Large-scale cluster management at Google with Borg

By: Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, John Wilkes

Papers We Love - Brasília

December $15^{th}, 2016$

What is Borg?

- A cluster management system developed by Google

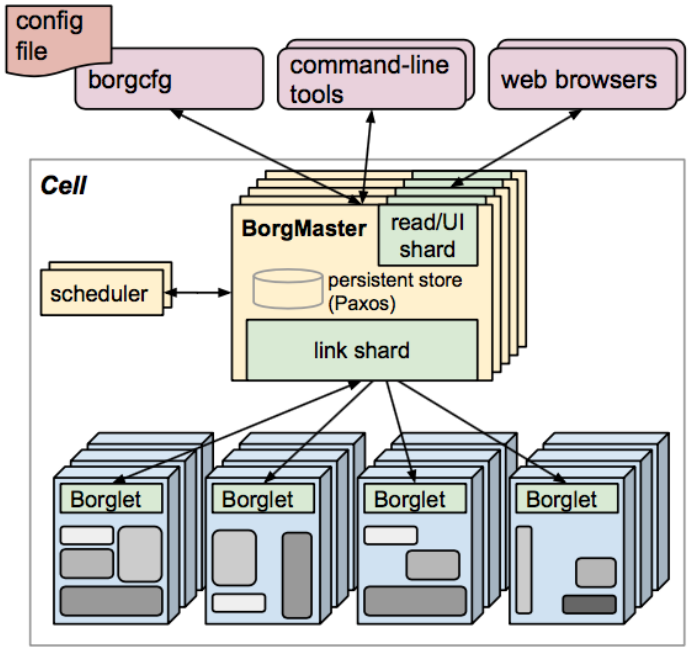


Figure 1: Borg's high-level architecture

What is Borg?

- A cluster management system developed by Google
- Its motivations are:

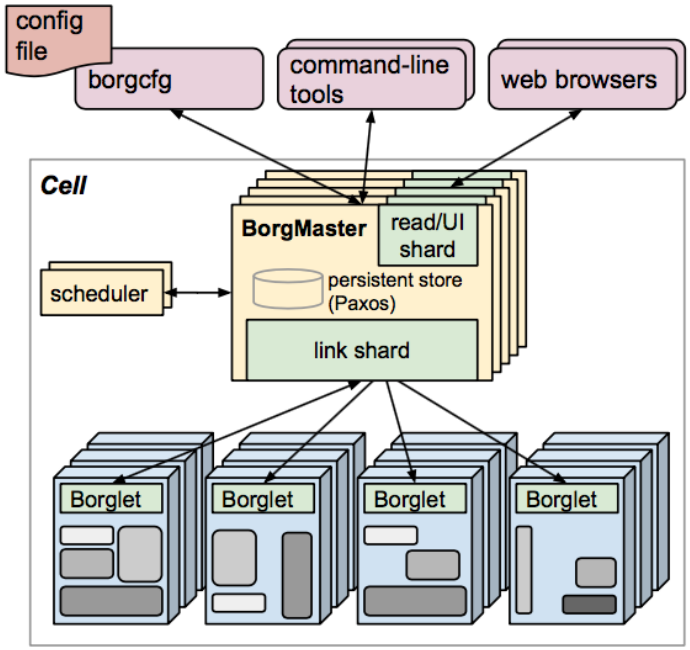


Figure 1: Borg's high-level architecture

What is Borg?

- A cluster management system developed by Google
- Its motivations are:
  1. hide the details of resource management from the users. In other words, to help the software engineers to focus on writing code, instead of dealing with application deployment

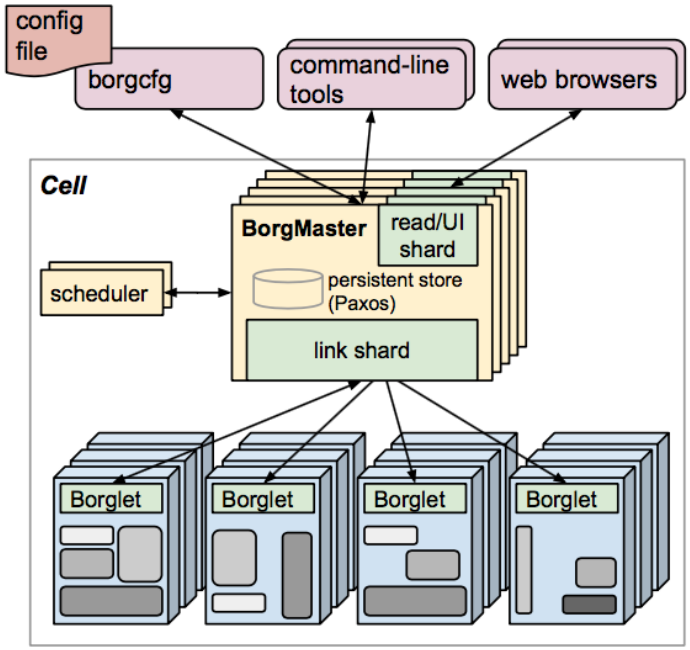


Figure 1: Borg's high-level architecture

What is Borg?

