences of these ideas

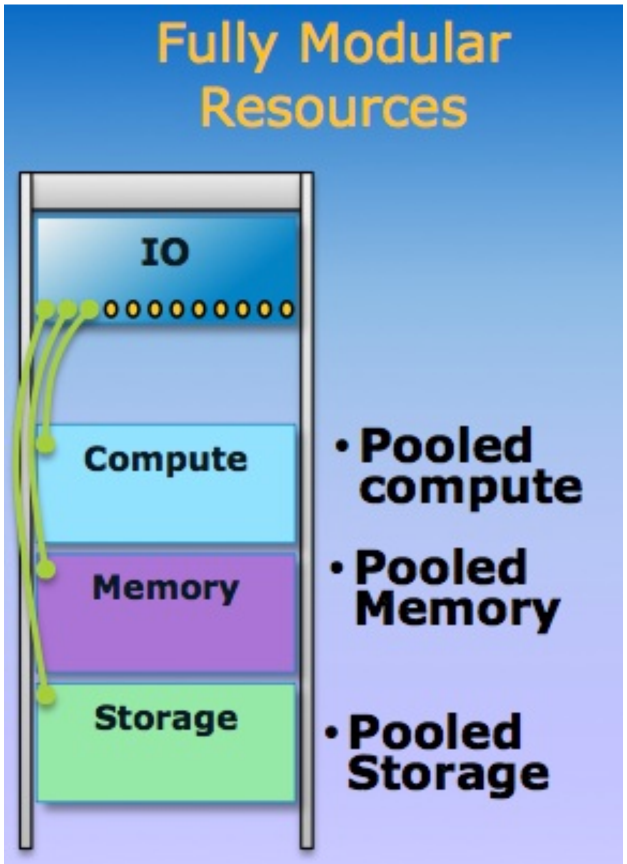
- The goal posts are moving in terms of what we design systems for.
- Human costs associated with running our systems are a bigger expense and inconvenience, at all levels, than the piecewise performance of components.
  - They are actually a barrier.
- The end of scarcity marks the beginning of a push for efficient predictability.
  - This is why storage customers by flash. It's also a hard systems problem.

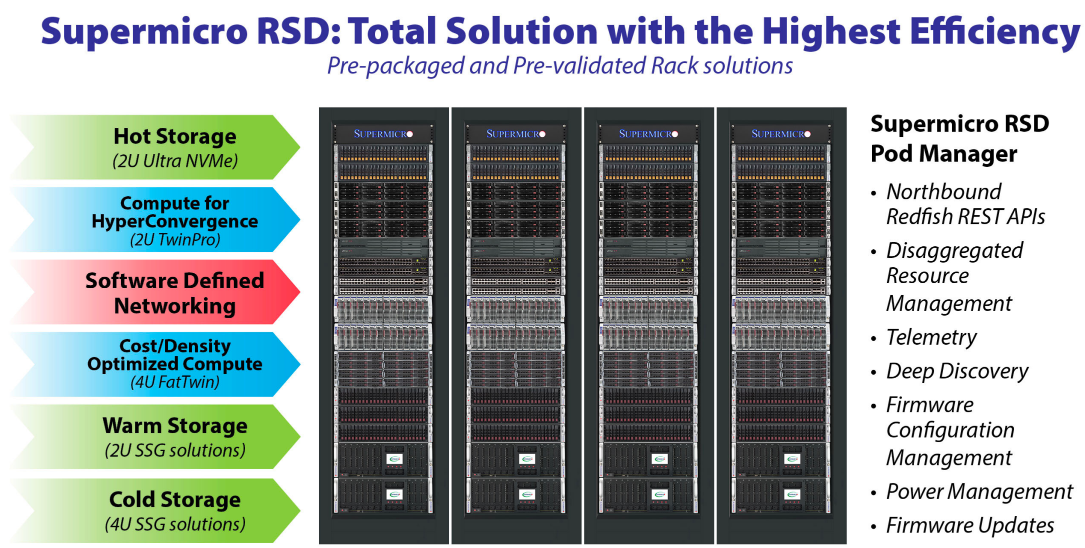
So what do we need to understand, as systems researchers, to help?

One significant hardware chage: Rack scale



This is a google data center circa 2001. GFS (2003): largest deployments had over 1,000 storage nodes, hundreds of clients, 300 TB of storage space





https://www.supermicro.com/solutions/SRSD.cfm

http://itq.nl/intels-take-open-compute-project-rack-scale-architecture/

What is "rack scale"?

