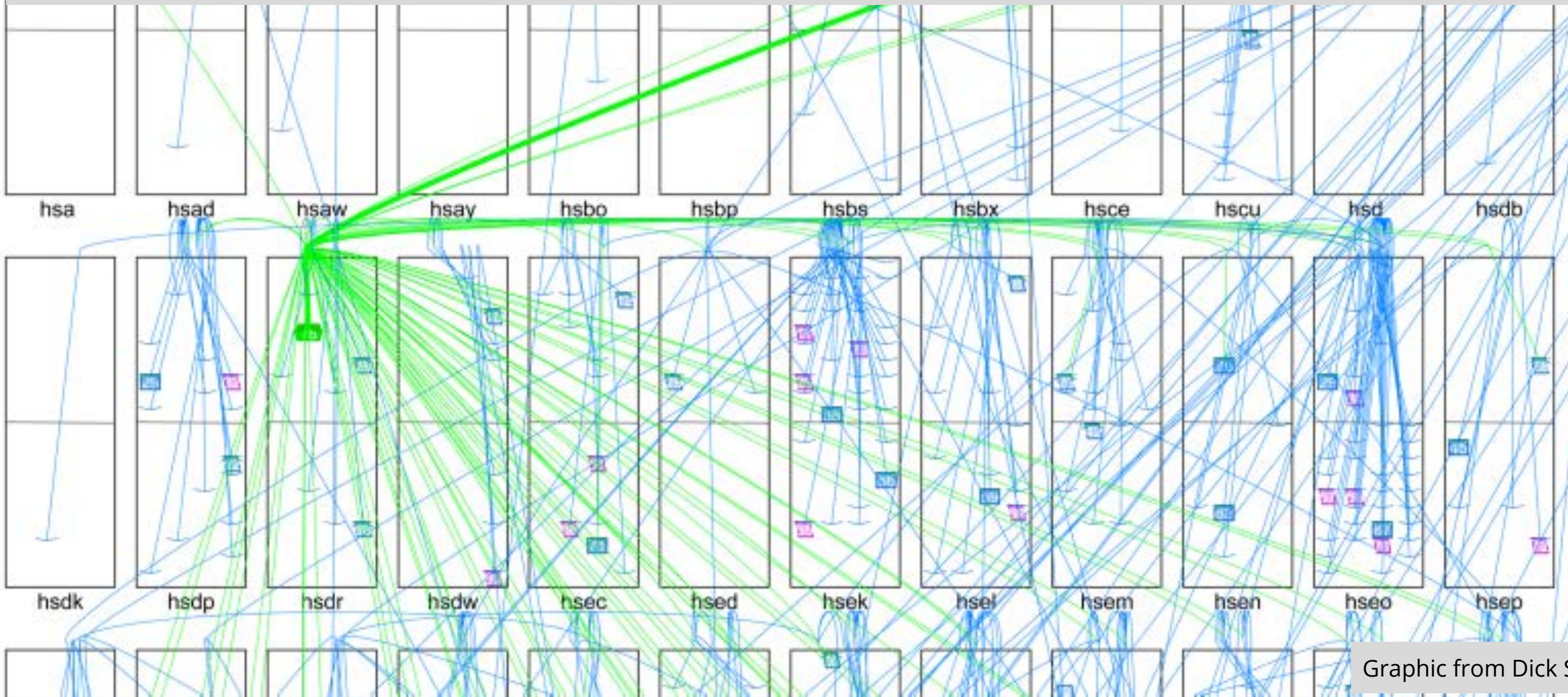
Think outside the rack

2015-04-21 WRSC

john wilkes / johnwilkes@google.com, Parthasarathy Ranganathan, Steven Hand
Google Inc.

Datacenter loads are not SPEC benchmarks

Single query across multiple racks of multiple servers: graph of one query and associated RPCs for work distribution (only two levels shown); other queries going on, but not shown.



Graphic from Dick S

Good news! lots of new technologies

Silicon/hardware is getting ever more inventive

- forced to move to parallelism to track Moore's "law"

Main memory: lots of volatile RAM, new non-volatile h/w
Computation: oodles of cores, specialized accelerators
Storage: flash/SSD, [magnetic disks still kicking]
Networking: high bandwidth + low latency + lossless(?)

