Borg

User Perspective

- Heterogenous Workload: production and non-production jobs
- Cell: belong to a cluster with 10k machines
- Jobs and tasks: operates by issuing remote procedure call

User Perspective

- Alloc: a reserved set of resources on a machine
- Priority, quota, and admission control
- Naming and monitoring

Job and tasks

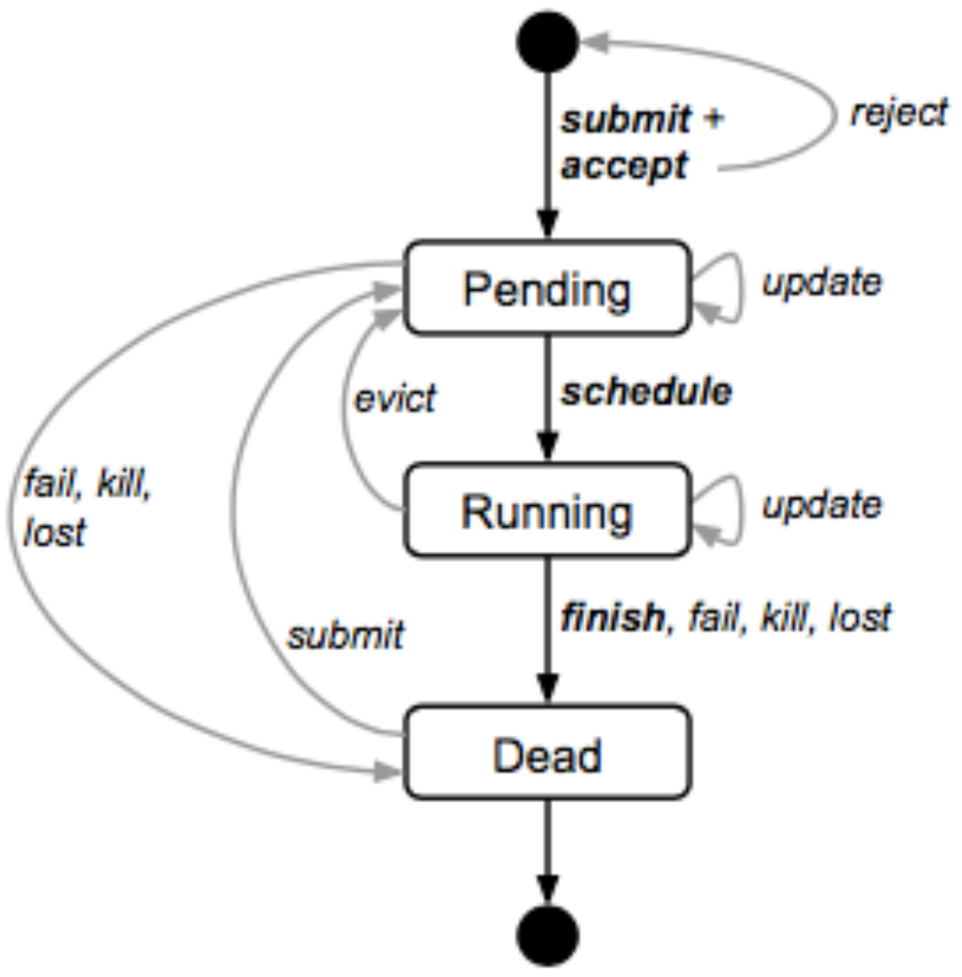


Figure 2: The state diagram for both jobs and tasks. *Users can trigger submit, kill, and update transitions.*

