



# Cluster management at Google with Borg - coping with scale

2016-11

john wilkes / [johnwilkes@google.com](mailto:johnwilkes@google.com) Principal Software Engineer

presented by Filip Czaplicki

Derived from EuroSys'15 paper (<http://goo.gl/1C4nuo>)

CC-BY-NC-ND Creative Commons [license](#)







New one

"Old" one

tree

car

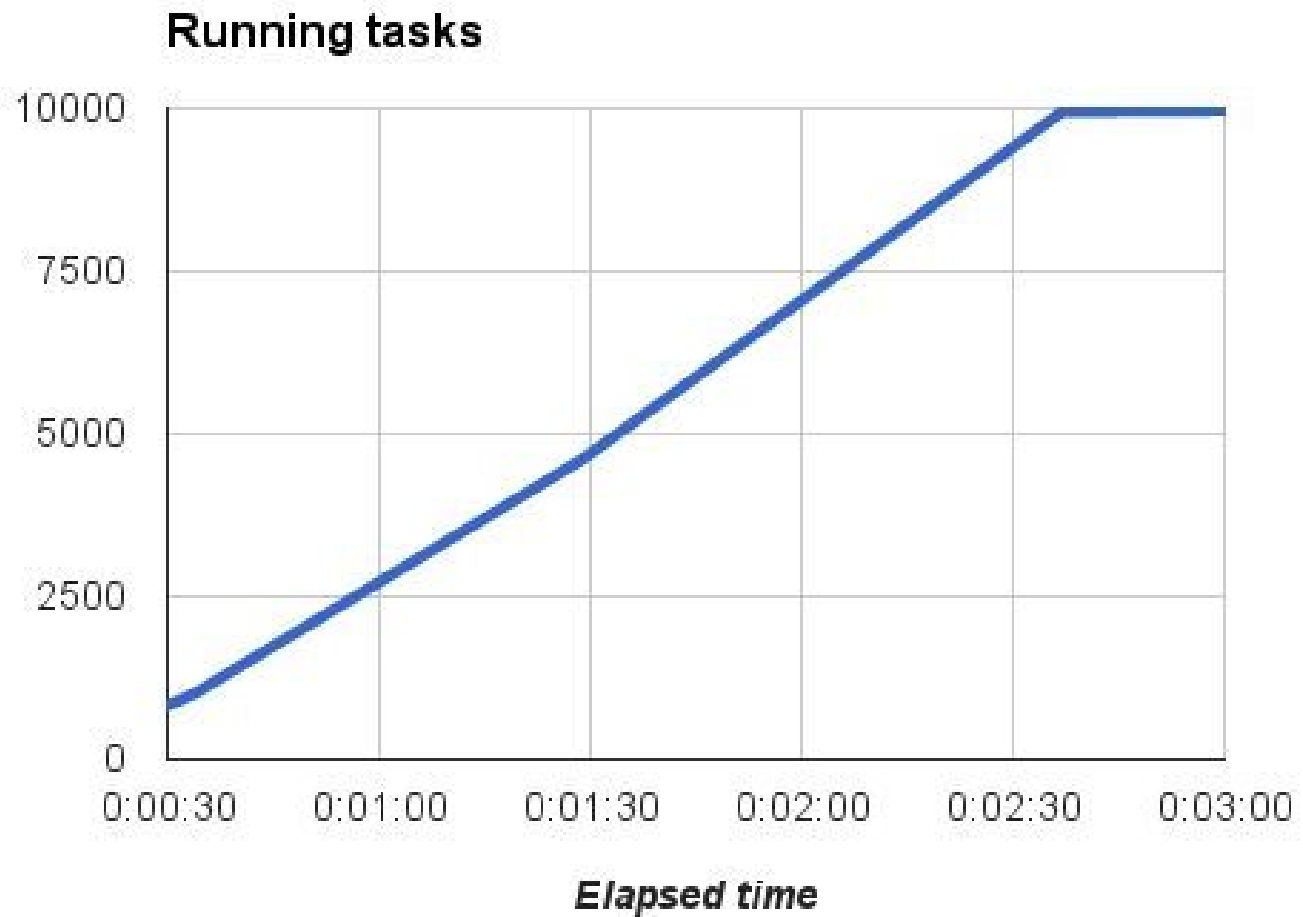
another tree



# User view

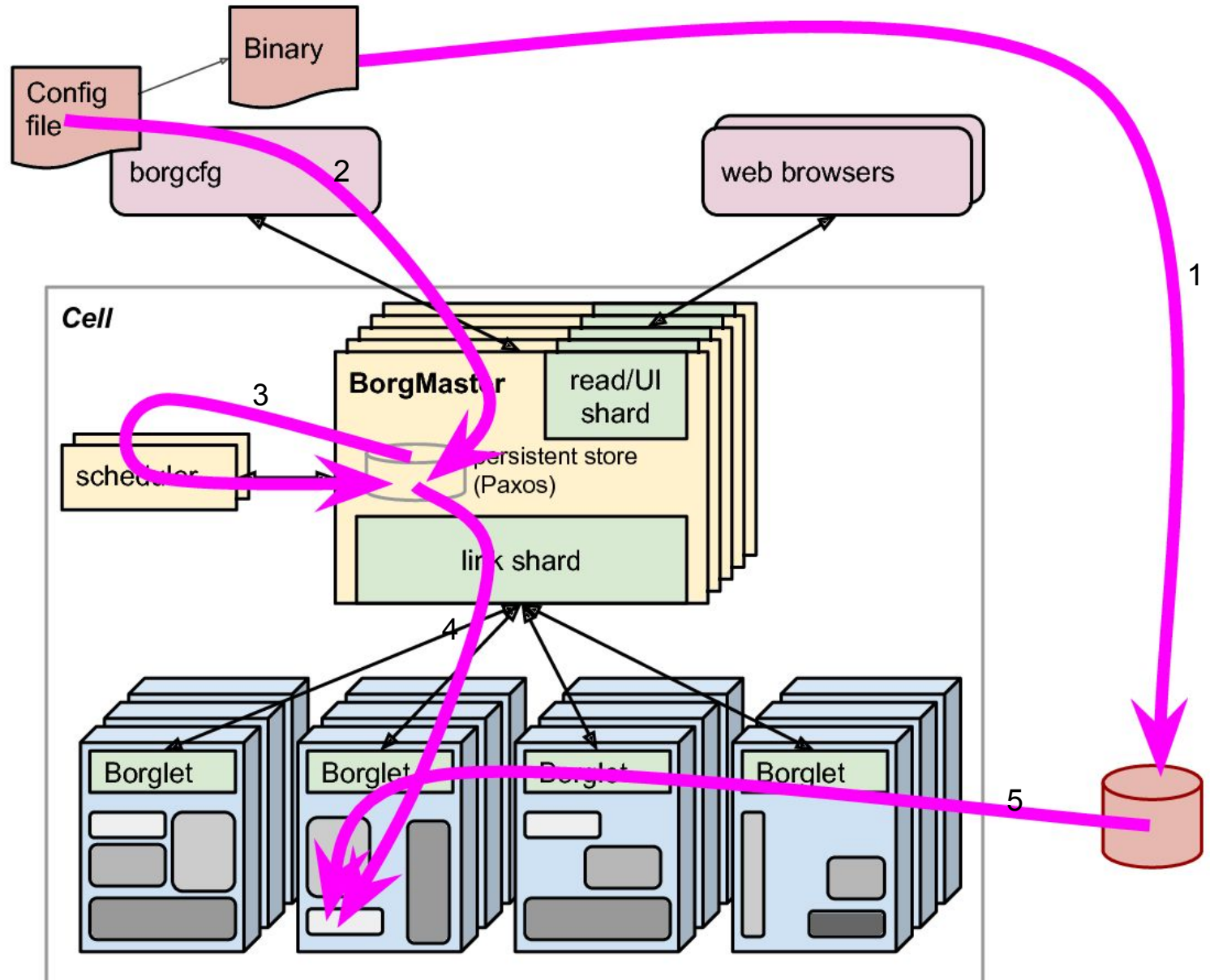
```
job hello_world = {  
  runtime = { cell = 'ic' }           // Cell (cluster) to run in  
  binary = '../hello_world_webserver' // Program to run  
  args = { port = '%port%' }         // Command line parameters  
  requirements = {                   // Resource requirements (optional)  
    ram = 100M  
    disk = 100M  
    cpu = 0.1  
  }  
  replicas = 10000 // Number of tasks  
}
```

