Large-scale cluster management at Google with Borg

Abhishek Verma, Luis Pedrosa, Madhukar Korupolu,
David Oppenheimer, Eric Tune, John Wilkes
Google Inc.

Slides heavily derived from John Wilkes's presentation at EuroSys, this year

Borg at Google

- Cluster management system at Google that achieves high utilization by:
  - Admission control
  - Efficient task-packing
  - Over-commitment
  - Machine sharing

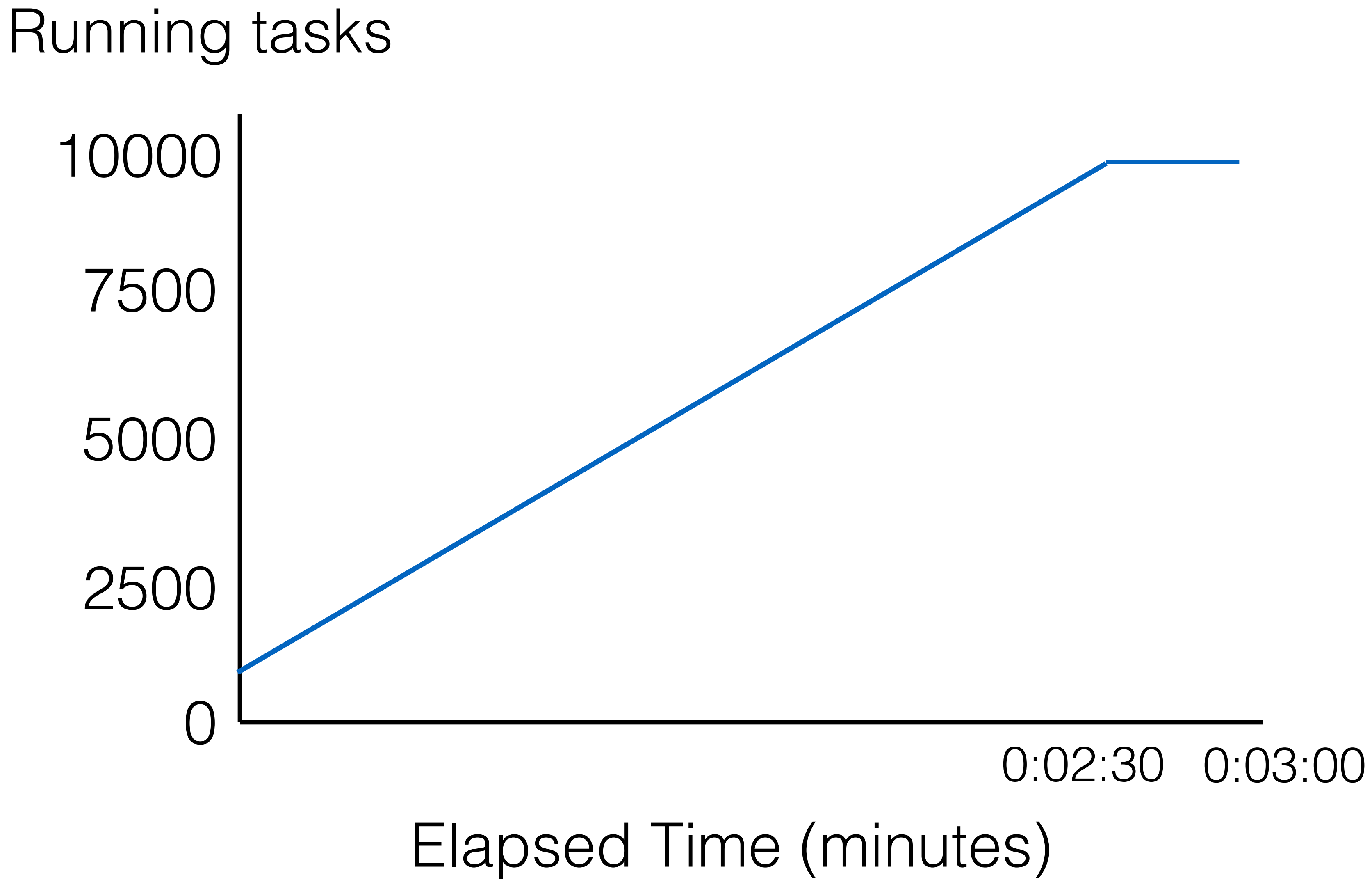
The User Perspective

- Users: Google developers and system administrators mainly
- The workload: Production and batch, mainly
- Cells
- Jobs and tasks
- Allocs and Alloc sets
- Priority, quota and admission control
- Naming and monitoring

The User Perspective

-
  ```
  job hello_world = {
      runtime = { cell = “ic” } //what cell should run it in?
      binary = ‘../hello_world_webserver’ //what program to run?
      args = { port = ‘%port%’ }
      requirements = {
          RAM = 100M
          disk = 100M
          CPU = 0.1
      }
      replicas = 10000
  }
  ```

