EC/CS 528: Cloud Computing

Resource Management and Sprint Demos

Instructor: Alan Liu

BU Department of Electrical & Computer Engineering

BOSTON UNIVERSITY

Announcements

- Talk about OpenShift/Kubernetes from Daniel McPherson: Wednesday, Nov. 16, 2022

- Reaching out for mid-term

Framework Isolation

- Mesos uses OS isolation mechanisms, such as Linux containers and Solaris projects
- Containers currently support CPU, memory, IO and network bandwidth isolation
- Not perfect, but much better than no isolation

BOSTON UNIVERSITY

Analysis

- Resource offers work well when:
  - Frameworks can scale up and down elastically
  - Task durations are homogeneous
  - Frameworks have many preferred nodes
- These conditions hold in current data analytics frameworks (MapReduce, Dryad, …)
  - Work divided into short tasks to facilitate load balancing and fault recovery
  - Data replicated across multiple nodes

BOSTON UNIVERSITY

Large-scale cluster management at Google with Borg

By: Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, John Wilkes

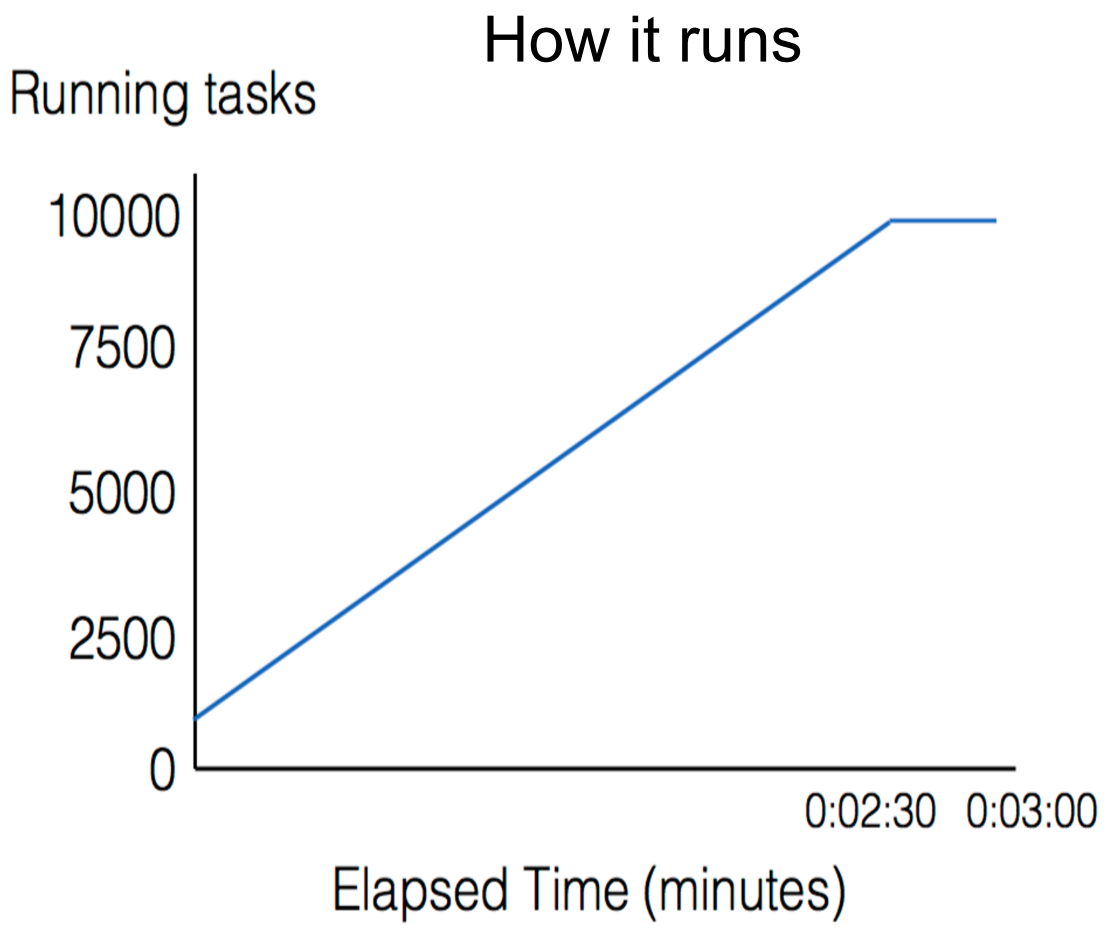
Google Inc

BOSTON UNIVERSITY

Example

```
job hello_world = {
  runtime = { cell = 'ic' }                    // What cluster should we run in?
  binary = '.../hello_world_webserver'    // What program are we to run?
  args = { port = '%port%' }                 // Command line parameters
  requirements = {      // Resource requirements
    ram = 100M
    disk = 100M
    cpu = 0.1
  }
  replicas =  10000   // Number of tasks
}
```

Example



BOSTON UNIVERSITY

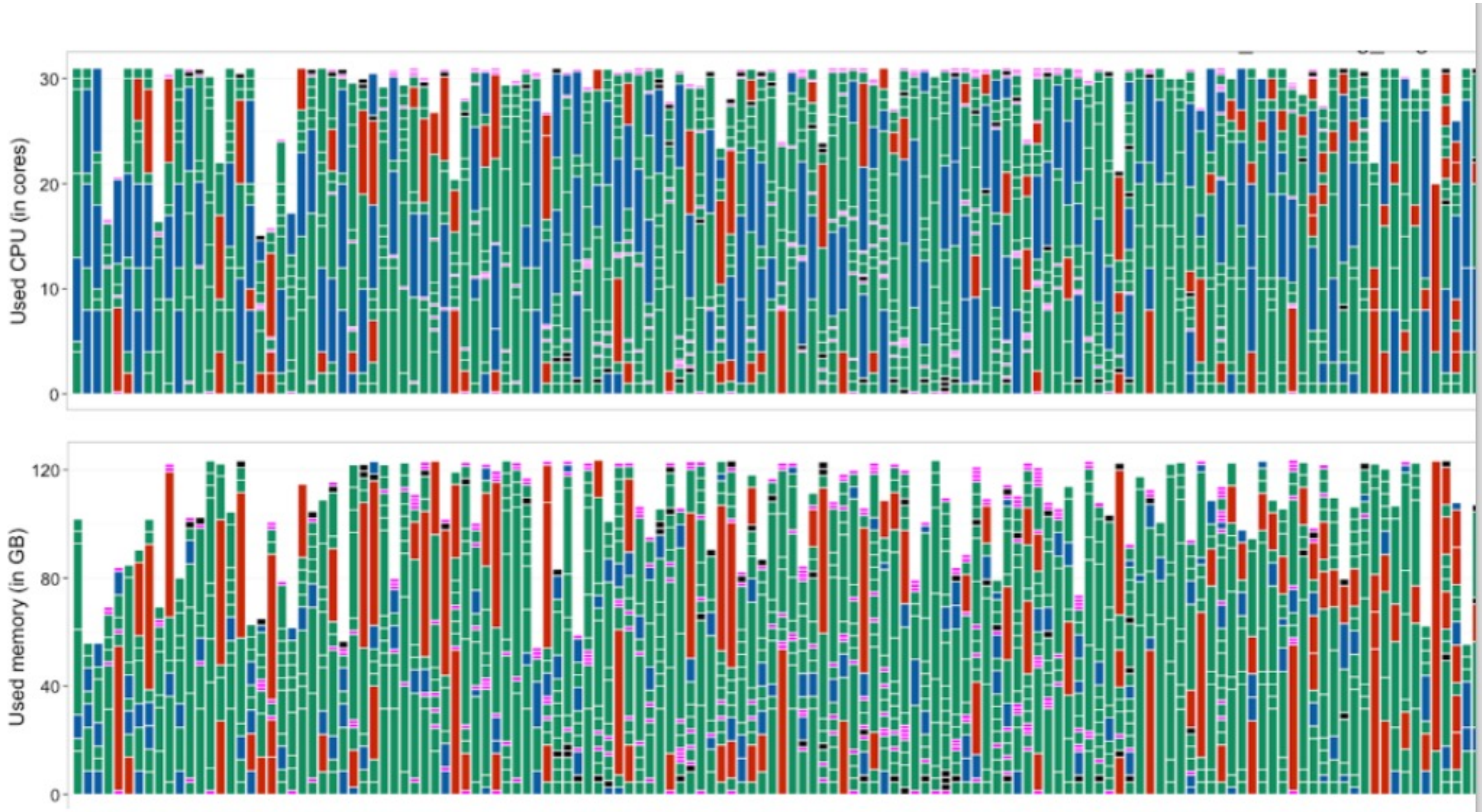
Main Benefits

- Scalability to thousands of machines, efficiently shares the machines
- Abstracts away details of resource management, monitoring, fault handling from users
- Operates with high reliability and availability