# User view

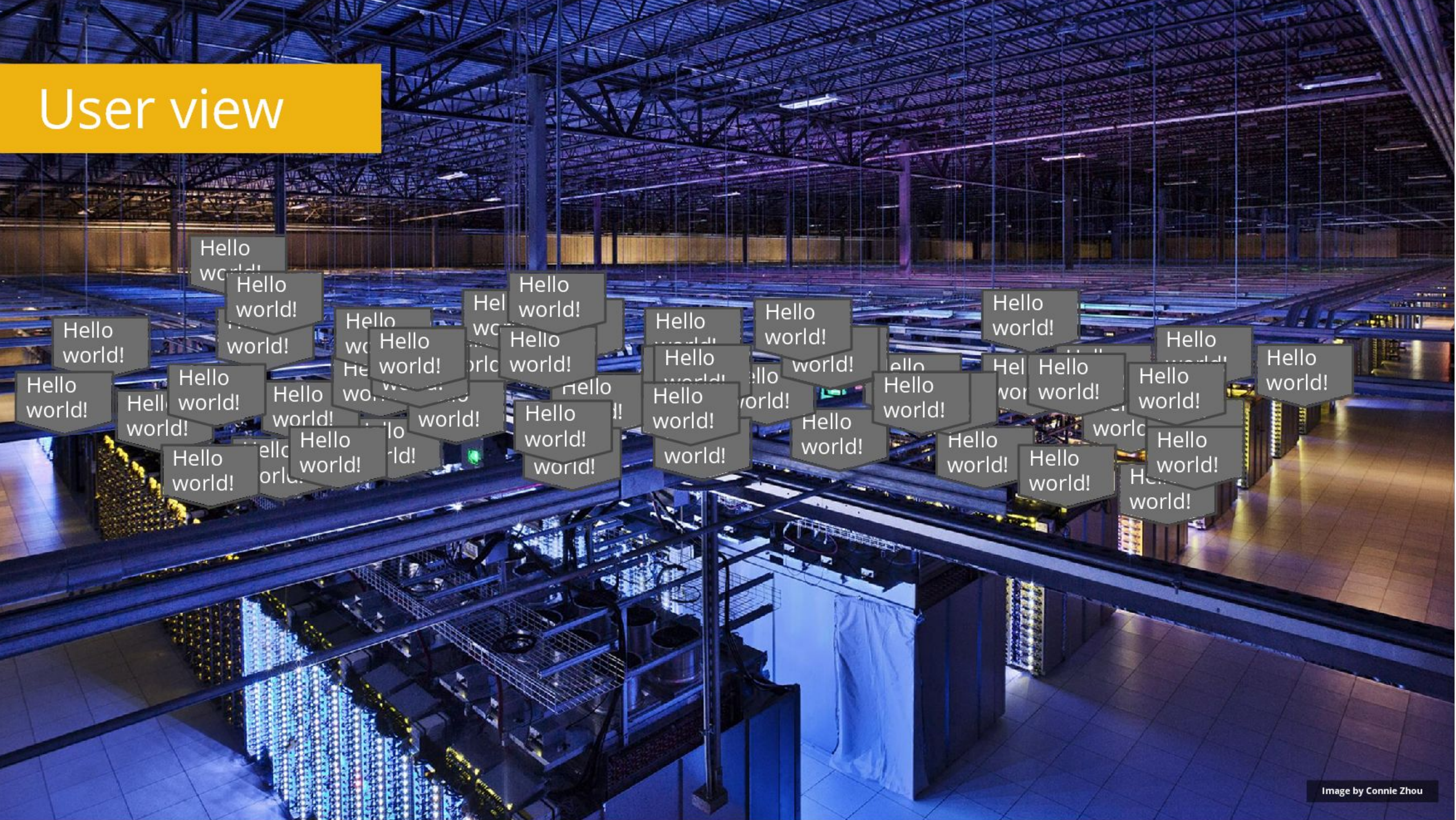


# User view

What just happened?



