Ωmega

Cluster management at Google

john wilkes

Oct/Nov 2011
johnwilkes@google.com

Google

Cluster management: what is it?

- A fleet of *machines* live in *datacenters* placed in different *regions & countries*

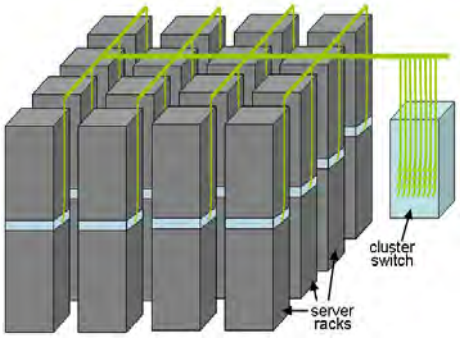
Google

Cluster management: what is it?

- A datacenter contains 1 or more *clusters*

machine

rack: 40-80 machines
+ Ethernet switch

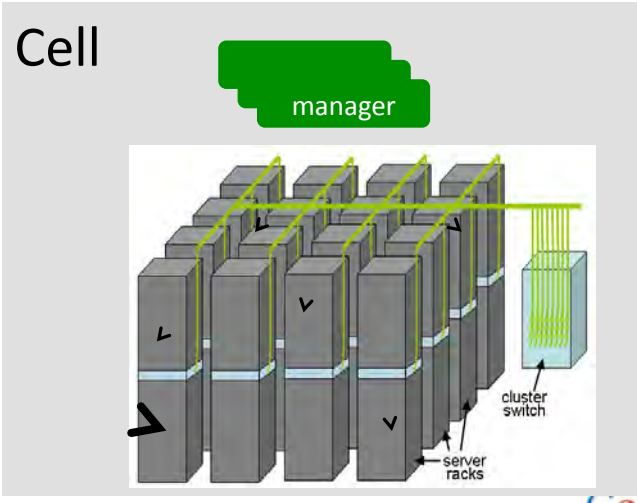


cluster

Google

Cluster management: what is it?

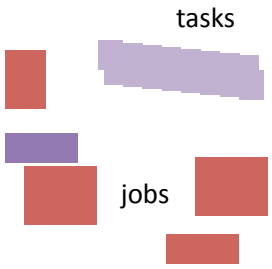
- Clusters are managed as 1 or more *cells*



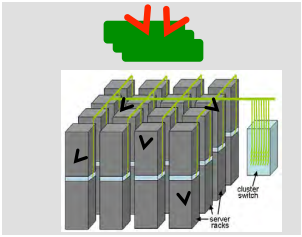
Google

Cluster management: jobs

- Users submit *jobs* to a cell
- A job has up to a few thousand *tasks*



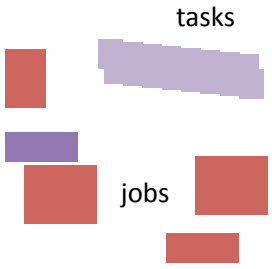
- Jobs & tasks have *requirements* (sizes, constraints)

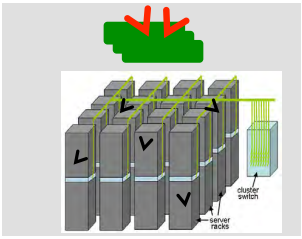


Google

Cluster management: jobs

- Jobs & tasks have *requirements*
  - resource *shape* (e.g., how much CPU, RAM, …)
  - constraints (e.g., machine type, external IP)
  - software to run (“package”)
  - preferences





Google

Cluster management: jobs

Job types:

- *services*; e.g., user-facing (latency-sensitive)
- *batch*; e.g., MapReduce (throughput sensitive)

  – important or not
  – one-off or periodic
  – standalone or coprocessor (e.g., BigTable)
  – inter-job dependencies

Google

Cluster management: other stuff

- Machine lifecycles
  – provisioning; testing; repairs; upgrades
- Software lifecycle
  – e.g., OS install + upgrades + downgrades
- Cluster maintenance
  – Planned Change Requests (PCRs)
  – scheduling; draining; restoring
- Monitoring (stats, events, usage, …)

Google

Cluster management: storage

- **Persistent storage** is distributed (GFS, Colossus/CFS, BigTable)
  - spread across machines in cluster
- **Local storage** is volatile
  - cache for data being served
  - logging: (1) write locally; (2) async copy to persistent store

Google

Cluster management: scale

- Big: the storage system **holds a Petabyte**
- Google: the storage system pages you when there's **only a few Petabytes of space left**