Borg comparison to other resource managers

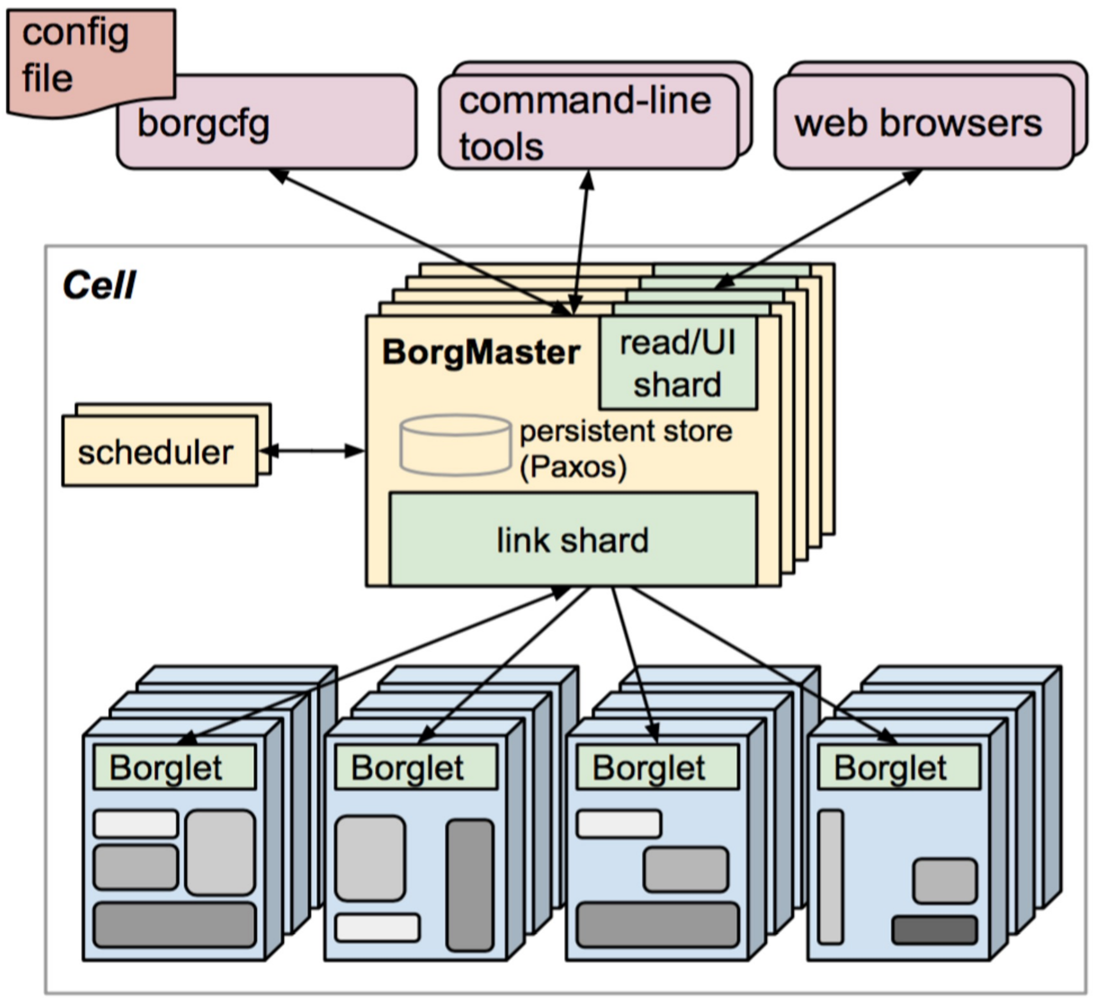
- Infrastructure for scale, packaging…
- Containers
- Master that distributes work

Efficiency: multiple resources



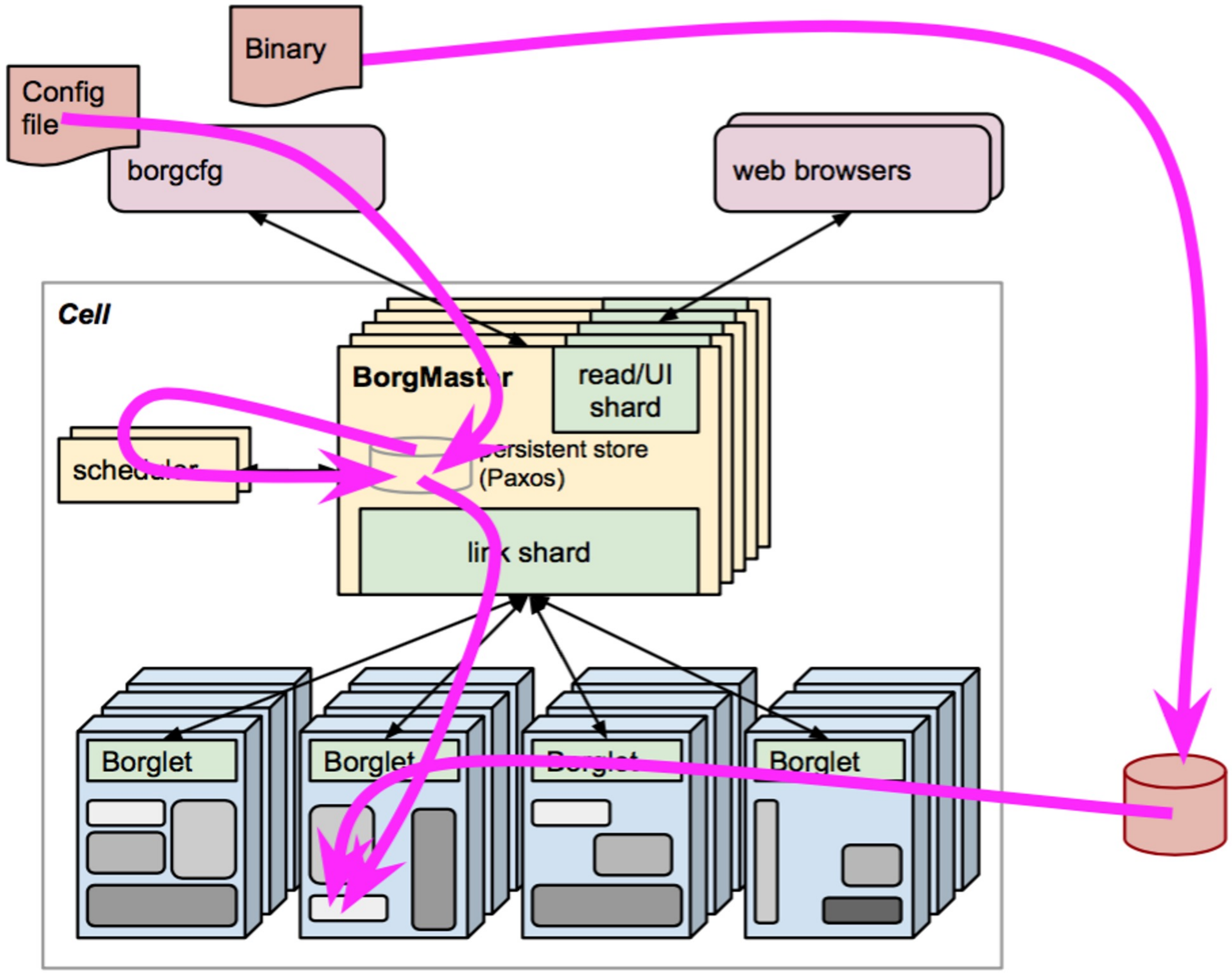
BOSTON UNIVERSITY

Architecture



BOSTON UNIVERSITY

Control Flow



BOSTON UNIVERSITY

Abstractions

- Cell – group of tightly coupled nodes
- Job – name, owner, collection of identical tasks
- Task – set of linux processes running in a container
- Allocs and Alloc sets - allocation of resources on one machine, or for one job

