Cluster management at Google

LISA 2013

john wilkes (johnwilkes@google.com)
Software Engineer, Google, Inc.
We own and operate data centers around the world
A cluster is managed as 1 or more cells
A cell runs **jobs**, made up of (1 to thousands) of **tasks**, for internal users.
workload: job inter-arrival times

CDF

Batch
Service

Job inter-arrival time [log]
Cluster A
Medium size
Medium utilization

Cluster B
Large size
Medium utilization

Cluster C
Medium (12k mach.)
High utilization
Public trace

Jobs/tasks: counts
CPU/RAM: resource seconds [i.e. resource job runtime in sec.]
Placing tasks onto machines is just a *knapsack problem*, with constraints.
Cluster management: what is it?

Heterogeneity and dynamicity of clouds at scale: Google trace analysis. SoCC’12
Want to try it yourself?

- search for [john wilkes google cluster data]

29 day cell workload trace from May 2011

- ~12.5k machines
- every job/task/machine event
- task usage every 5 mins
- (obfuscated data)
Diagram from an original by Cody Smith.
Make an app work right once: *simple*

Google Docs uses ~50 systems and services

Make it run in production: *priceless*

- run it in 6 cells
- fix it in an emergency
- move it to another cell

http://melinathinks.com
If you aren't measuring it, it **is** out of control ...
SLI/SLOs for cluster manager

1. Availability
   ● % time master is up
   ● max outage duration

2. Performance
   ● RPC request latencies
   ● Time to schedule+start job

3. Evictions
   ● % jobs with too high an eviction rate
   ● % jobs with too many concurrent evictions
   ● % tasks evicted without proper notice
The current system was built 2003-4. It works pretty well 😊

But ...

- growing scale (#cores)
- inflexibility (adding features)
- internal complexity (adding people)
Cluster management: *pre-Omega*

- **RPCs**
  - Blue line: RPCs
  - Red line: Net Removed

- **Protobuf fields**
  - Blue line: Fields
  - Red line: Net Removed

Brian Grant, Oct 2013
The “second system” …

- Main user goals: predictability & ease of use
- Main team goal: flexibility

Omega is currently being deployed + prototyped
**CellState:**
- minimum necessary data
- no policies
- just enforces invariants

**Calendar:** everything is a scheduling event!
Monolithic scheduler

Omega: flexible, scalable schedulers for large compute clusters. EuroSys 2013
Workers per job [log scale]

Jobs

Omega: flexible, scalable schedulers for large compute clusters. *EuroSys 2013*
Relative speedup $[\log_{10}]$

CDF

$60\%$ of MapReduce jobs

better

3-4x speedup!

Omega: flexible, scalable schedulers for large compute clusters. EuroSys 2013
multiple applications per machine

(a) CDF of tasks/machine

(b) CDF of threads/machine

CPI²: CPU performance isolation for shared compute clusters. *EuroSys 2013*
multiple applications per machine

Outliers => victims

CPI data for all the tasks in a job
- per-cluster
- per-platform
- mean ($\mu$)
- stddev ($\sigma$)

CPI$^2$: CPU performance isolation for shared compute clusters. EuroSys 2013
Workload variation
CPU and RAM usage as a function of offered load

Omega goals: ease of use
“Here’s a binary … run it”

Why didn’t it run?
Open issues: *smart explanations*

It’s 3am and your pager goes off
- are we in trouble?
- are we about to get into trouble?

→ what should you do about it?
Large-scale systems are exciting beasts

- scale $\rightarrow$ failures
- QoS $\rightarrow$ new designs
- configuration may be the next big challenge

DO COOL THINGS THAT MATTER

We’re hiring!  www.google.com/jobs

johnwilkes@google.com