Good news! resource disaggregation

Conceptually it's wonderful:

- Build a "rack computer" from a kit of parts (*)
  - a single big, disaggregated machine
  - all the benefits of a unified OS

- Build a "datacenter in a rack" (*)
  - a single, scaled-down distributed system, like the big guys use
  - all the benefits of shared-nothing distributed systems

(*) OK - build at least two, for reliability

Good news! a RackScale foo is *both* of these

Upsides:

- meet all the needs of all but the largest organizations

- buy just what you need (save money)
- build just what you want (go fast)
- tune for peak performance (go fast; save money)

- conceptually similar to existing programming models

What could *possibly* go wrong?

An RSfoo breaks *everything*

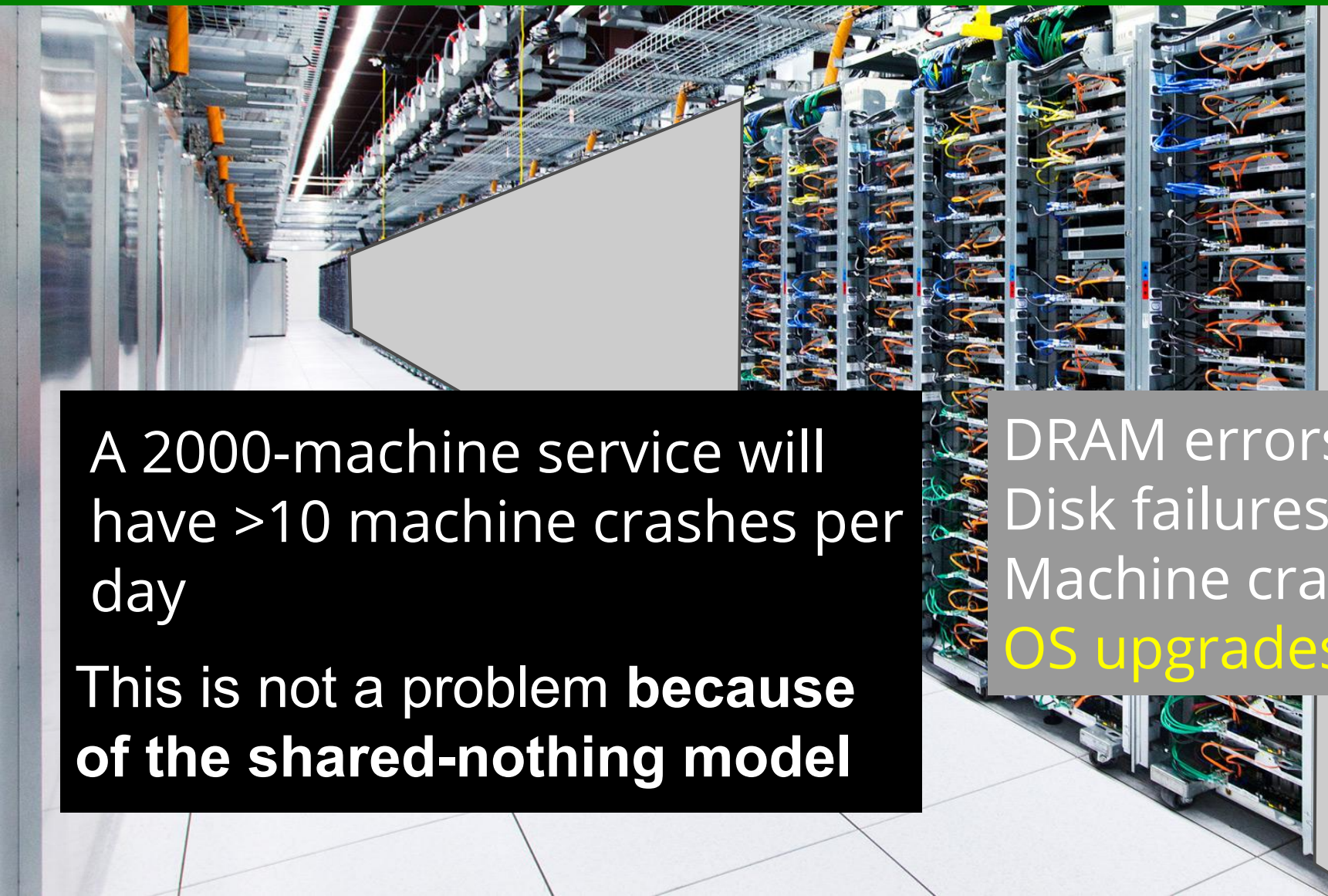
An RSfoo is *not* the same as a computer