-- Luiz Barroso

Google

Cluster management: faults ☹

> 1%	Rate of uncorrectable DRAM errors/ machine/year
2-10%	Annual failure rate of disk drives
~2	Crashes/machine/year
2-6	OS upgrades/machine/year
> 1	Power utility events per year

-- Luiz Barroso Google

Cluster management: faults ☹

A 2000-machine service sees
>10 machine crashes per day

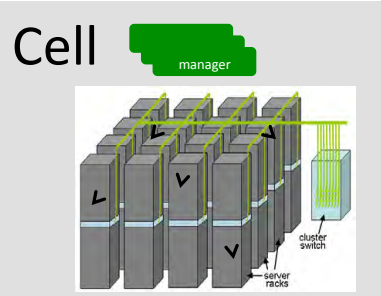
Main causes of service outages:
networking, power, "oops"

Rarer events:
wild dogs, sharks, dead horses,
copper thieves, drunken hunters, …

-- Luiz Barroso/Jeff Dean
Google

Cluster management: goals

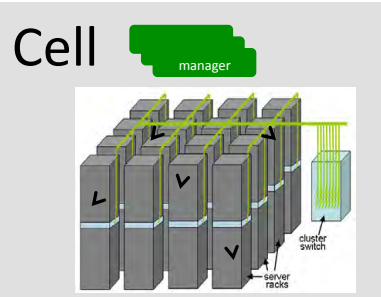
1. Run everything :-)
2. High utilization
3. Predictable, understandable behavior



Google

Cluster management: goals

4. Fine control for the big guys (resource efficiency)
5. Ease of use for others (innovation efficiency)
6. Keep going (failure tolerance)



7. At large scale
8. With few people

Google

Cluster management: goals

- Q: why not energy/power?
- A: we *do* care about energy/power proportionality

Google

Cluster management: goals

- Q: why not energy/power?
- A: we *do* care about energy/power proportionality

- But …
  - best way to save energy is to write good software
  - Google PUE was 1.13 in Q1'2011 (3-month weighted average)
  - don't buy idle machines!

Google

Cluster management: pre-Omega

- Current system was built 2003-4
- Works pretty well ☺

- Beginning to run out of steam …
  - scale (largest clusters)
  - inflexibility (ease of adding new features)
  - internal complexity (ease of adding new people)

Google

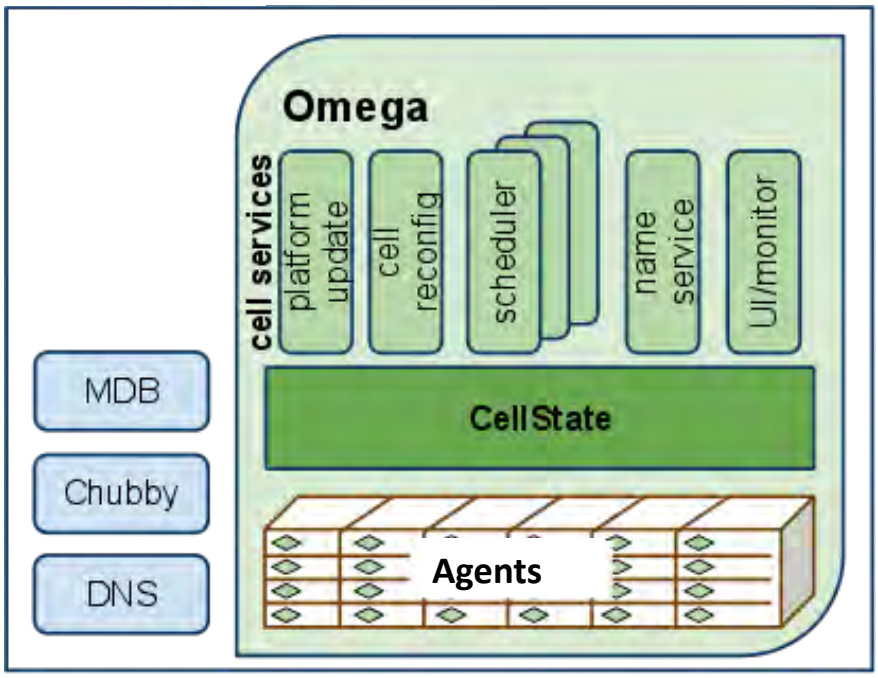
Ωmega

- The "second system" …
- Main *user* goals: predictability & ease of use
- Main *team* goal: flexibility

- Caveat: Omega is currently being prototyped
  - not in production!
  - many things will change!
  - it may never be deployed!