High-level Architecture

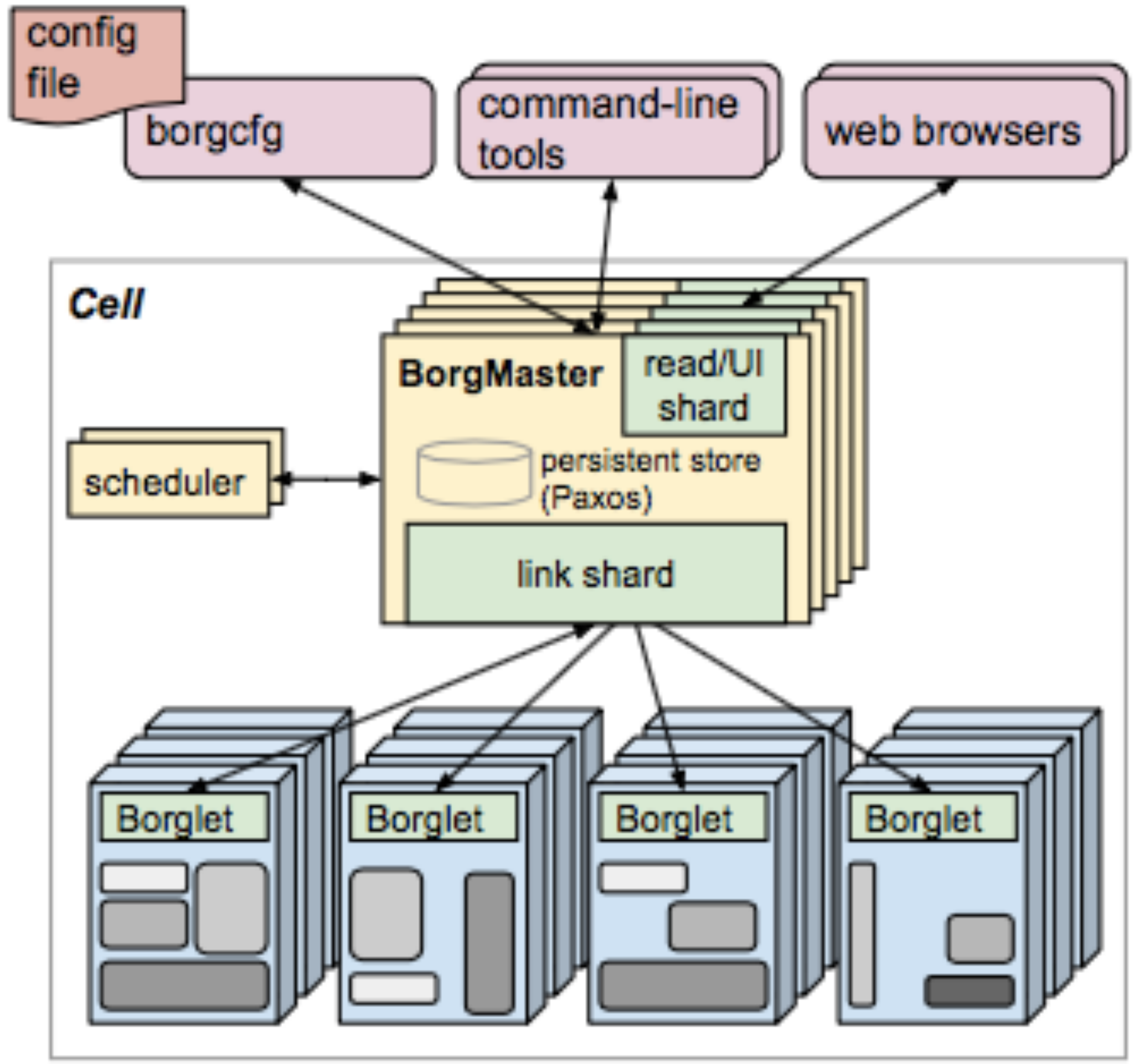
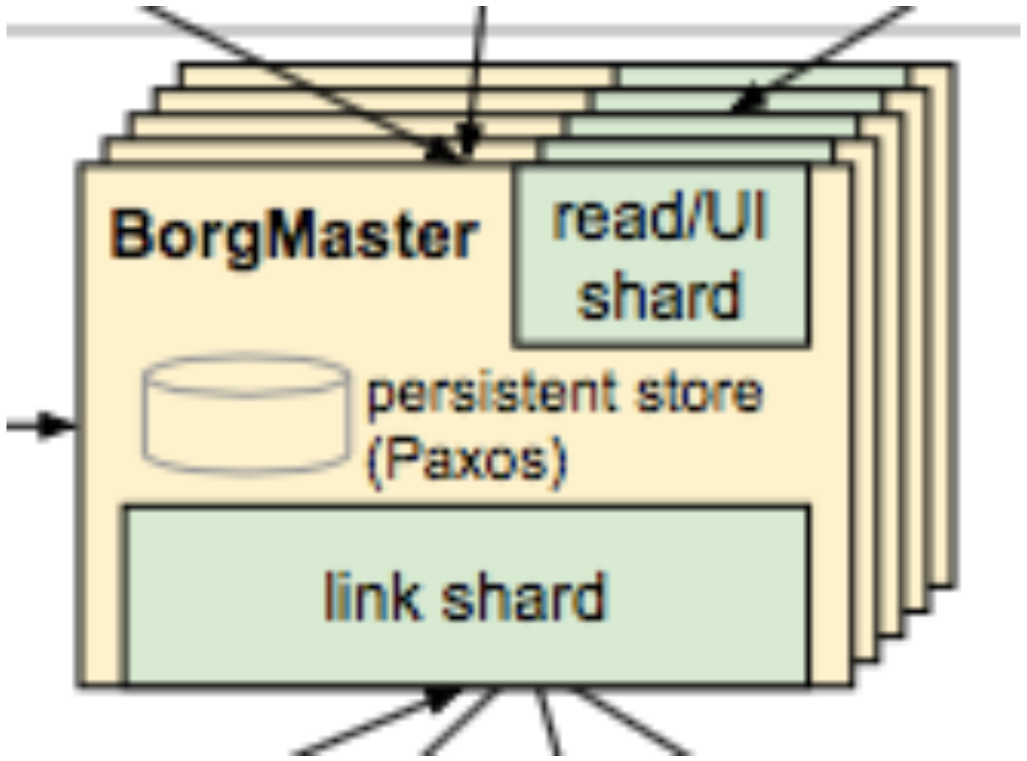


Figure 1: The high-level architecture of Borg. *Only a tiny fraction of the thousands of worker nodes are shown.*