BOSTON UNIVERSITY

Abstractions

- Priority – relative priority of jobs running or waiting to be run
  - Higher will preempt/kill lower
  - Production do not preempt each other
  - Bands: 1) monitoring, 2) production, 3) batch, 4) best effort
- Quota – used to decide which jobs to admit for scheduling; max resources job can ask for
  - Users tend to buy more than they need
  - System sells more lower-priority than it has
  - Systems tends to be oversubscribed

BOSTON UNIVERSITY

Characteristics of infrastructure

- Medium cell size around 10K machines
- Heterogenous in resources (core, memory, storage) & capabilities, e.g., external IP

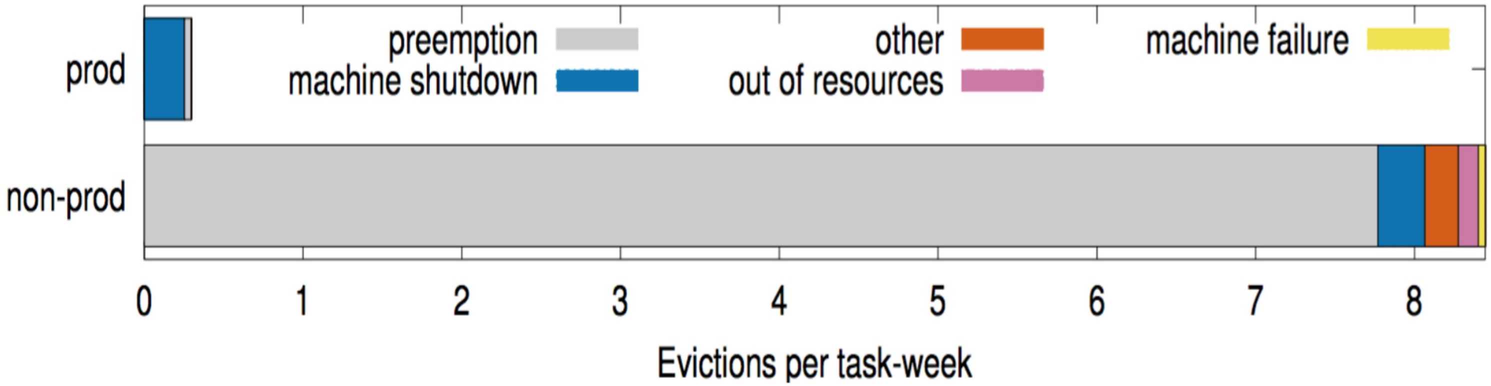
Characteristics of workload

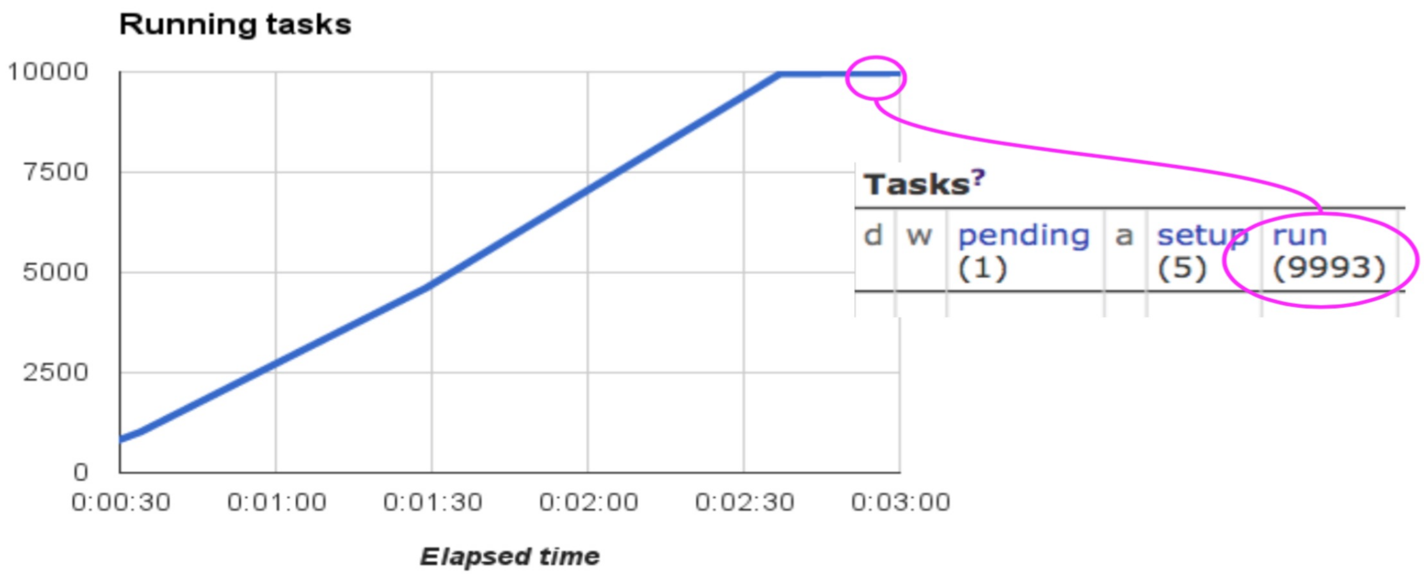
- Long running services
  - Latency sensitive, e.g., Gmail, Google docs, search, BigTable
  - Many frameworks (MapReduce, Pregal…)
  - Diurnal usage pattern
  - Usually controller with master job, one or more worker job
  - Runs as Prod
- Batch jobs
  - Take from seconds to days
  - Insensitive to performance fluctuations

Characteristics of app execution environment

- Everything run in container
- Binaries are statically compiled: dependencies
- Tasks have built in HTTP server
  - Health, and thousands of performance metrics
  - Borg can monitor, and restart tasks don’t respond
- Tasks assumed to handle failures

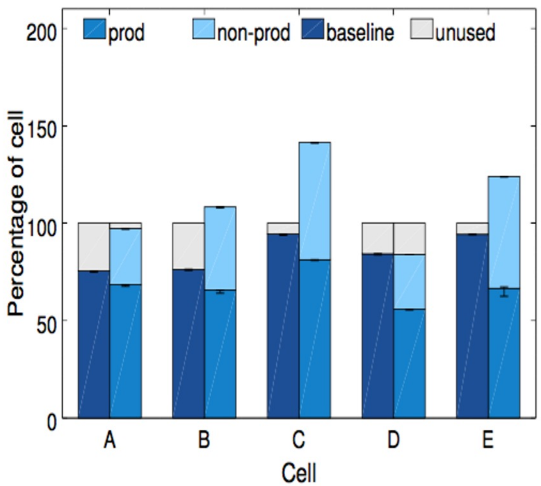
Availability



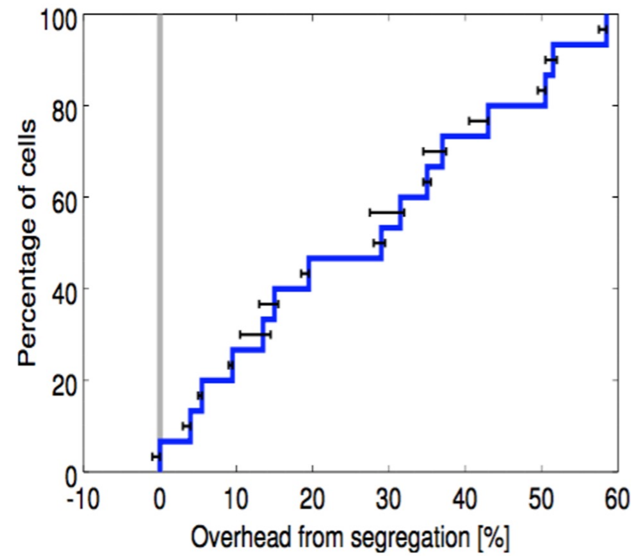


Utilization

Cell sharing: Segregating prod and non-prod work into different cells would need more machines