Google

Ωmega the general approach



Google

Ωmega the general approach

- Dedicated "**vertical***s*" for different needs
  - services, batch, machine management
- Central shared **state**
  - calendar of allocation decisions
  - minimal necessary data
  - no policies; just enforces invariants
- **Failures** are a first-class property
  - the resource model

Google

Omega issues: *intentions*

Avoid detailed specifications of *how*

- **not**: "place 40 tasks on that rack, 20 on this one"
- **not**: "I need 4.6 CPUs of processor type *p1*"

Use *goals* (SLOs/SLAs)

- "<=2 tasks down at a time (99.9%ile)"
- "<25ms request latency (95%ile)"

Google

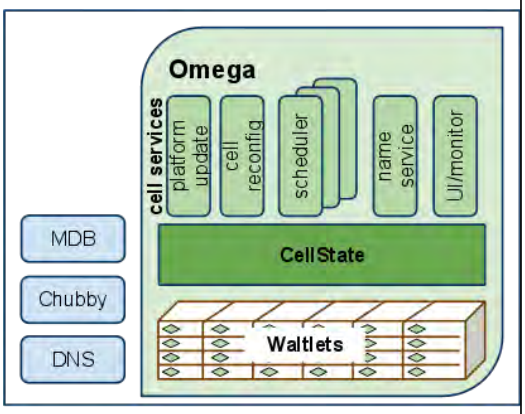
Omega issues: *failure tolerance*

- Goal: limit the number of concurrent outages
  - topology-aware scheduling
  - surety: likelihood of resources being available
- Fault detection
  - real fault, or just lost touch?
  - time to detect *vs.* false positives
  - correlated failures

Google

Omega issues: *master scalability*

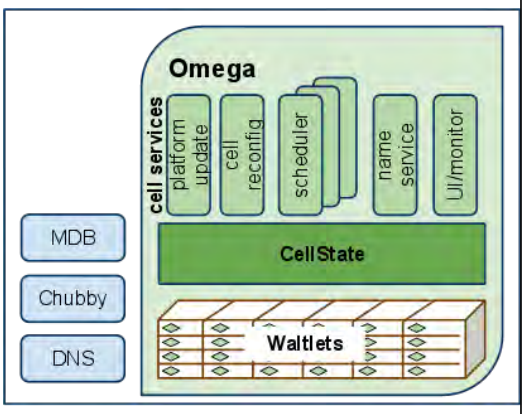
- Calendaring
  - super-efficient "does it fit?" checks
  - scheduling horizon? edge effects?



Google

Omega issues: *master scalability*

- Multiple scheduler verticals
  - livelock / mutual interference
  - optimistic concurrency?
  - what needs to be communicated?



Google

Omega issues: *predictable behavior*

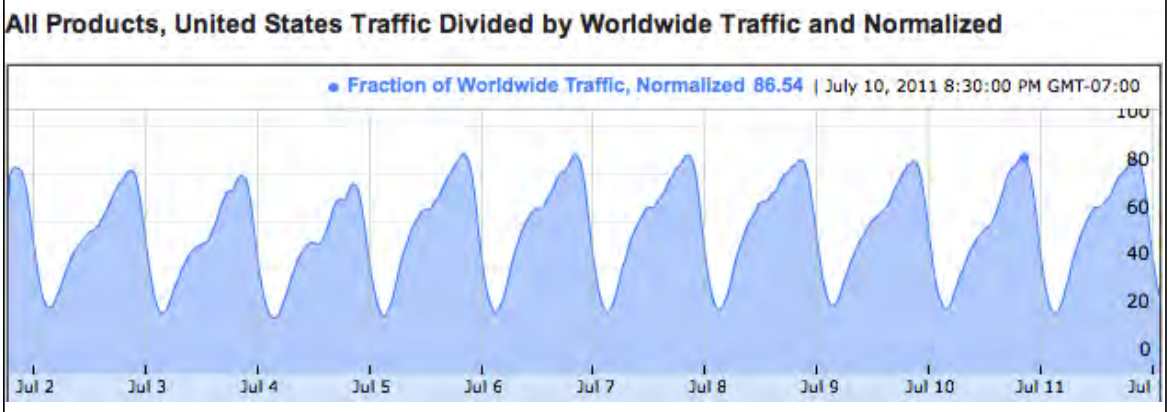
In the machine

- normalized performance (CPU, memory/NUMA)
- **performance isolation**
  - low latency **and** high throughput
  - storage?
  - caches?
- security isolation (PII, SOX)

Google

Omega issues: *predictable behavior*

Under load



Google

Omega issues: *predictable behavior*

Across the cell

- "why was my job not scheduled?"
- "where should I provision a new service?"

