Ωmega

Cluster management at Google

john wilkes

Faculty Summit, 2011-July-14
johnwilkes@google.com

Google

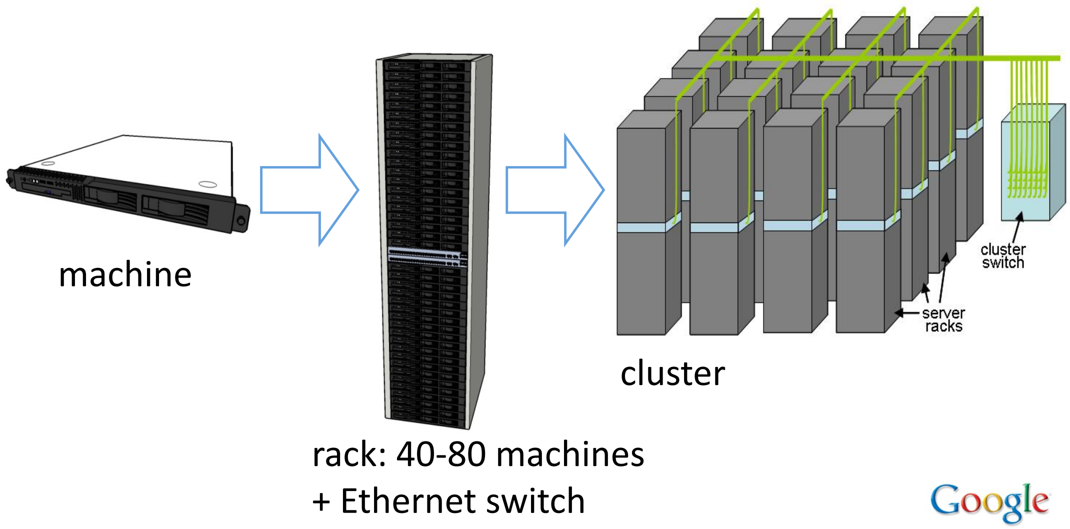
Cluster management: what is it?

- 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*, and has a *network* and a *power topology*



Cluster management: what is it?

- Clusters are managed as 1 or more *cells*
  - Each cell has a (replicated) central *manager*
  - Each machine has a local *agent*

Cluster management: scale

- Scale => "Your storage system pages you because there are only a few Petabytes of free space left"

-- Luiz Barroso

Cluster management: jobs

- Users submit *jobs* to a cell, comprising one or more *tasks*
- Jobs & tasks have *requirements*
  – resource *shape* (e.g., how much CPU, RAM, …)
  – constraints (e.g., machine type, external IP)
  – software to run ("package")
  – preferences

Cluster management: jobs

- *services*; e.g., user-facing (latency-sensitive)
- *batch*; e.g., MapReduce (throughput sensitive)
- up to thousands of tasks
- run for few seconds to many weeks

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

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, …)

Cluster management: faults ☹

99-99.9%	Internet availability
> 1%	Rate of uncorrectable DRAM errors/machine/year
2-10%	Annual failure rate of disk drives
~2	Machine crashes/year
> 1	Power utility events per year

- 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

Cluster management: goals

1. run everything :-)
2. high utilization
3. predictable, understandable behavior
   - fine control for the big guys (resource efficiency)
   - ease of use for others (innovation efficiency)
4. keep going (failure tolerance)

… all at large scale, with low operator effort

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!
  – dispersed storage => hard to turn machines off
  – complex interactions with failures

Google

Cluster management: pre-Omega

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

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

Google

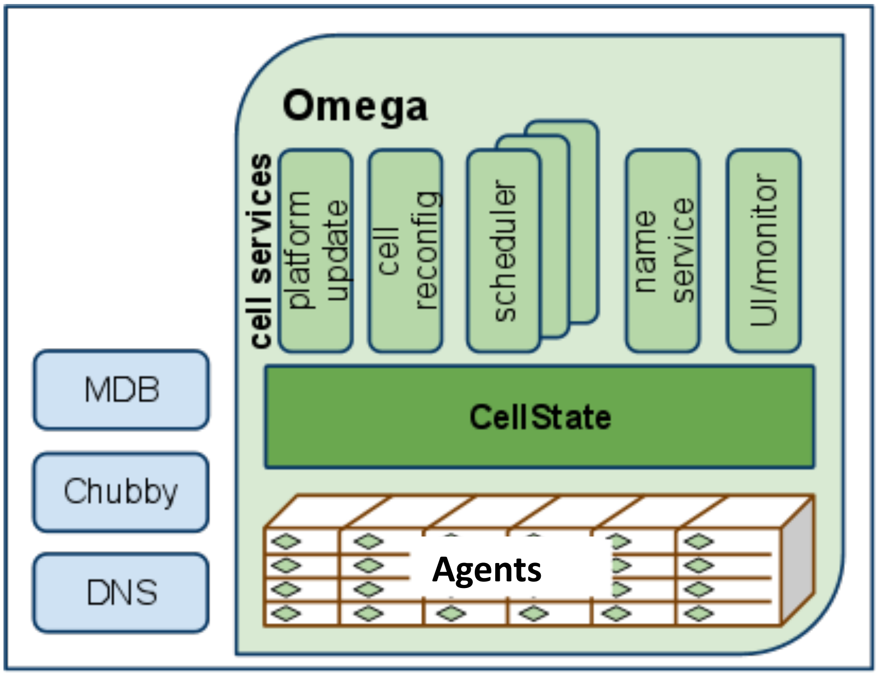
Ωmega

- 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



Ωmega the general approach

- Dedicated "**verticals**" 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