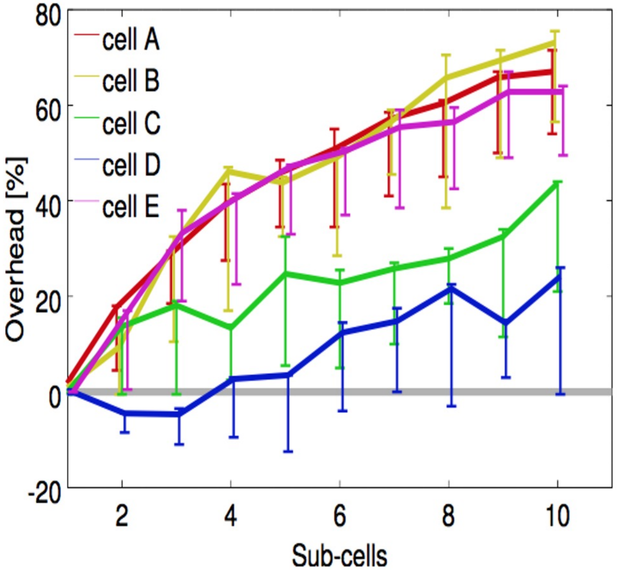
(a) *The left column for each cell shows the original size and the combined workload; the right one shows the segregated case.*



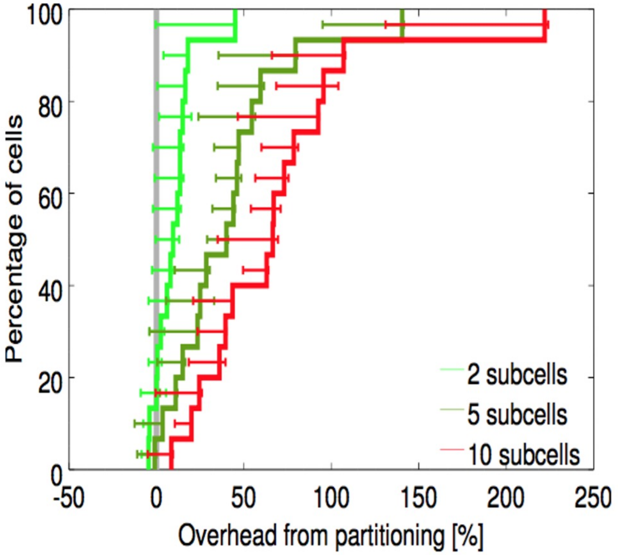
(b) *CDF of additional machines that would be needed if we segregated the workload of 15 representative cells.*

Utilization

Cell size: subdividing cells into smaller ones would require more machines



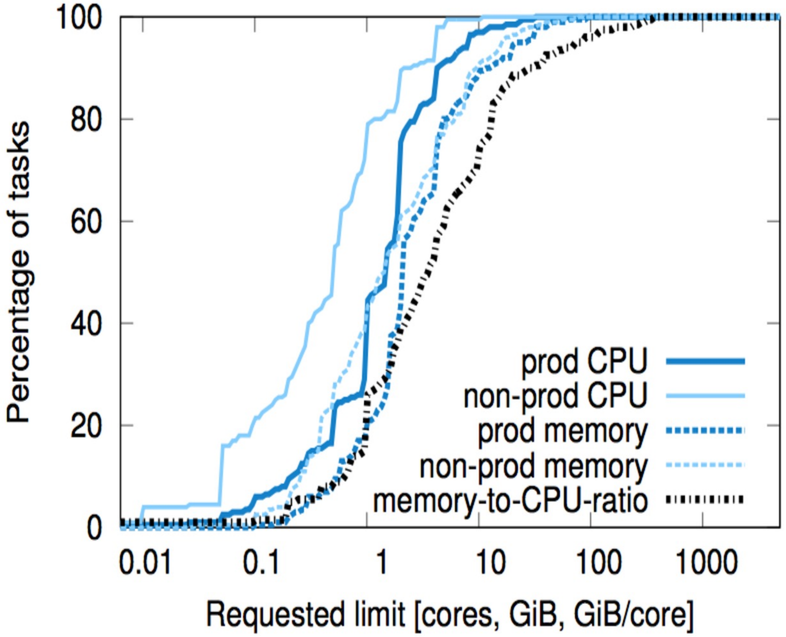
(a) *Additional machines that would be needed as a function of the number of smaller cells for five different original cells.*

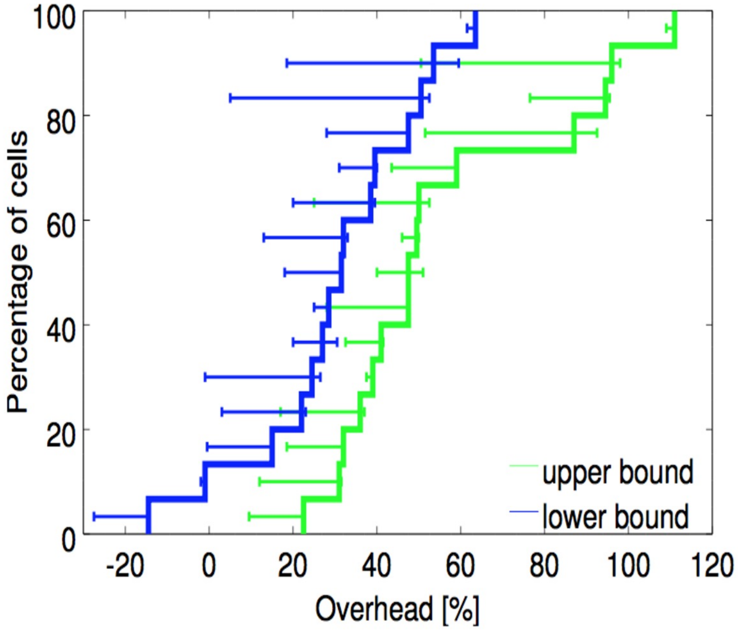


(b) *A CDF of additional machines that would be needed to divide each of 15 different cells into 2, 5 or 10 cells.*

BOSTON UNIVERSITY

Task use/Bucket size





Lessons

- Bad
  - Jobs are restrictive as the only grouping mechanism for tasks
  - One IP address per machine complicates things
  - Optimizing for power users at the expense of casual ones – hard to use
- Good
  - Allocs are useful
  - Cluster management is more than task management
  - Introspection is vital – expose everything
  - The master is the kernel of a distributed system

Comparison to Mesos

- Infrastructure for scale, packaging…
- Containers
- Master that distributes work

BOSTON UNIVERSITY

Differences

- Borg/Kubernetes
  - more prescriptive monitoring…
  - Scheduler that looks at all constraints, more efficient
  - Can schedule arbitrary jobs
- Mesos
  - exokernel like model, exploits/works with frameworks; not arbitrary jobs
  - Doesn’t talk about jobs that require more resources than available
  - Much much simpler
  - Arguably more general
  - Probably more scalable

More thoughts

- Borg wasn't published, and Kubernetes was not created, until after Mesos was published