- A cluster management system developed by Google
- Its motivations are:
  1. hide the details of resource management from the users. In other words, to help the software engineers to focus on writing code, instead of dealing with application deployment
  2. provide high reliability and availability environment for the applications

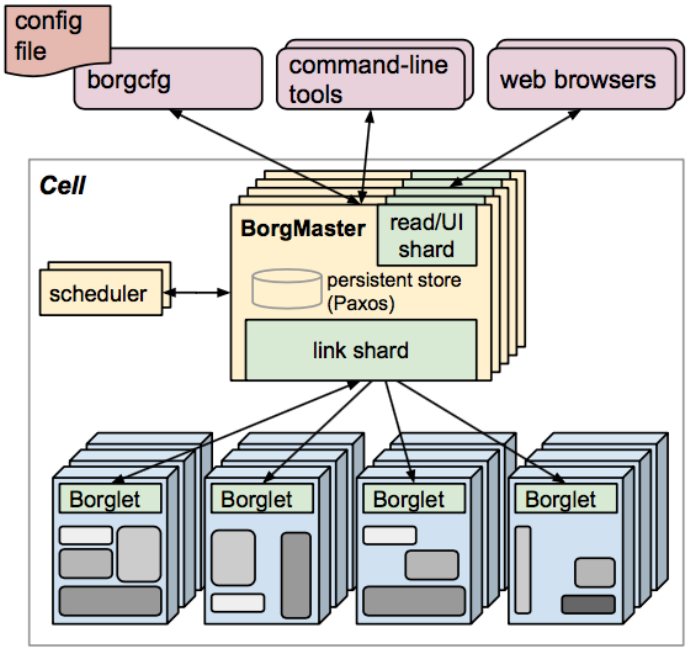


Figure 1: Borg's high-level architecture

What is Borg?

- A cluster management system developed by Google
- Its motivations are:
  1. hide the details of resource management from the users. In other words, to help the software engineers to focus on writing code, instead of dealing with application deployment
  2. provide high reliability and availability environment for the applications
  3. improve resource sharing

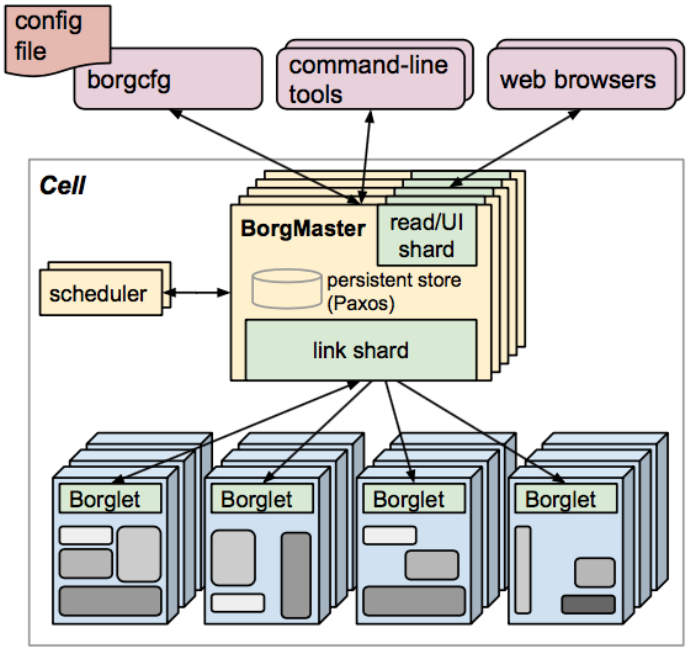


Figure 1: Borg's high-level architecture

The users are mainly software engineers and site reliability engineers (SRE)

- Borg's workload comprises:

The users are mainly software engineers and site reliability engineers (SRE)

- Borg's workload comprises:
  1. long-running services: should "never" go down

The users are mainly software engineers and site reliability engineers (SRE)

- Borg's workload comprises:
  1. long-running services: should "never" go down
  2. batch jobs: take few seconds to few days to complete

The users are mainly software engineers and site reliability engineers (SRE)

- Borg's workload comprises:
  1. long-running services: should "never" go down
  2. batch jobs: take few seconds to few days to complete
- Borg provides a DSL to allow the users to describe the requirement for their jobs