Omega issues: *intentions*

- Avoid detailed specifications of *how*
  - **not**: "place 40 tasks on that rack, 20 on this one" to achieve failure tolerance
  - **not**: "I need 4.6 CPUs of processor type *p1*" to achieve adequate throughput/latency
- But … what to say instead?
  - goal (SLO) specifications

Google

Omega issues: *failure tolerance*

- Goal: limit the number of concurrent outages
  - topology-aware scheduling
    (multiple topologies? competing objectives?)
  - surety: quantify likelihood of resources being available
- Detection
  - real fault, or just lost touch?
  - time to detect *vs.* false positives
  - correlated failures

Google

Omega issues: *master scalability*

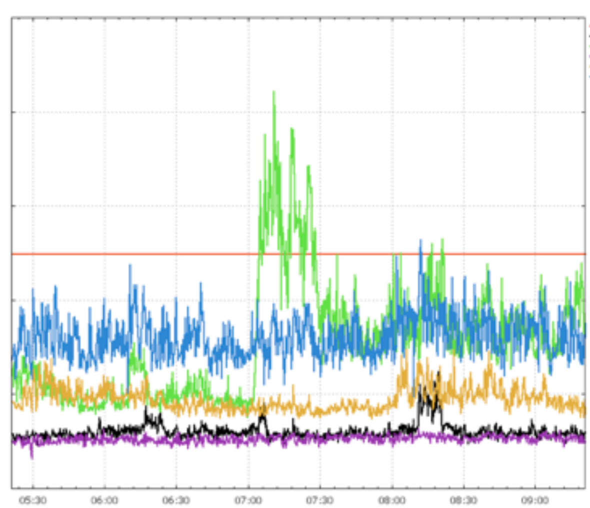
- Calendaring
  - super-efficient "does it fit?" checks
  - scheduling horizon? edge effects?

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

Omega issues: *predictable behavior*

In the machine

- normalized performance (CPU, memory/NUMA)
- **performance isolation** (caches?)
- storage (especially disk I/O): need *both* low-latency and high-bandwidth
- security isolation (PII, SOX)

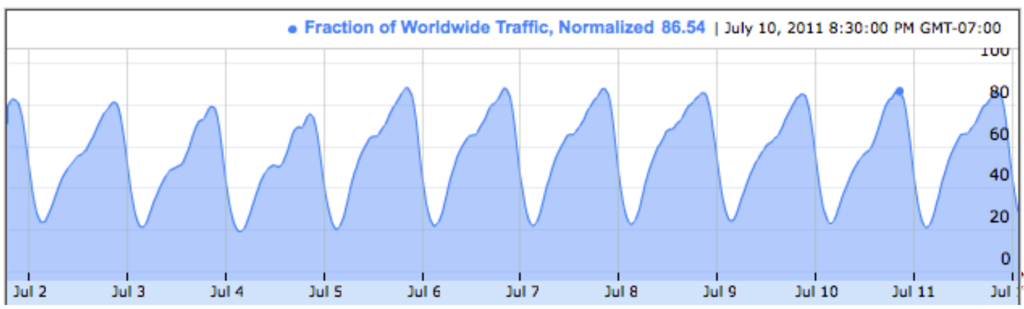


Omega issues: *predictable resources*

In the master

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

All Products, United States Traffic Divided by Worldwide Traffic and Normalized



Omega issues: *objectives*

- SLOs and SLAs
  - what can/should be offered?
  - how can they be controlled for at runtime?
  - handling evolution
- Objective functions
  - is "fairness" useful/important?
    (reality is more complicated)

Google

Omega issues: *ease of use*

- Can we simplify things for the little guy?
  - "here's a binary … run it"
  - predictions based on prior history may help
- But … how to specify (or infer):
  - good behavior
  - dependencies
  - the degrees of control freedom

Google

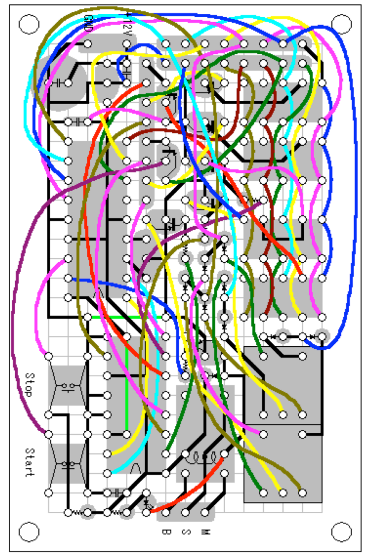
Omega issues: *cell management*

- are we in trouble?
- are we about to get into trouble?
- what should we do about it?
  - "it's 3am and your pager goes off …"

Google

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

Summary

- Large-scale systems have some fun problems
- Configuration may be **the** next big challenge