Building warehouse-scale computers

or … what's it like to supply *exponential growth*

john wilkes 2017-03

Google Cloud Platform

BU Department of Electrical & Computer Engineering

BOSTON UNIVERSITY

$29.4B

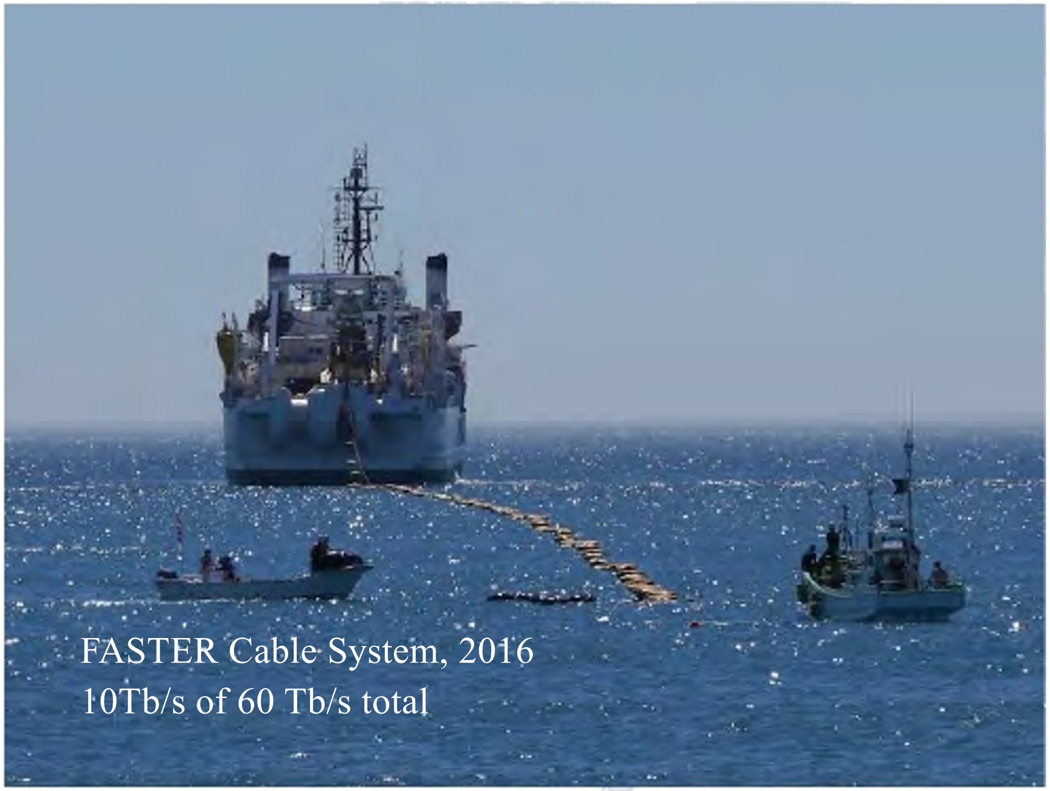
3-year trailing CapEx, as of March 2017

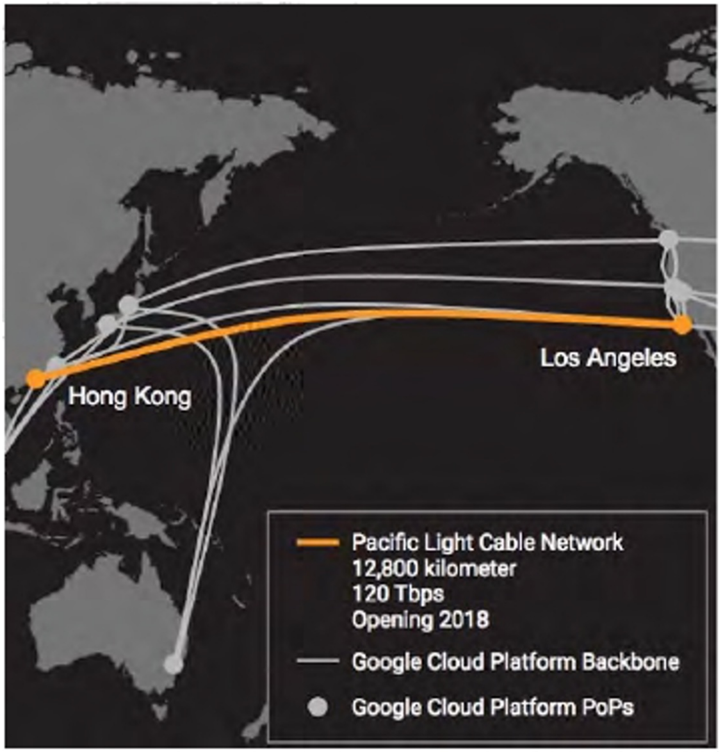
GCP infrastructure

2016-12: 6 regions, 18 zones, over 100 points of presence, and a well-provisioned global network comprised
of hundreds of thousands of miles of fiber optic cable.

Planning for network

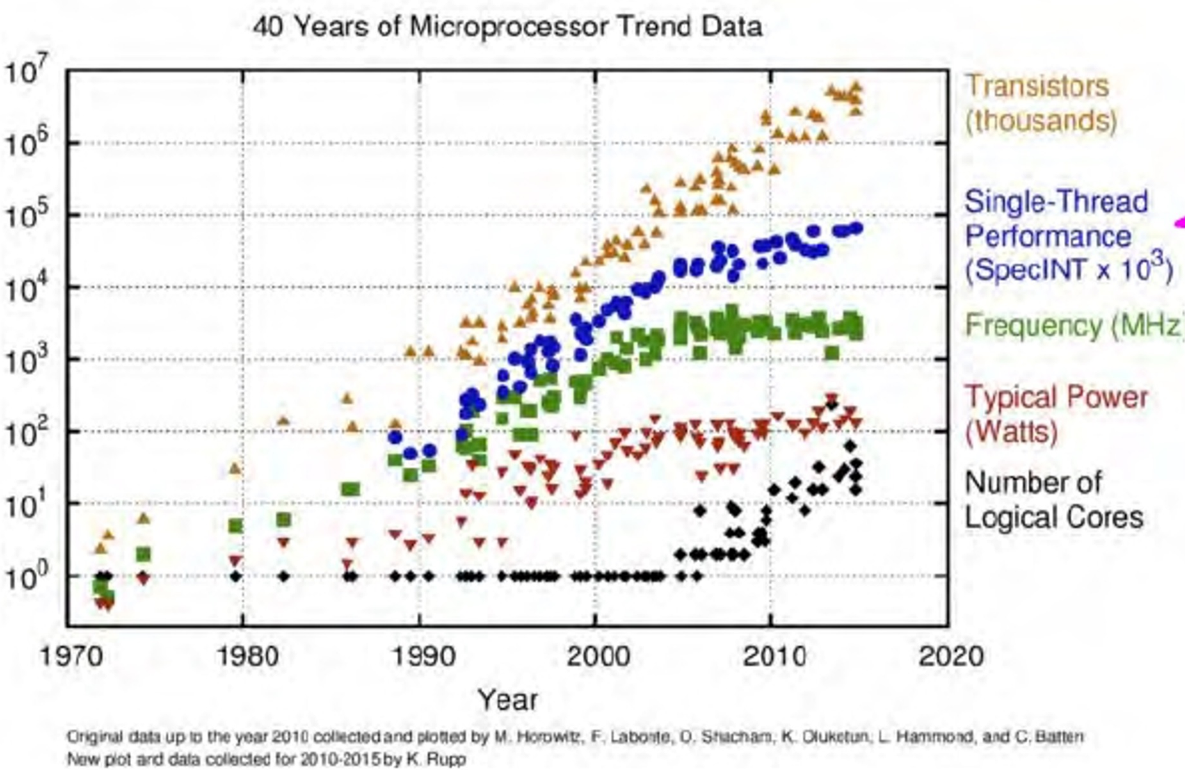
Google network: 12 years to build and still growing





Google Cloud Platform

Meanwhile – what’s up with Moore’s law?



Single-core performance plateauing after decades of exponential growth

Google Cloud Platform

Graph from [40 Years of Microprocessor Trend Data](), Karl Rupp,

Meanwhile – what's up with Moore's law?