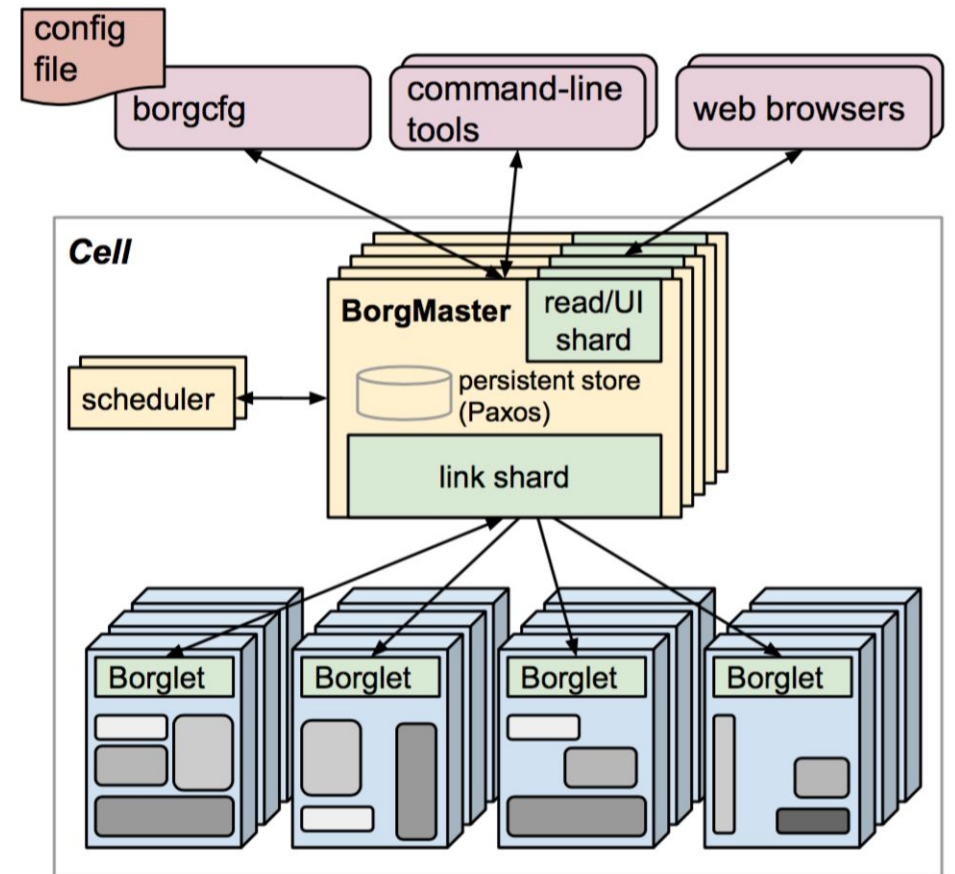
# User view





# Borg Architecture

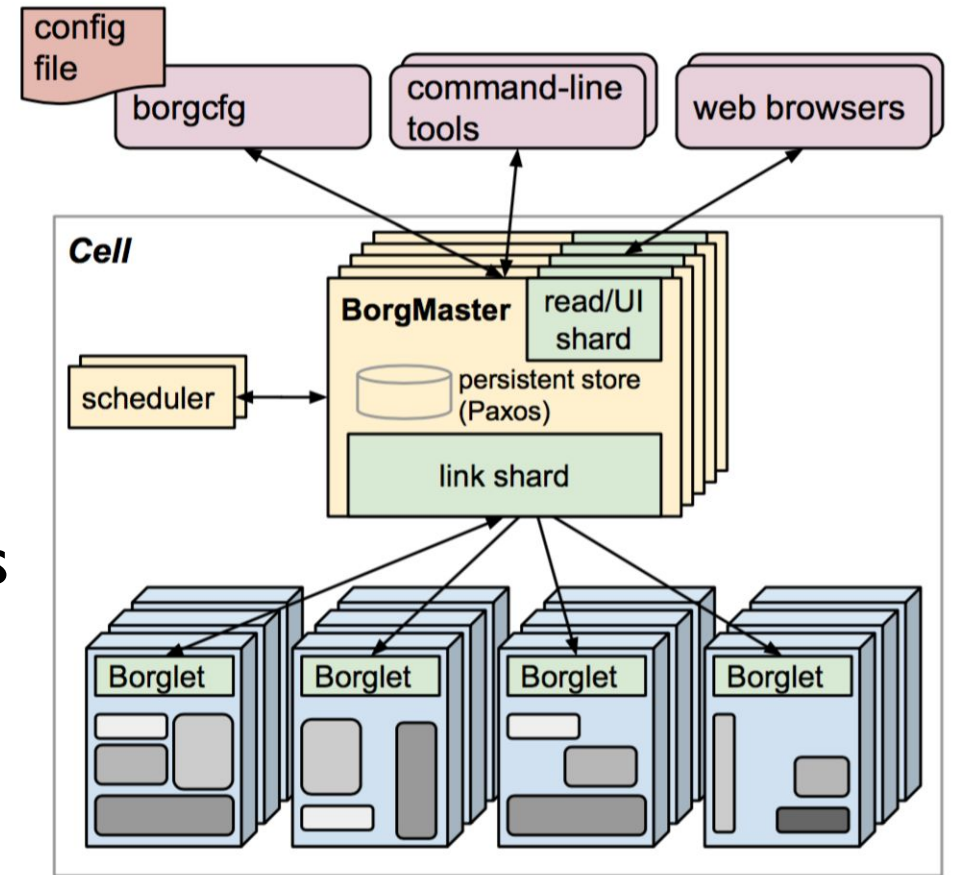
- **Borgmaster**
  - Main Borgmaster process & Scheduler
  - Five replicas
- **Borglet**
  - Manage and monitor tasks and resource
  - Borgmaster polls Borglet every few seconds



The high-level architecture of Borg(A Cell)