Approaches

- admission control
- explanations that make sense
- avoid the problem

Google

Omega issues: *objectives*

SLOs and SLAs

- what can/should be offered?
- how can they be controlled for at runtime?
- handling evolution

"Fair" objective functions

- is fairness useful/important?
  (reality is more complicated)

Google

Omega issues: *cell management*

It's 3am and your pager goes off

- are we in trouble?
- are we about to get into trouble?

- what should we do about it?

Google

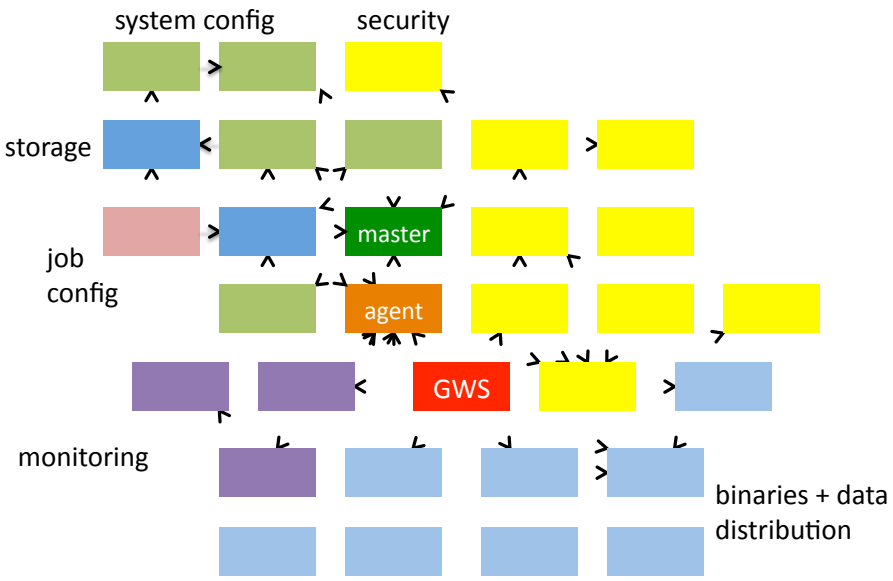
Omega issues: *ease of use*

How much can we simplify things?

- "here's a binary … run it"
- predictions based on prior history may help
- parameters?

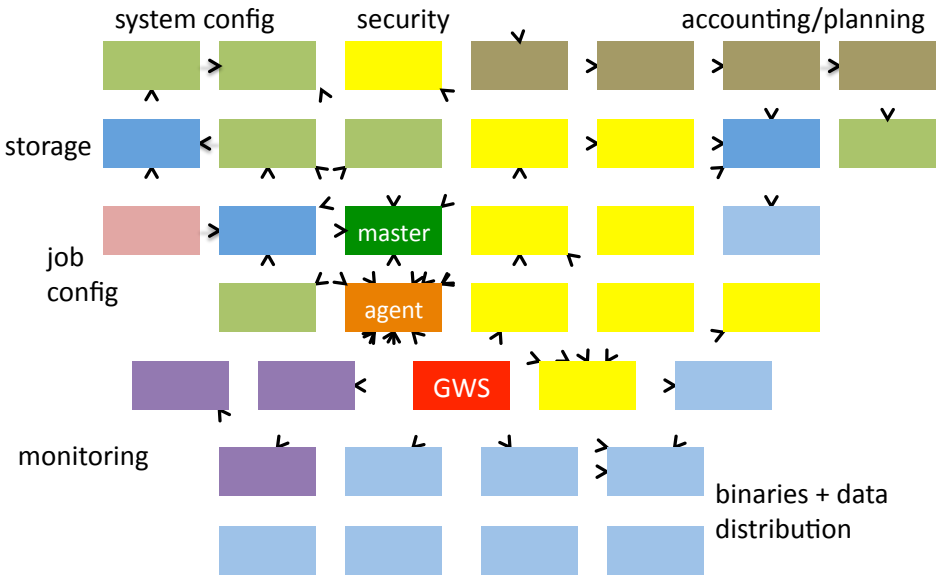
Google

Omega issues: *configuration*



Google

Omega issues: *configuration*



Google

Issues: *configuration*

- Make an app work right for one instance: *simple*
  - Google Docs uses ~50 systems and services
- Make an app work right in production: ***priceless***
  - run it in half a dozen cells
  - release a new version
  - fix it on the fly in an emergency
  - move one copy to another cell

http://melinathinks.com

Google

Summary

Ωmega

- Large-scale systems have some fun problems
  - driven by scale + failures
  - enormously fruitful ground for CS research
- Configuration may be **the** next big challenge

Google