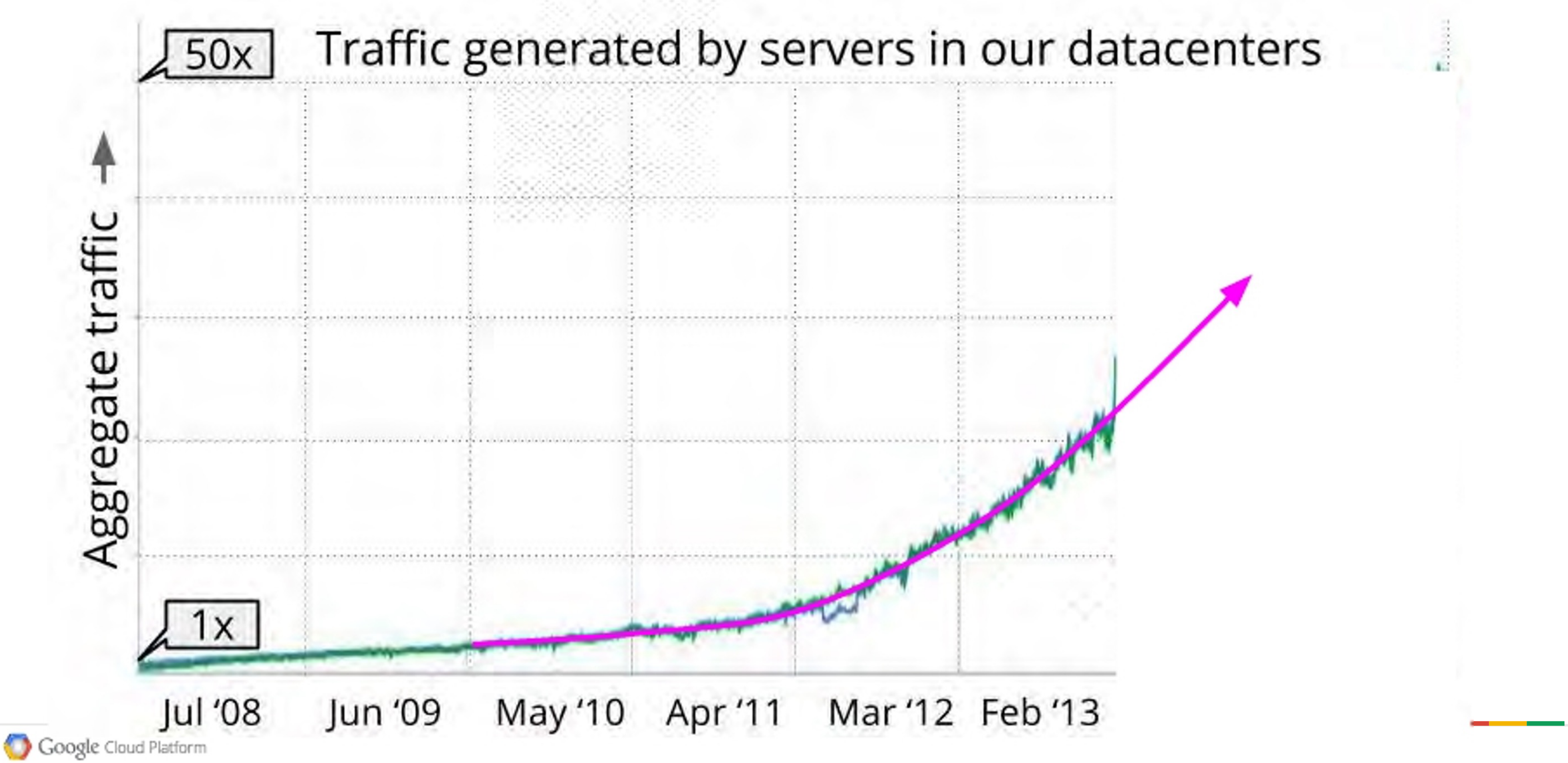
Accelerators to the rescue!

	Intel®Xeon® ProcessorE7-8890 v4CPU	NVIDIAK80GPU (perGPU)	AMDS9300x2GPU (perGPU)	NVIDIAP100GPU
Cores	24 (48threads)	2496 streamprocessors	4096 streamprocessors	3584 streamprocessors
Memory Bandwidth	85GBps	240GBps	512GBps	732GBps
Frequency (boost)	2.2(3.4)GHz	562MHz(875MHz)	850MHz	1.13(1.30)GHz

20% per year: CPU processing power
40% per year: data growth
50% per year: GPU processing power

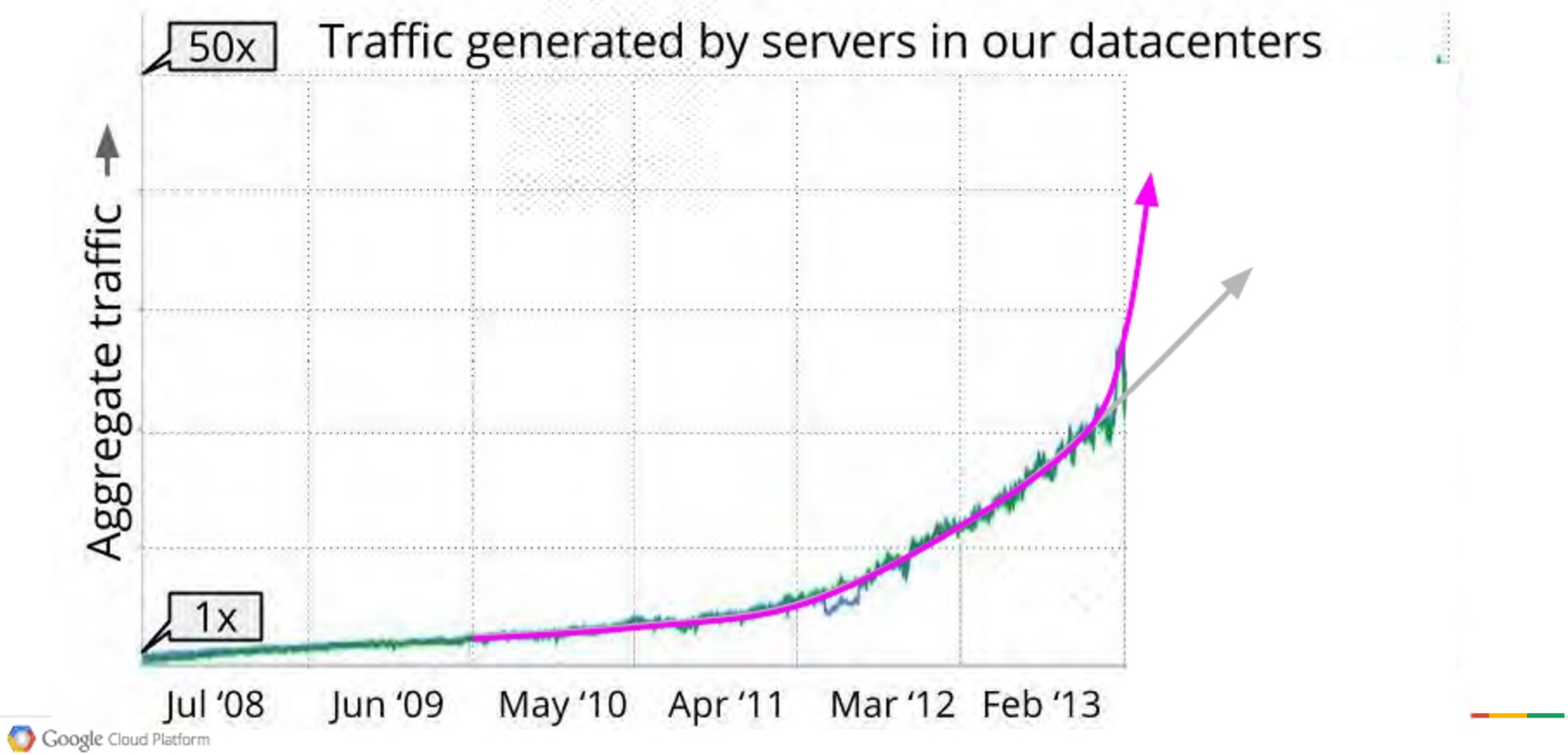
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. SIGCOMM 2015.