- **Everything in a rack will share a high performance bus.**
  - Within a rack, optical interconnects are expected to reach terabit bandwidth in the near term with sub-microsecond latencies.
- **The server as we know it will be completely disaggregated.**
  - CPUs, GPUs, storage, network interfaces, and volatile memory will each move to independent physical enclosures. Arbitrary composition and independent scale.
- **Rack resources will be very dense.**
  - Like, *really* dense.
  - As a ballpark, within a rack we are likely to see thousands of cores, tens of petabytes of persistent memory, and terabytes of RAM.

- In short, a single datacenter rack with a capital value in the low millions of dollars, will be as capable as entire first-generation (e.g. 2003-era) "warehouse" datacenters from public cloud providers

Consequences of the rack scale trend on software design.

What’s changing?

1. Storage is becoming dense.
   - Problematically dense!
2. The memory hierarchy is having an identity crisis.
3. Application latency is a cruel taskmaster.

Trend 1: **Dense nonvolatile storage capacity.**

Dense Nonvolatile Capacity

- Flash vendors have finally started to relax about the durability problem.
- The jaw dropping bit: we will see 4PB in 1u in a small number of years.
  - At a price that approaches spinning disk.
- The bad news: in the immediate term, interconnection will be a problem.
  - And in the longer term it may not get a whole lot better.

Trends

SSD	Cap / 1u	Xput per data
2 TB	64TB	312MB/s/TB
8 TB	256TB	78 MB/s/TB
32 TB	1PB	20 MB/s/TB
128 TB	4PB	5 MB/s/TB

Trends

SSD	Cap / 1u	Xput per data
2 TB	64TB	312MB/s/TB
8 TB	256TB	78 MB/s/TB
32 TB	1PB	20 MB/s/TB
128 TB	4PB	5 MB/s/TB

NVMe device: x4 PCIe
Broadwell CPU: 40 PCIe lanes

Trends

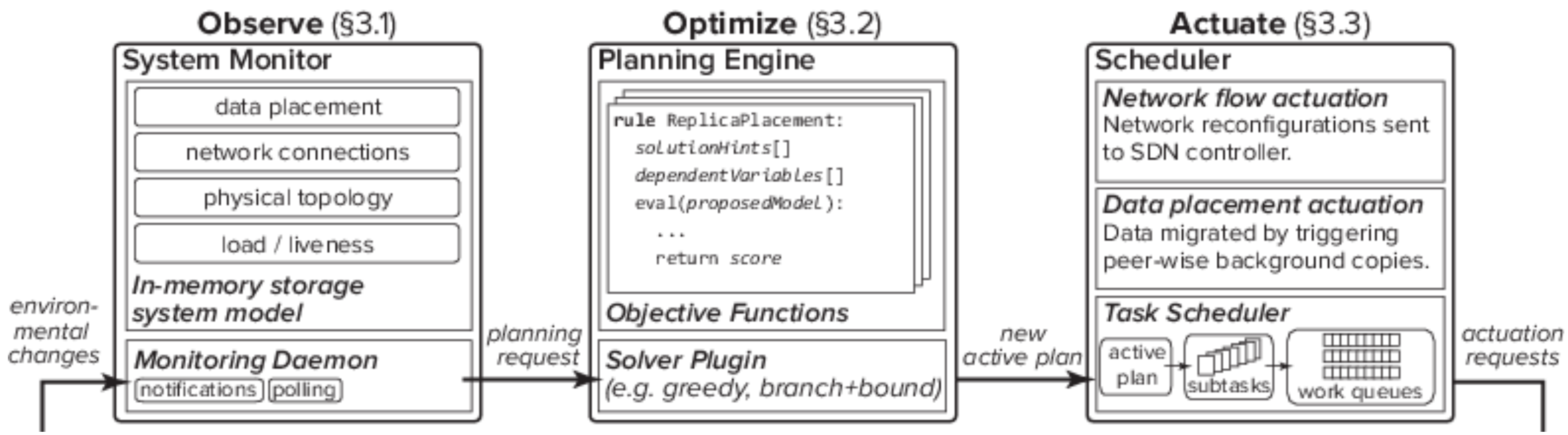
SSD	Cap / 1u	Xput per data
2 TB	64TB	312MB/s/TB
8 TB	256TB	78 MB/s/TB
32 TB	1PB	20 MB/s/TB
128 TB	4PB	5 MB/s/TB

NVMe device: x4 PCIe
Broadwell CPU: 40 PCIe lanes
TOR cross-rack links typically oversubscribed at 3 or 4:1

This is very different from all the storage systems that we've built in the past.