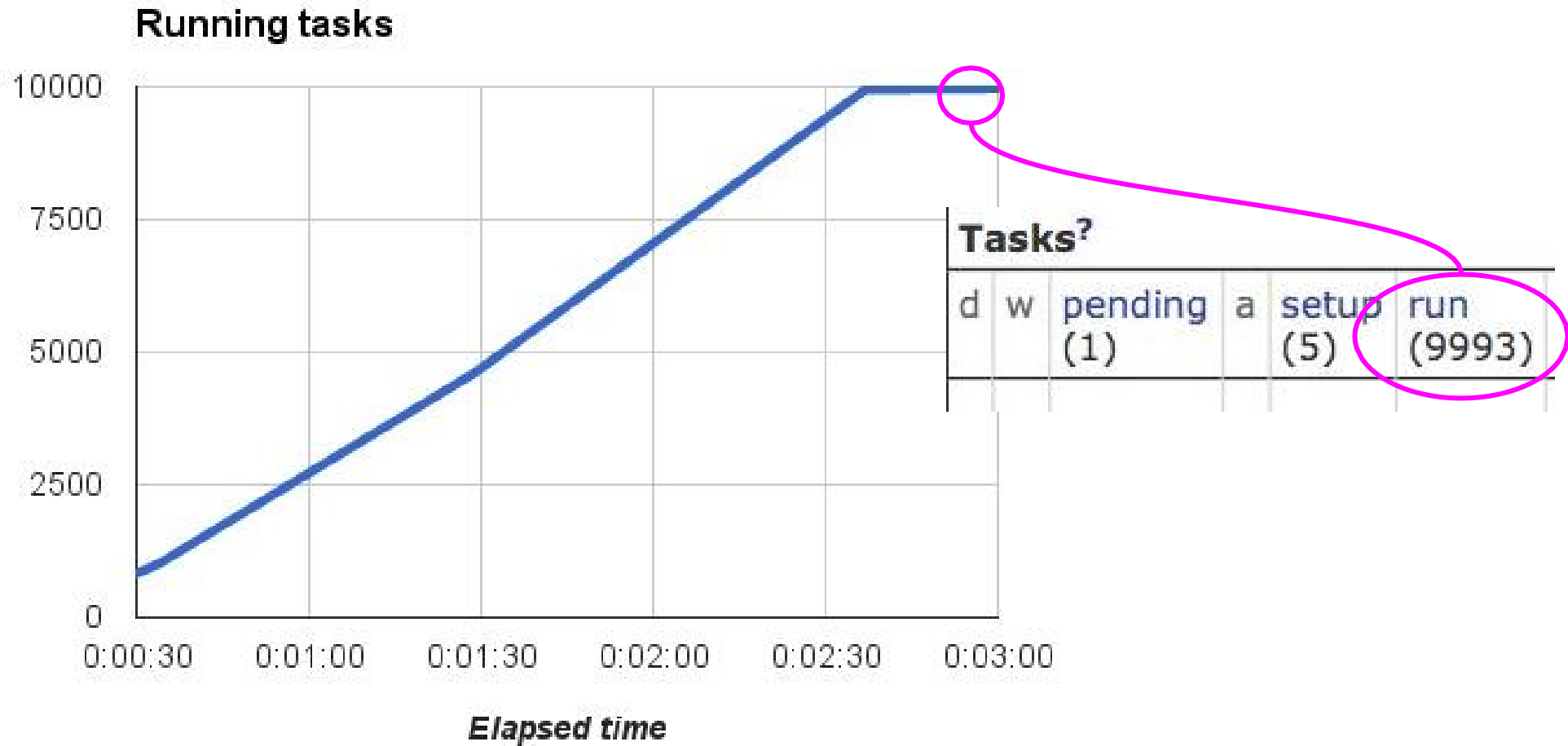
# Scalability

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

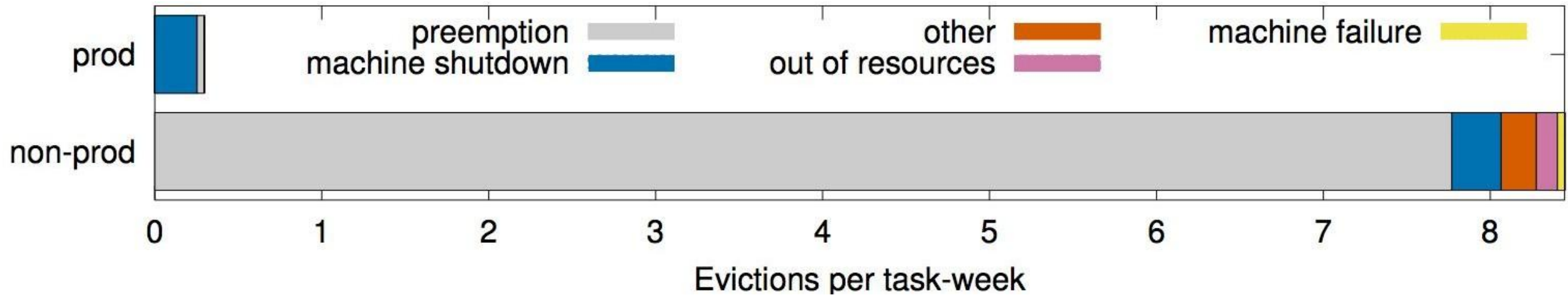


The high-level architecture of Borg(A Cell)

# User view



# Failures



task-eviction rates  
and causes

# Failures

A photograph of a server room aisle. The aisle is long and narrow, with rows of server racks on both sides. The racks are filled with server hardware, and a dense network of colorful cables (blue, green, orange, yellow) is visible, connecting the servers. The floor is light-colored and reflective. The ceiling has various pipes and conduits. The perspective is from the end of the aisle, looking down its length.

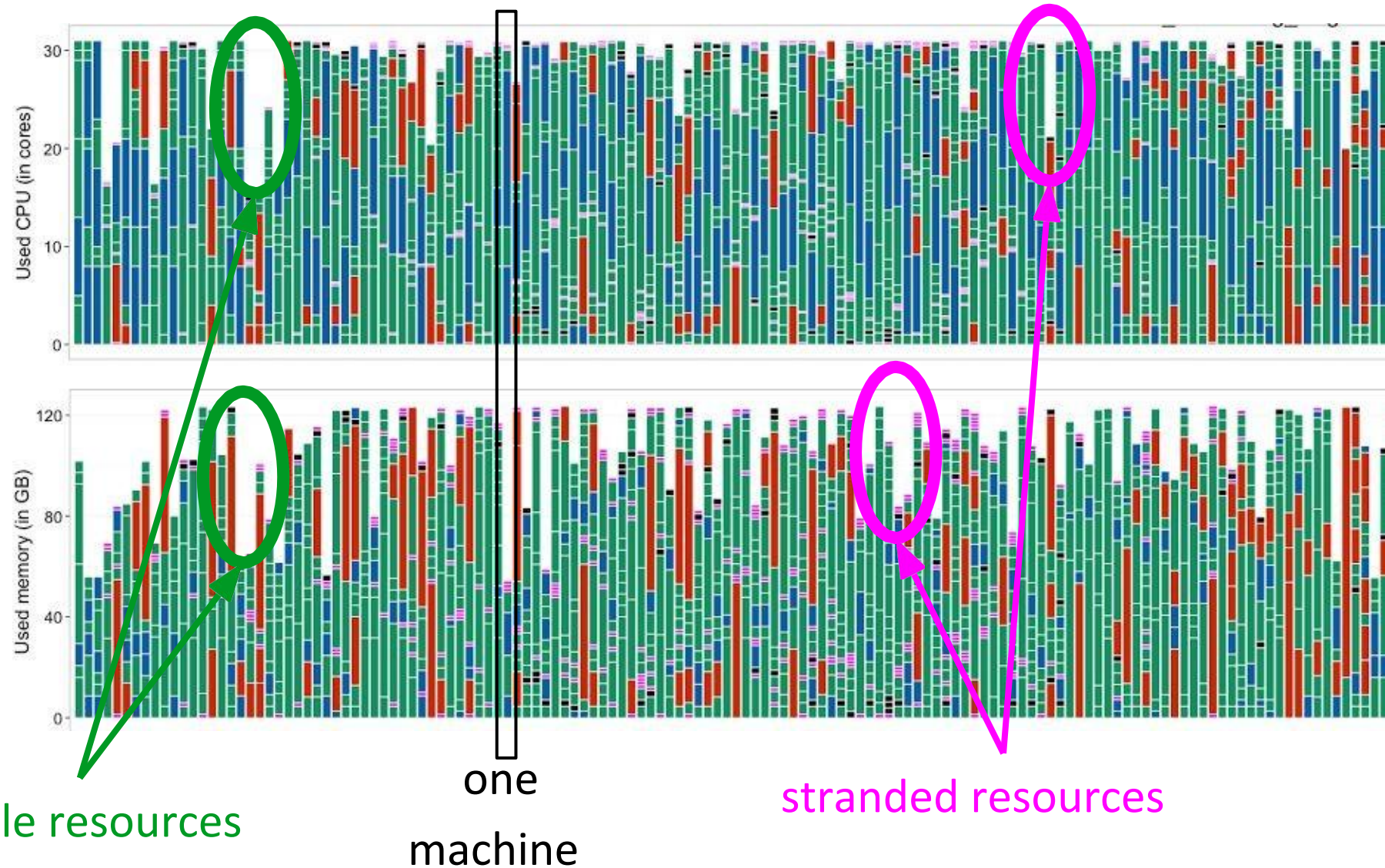
A 2000-machine service will have  $>10$  task exits per day

