ECE 473/573
Cloud Computing and Cloud Native Systems
Lecture 16 Resource Management II

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 16, 2024

Outline

Reading Assignment

- This lecture: Large-scale cluster management at Google with Borg https://storage.googleapis.com/pub-tools-public-publication-data/pdf/43438.pdf
- Next lecture: Kubernetes https://kubernetes.io/docs/concepts/

Outline

Borg

Google Borg

- An internal cluster management system developed by Google.
  - Across a number of clusters each with up to tens of thousands of machines.
  - Support hundreds of thousands of jobs from many thousands of different applications.
- Benefits
  - Hide details of resource management and failure handling so users can focus on application development.
  - High availability and reliability, and support applications that do the same.
  - Operating at scale while providing resiliency and completeness.

User Perspective

- Users of Borg are developers and SREs (system administrators as site reliability engineers).
- Unit of management is a Borg cell.
  - Users submit work to Borg as jobs.
  - Each job consists of tasks all run the same binary program.
  - The Borg cell refers to the set of machine the job runs in.
- Physically, machines in a cell belong to a single cluster.
  - In a single datacenter building, connected by high-performance datacenter-scale network.
  - Machines are heterogeneous: CPU etc. can be all different.
- Borg manages physical machines and hides their differences and failuers from users.
  - Install programs and dependencies.
  - Health monitoring.
  - Restart failed machines.

Workloads

- End-user-facing services.
  - Sensitive to latency.
  - Usually short-lived: us to sub-second
- Batch jobs.
  - Take longer time to complete: seconds to days
  - Not sensitive to short-term performance fluctuations.
- The workload mix varies across applications and over time.

Job and Task Management

- Each job has a name, an owner, and the number of tasks.
- Each job can in addition have constraints.
  - Force its tasks to run on machines with particular attributes like processer type and OS version.
  - Hard constraints must be satisfied; soft one are preferences.
- Each task maps to a set of Linux processes running in a container on a machine.
  - Task specifies its resource requirement.
  - Task also knows its index within the job.
- Jobs and tasks in the system are in one of the three states: Pending, Running, Dead.
  - Users can submit new jobs or resubmit Dead jobs, which move into Pending state if accepted.
  - Users can kill Pending and Running jobs into Dead state.
  - Users can update Pending and Running jobs without interrupting them.
  - Borg takes care of the rest, e.g. to schedule a Pending job into Running state, and move jobs to Dead for failures.

Quota and Priority

- Each job has a priority.
  - Express the relative importance of jobs.
  - e.g. monitoring > production > batch > best effort
- Quota is used to decide which jobs to admit for scheduling.
  - A vector of maximum resource usage for a period of time (typically months) at a given priority for an user.
  - e.g. 20TB of memory for production for the rest of the month.
  - Jobs with insufficient quota are rejected upon submission.
- Once admitted, higher priority jobs may preempt lower priority ones to obtain resources.
- Higher priority quota is limited to available resources.
  - However, users tend to overbuy higher priority quota to avoid future shortages – waste of resource!
  - Quota for lowest priority is set to infinite for all users.
  - Jobs of lower priorities may be admitted but need to wait resources to become available.

Naming and Monitoring

- A “Borg name service” (BNS) name allows to identify a task via the cell name, job name, and task index.
  - The BNS name is further used in the DNS name for the task.
- Almost every task uses its own HTTP server to report its health and performance metrics.
  - Borg restarts a task if its HTTP server stops to respond.
  - Monitoring tools track these data for visualization and notifications.

Borg Architecture

- Recall that for each job there will be a Borg cell which includes all machines the job runs on.
- Each Borg cell has a controller named Borgmaster.
  - Consist of two processes: the main Borgmaster process and a separate scheduler.
- Borglet, an agent process, runs on each machine in the cell.
  - Start, stop, restart tasks and manage local resources.
- The Borgmaster main process interfaces with users and Borglets, and manages states for tasks and machines.
  - With multiple replicas supported by Paxos consensus.
  - These replicas stores checkpoints, which consist of state snapshots and change logs at a point in time.
  - Checkpoints are used for fault recovery, troubleshooting, offline simulation etc.

Scheduling

- Once a job is accepted by the Borgmaster main process, its tasks are queued for scheduling by the scheduler.
- The scheduler needs to evaluate task-machine relationships to schedule tasks to machines.
- Feasibility checking: a task is feasible to run on a machine if there are sufficient available resources.
  - Plus additional constraints from the job.
  - May consider to evict lower-priority tasks.
- Scoring: decide which tasks to run if many are feasible and decide where to run them.
  - Consider priority and fairness, data and package availability, power and failure domains, packing quality for load spike etc.

Optimizations for Scalability

- Functional partitioning: use separate threads for Boglet to Borgmaster communications and read-only queries.
  - Sharding further distributes these works to replicas.
- Score caching: recompute scores for tasks and machines only when there are changes.
- Equivalence classes: handle similar tasks in a job as a whole so that feasibility checking and scoring only need to run once.
- Relaxed randomization: for a single task, avoid to evaluate it on all machines for feasibility checking and and scoring.
  - If enough machines evaluated following a random order are feasible, then the best score so far is good enough.

Techniques for Availability

- Keep tasks running even if Borgmaster or Boglets are down.
- Automatically reschedule evicted tasks
  - Reduce correlated failures by spreading tasks of a job across failure domains such as machines, racks, and power domains.
  - Rate-limit to find new machines for tasks as it could be either due to large-scale machine failure or network partitioning.
  - Avoid repeating task-machine schedulings that lead to crash.
- Limit task disruptions within a job during maintenance.
- Use idempotent operations to support retries.

Isolation

- Security isolation is achieved by a combination of Linux chroot jail, cgroup (container), and VM for software from various sources.
- Performance isolation is supported via containers.
  - In order to limit resource usages of tasks.
  - Use appclass to indicate needs of tasks: latency-sensitive vs batch.
  - Separate compressible resources like CPU and I/O bandwidth, from non-compressible resources like memory capacity.
    - Compressible resources can be reclaimed by rate-limiting.
    - Kill tasks requiring more non-compressible resources than allowed, or when such resources are over-committed.
  - Improve standard Linux CPU scheduler for both low latency and high utilization.

Summary

- People learned a lot from building Borg to support cluster computing needs in Google, which are eventually applied in the development of Kubernetes.