Borg

Main idea

Goolgle's Borg system is a cluter manager which runs hunderds of thousands of jobs of Google on many clusters, each consisting of tens of thousands of machines.

> We present a summary of the Borg system architecture and features, important design decisions, a quantitative analysis of some of its policy decisions, and a qualitative examination of lessons learned from a decade of operational experience with it.

Some keywords

- **prod** higher-priority Brod jobs
- **non-prod** lower-priority ones
- **cell** tens of thousands of machines in a datacenter building
- **cluster** a cluster is made up of a few cells
- **task** a set of Linux processes
- **job** a set of tasks
- **aloc** a reserved set of resources on a machine in which one or more tasks can be run
- **quota** describes the resource a job needs, used to decide which jobs to admit for scheduling
- **BNS** Borg Name Service, used to find a task
- **Chubby** used by RPC to find task end point
- **Sigma** provides a UI to users that help them monitor their tasks

Cell-level management: Centralized archietecture

A Borg cell-level management is made up of a centralized controller called Borgmaster and an agent process called Borglet for each machine.

Borgmaster

Borgmaster consists of main Borgmaster process and a seperate scheduler. The main Borgmaster process communicates with each Borglet and provides information to users.

Crash consistency for Borgmaster

To achieve crash consistency, the Brogmaster has 5 replicas. Besides, Borgmaster keeps a checkpoint periodically, which helps it to restore states.

Debug in Borgmaster

Fauxmaster is a Borgmaster simutalor, which can act exactly like a Borgmaster. It can be used to debug, simulate Borgmater changes.

Scheduler

When a job is submitted to pending queue by Borgmaster, scheduler decides if there is sufficient resources to conduct the job.

Feasibility checking of scheduler

Scheduler checks if machine has enough resource. It is noticable that jobs of lower priority can be evicted to gain more resrouces.

Scoring of scheduler

The current scoring model is a hybrid one that tries to reduce the amount of stranded resources. It provides about 3–5% better packing efficiency than best fit for the workloads. It is noticable that since the installation can cause a great deal of latency, so scheduler prefers to assign jobs to machines that have necessary packages installed.

Borglet

A machine-level management process, controlled by Borgmaster. Borgmaster is logically center of all Borglets in a Cell, it keeps states of Borglets. Borgmaster polls Borglets' states periodically. If the poll fails a few times in a row, the master will reorganize jobs.

Deal with scalability

To deal with large scale, the paper came up with following ideas:

1. Split the scheduler into a seperate process. Also use a replica on a chached copy of cell state.
2. Add seperate threads to talk to Borglets and respond to read-only RPCs to improve response time.
3. Split Borgmaster functions among its five replcas.
4. Since scoring a machine is expensice, Borg caches scores until properties of the machine or task changes.
5. Since tasks in a Borg job usually have same requirements, Borg only does feasibility and scoring for one task per equivalence class – a group of tasks with identical requirements.
6. Examine machines in a random order until it has found “enough” feasible machines to score, and then selects the best within that set.

Keep availability

To keep availability, already-running jobs will continue to run even if the Borgmaster or Borglet of a machine goes down.

Utilization: make efficient use of Google’s fleet of machines

The driver to improve utilization is that increasing utilization by a few percentage points can save millions of dollars for Google.

Cell Sharing

They found that segregating prod and non-prod jobs would result in using 20-30% more machines because prod jobs often don't make full use of resources. Packing unrelated jobs (even from unrelated users) help to increace utilization.

Large Cells

Google tests to use large cell or multiple small cells for the same workload, and finds that using a large cell consumes far fewer machines.

Finer-grained CPU-level request

Experiments have shown that finer level of CPU request can reduce resource cost.

Resource Reclaimation

Because an overestimated allocation of resource can be a lot of waste, the Borgmaster conducts a computation of the real needs of jobs every few seconds.