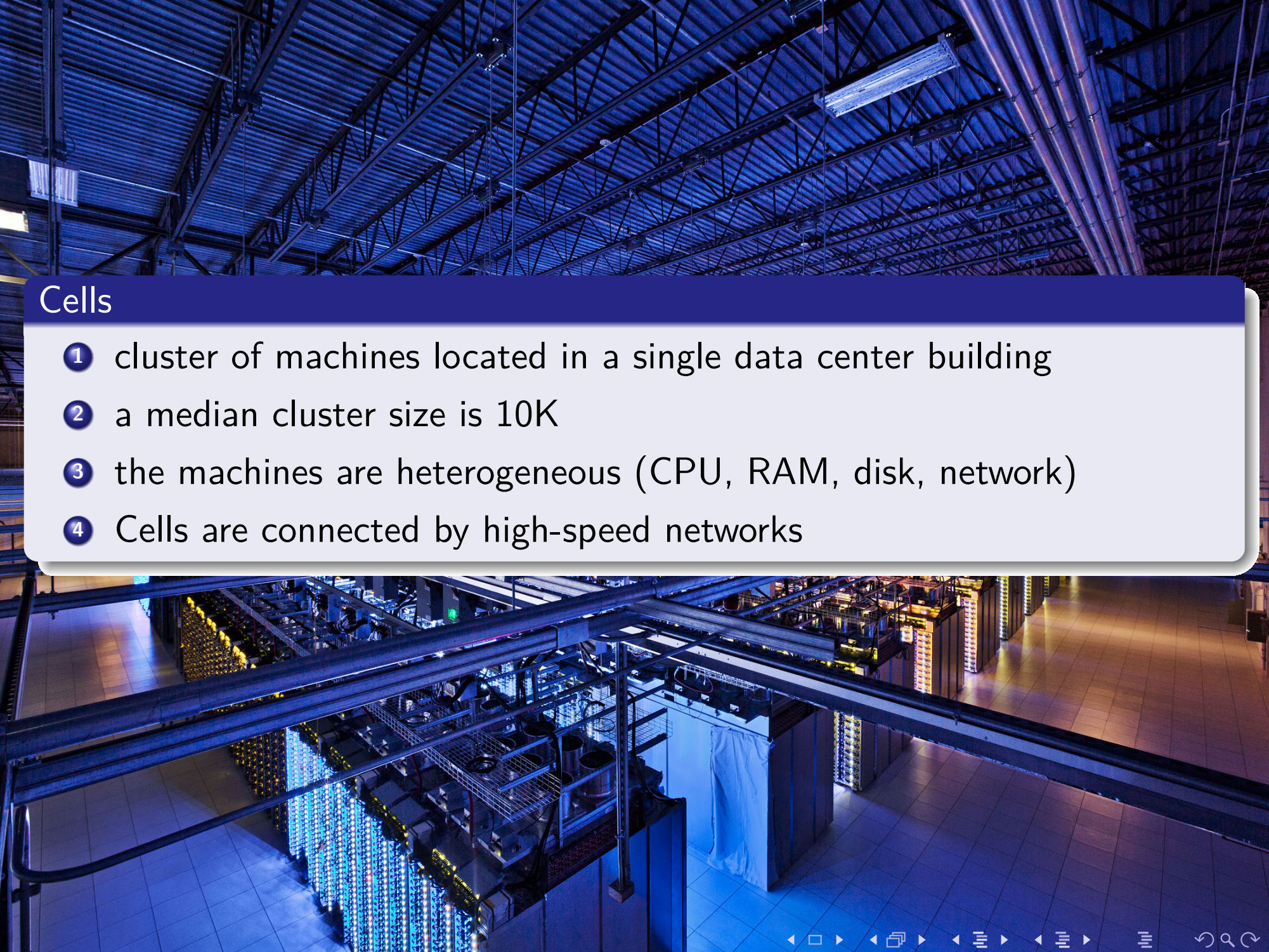
```
job hello_world = {
   runtime = { cell = 'ic' }                // What cluster should we run in?
   binary = '.../hello_world_webserver'     // What program are we to run?
   args = { port = '%port%' }               // Command line parameters
   requirements = {     // Resource requirements
   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

The users are mainly software engineers and site reliability engineers (SRE)

- Borg's workload comprises:
  1. long-running services: should "never" go down
  2. batch jobs: take few seconds to few days to complete
- Borg provides a DSL to allow the users to describe the requirement for their jobs

