Cluster Frameworks and Kubernetes

March 11th 2021

Suresh Marru & Marlon Pierce

Reference Papers

- Verma, Abhishek, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. "Large-scale cluster management at Google with Borg." In *Proceedings of the Tenth European Conference on Computer Systems*, pp. 1-17. 2015.
    - https://dl.acm.org/doi/pdf/10.1145/2741948.2741964
- Burns, Brendan, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. "Borg, omega, and kubernetes." *Queue* 14, no. 1 (2016): 70-93.
    - https://dl.acm.org/doi/pdf/10.1145/2898442.2898444
    - The lecture borrows concepts and content from this paper.

Cluster Computing Frameworks

Hadoop Map Reduce

Google Pregel

Pig

RAILS

Dryad

Input files

Stage

Channels

Vertices (processes)

Output files
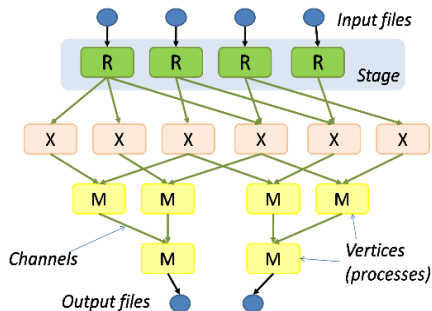
CIEL

Google Percolator

S4 distributed stream computing platform

MPI 2

Container recap

- "Containers encapsulate the application environment, abstracting away many details of machines and operating systems from the application developer and the deployment infrastructure."

- "Because well-designed containers and container images are scoped to a single application, managing containers means managing applications rather than machines. This shift of management APIs from machine-oriented to application oriented dramatically improves application deployment and introspection."

The Challenge

Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2

User DB
postgresql + pgv8 + v8

Queue
Redis + redis-sentinel

Analytics DB
hadoop + hive + thrift + OpenJDK

Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend
Ruby + Rails + sass + Unicorn

API endpoint
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Development VM

QA server

Public Cloud

Production Cluster

Disaster recovery

Customer Data Center

Contributor's laptop

Production Servers

The Matrix From Hell

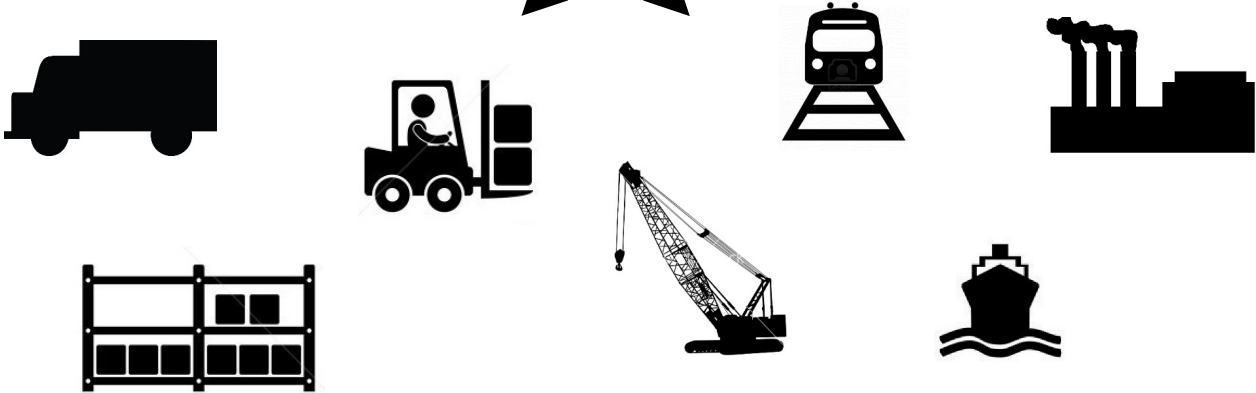|	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
Background workers	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

Cargo Transport Pre-1960

Multiplicity of Goods

Multipilicity of methods for transporting/storing

Do I worry about how goods interact (e.g. coffee beans next to spices)

Can I transport quickly and smoothly (e.g. from boat to train to truck)

A matrix from hell

Solution: Intermodal Shipping Container

Multiplicity of Goods

A standard container that is loaded with virtually any goods, and stays sealed until it reaches final delivery.