How much capacity do we need?



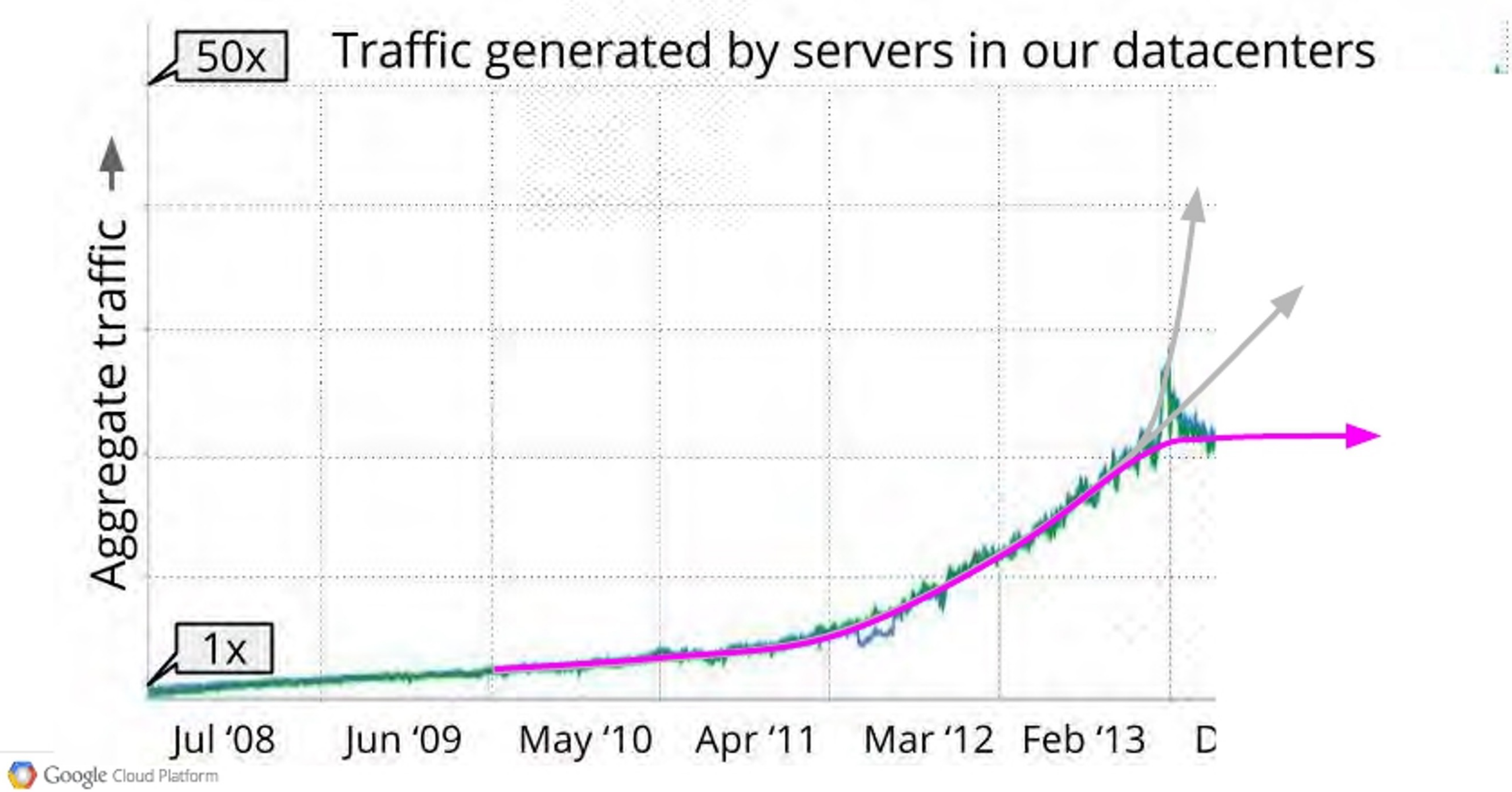
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. SIGCOMM 2015.

How much capacity do we need?



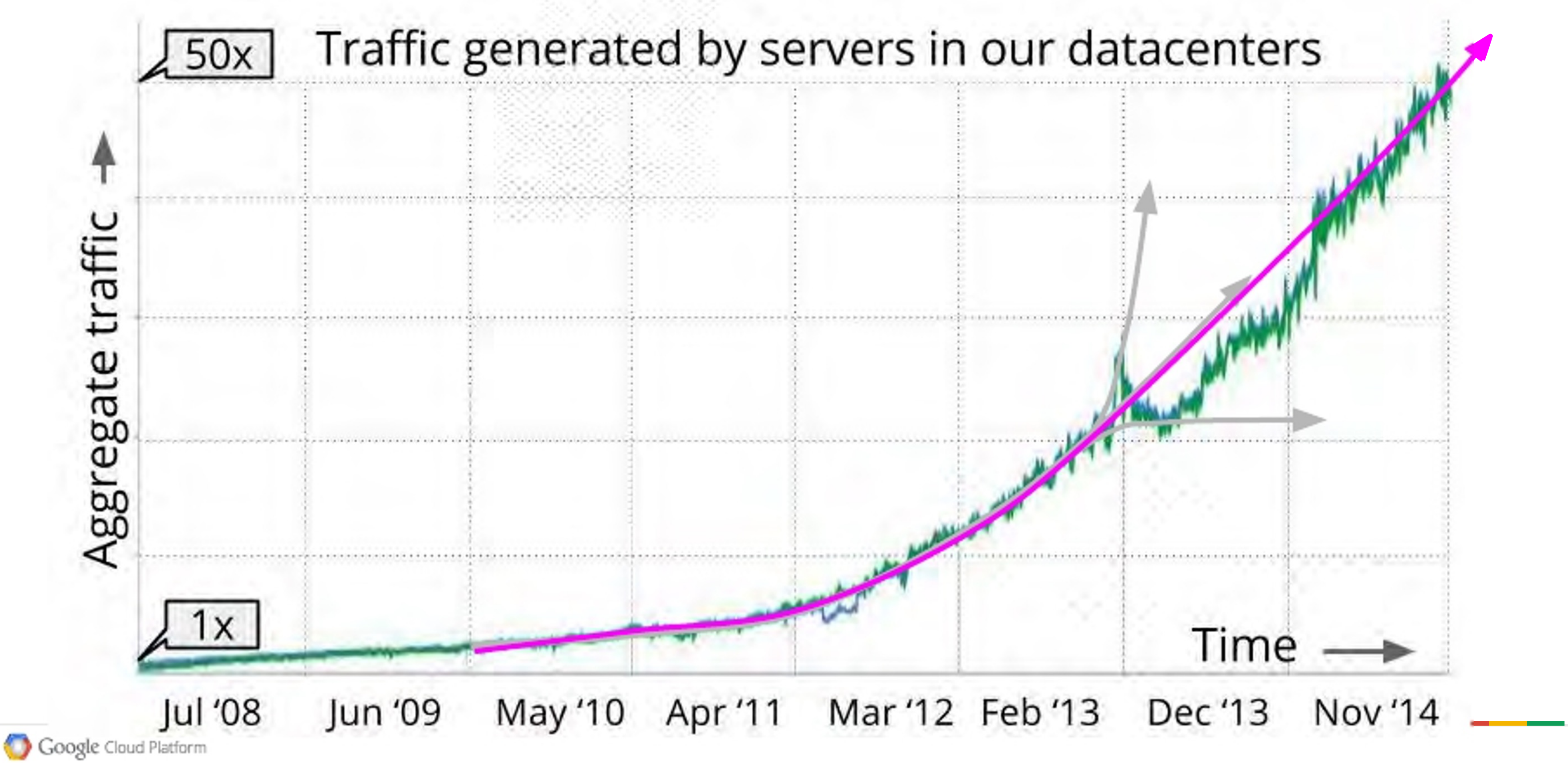
[Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network.] SIGCOMM 2015.