- **No seek penalty.**
  - Means that background I/O is actually reasonable to do.
  - Migration for performance.
  - Alternate representations (e.g. materialized views, intentional DUPlication) often for performance.
  - Metadata all day long.
- **Sprinkler heads are a problem.**
  - 4PB is an awfully scary failure domain.
  - Sensible application of erasure coding needs five or more nodes.
  - East/west traffic is constrained.
- **Capacity-motivated deletion is silly in most cases.**
  - But real deletion probably needs to be encryption based.

Mirador (FAST '17)



Centralized three-stage pipeline continuously optimizes placement

Trend 2: **The magic of persistent memory.**

Persistent Memory

- Everyone is excited about 3D Xpoint.
  - (What the heck is 3d xpoint?)
- Bad news: persistent RAM is a total PITA.
  - Because it's not really persistent RAM: ram as you think about it is a total ilusion.
  - It's really a super duper fast disk.
  - In fact, it's a super duper fast *single* *unreliable* disk. But more on this in a sec.
- But wait, this doesn't mean that XPoint isn't a spectacularly good idea.
  - With it, RAM is about to break through the memory wall (core to capacity ratio).
  - Technologies like XPoint will give us a multiplier on working set.
  - Persistence will massively speed up restart times, especially for read-only data.

One more spanner: Disaggregation.

- Some significant amount of memory is about to move off host.
- Nobody seems to agree on how this is going to happen.
  - "remote" memory vs shared physical bus vs Rack-scale CC NUMA
- All of these things are interesting in two ways.
  - First, failure domains are very different... in ways that Apps and OSes are NOT used to reasoning about.
  - Second, they afford an entirely new (and exciting!) form of dynamism.
    - Map Reduce and Spark have a good but very coarse-grained notion of partitioning.
    - These systems have the potential to be so much more dynamic.
    - Same for scale out data stores.
    - Same for state replication and HA

So what's going to happen here

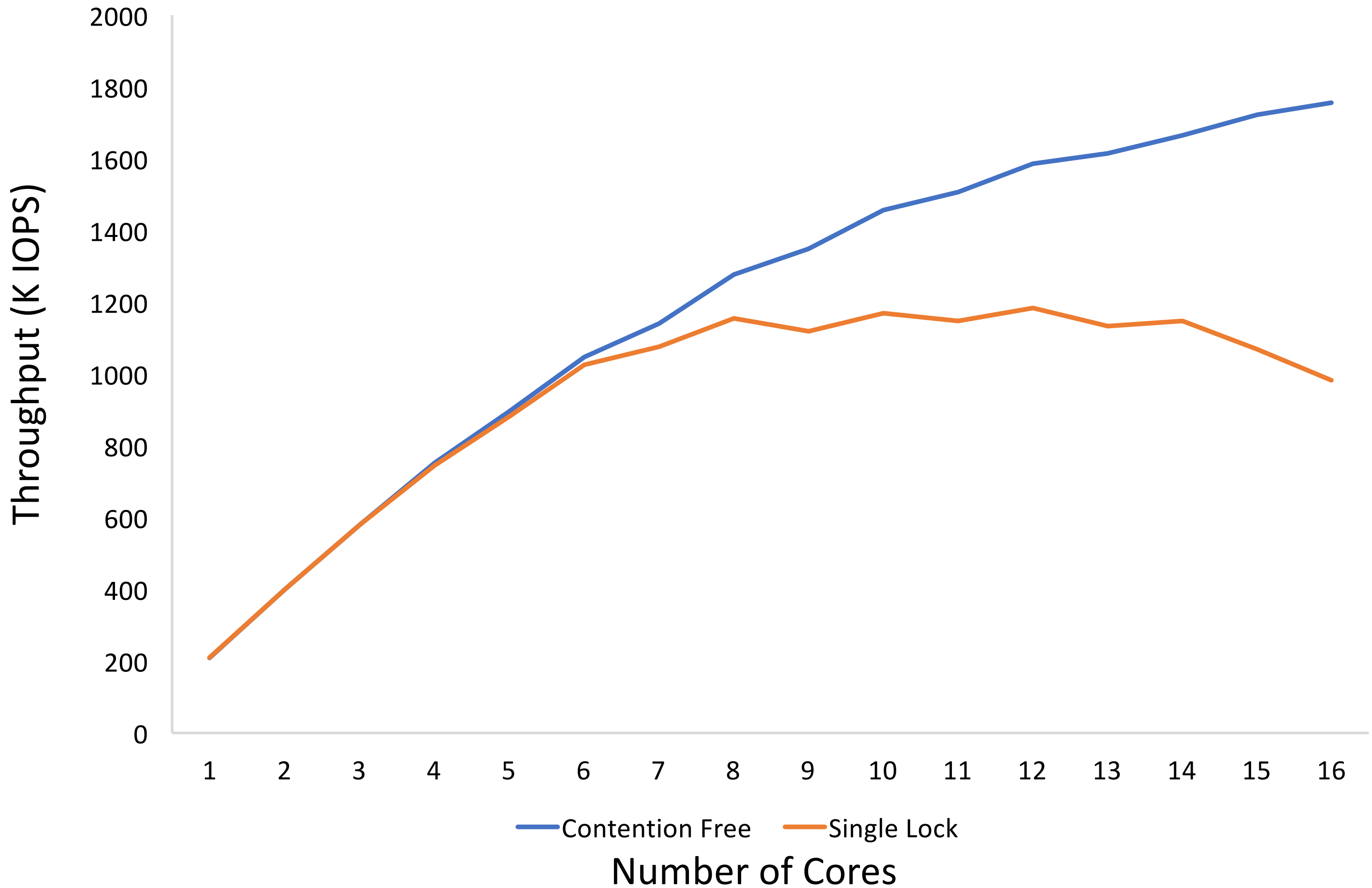
- Total chaos.
- Persistent memory looks like a really fast disk. Disaggregated memory looks like an extension of the cache hierarchy.
- Our view of memory, locality, and persistence is in trouble.
- Interfaces and abstractions really need to change in support of this.
- One prediction: file system and virtual memory will merge.
  - Loads of reasons to do this -- serialization overheads, reboots, sharing.
  - but still many open questions.