Do I worry about how goods interact (e.g. coffee beans next to spices)

…in between, can be loaded and unloaded, stacked, transported efficiently over long distances, and transferred from one mode of transport to another

Multiplicity of methods for transporting/storing

Can I transport quickly and smoothly (e.g. from boat to train to truck)

Docker is a shipping container system for code

Static website   User DB   Web frontend   Queue   Analytics DB

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

Development VM   QA server   Customer Data Center   Public Cloud   Production Cluster   Contributor's laptop

Docker: Infrastructure as Code

- In short, Docker lets you define in script files everything about each of your microservices.
- Combine this with CI/CD systems to deploy EACH microservice.
  - Your development to test to production environments should be identical and reproducible.
  - Testing and production deployments for each service should be infinitely clone-able.
- This is not elasticity, but it is a prerequisite.
- Docker and other containers have much less overhead

Changes and Updates

App A

Bins/ Libs

Base Container Image

App Δ

Container Mod A'

Container Mod A''

Push

Docker Container Image Registry

Update

App Δ

App A

Bins/ Libs

Docker Engine

Host running A wants to upgrade to A''. Requests update. Gets only diffs

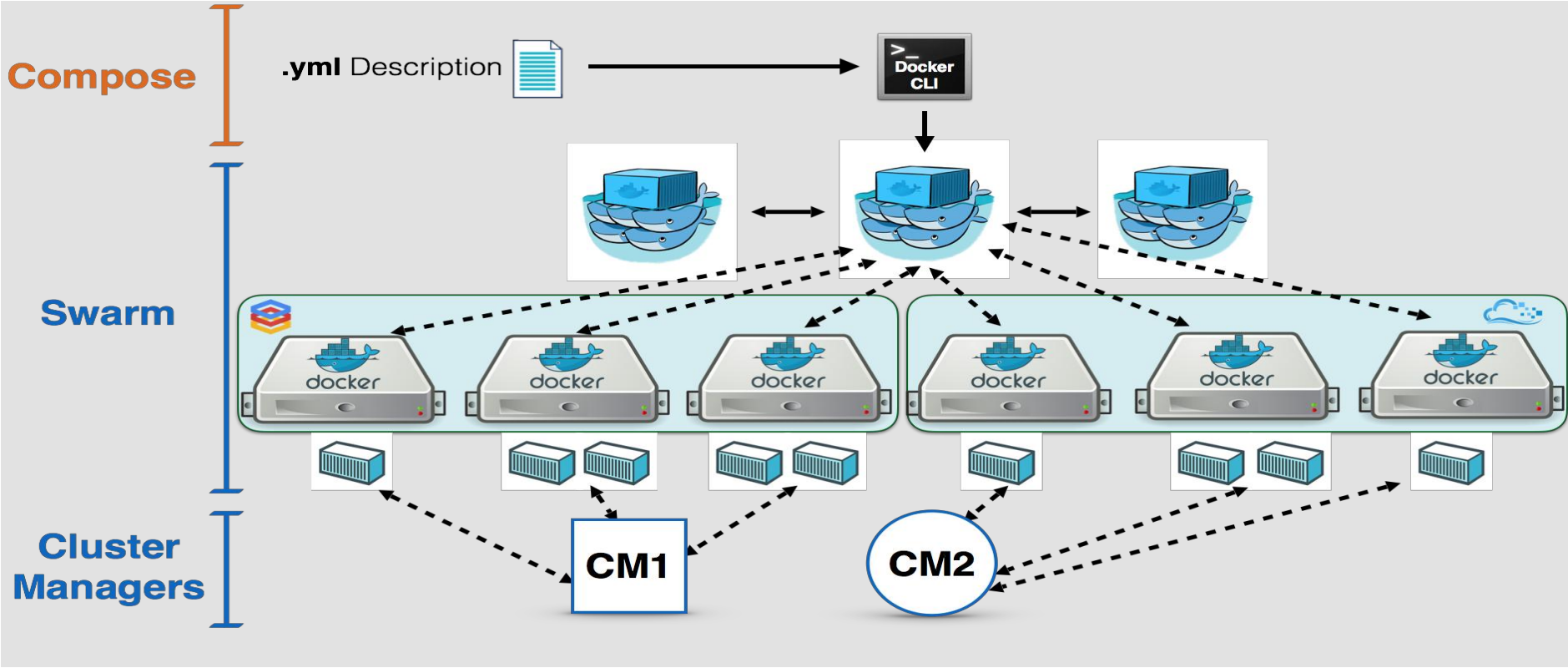App A''