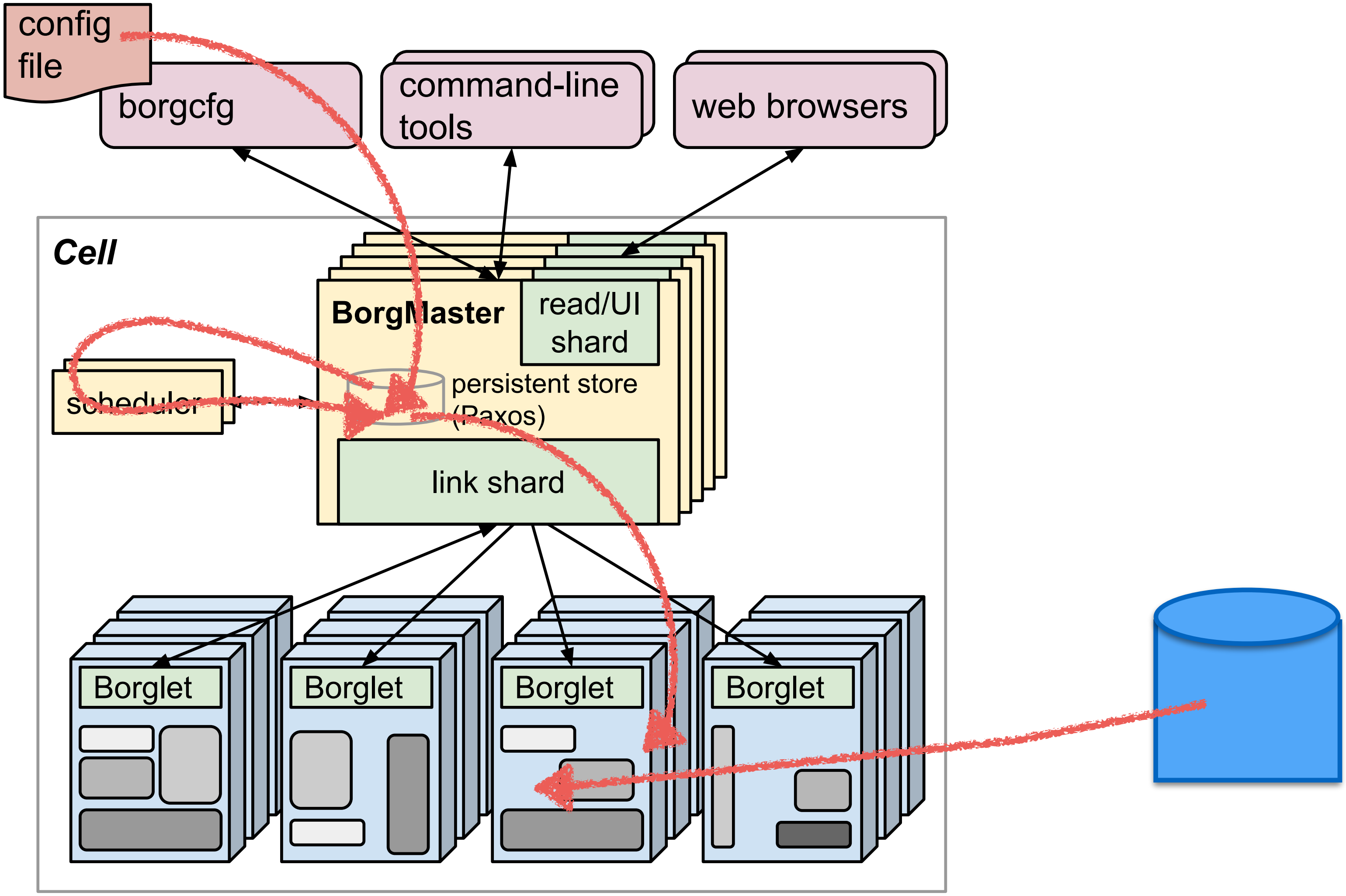
The User Perspective

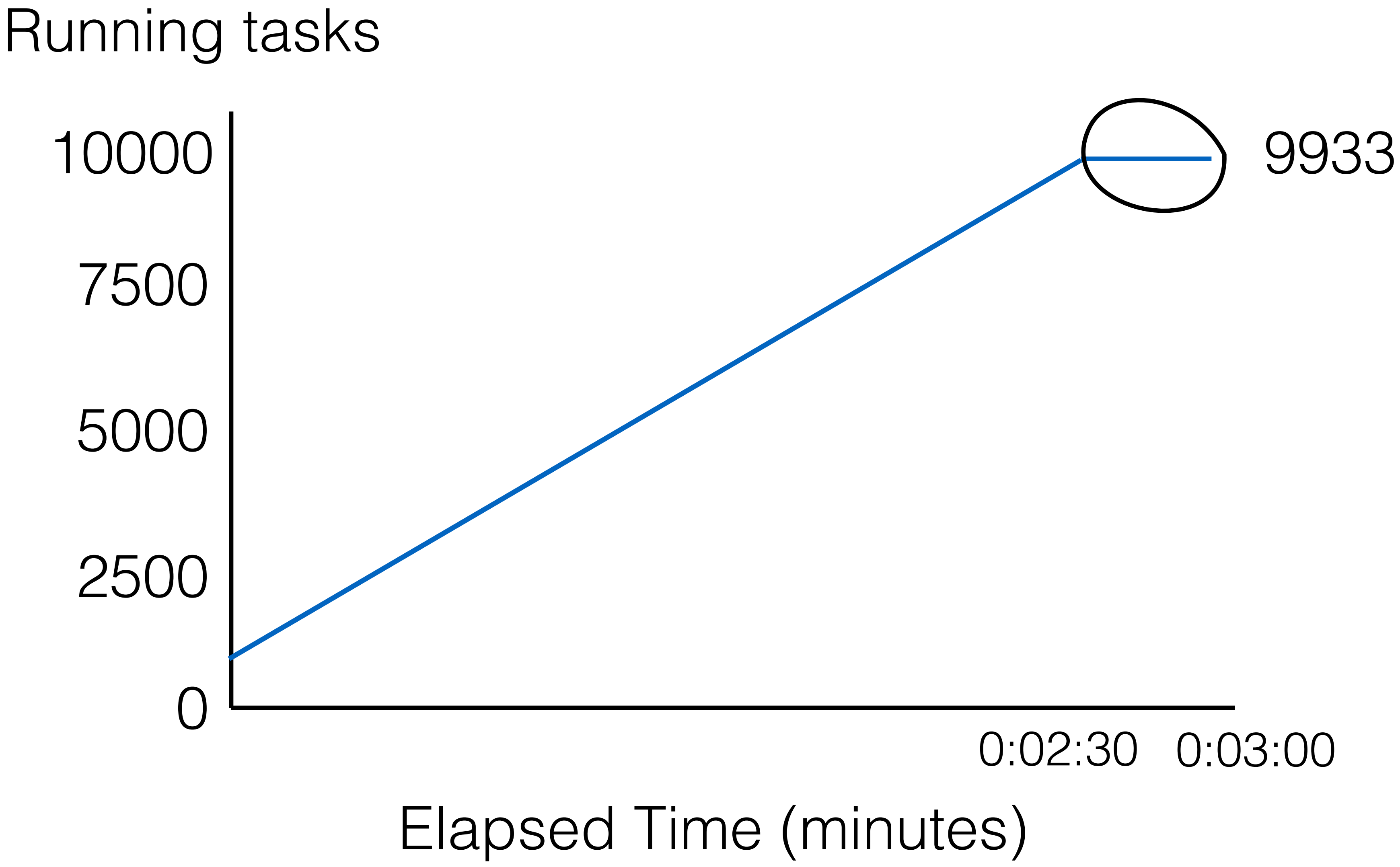


Main Benefits

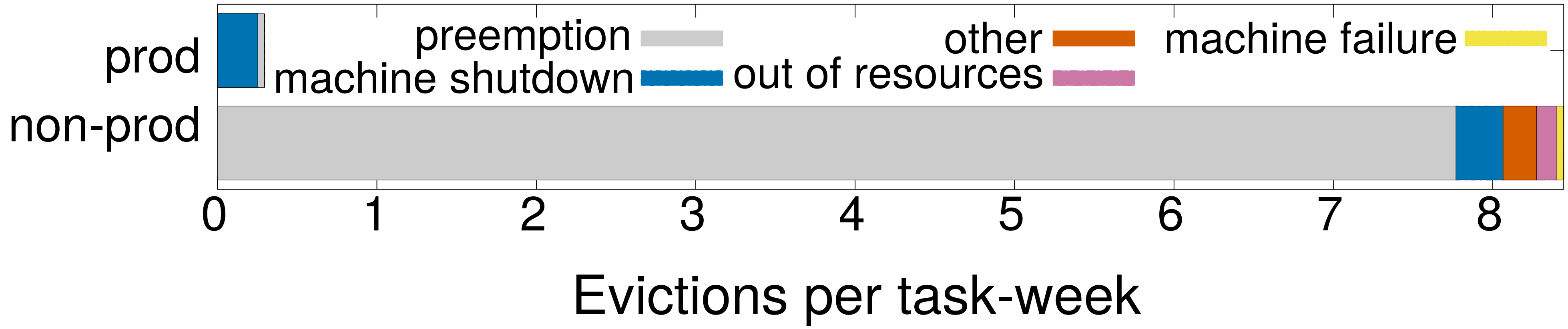
- Provides scalability to run workloads across thousands of machines
- Abstracts away the details of resource management and fault handling from users
- Operates with high reliability and availability

High-level Architecture




Running tasks
10000
7500
5000
2500
0
9933
0:02:30
0:03:00
Elapsed Time (minutes)