```
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
}
```

- Likewise, the jobs are classified as production (i.e., higher-priority) and non-production



Cells

1. cluster of machines located in a single data center building
2. a median cluster size is 10K
3. the machines are heterogeneous (CPU, RAM, disk, network)
4. Cells are connected by high-speed networks

1. **Allocs**
   - Reserved set of resources
2. **Priority, quota, and admission control**
   - job has a priority
   - Quota is used to decide which jobs to admit for scheduling
3. **Naming and Monitoring**
   - 50.jfoo.ubar.cc.borg.google.com
   - Monitoring health of the task and thousands of performance metrics

Borg follows a master/slave architecture

1. Borgmaster
   - Borgmaster process and scheduler
   - Replicated
2. Borglet
   - Manage and monitor tasks and resource
   - Borgmaster polls Borglet every few seconds

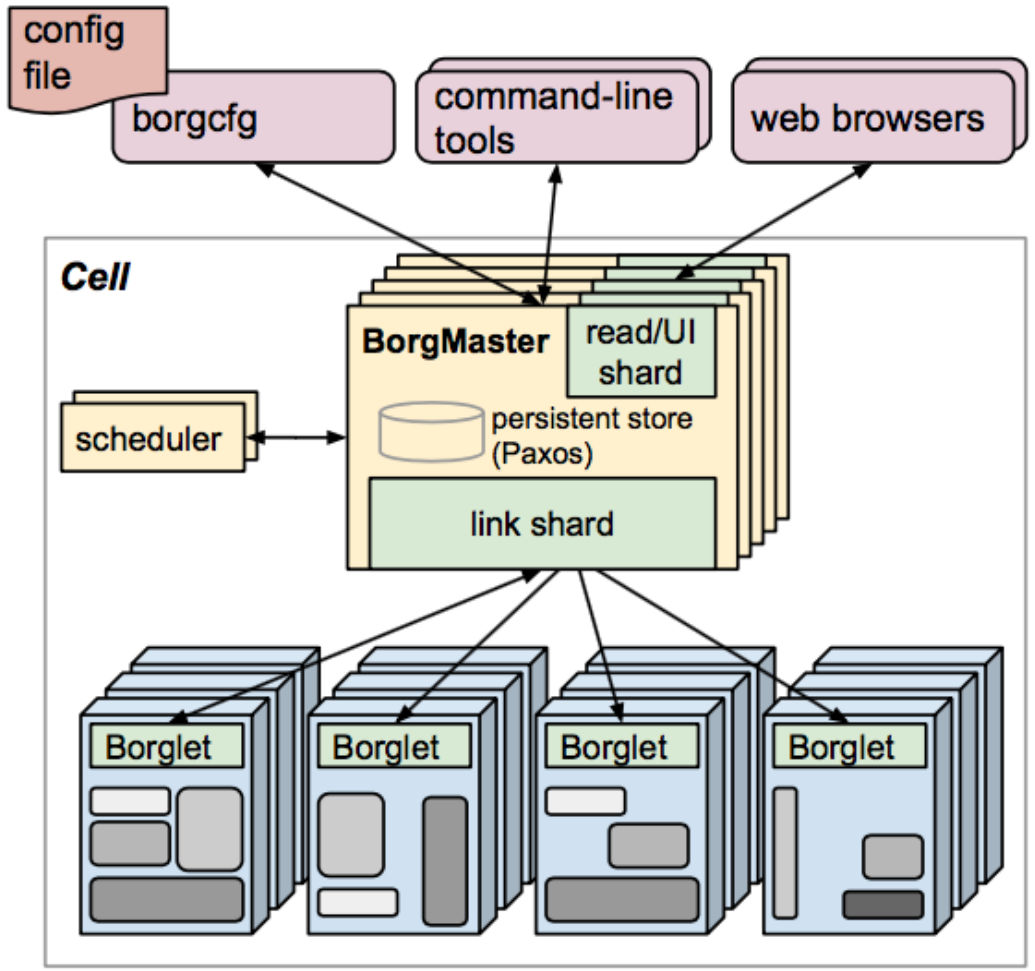


Figure 2: Borg's high-level architecture

A scoring system combines user-specified preferences & build-in criteria

1. Scheduling
   - **feasibility checking**: find machines on which the task could run
   - **scoring**: pick a machine considering different criteria

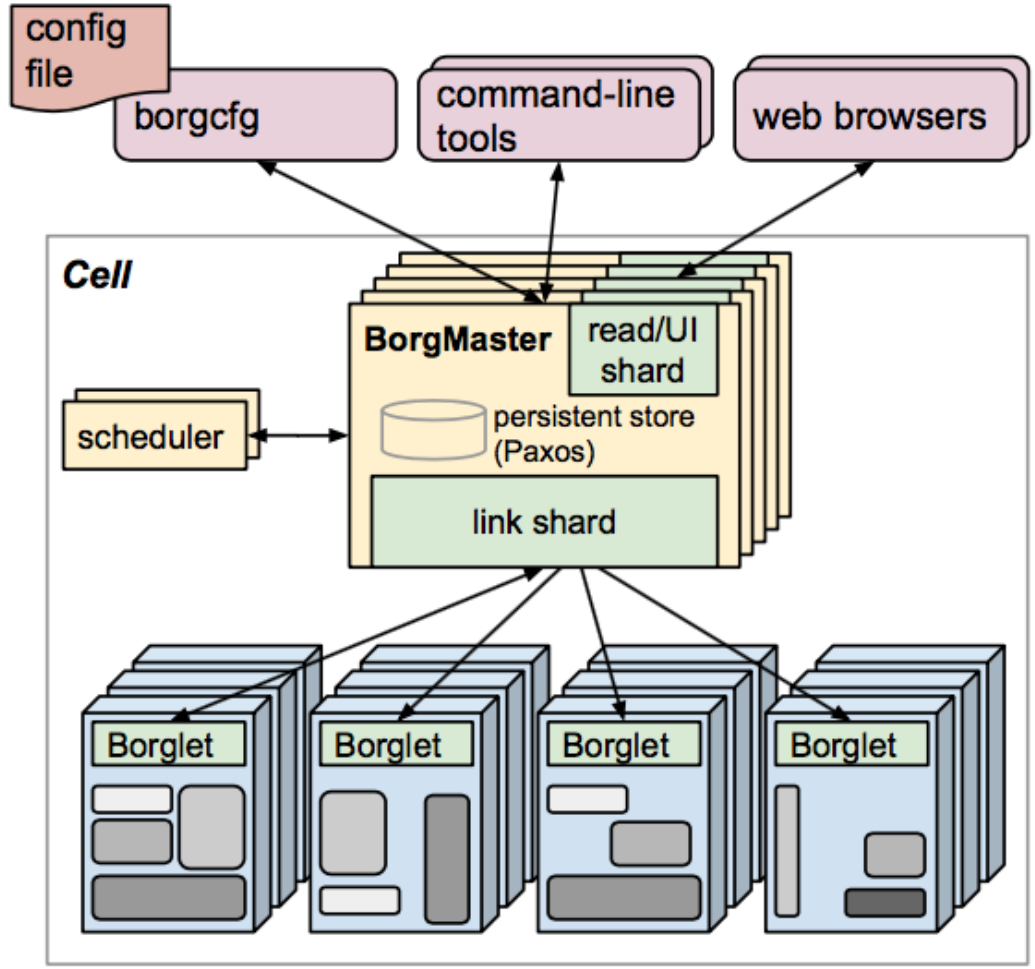


Figure 3: Borg's high-level architecture

Borg employs a modular approach to run its functions across different processes

1. Separate scheduler
2. Separate threads to poll the Borglets
3. Partition functions across the five replicas