Bins/ Libs

Docker Engine
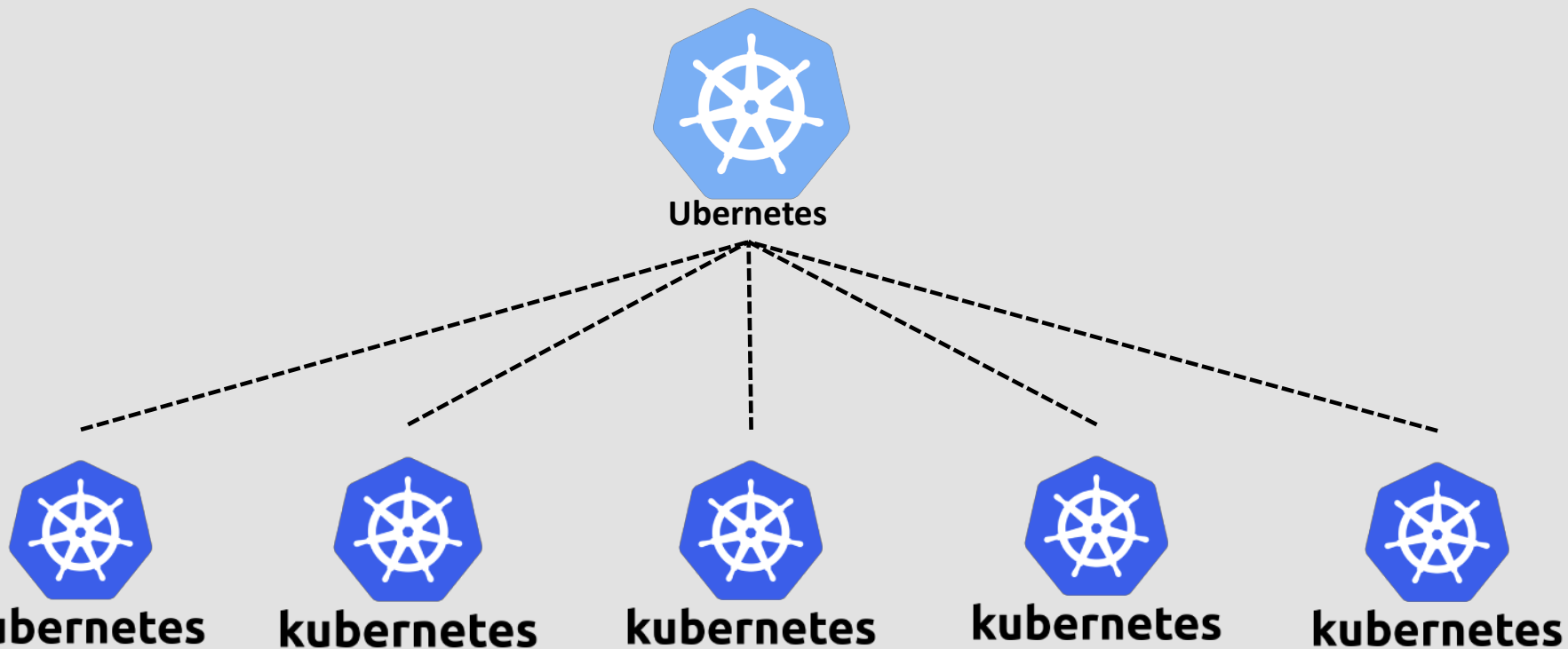
Host is now running A''
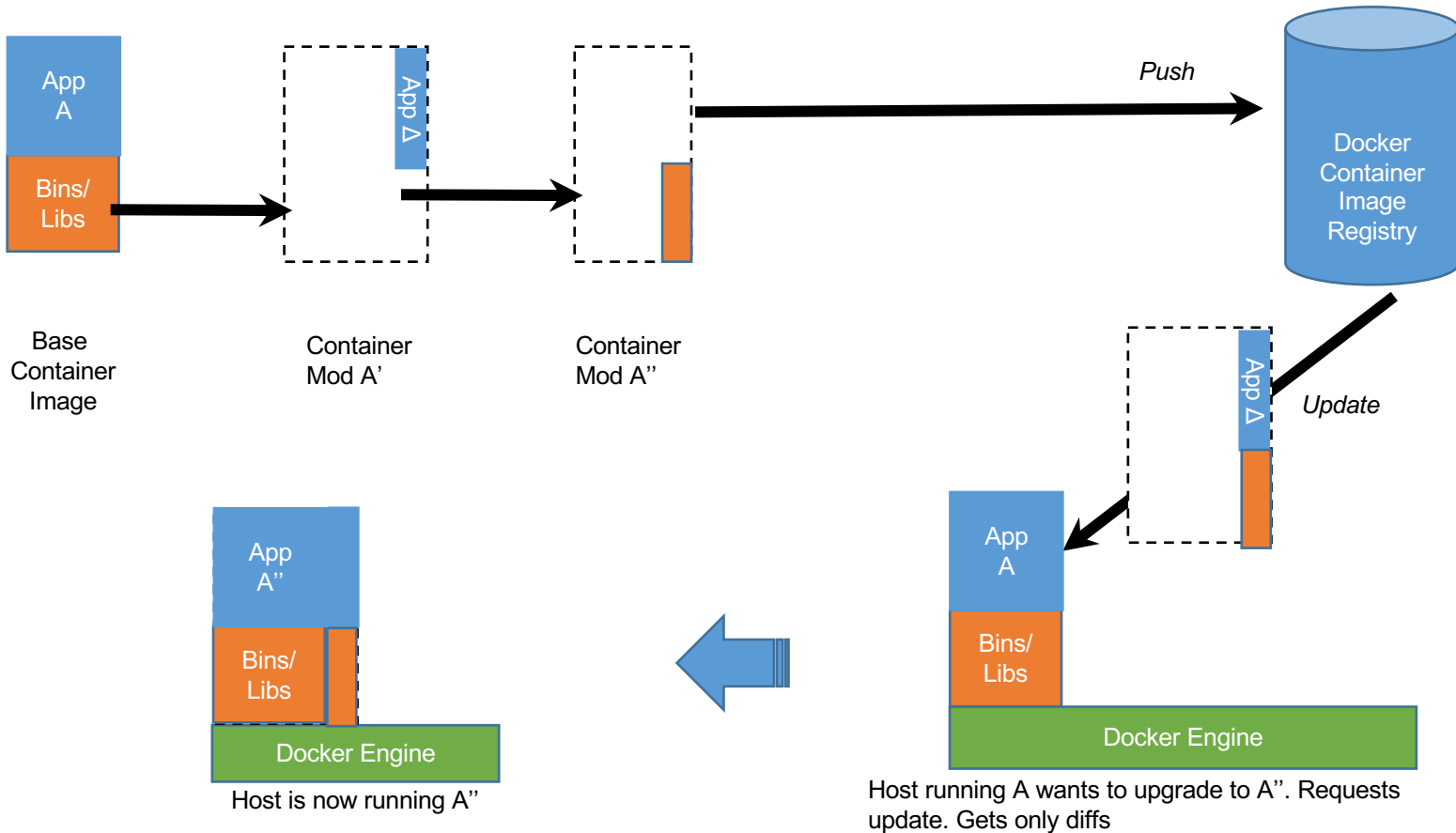
Cloud Federation