Failures

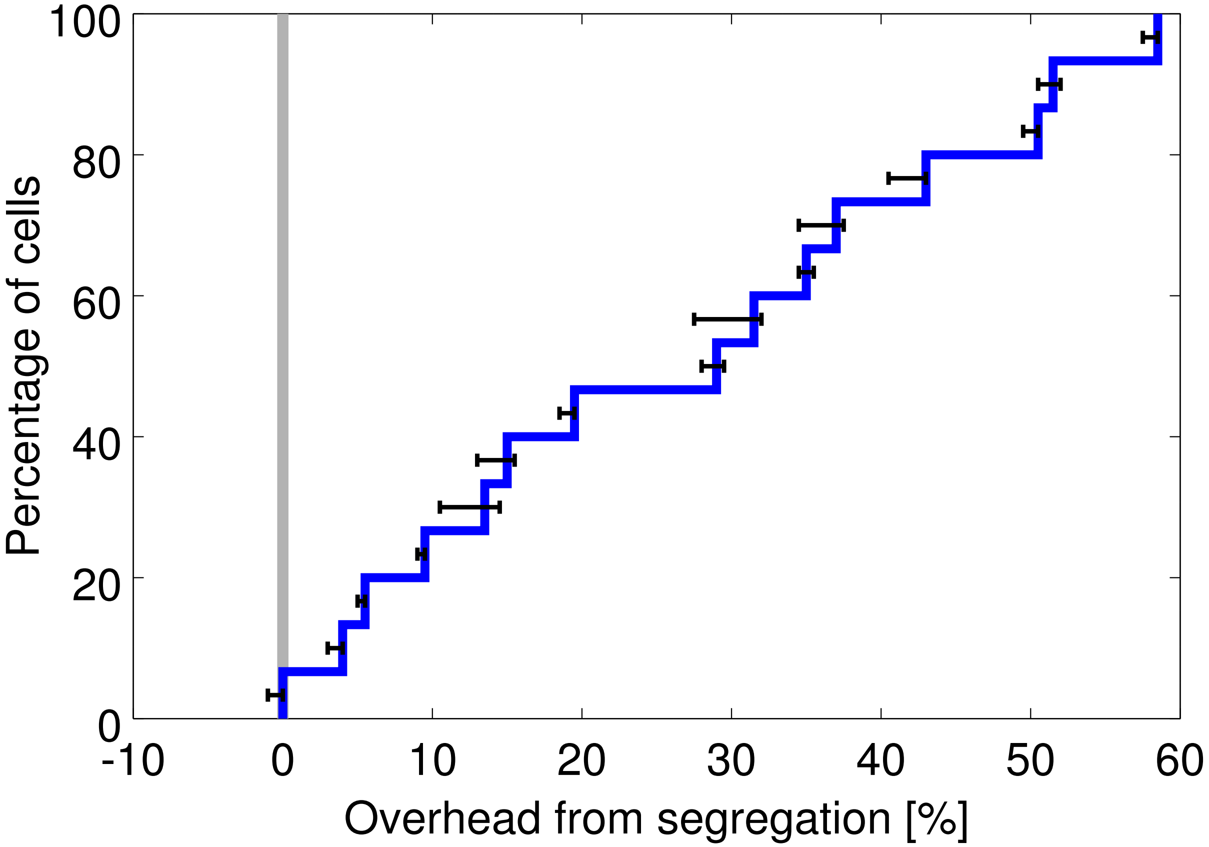

prod
non-prod
preemption
machine shutdown
other
out of resources
machine failure
0
1
2
3
4
5
6
7
8
Evictions per task-week

Efficiency: Is Borg’s policy the best for utilizing clusters?

- Advanced Bin-Packing algorithms:
  - Avoid stranding of resources
- Evaluation metric: Cell-compaction
  - Find the smallest cell that we can pack the workload into…
  - Remove machines randomly from a cell to maintain cell heterogeneity
- Evaluated various policies to understand the cost, in terms of extra machines needed for packing the same workload

Should we share cluster?

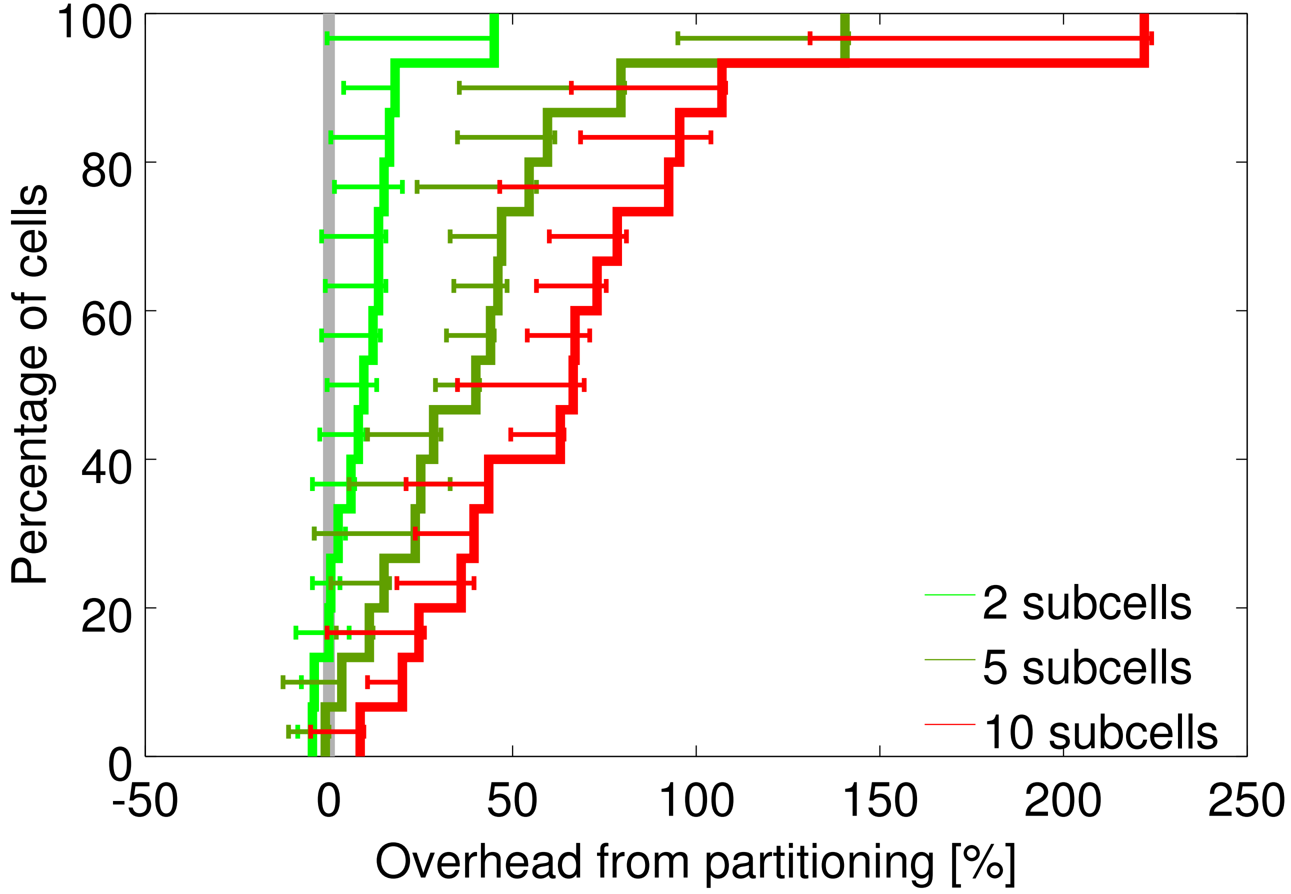
- …between production and non-production workloads?