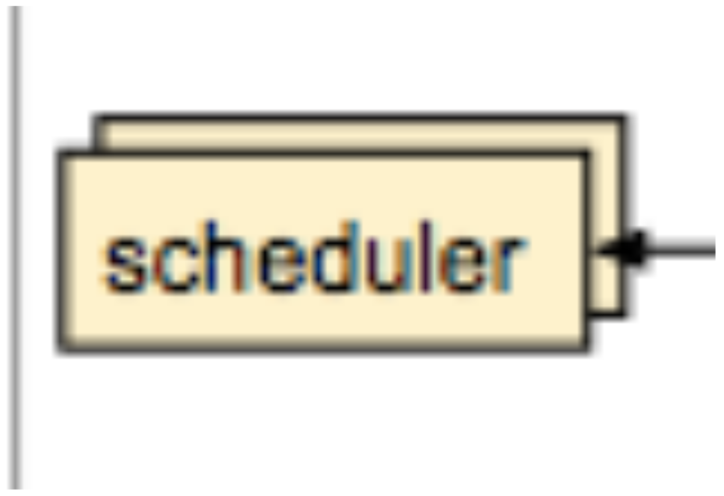
BorgMaster

- Handle clients RPCs
- Manage state machines for objects (machines, tasks, allocs, etc)
- Communicates with the Borglets
- Offer a web UI
- Replicated five times



Scheduling

- Scan jobs in queue and assign tasks asynchronously
  - Feasibility checking: find machines on which tasks can run
  - Scoring: pick one of the feasible machines

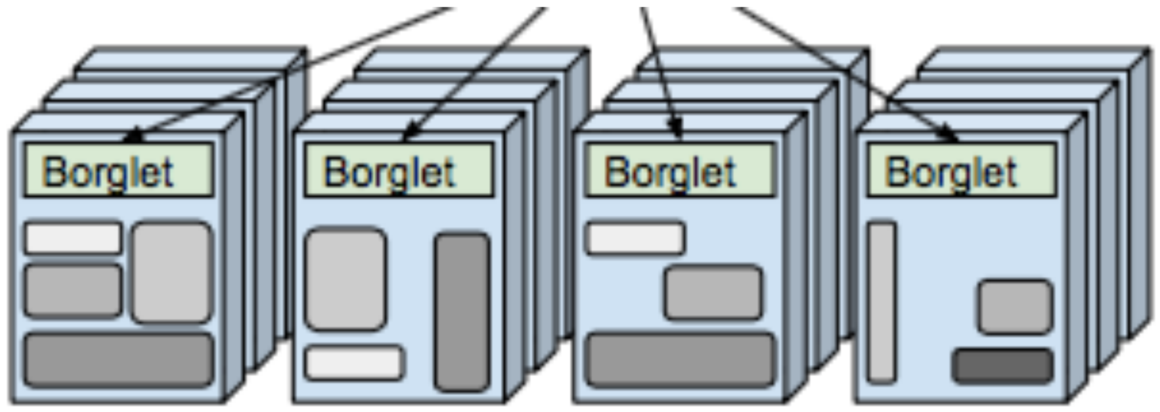


Borglet

- Manage tasks

- Manage local resources

- Report the state of machines



Scalability

- Distributed scheduling: optimistic concurrency control like Omega
- Replicated BorgMaster to respond to read-only RPCs
- Scheduler: score caching; equivalence classes; relaxed randomisation

Availability

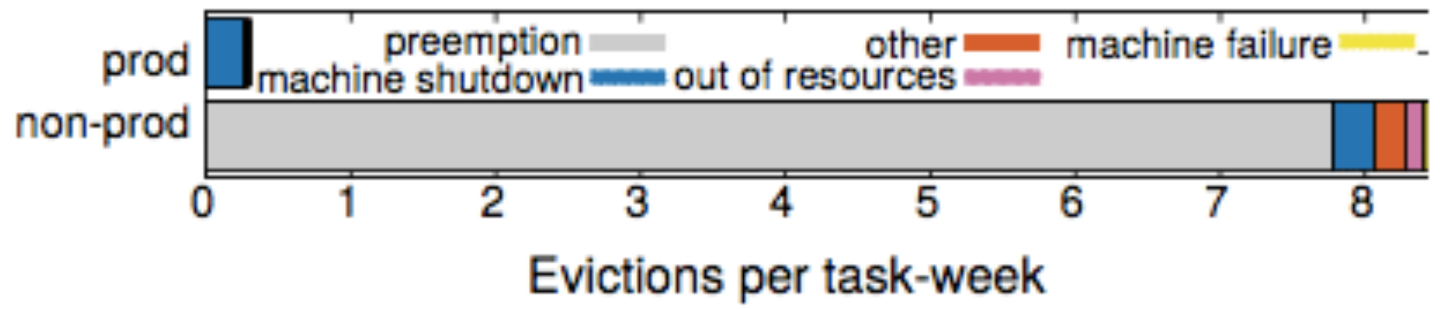


Figure 3: Task-eviction rates and causes for production and non-production workloads. *Data from August 1st 2013.*

- Key feature: already-running tasks continue to run even if the Borgmaster or a Borglet goes down

Utilisation

- Evaluation methodology
- Cell sharing
- Large cell
- Fine-grained resource requests
- Resource reclamation