- multiple internal failure domains
- non-uniform resource access costs

An RSfoo is *not* the same as a datacenter

- shared nothing => disaggregated resources
- existing programming models don't work

Datacenter experiences are relevant



A 2000-machine service will have >10 machine crashes per day

This is not a problem **because of the shared-nothing model**

DRAM errors (1% AFR)
Disk failures (2-10% AFR)
Machine crashes (~2/year)
OS upgrades (2-6/year)

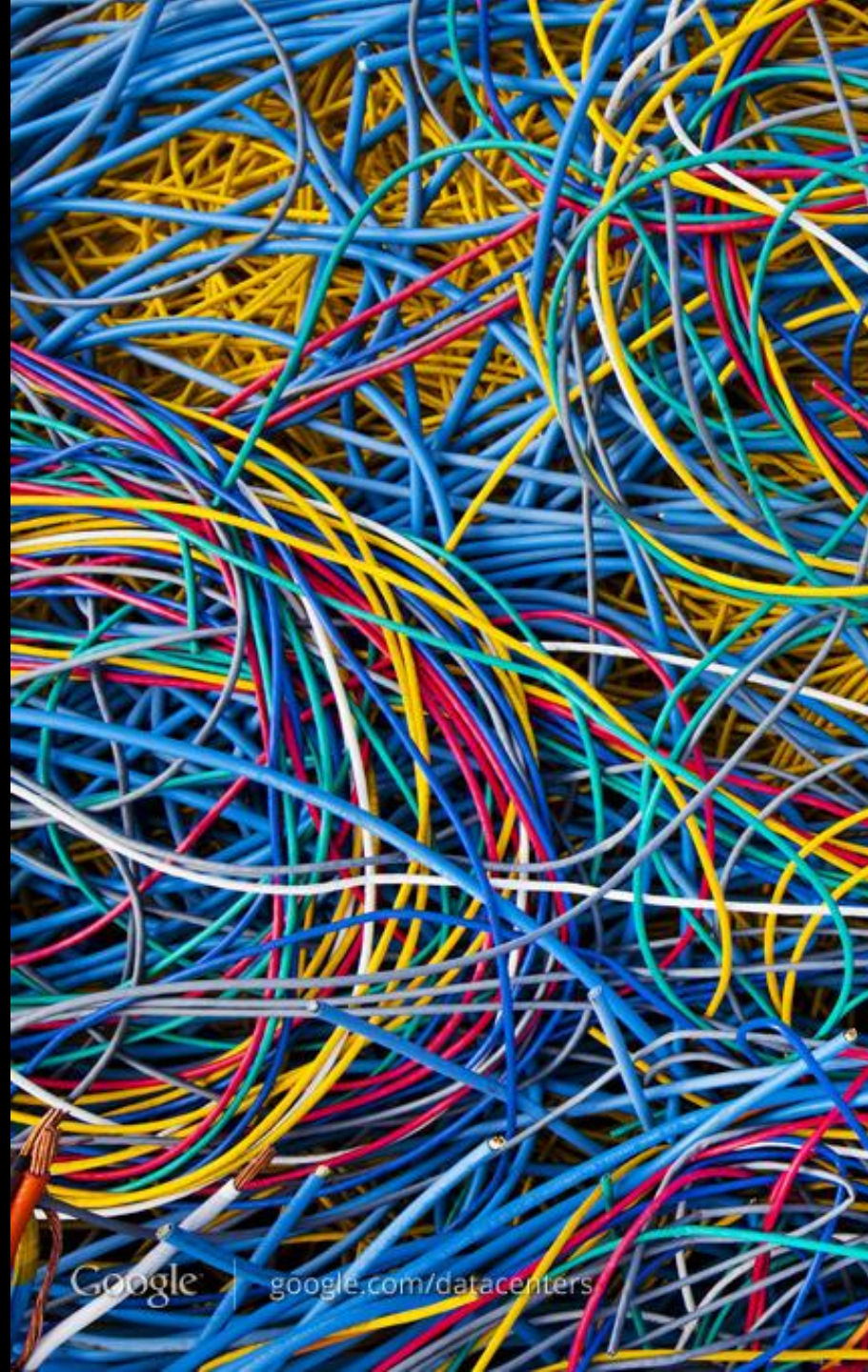
Images by Connie Zhou

RSfoo failures

If disaggregation is used

- each component failure
  ⇒ partial system failure
  ⇒ visible at the app level
- fault propagation at the speed of light

Apps aren't designed to handle this today

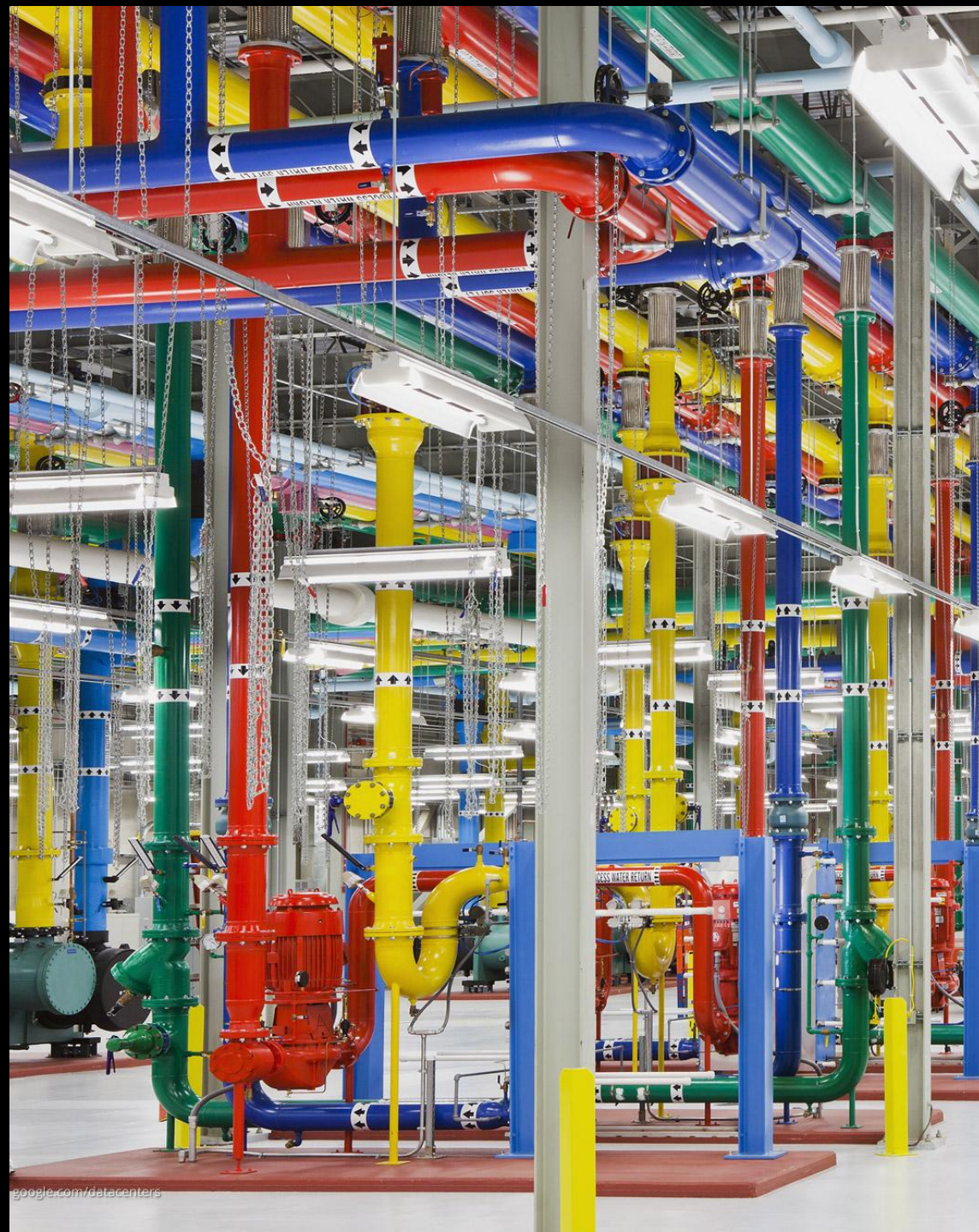


RSfoo provisioning

How much of what to buy?