Trend 2: **Application latency.**

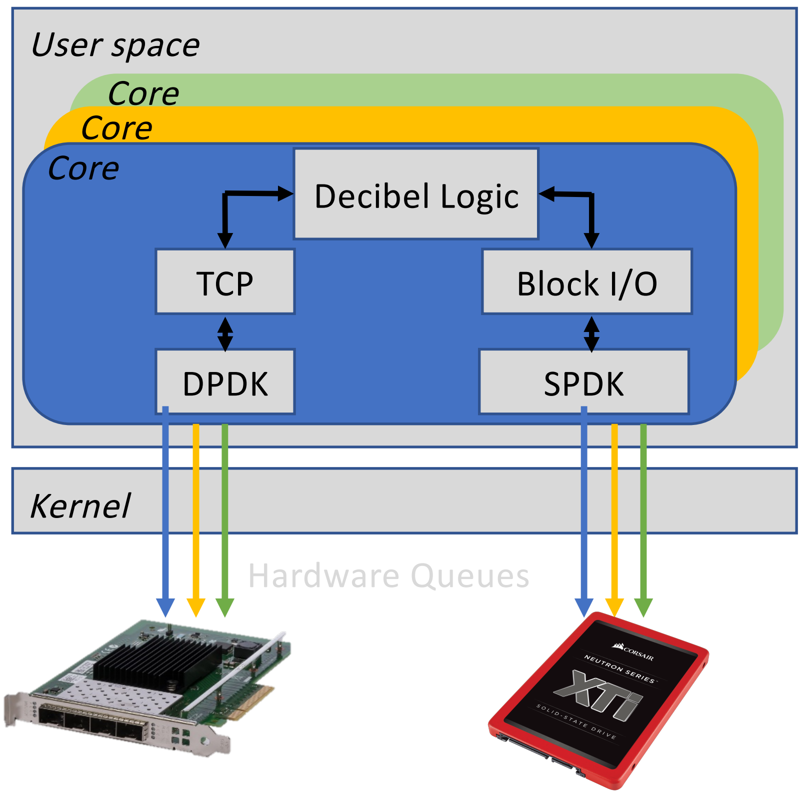
Latency

- Tell me if you’ve heard this one before: CPUs aren't getting faster
- I/O is getting faster and wider.
- Latency is becoming a dominant metric.
  - Direct impact on e.g. purchase probability.
  - But it's a much harder metric to work with than throughput.
- Shrinking I/O latencies results in increased computational density.
  - Because I/O wait goes away (e.g. DBMS)
- But a latency focus imposes a lot of constraints on software design.
  - Especially tail-latency SLOs.
  - Need to reason about the slow path as a common case.