Evaluation methodology

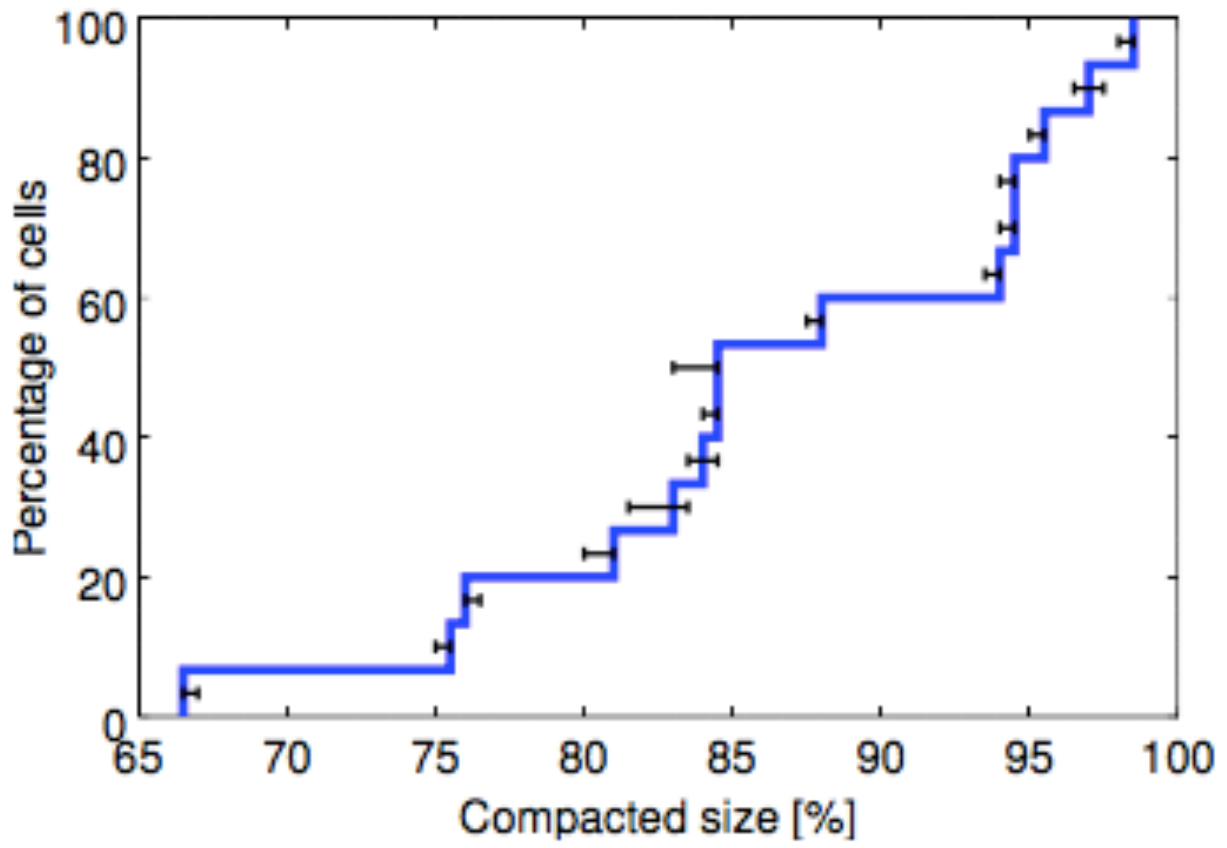
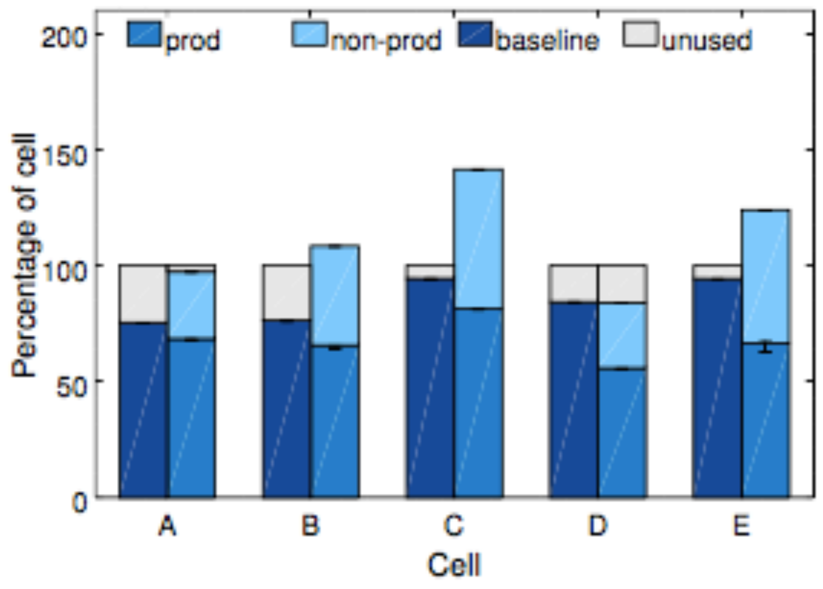