**This is not a problem: it's normal**

# Efficiency

Advanced  
bin-packing  
algorithms

Experimental placement of  
production VM workload,  
July 2014



available resources

one

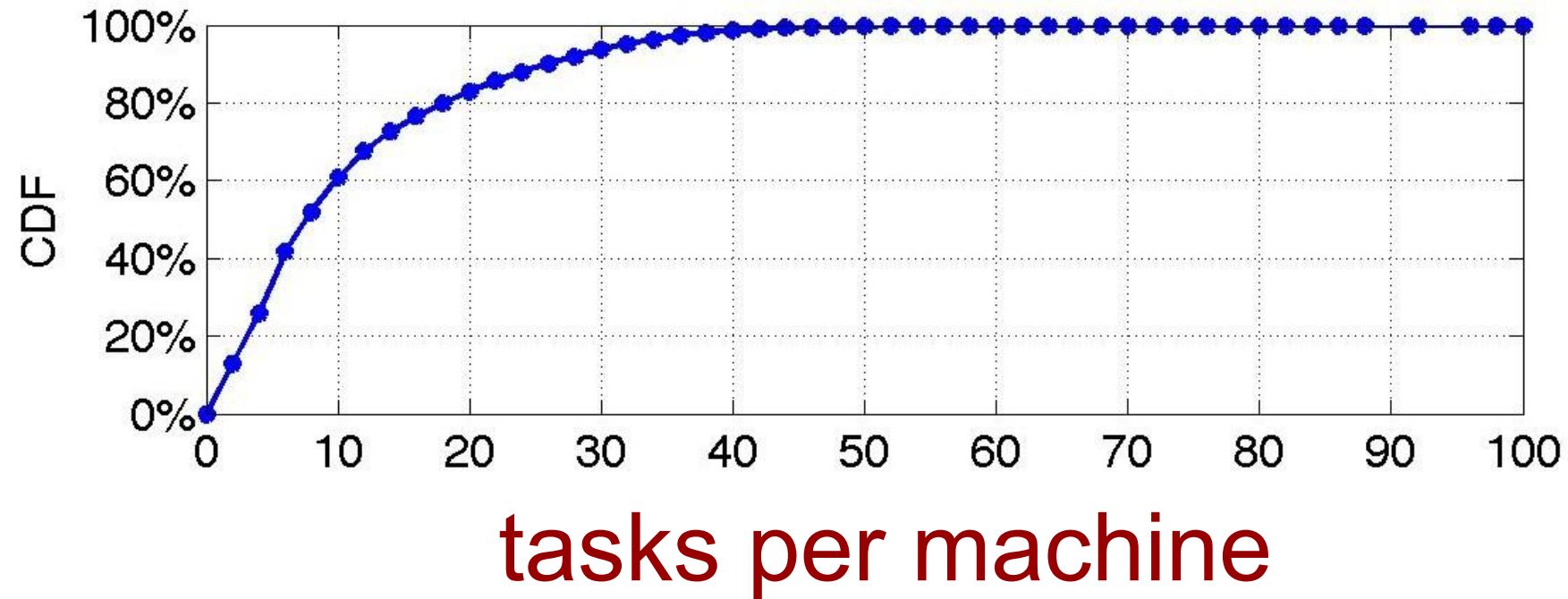
machine

stranded resources

# Efficiency

Multiple applications per machine