THE COST OF CONTENTION

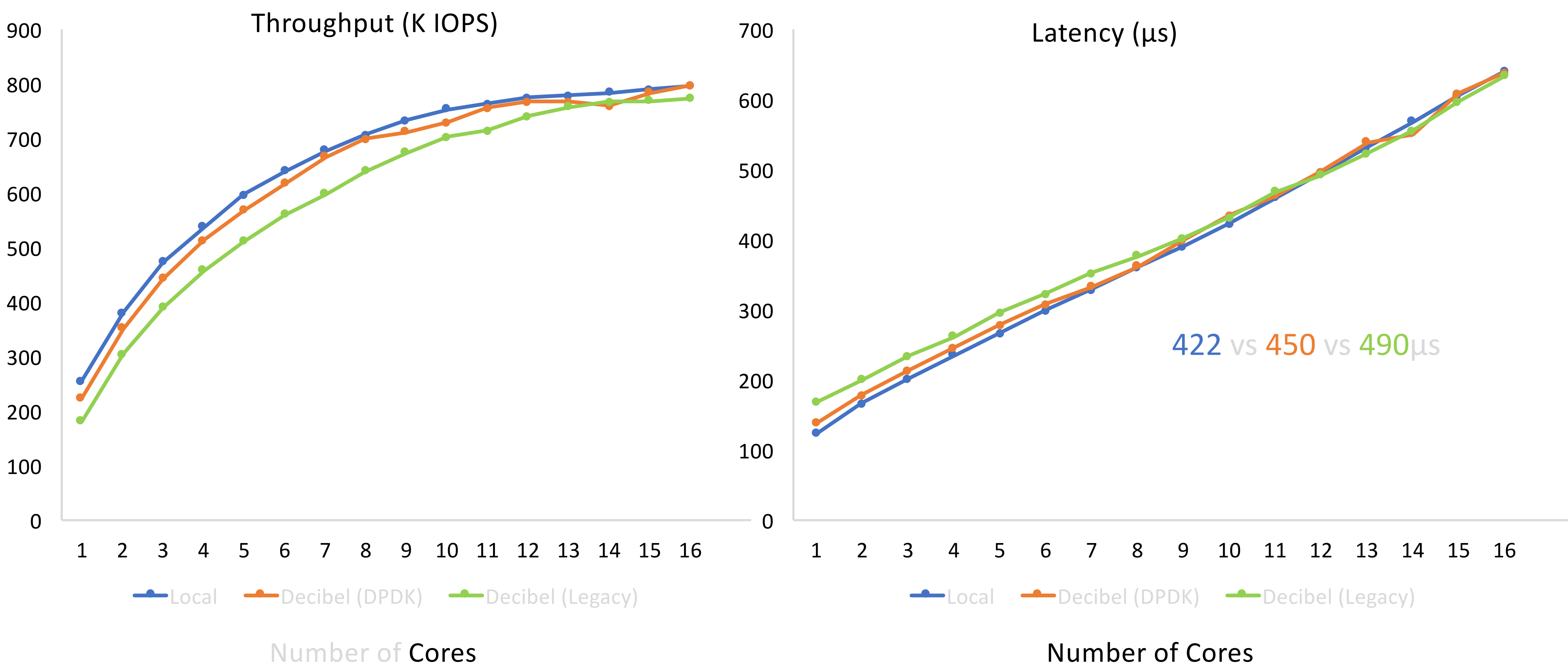


Decibel (NSDI '17)



- How should we structure a storage system to provide virtual local disks?
- Partition like crazy, crusade against latency, push all unnecessary functionality up the stack.
- This generalizes to applications.

Decibel Performance (70/30 Mixed Workload)



Everything hurts latency

- Redundancy is a good example of why this gets hard.
  - For in memory, network RTT will approach media store time.
  - So a remote write doubles the cost.
  - Worse: Replication at lower layers of the system is invariably amplified.
  - This is why emerging data stores don't do it.
- A real latency focus drives software architecture in a very specific direction.
  - Contention is a source of hard-to-reason-about performance variance.
  - So avoid contention at all costs. Design it out up front.
  - (If you do this right, you benefit from not having to hire developers that understand locking.)
  - Doing this right means designing data and code-level partitioning very carefully.
  - Less academically rewarding than OCC and lock freedom, but see parenthetic point above.

And with that, I’m mostly done.

Here are the high-level trends/ideas

1. Diminishing scarcity.
2. Practical/sensible to own your own hardware again.
3. Software needs to change.

Closing thought.

- Nobody is going to adopt your stuff unless you make it as easy as heck for them to do it.
- Expose your research results as a service, or as something as close to a service as is possible.
  - Put them in containers, host them on AWS.
- Solve application problems.
  - Early experiences working with physical scientists.