Docker Swarm (Cluster Federation)

Cloud Federation

Ubernetes (Google)

Google Borg

- Borg: An OS of Cluster (Datacenter)

- Motivation
    - Hide the details (programmer focus on App)
    - Provide resource sharing
    - Provide high reliability and availability for Cluster

The User Perspective

```
job hello_world = {
    runtime = { cell = 'ic' }                // What cluster should we run in?
    binary = '.../hello_world_webserver'     // What program are we to run?
    args = { port = '%port%' }               // Command line parameters
    requirements = {        // Resource requirements
        ram = 100M
        disk = 100M
        cpu = 0.1
    }
    replicas = 10000    // Number of tasks
}
```

> borgcfg .../hello_world_webserver.borg up

The User Perspective

- Allocs
  - Reserved set of resources

- Priority, Quota, and Admission Control
  - Job has a priority (preempting)
  - Quota is used to decide which jobs to admit for scheduling

- Naming and Monitoring
  - 50.jfoo.ubar.cc.borg.google.com
  - Monitoring health of the task and thousands of performance metrics
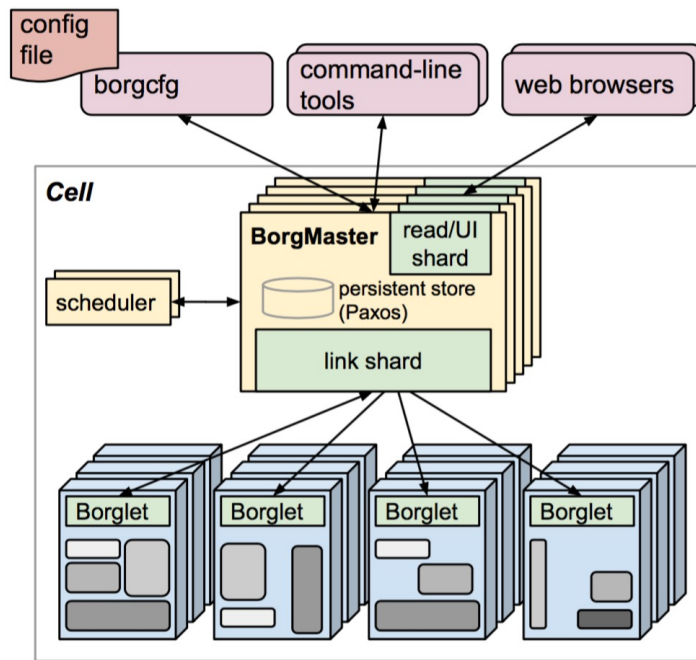
Borg Architecture