- Segregating them would need more machines!

Why such large cells?

- Should we split them into smaller cells?



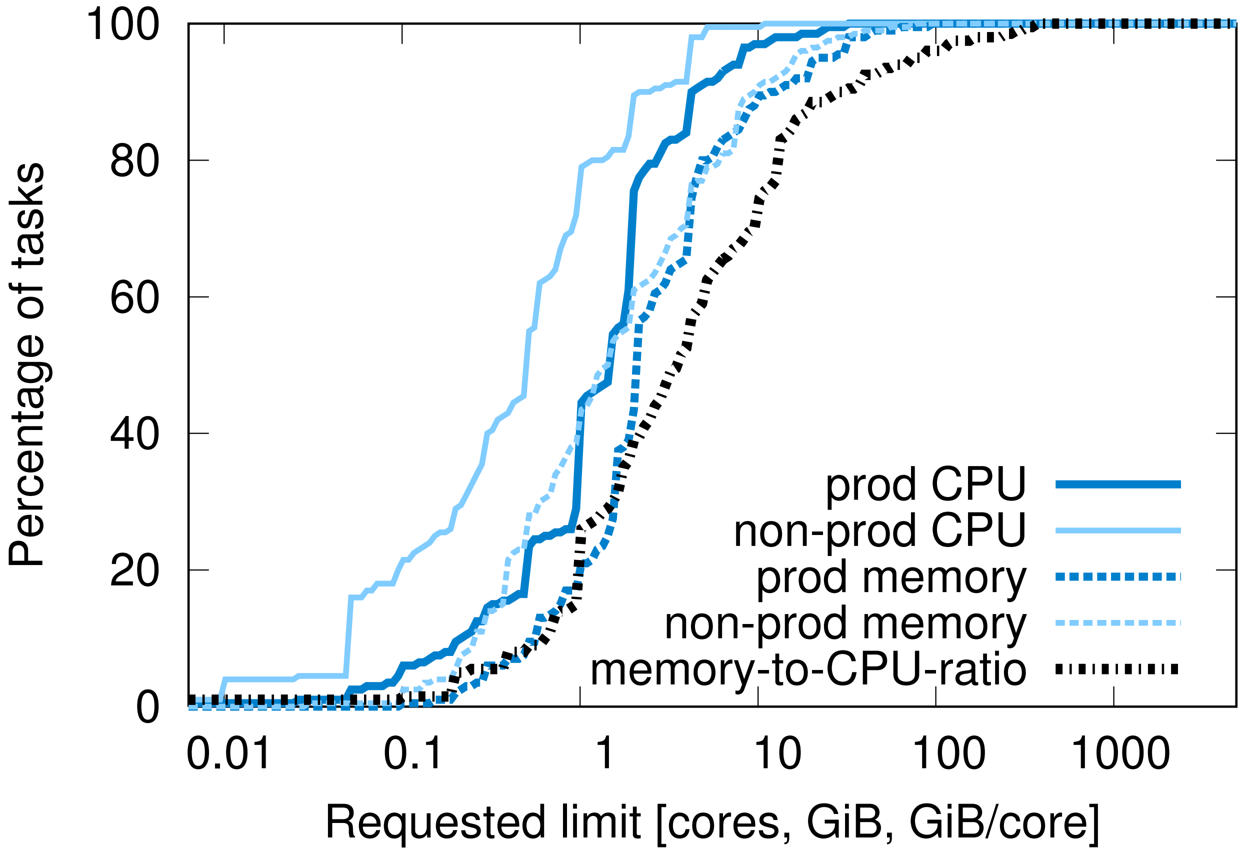
- …might end up having to partition workload across multiple sub-clusters
- would need more machines
- …might be useful to share a cell between users

Should we make cells even larger?