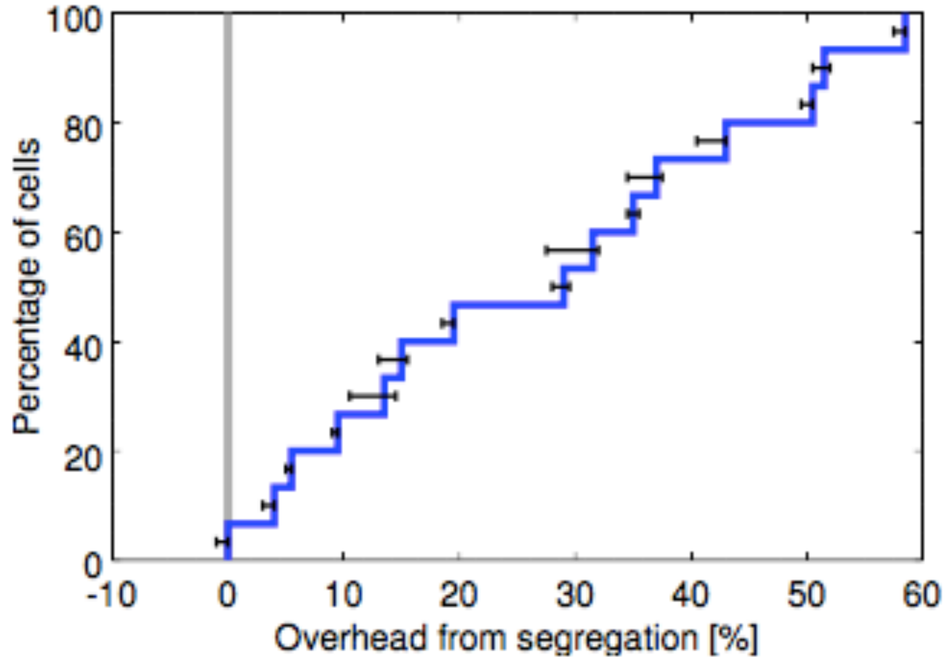
Figure 4: The effects of compaction. *A CDF of the percentage of original cell size achieved after compaction, across 15 cells.*

- Cell compaction: given a workload, removing machines until the workload no longer fitted

Cell sharing

- Question: why sharing resources between prod and non-prod jobs?

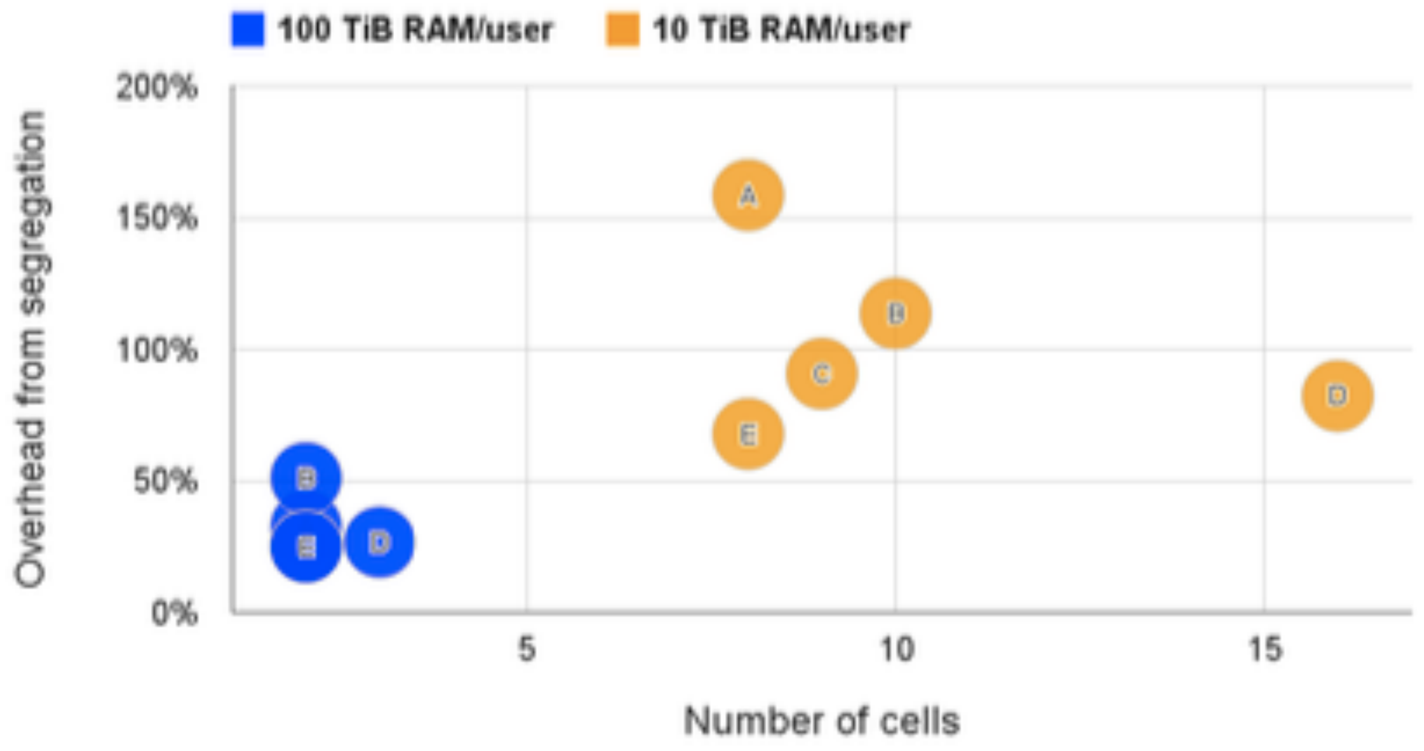




- Conclusion: need more resources if separate prod and non-prod jobs.