- Borgmaster
  - Main Borgmaster process & Scheduler
  - Five replicas
- Borglet
  - Manage and monitor tasks and resource
  - Borgmaster polls Borglet every few seconds
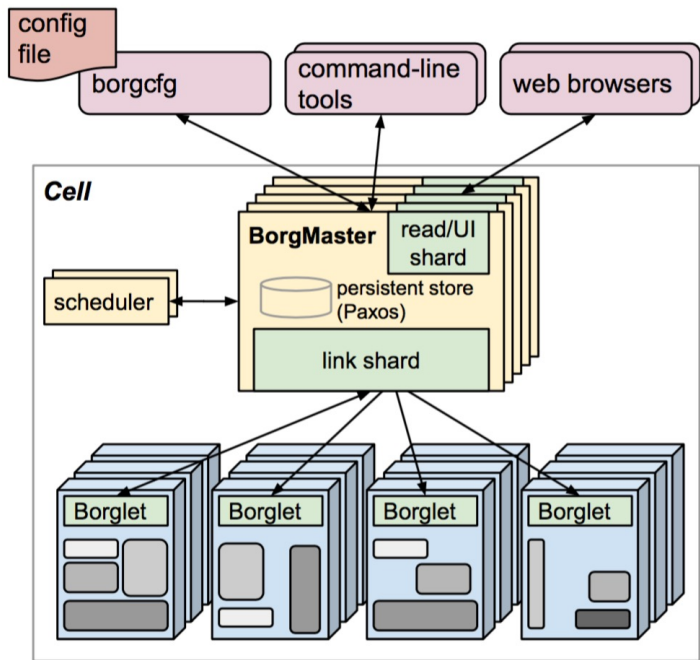
The high-level architecture of Borg(A Cell)
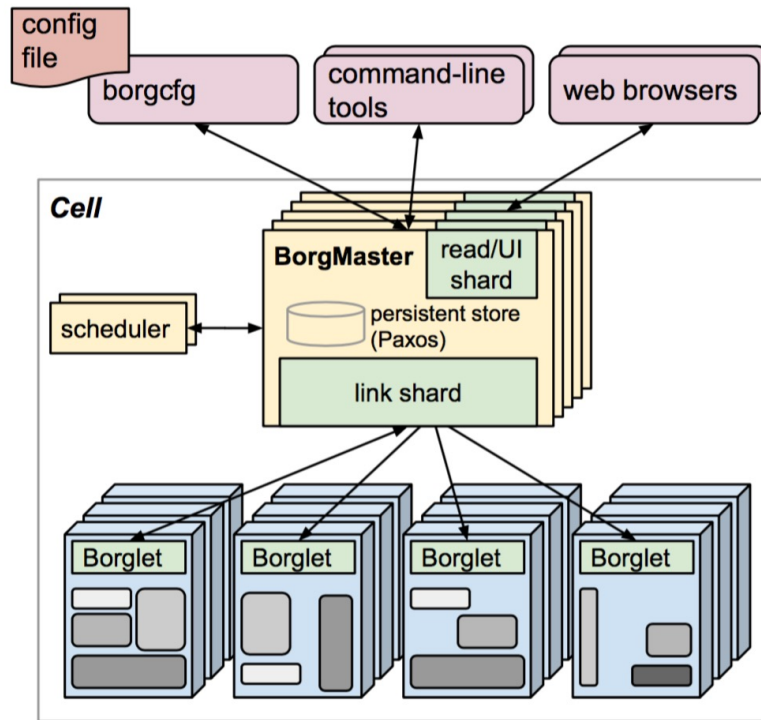
Borg Architecture

- Scheduling
  - feasibility checking: find machines
  - Scoring: pick one machines
    - User preferences & build-in criteria
    - E-PVM VS best-fit
    - Tradeoff

The high-level architecture of Borg(A Cell)

Scalability

- Separate scheduler
- Separate threads to poll the Borglets
- Partition functions across the five replicas
- Score caching
- Equivalence classes
- Relaxed randomization