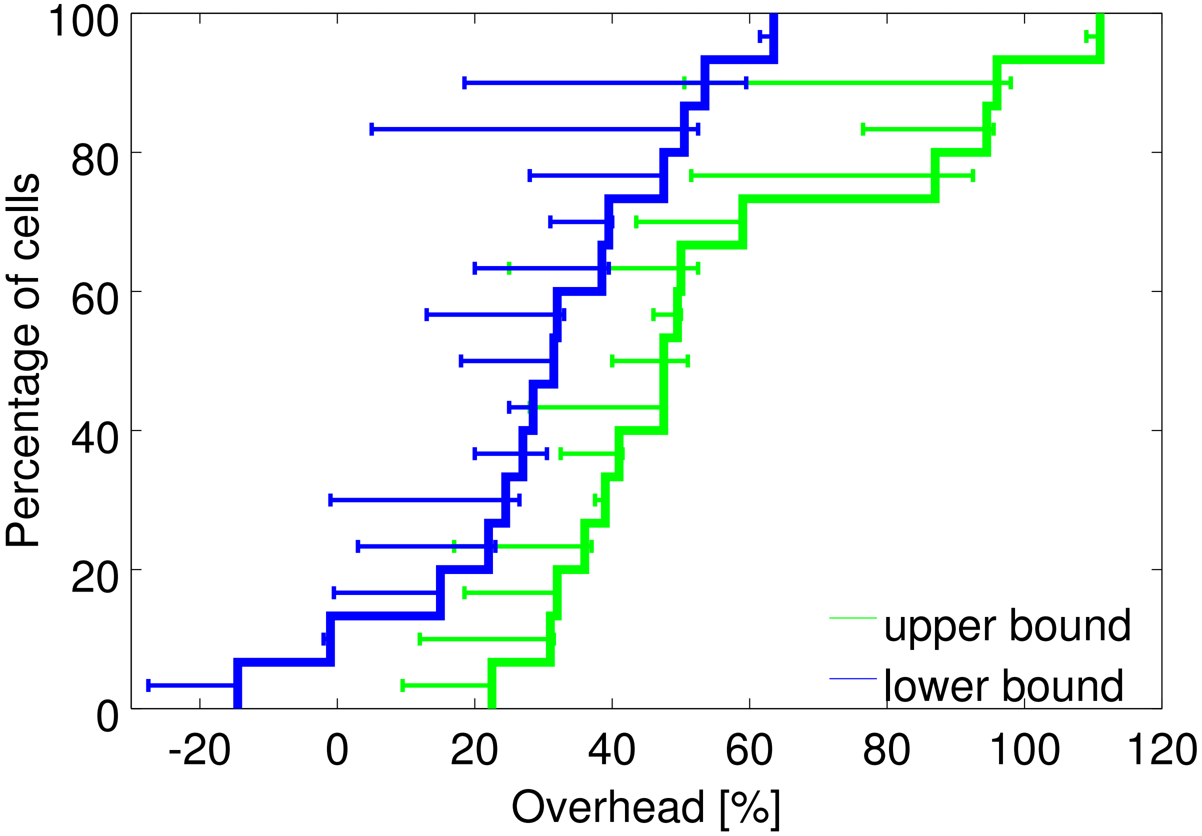
- Failure containment

Would fixed resource bucket sizes be better?