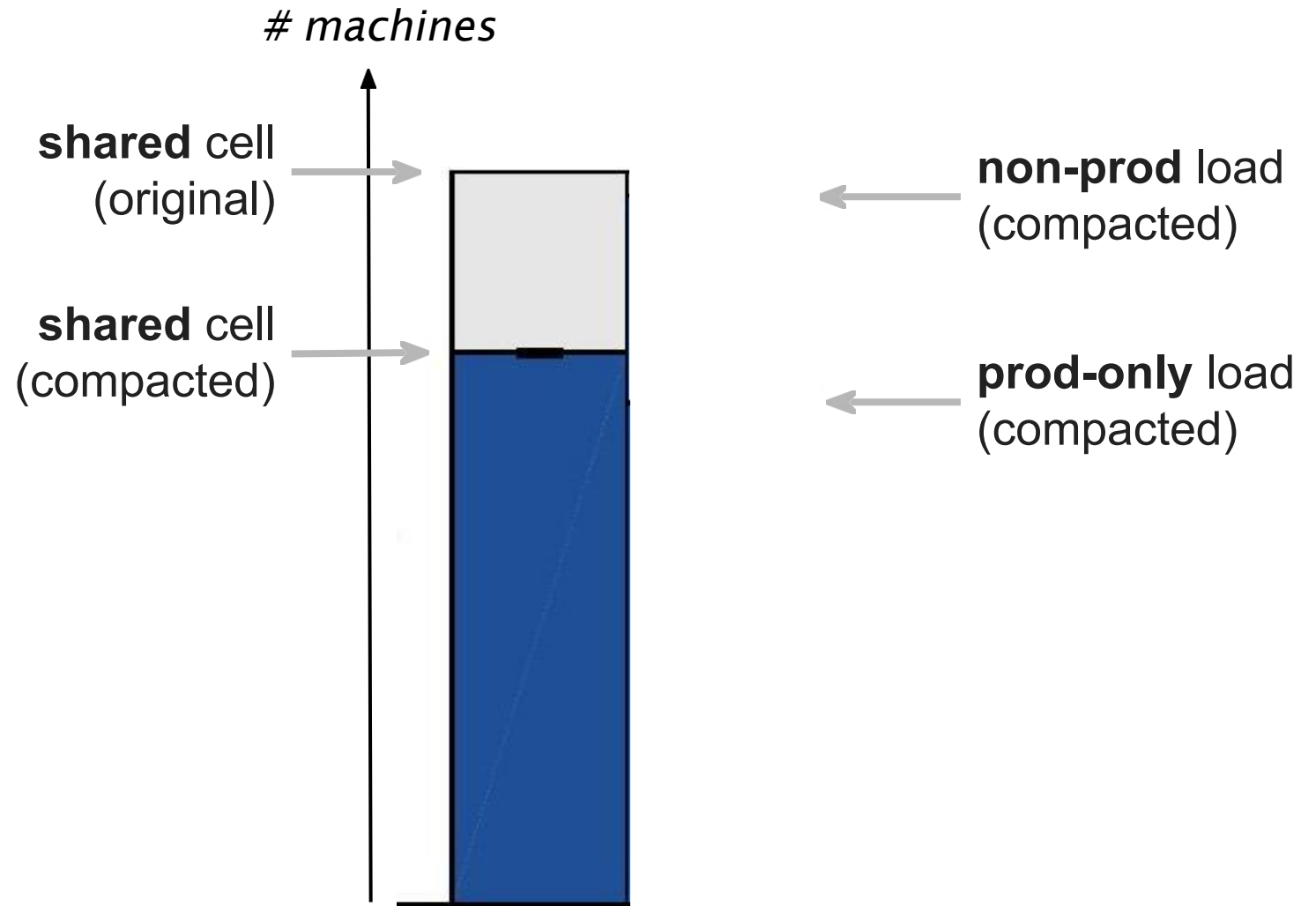
*CPI*<sup>2</sup> paper, EuroSys 2013



# Efficiency

Sharing clusters  
between  
prod/batch helps

Segregating them would need  
more machines

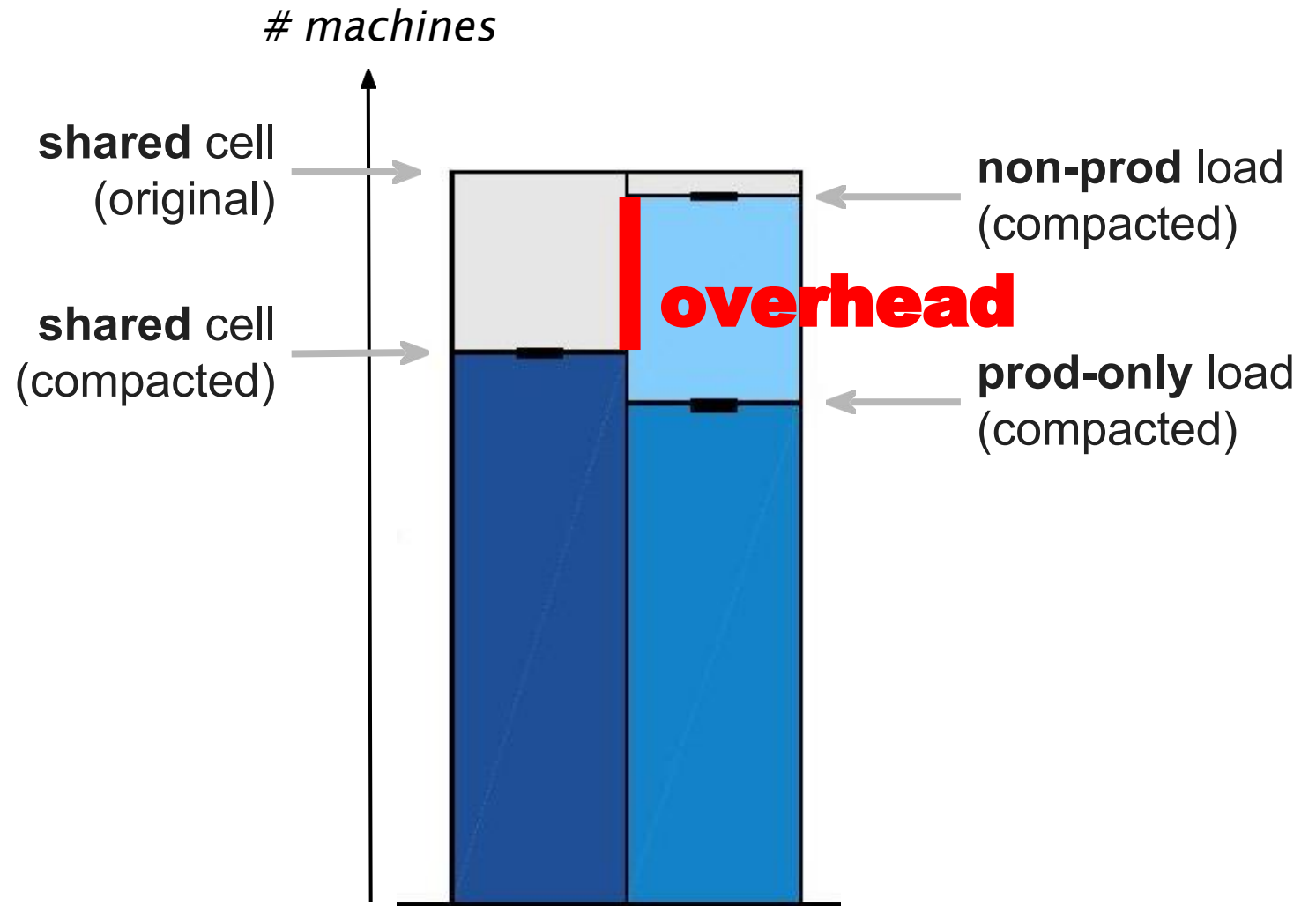




# Efficiency

Sharing clusters  
between  
prod/batch helps