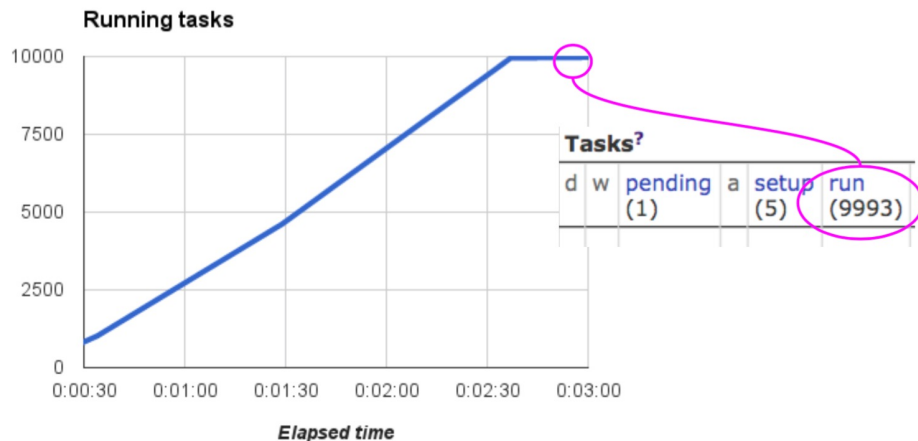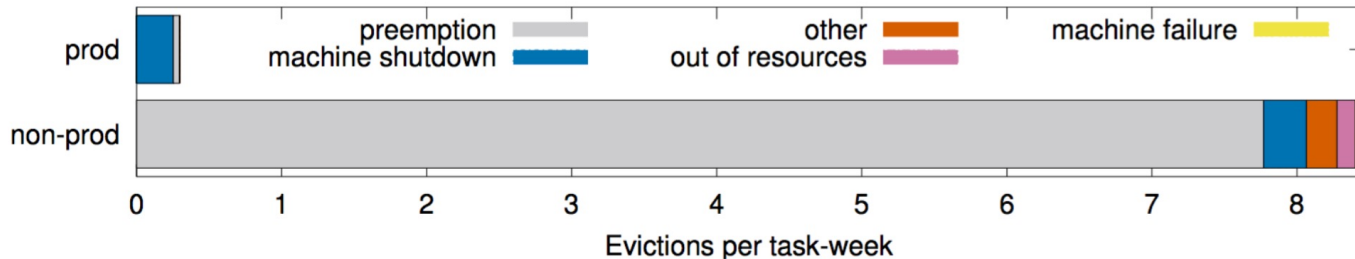The high-level architecture of Borg(A Cell)

Availability

- System
  - Borgmaster:
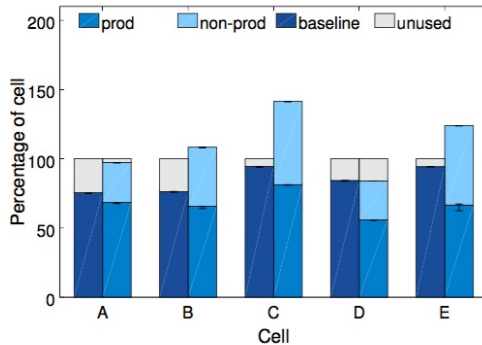 - Borglet: Borgmaster
  - Take checkpoint
- Job
  - Reschedules evicted tasks
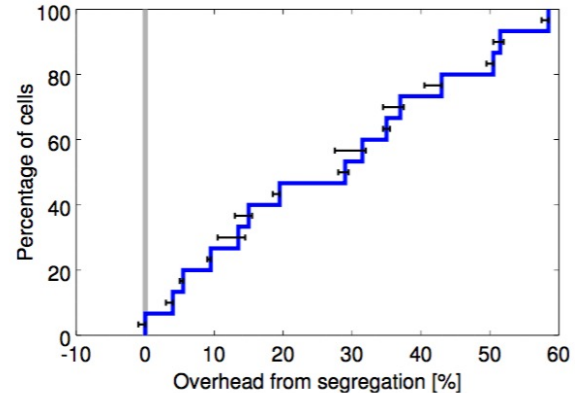 - Spreading tasks across failure domia
  - ...