- Borg offers flexible resource requirement specification

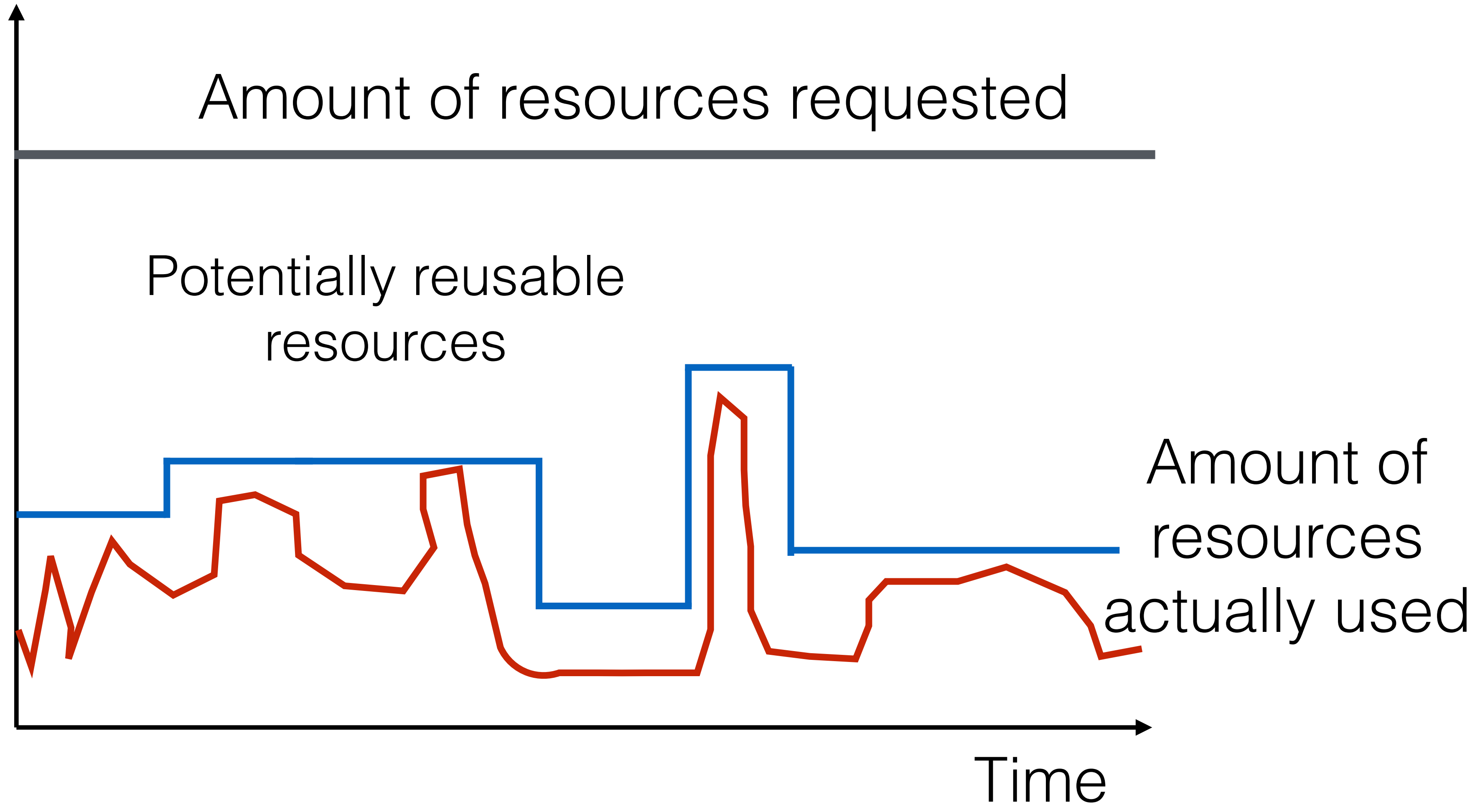


Bucketing resource requirements

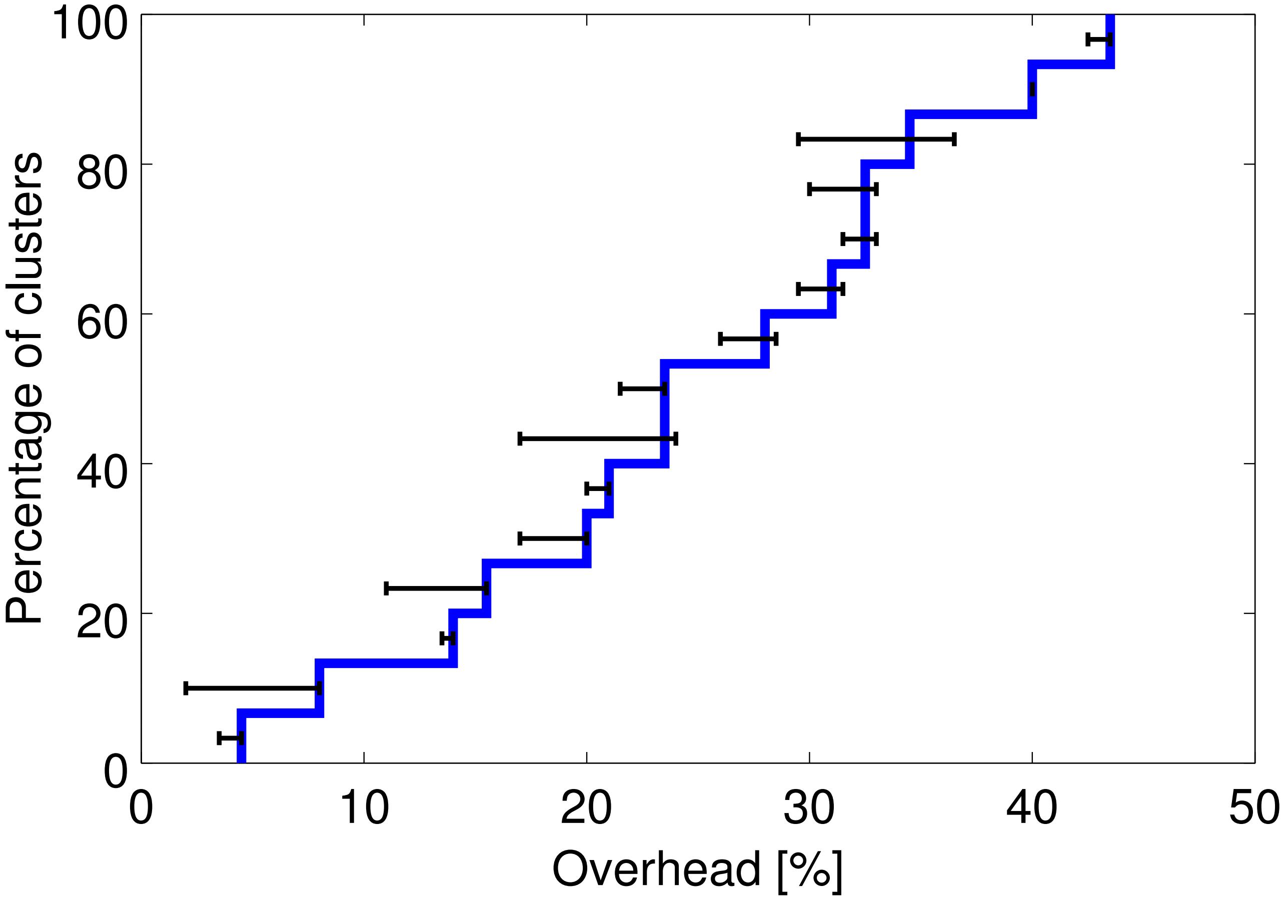


- …would need more machines

Resource Reclamation

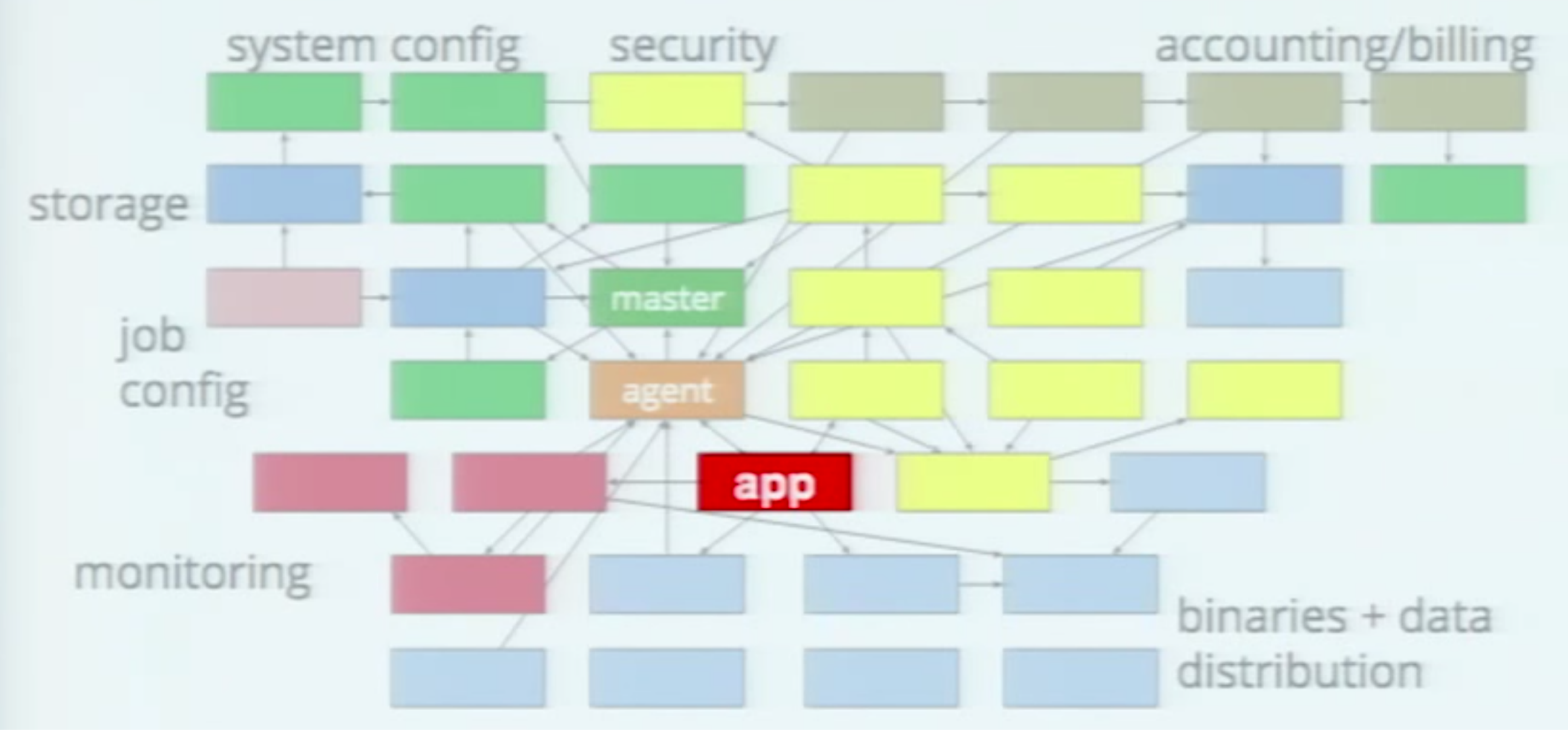


Effectiveness of resource reclamation



- would end up using more machines if resources aren’t reclaimed

Users can focus on their application



Containers

- Google runs everything inside containers, even their VMs
- Containers provide:
    - resource isolation
    - execution isolation

Kubernetes

- An open-source cluster manager derived from Borg
- Also runs on the Google Compute Cloud

- **Directly derived:**
  - Borglet => Kubelet
  - alloc => pod
  - Borg containers => docker
  - Declarative specifications
- **Improved:**
  - Job => labels
  - managed ports => IP per pod
  - Monolithic master => micro-services

Summary

- Resiliency: A lot of attention is given to fault tolerance
- Efficiency: share resources between users, between workloads, reclaim unused resources
- Kubernetes: containers enables users to focus on their applications