Segregating them would need  
more machines

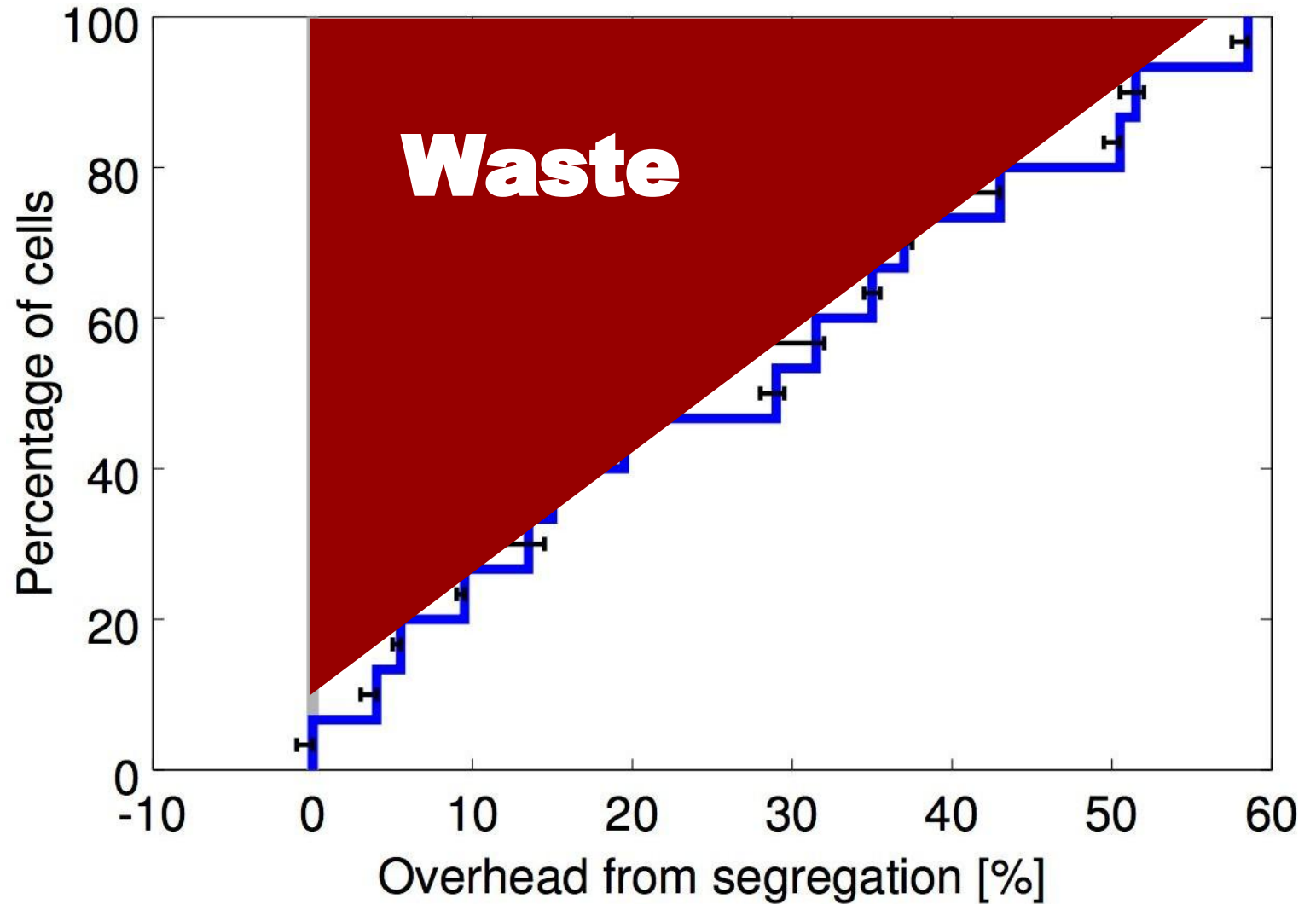


# Efficiency

Sharing clusters between prod/batch helps

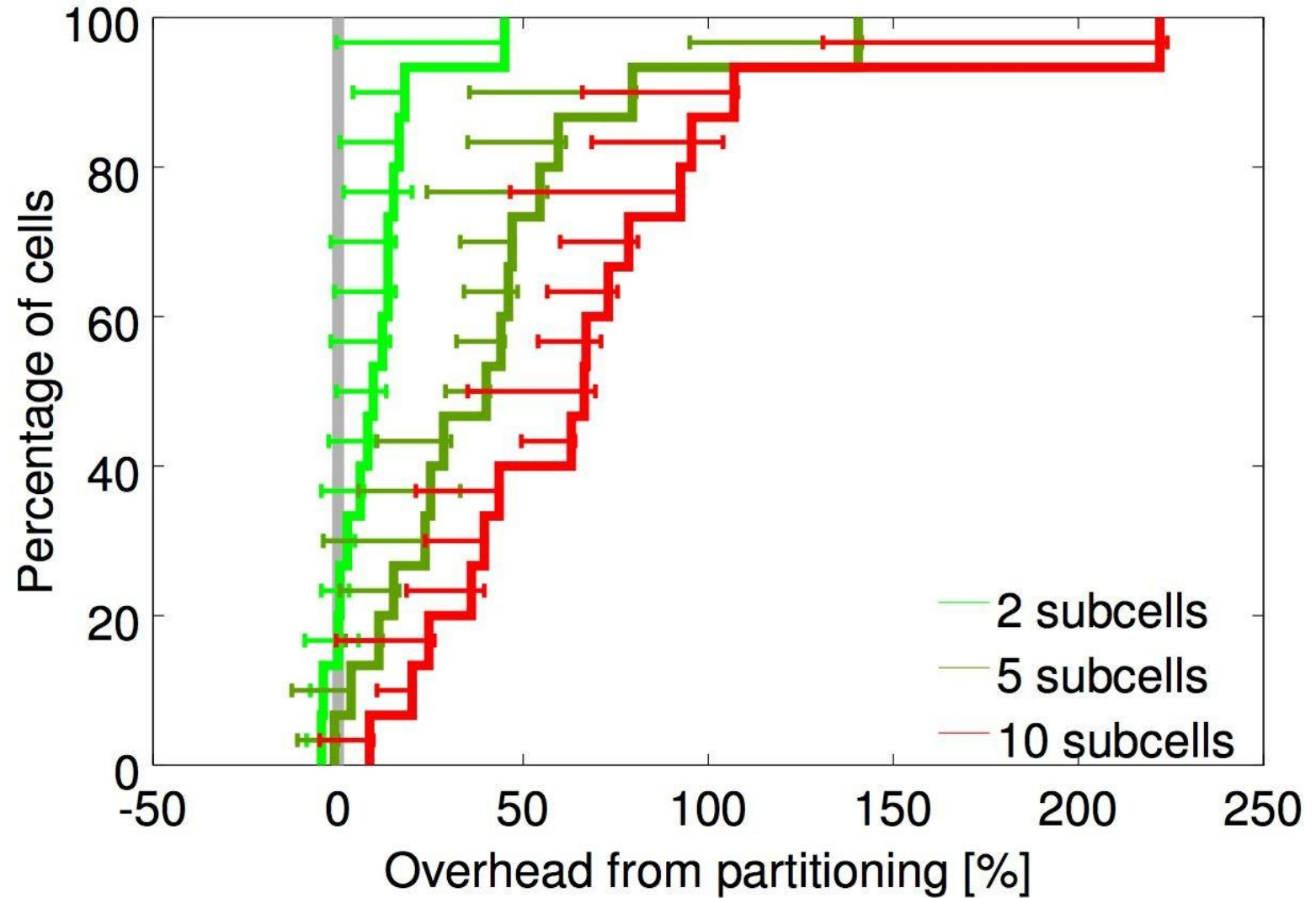
Segregating them would need more machines

15 production cells from a larger pool, omitting small ones (<5000 machines)