4. Score caching: Cache score, ignore small changes
5. Equivalence classes: scoring for one task per equivalence class
6. Relaxed randomization: don't scores all machines, choose random number of machines

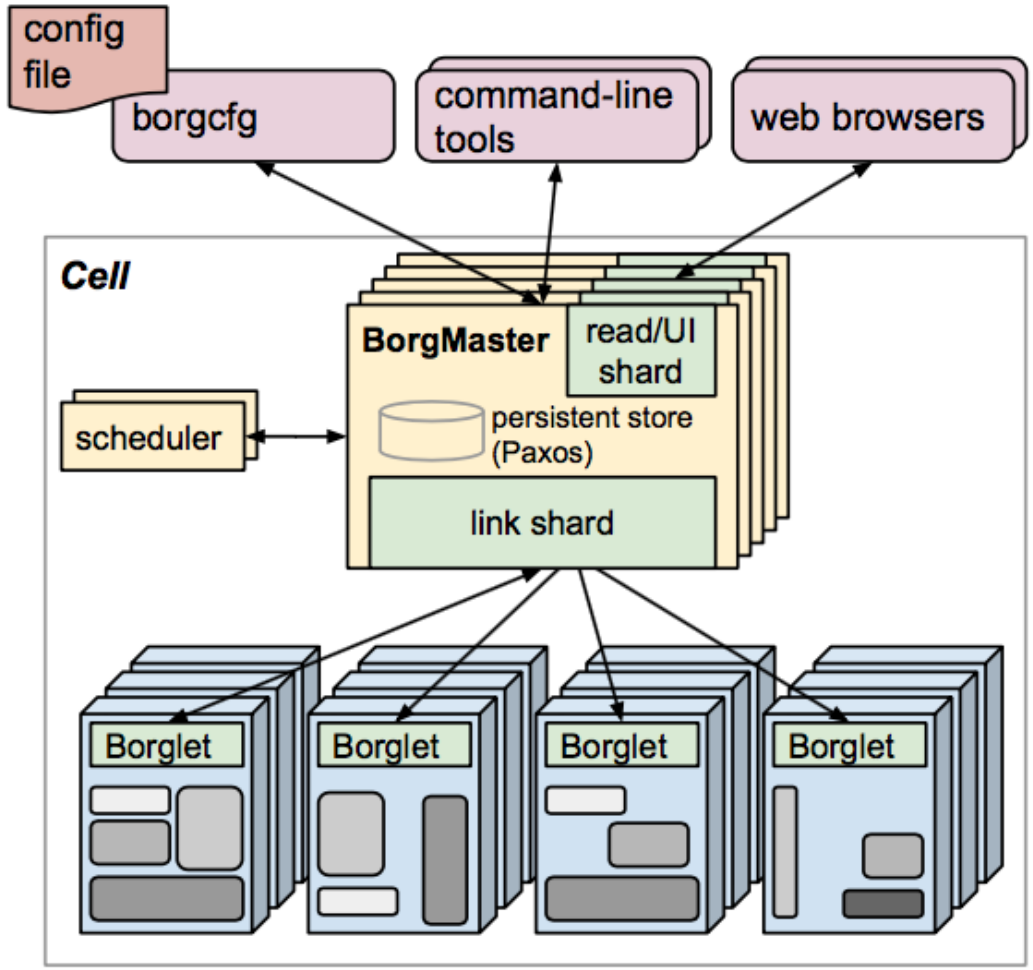
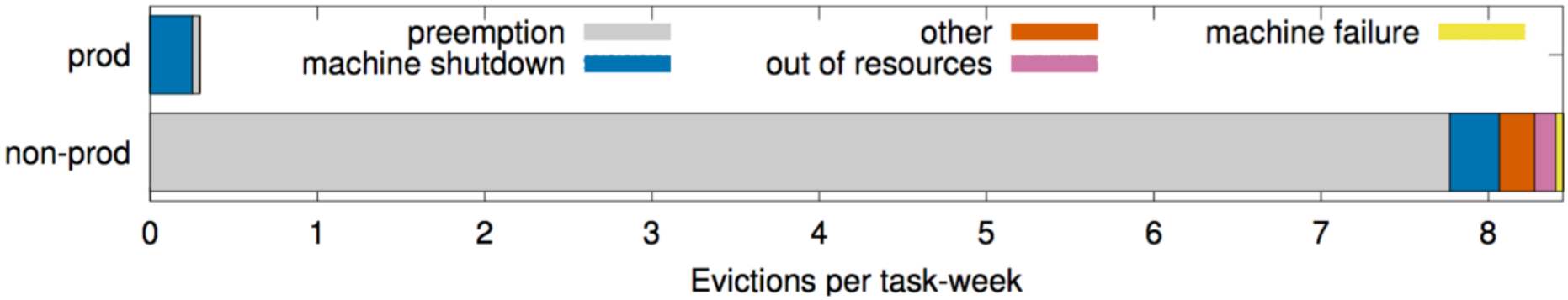
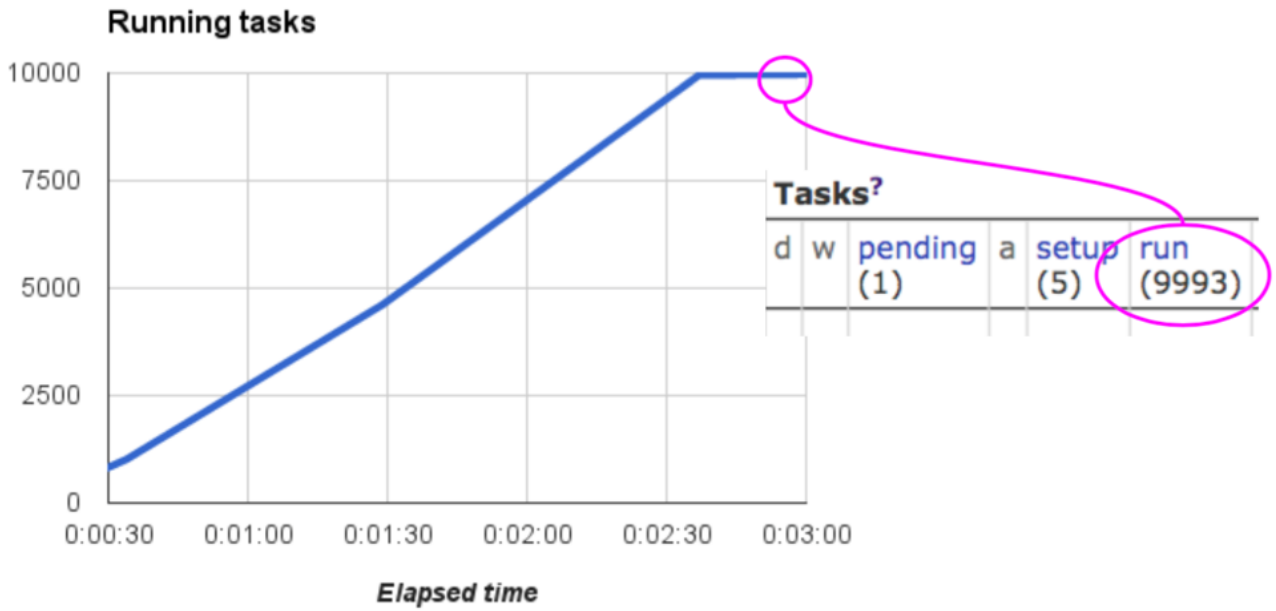


Figure 4: Borg's high-level architecture

Availability

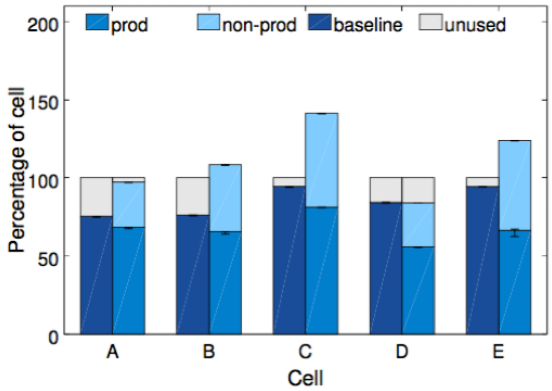


1. Implements checkpoint
2. Reschedules evicted tasks
3. Spreads tasks across failure domain

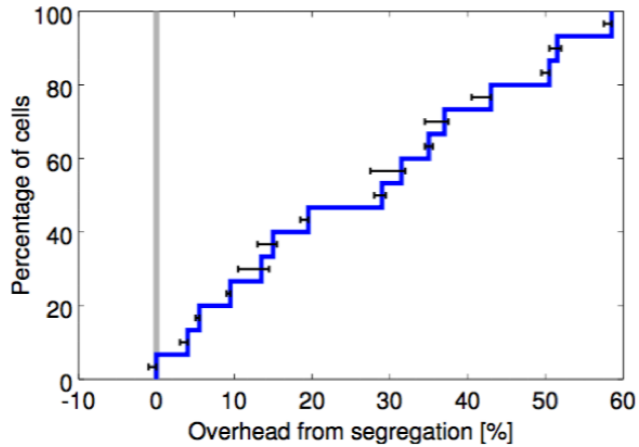


Utilization I

1. **Cell sharing**: segregating prod and non-prod works into different cells would need more machines



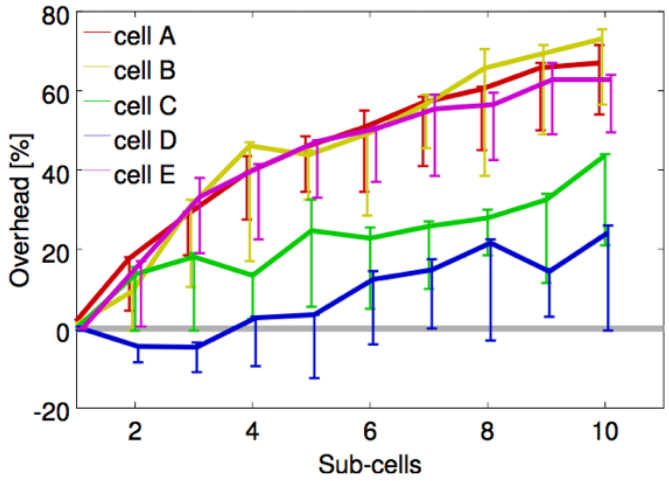