Utilization

- Cell Sharing
  - Segregating prod and non-prod work into different cells would need more machines
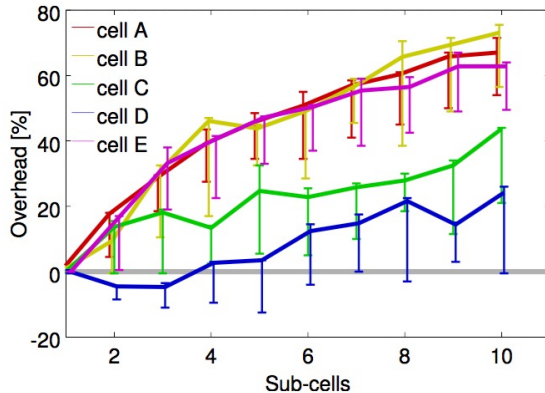

(a) *The left column for each cell shows the original size and the combined workload; the right one shows the segregated case.*
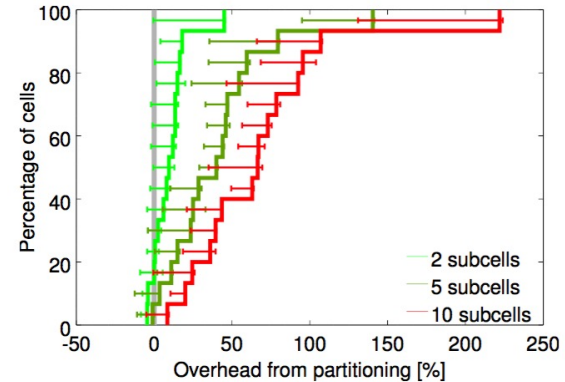

(b) *CDF of additional machines that would be needed if we segregated the workload of 15 representative cells.*

Utilization

- Cell Size
  - Subdividing cells into smaller ones would require more machines

(a) *Additional machines that would be needed as a function of the number of smaller cells for five different original cells.*
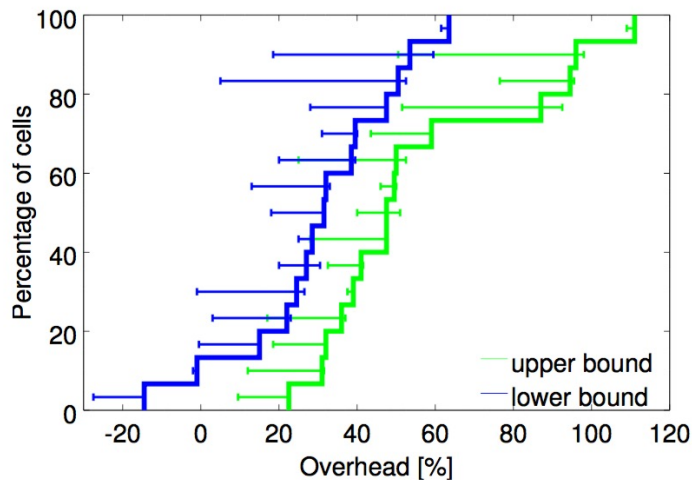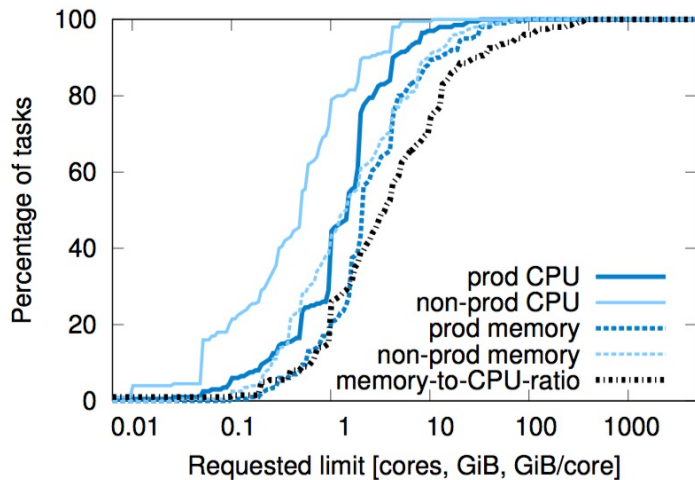
(b) *A CDF of additional machines that would be needed to divide each of 15 different cells into 2, 5 or 10 cells.*
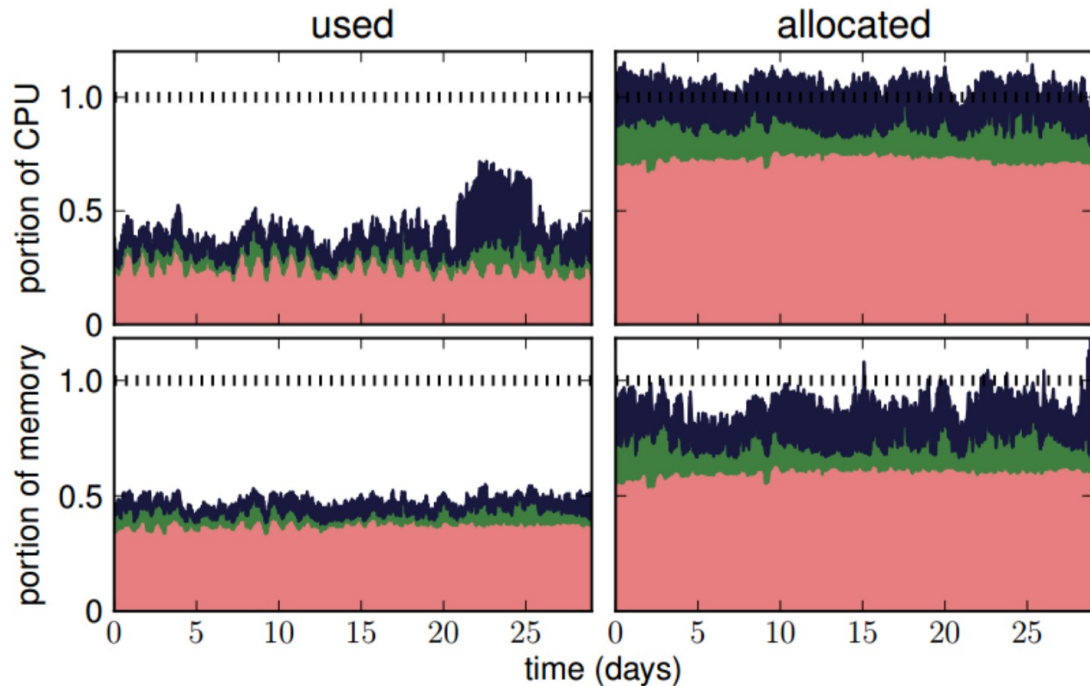
Utilization