Cell sharing

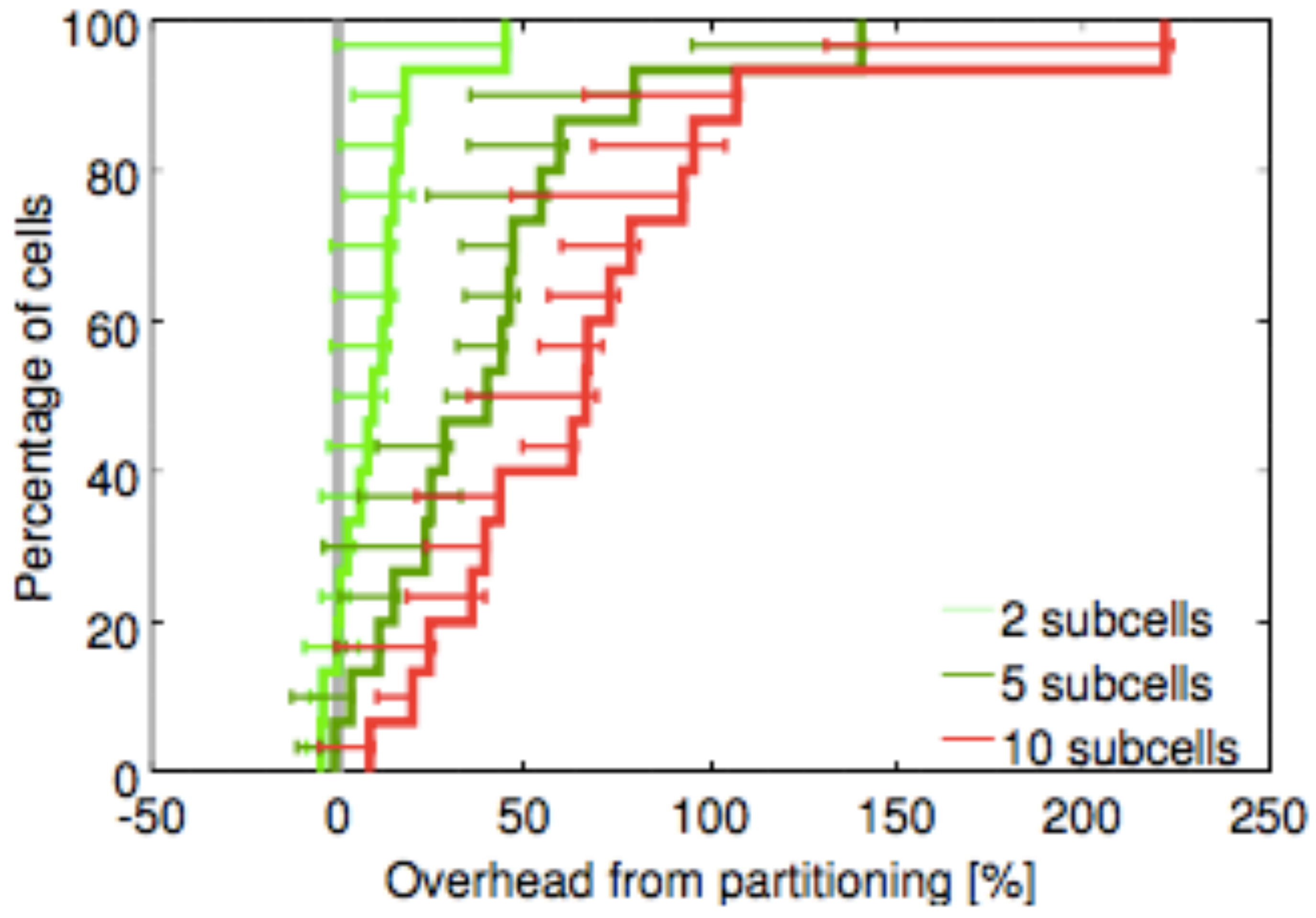
- Question: how about separating picky jobs to different cells?



- Conclusion: need more resources if separate tricky jobs to other cells.