(a) *The left column for each cell shows the original size and the combined workload; the right one shows the segregated case.*



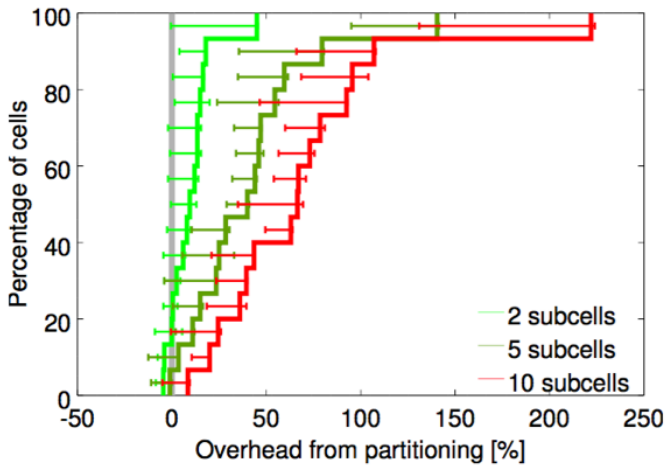
(b) *CDF of additional machines that would be needed if we segregated the workload of 15 representative cells.*

2. **Cell size**: subdividing cells into smaller ones would also require more resources

Utilization II



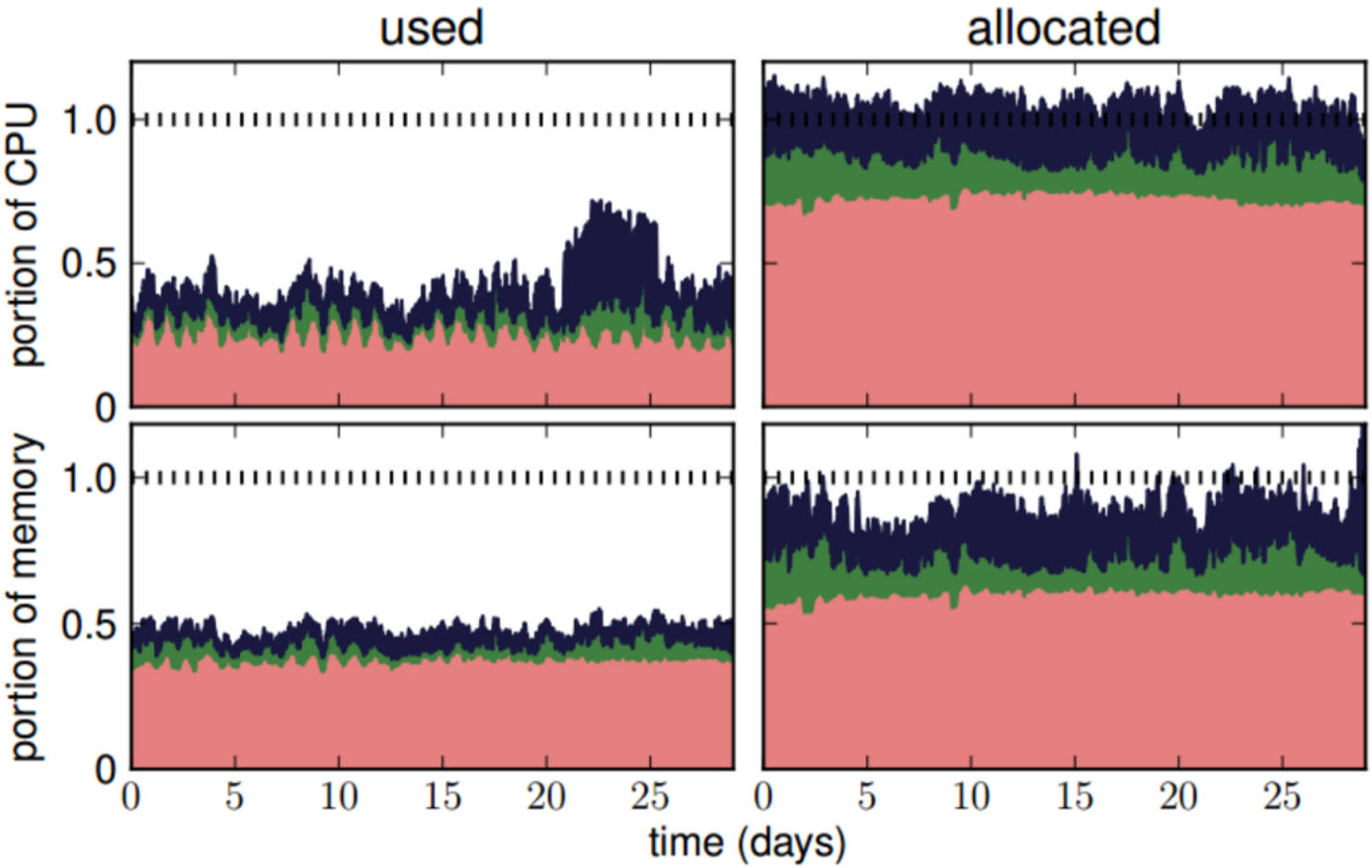
(a) *Additional machines that would be needed as a function of the number of smaller cells for five different original cells.*



(b) *A CDF of additional machines that would be needed divide each of 15 different cells into 2, 5 or 10 cells.*