- Fine-grained resource requests

No bucket sizes fit most of the tasks well. Bucketing resource would need more machines
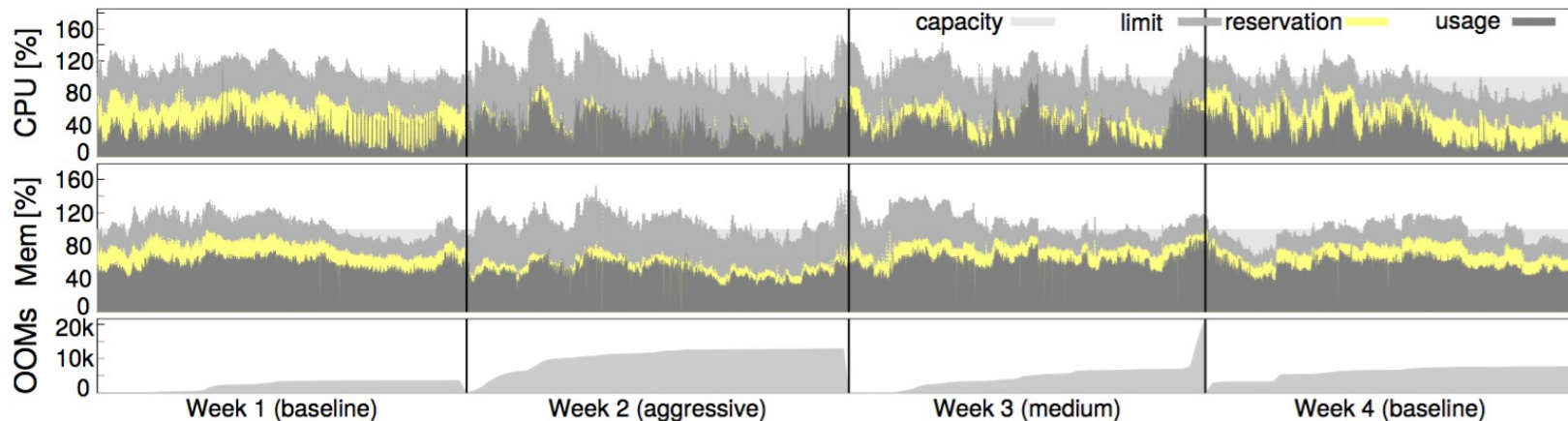
Utilization

- Resource reclamation
  - estimate how many resources a task will use and reclaim the rest for work
  - Kill non-prod if not available

Utilization

- Resource reclamation
  - Choose medium

Isolation

- Security isolation
  - chroot jail as the primary security isolation mechanism

- Performance isolation
  - High-priority LV task(prod) get best treatment
  - compressible resources: reclaimed
  - non-compressible resources: kill or remove task

Kubernetes

- Evolved from Borg

- An open-source system for automating deployment, operations, and scaling of containerized applications

- Pods: groups of containers

- Labels

- Replica controller

- Services