Large cell

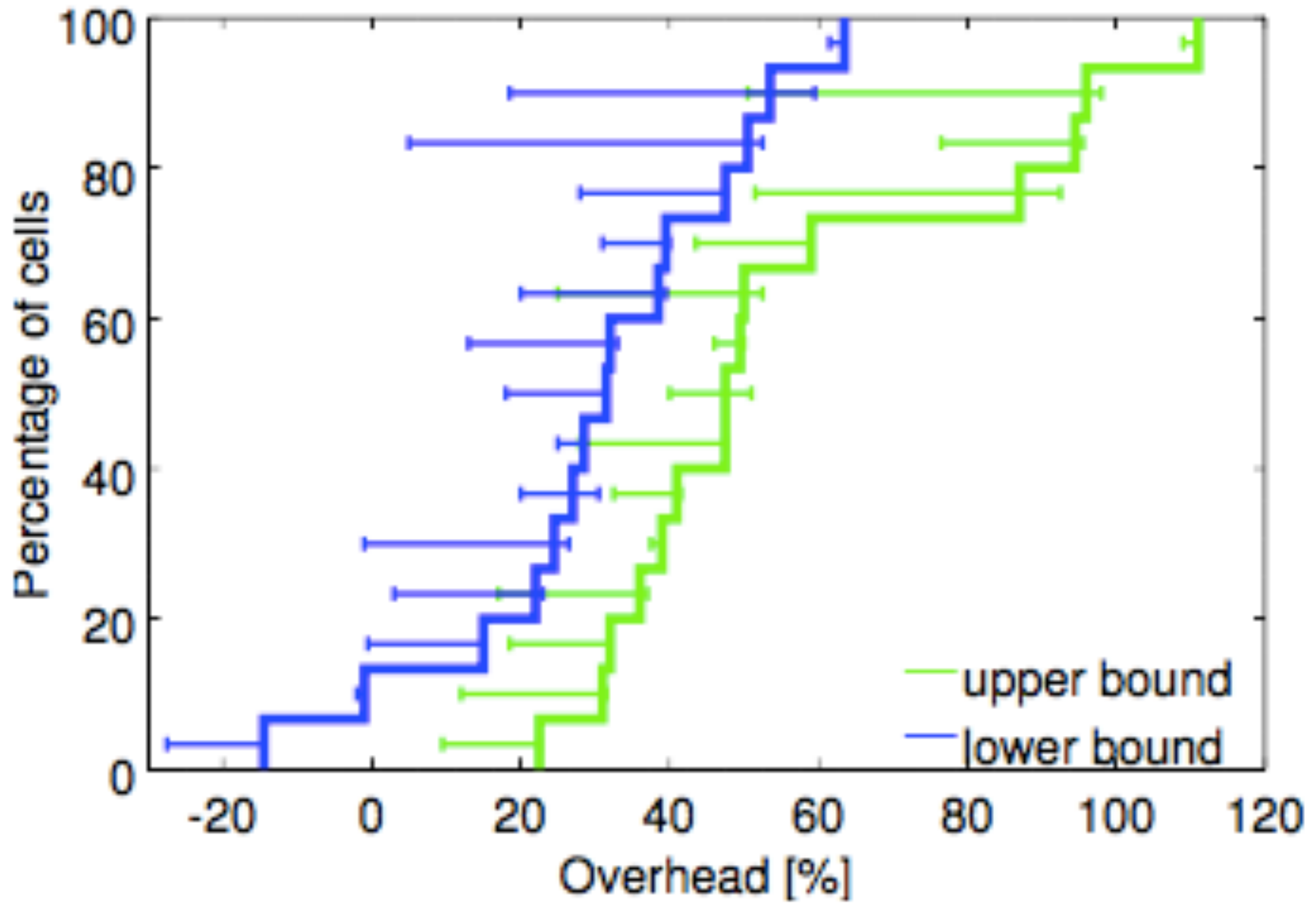
- Question: why use large cell with thousands of machines?



- Conclusion: need more resources if partition cells into subcells.

Fine-grained resource requests

- Question: why not fixed-size containers or virtual machines?