3. **Resource reclamation**: estimate how many resources a task will use and reclaim the rest for work

Utilization III

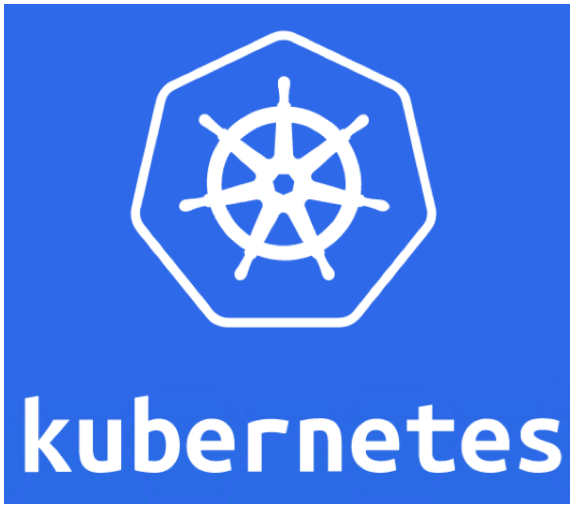


Lessons

1. Bad decisions
   - jobs are restrictive as the only grouping mechanism for tasks
   - one IP address per machine complicates things
   - optimizing for power users at the expense of casual ones
2. Good decisions
   - use of *allocs*
   - cluster management is more than task management
   - introspection is primordial
   - the master is the kernel of a distributed system

Kubernetes evolved from Borg

1. An open-source system for automating deployment, operations, and scaling of containerized applications
2. Pods: groups of containers
3. Labels
4. Replica controller
5. Services



That's all Folks!