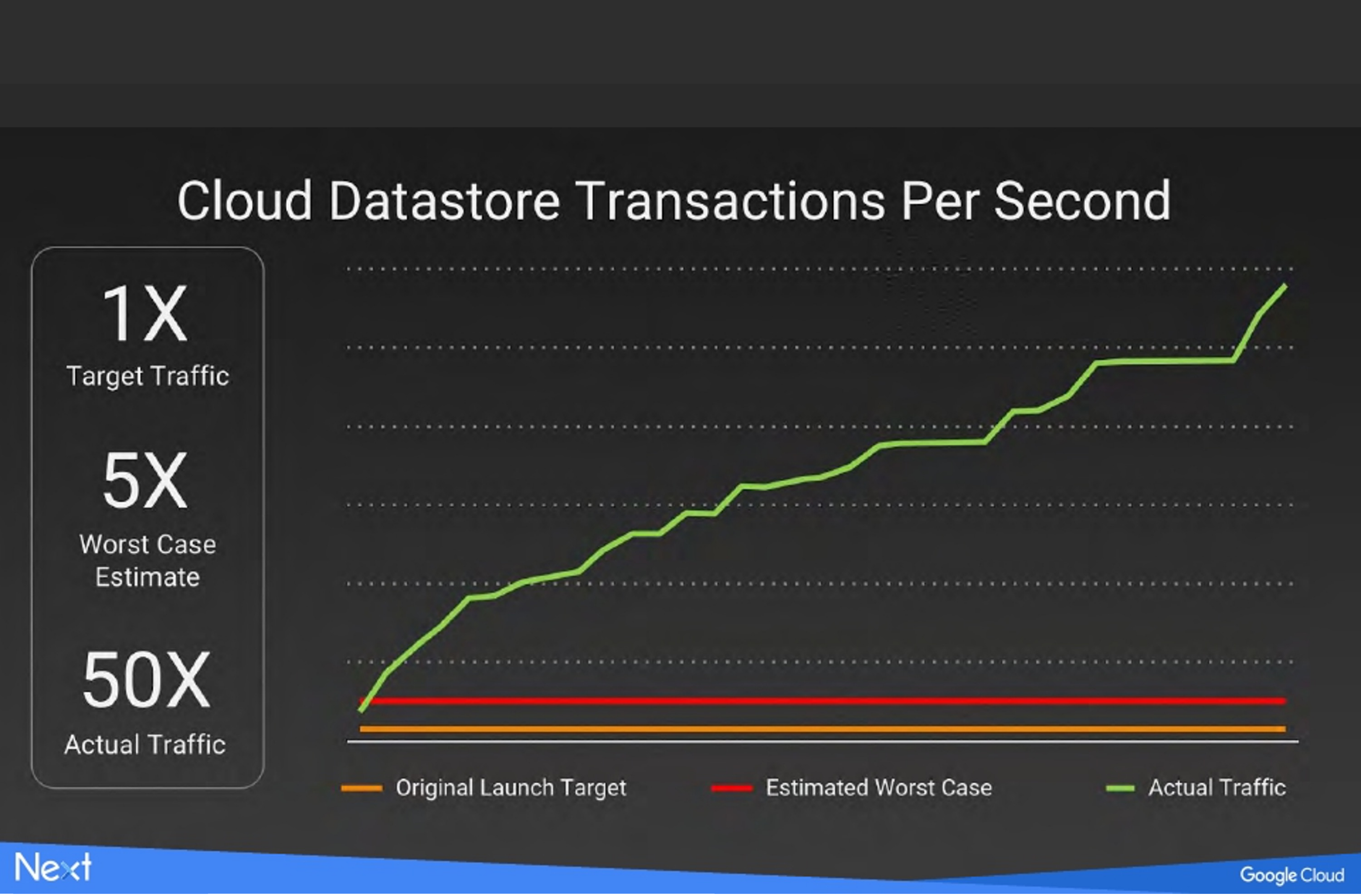
How much capacity do we need?



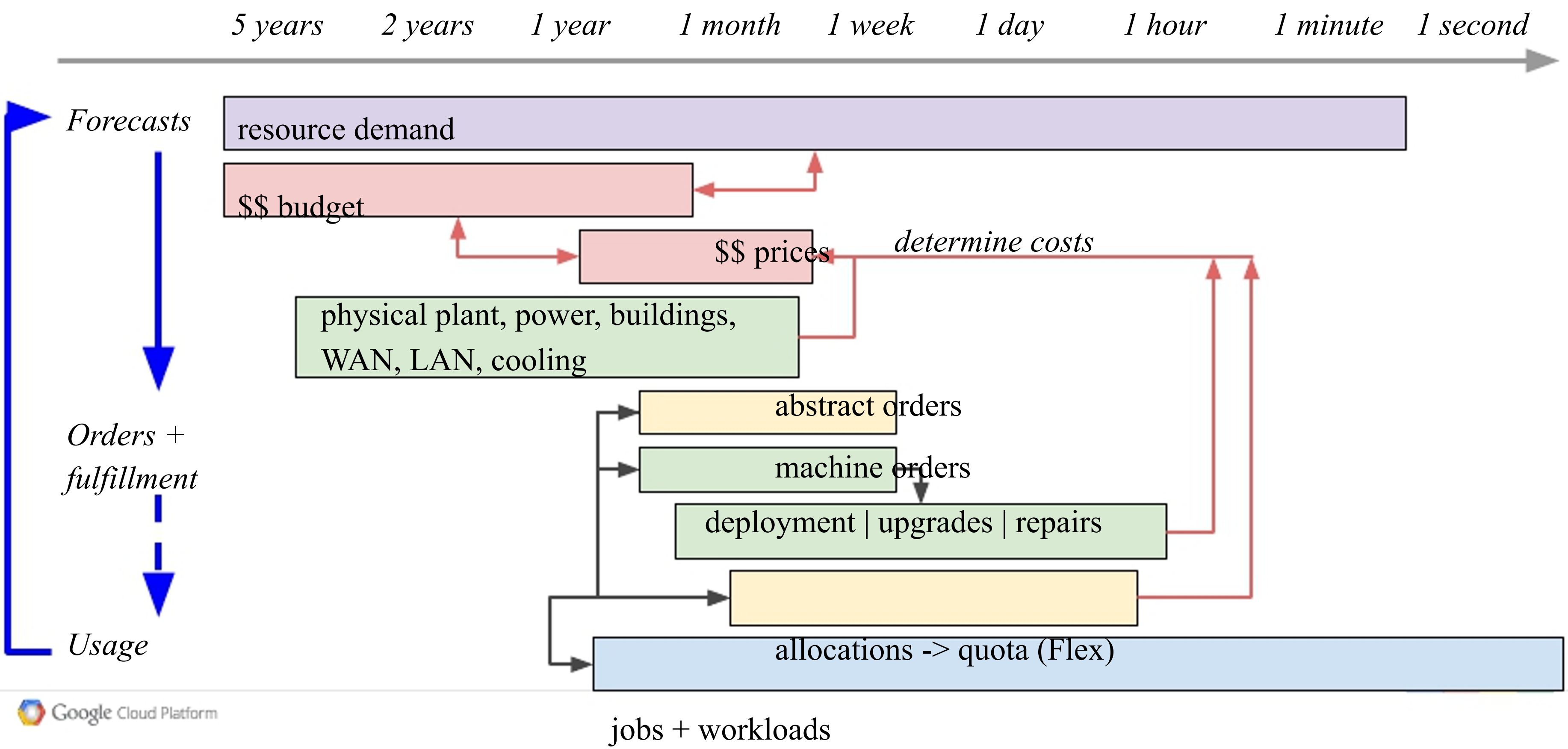
Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network. SIGCOMM 2015.

How much capacity do we need?




Cloud Datastore Transactions Per Second
1X
Target Traffic
5X
Worst Case
Estimate
50X
Actual Traffic
Original Launch Target
Estimated Worst Case
Actual Traffic
Next
Google Cloud

Putting it all together



Q&A