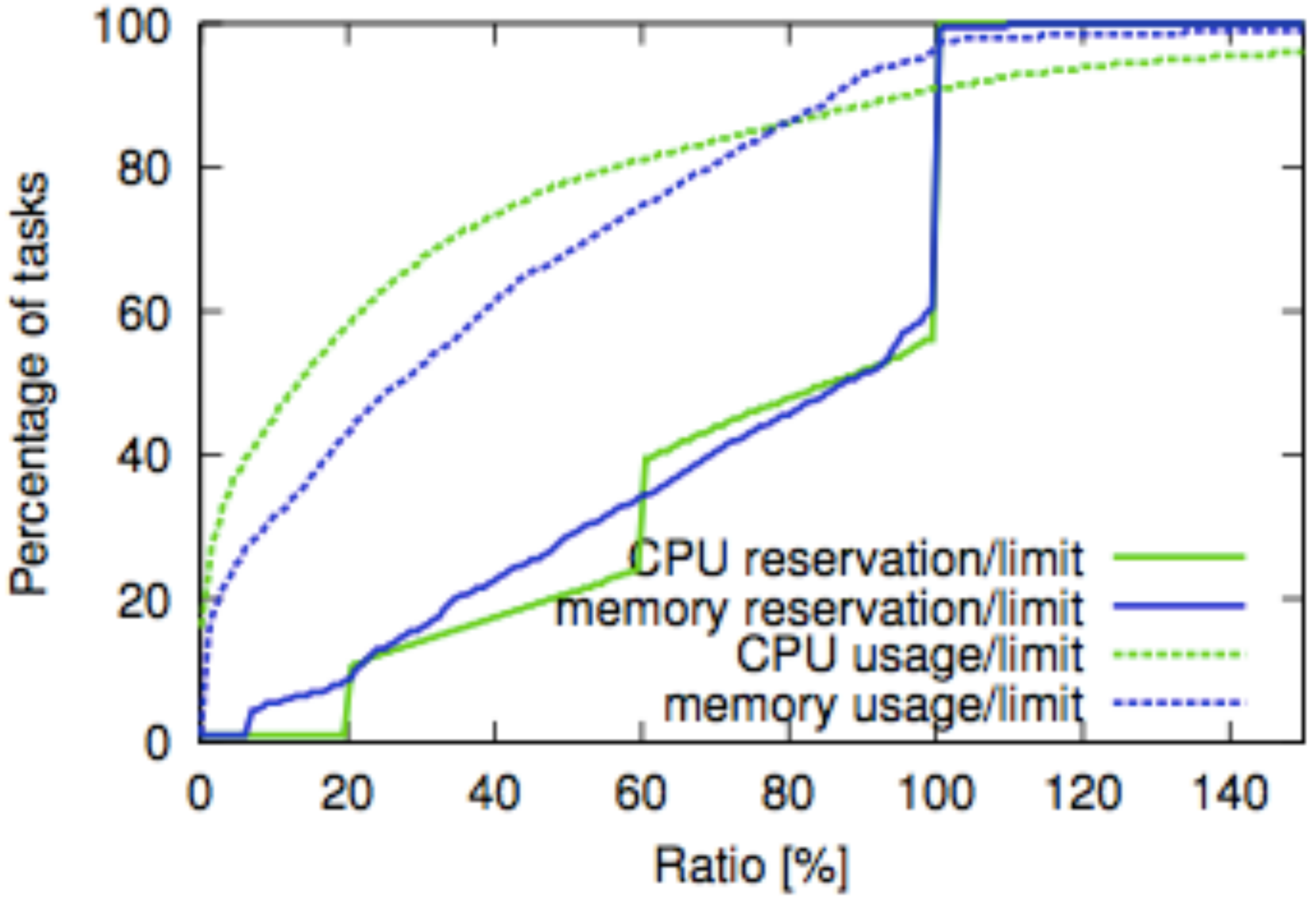


- Conclusion: more overhead if using bucketing resource requirements.

Resource reclamation

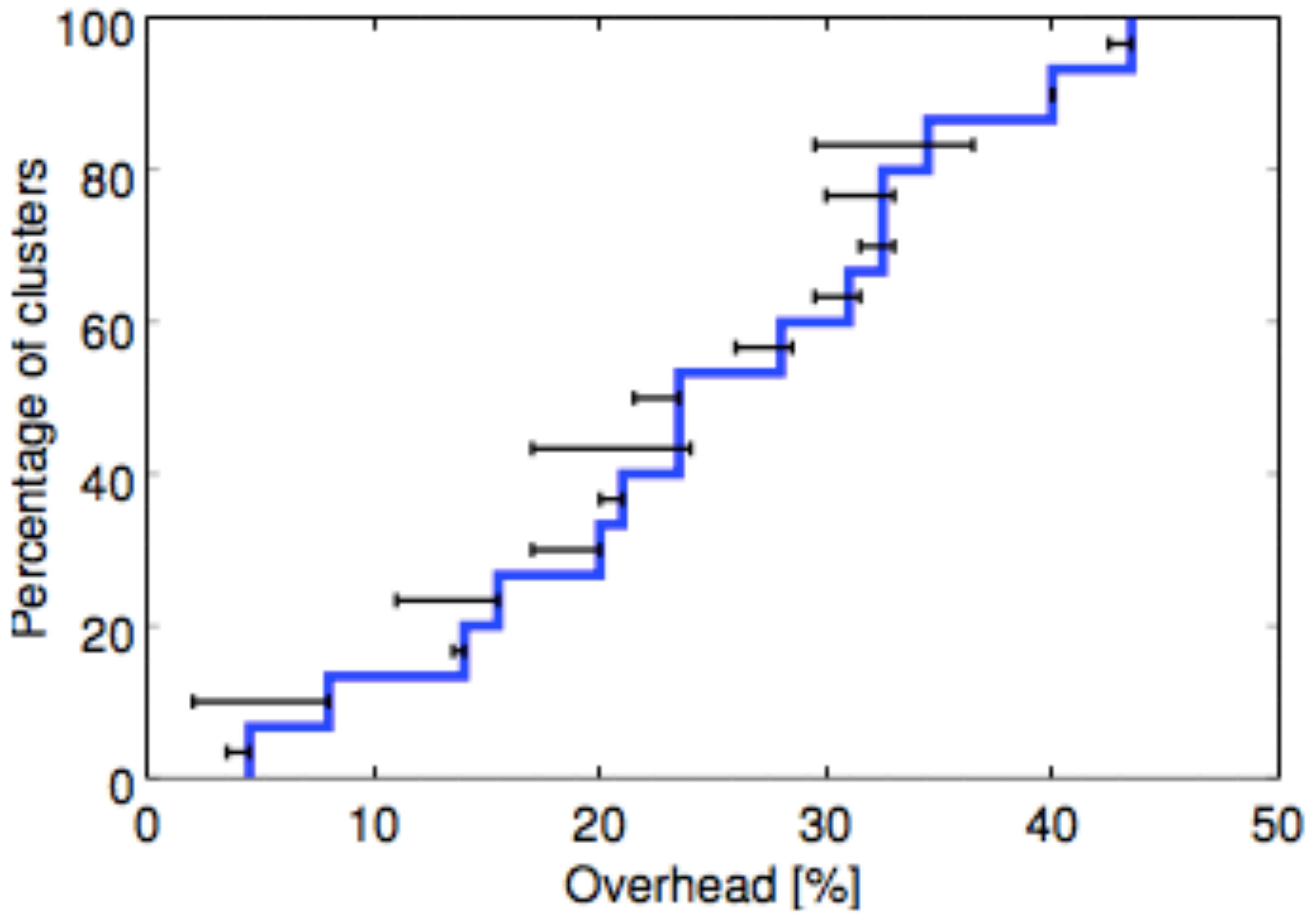
- Problem: Users allocate more resources than what they needs.
- Method:
  - Reservation: resource decays slowly every 300s
  - Reclamation of resources for work that can tolerate low-quality resources.

Resource reclamation



- Tasks claim more resources than what they needs.

Resource reclamation



- Conclusion: resource reclamation can save resources.