- workload lifetime << hardware depreciation cycle
- multiple esoteric resources
- requires (dynamic) hardware evolution

Apps + planning tools aren't designed to handle this today



RSfoo placement/scheduling

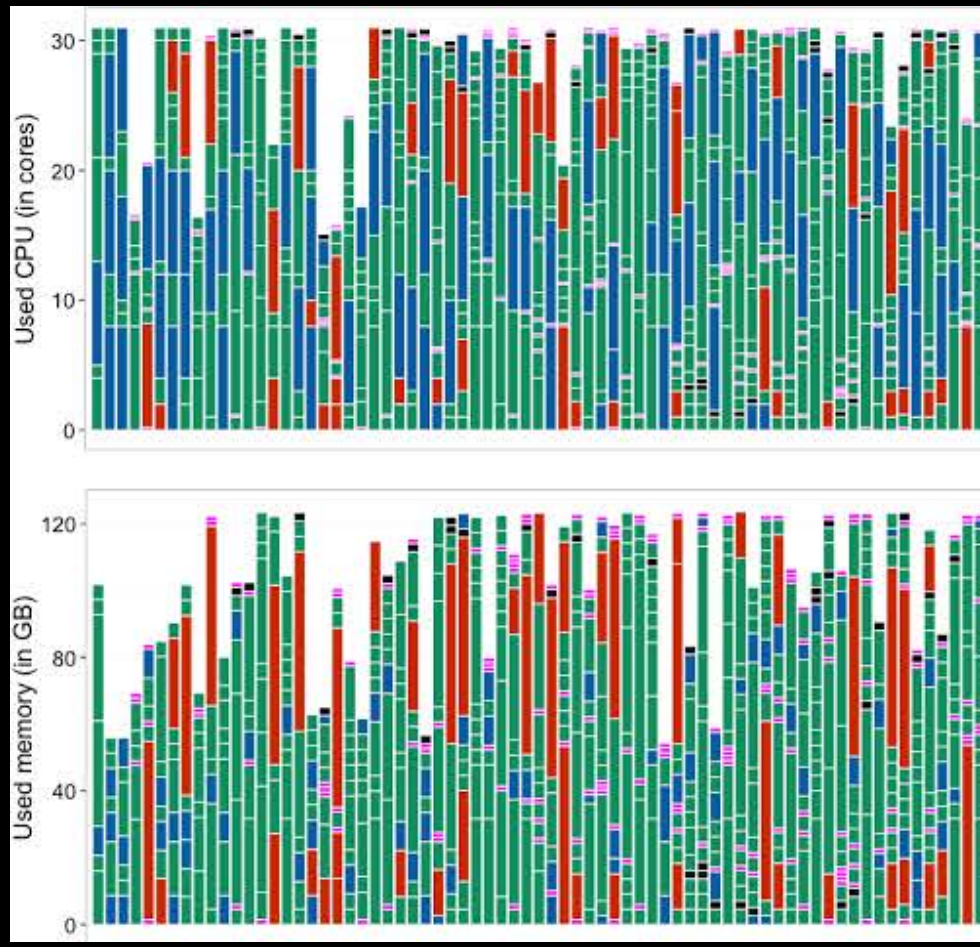
Avoid resource stranding

- disaggregation helps ...
- but RSfoo has more resource types

Avoid bad placement

- NUMA writ large
- dynamic interference

Existing placement / scheduling algorithms aren't good at this today



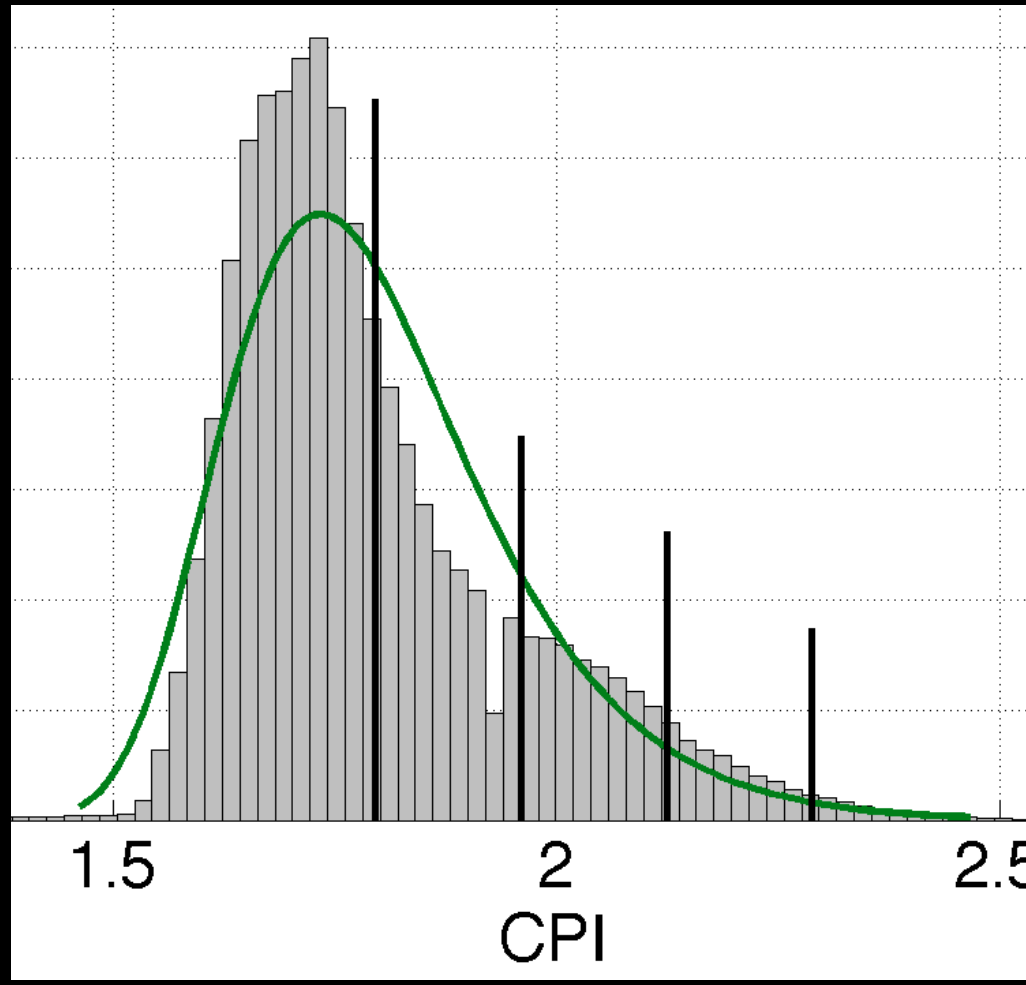
RSfoo inter-application interference

RSfoos are small-scale datacenters, so **will** run multiple apps

Disaggregated resources make ...

- *performance* isolation much harder
- *security* isolation much harder
- *failure* isolation much harder

Apps + systems aren't designed to handle this today



RSfoo groups

You still need multiple RSfoos ...

- control-plane failure
- datacenter / network / environment failure
- "big" workloads
- end-user latency

Existing inter-datacenter solutions (e.g., full replication) probably aren't ideal

Good news!

We'll have **job security** ;-)

Good news!

The solutions are in sight.

The problems are just beginning.

- Failures
- Provisioning and configuration hassles
- Interference
- Multi-RSfoo support

One possible approach

For each feature/property/behavior, start by asking:

- "is this a big computer, or a small datacenter?"
- (distributed systems techniques go a long way)

Thinking about timescales may help:

- seconds and up - datacenter control model
- below that: application-level

Introduce a feature *after* addressing issues identified here

- don't forget the programming model!