# Efficiency

Smaller cells would need more machines

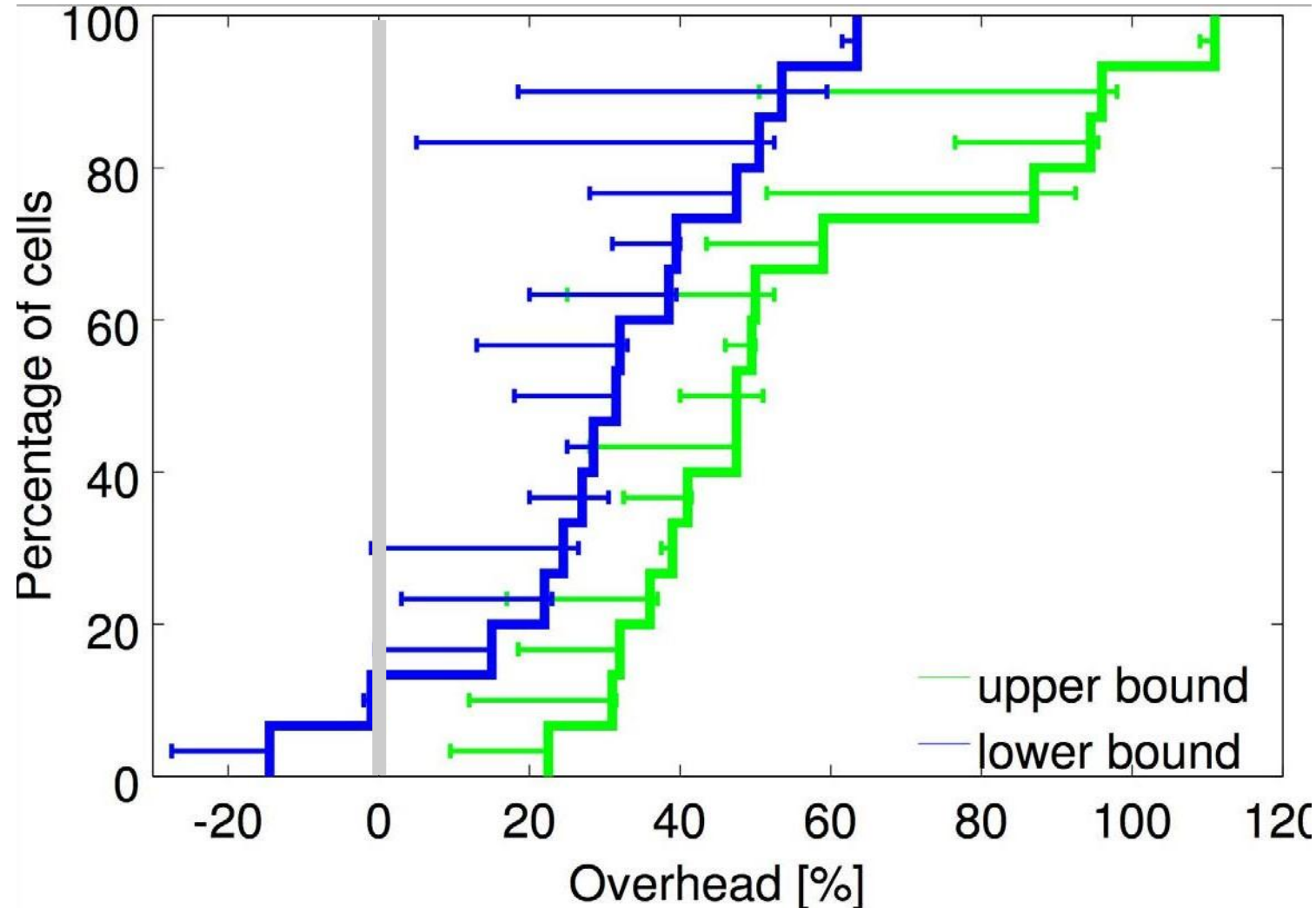


# Efficiency

Bucketing to next-largest power of 2 would need more machines

prod only,  
starting from 0.5 cores, 0.5GiB

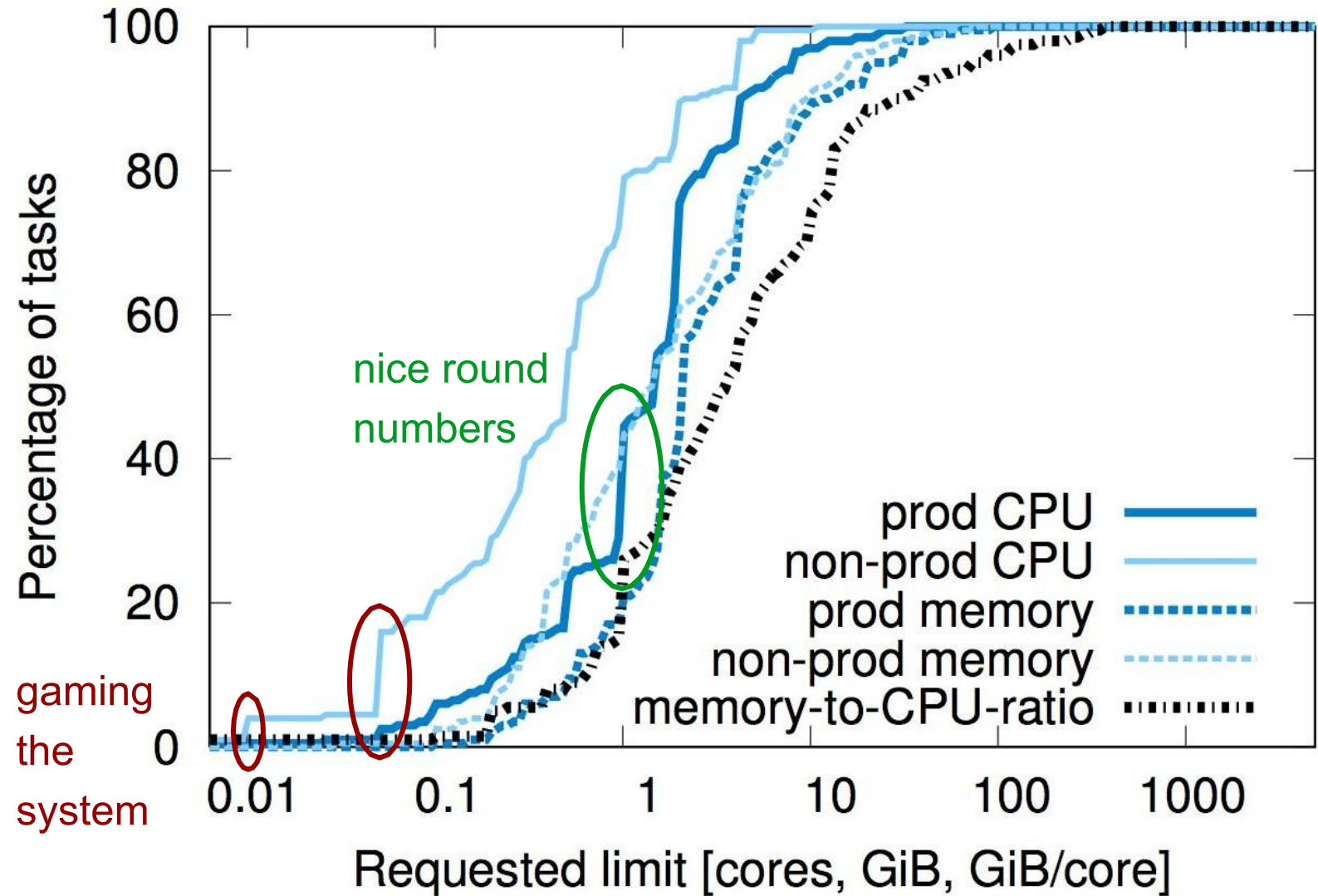
⇒ GCE *Custom machine types*



# Efficiency

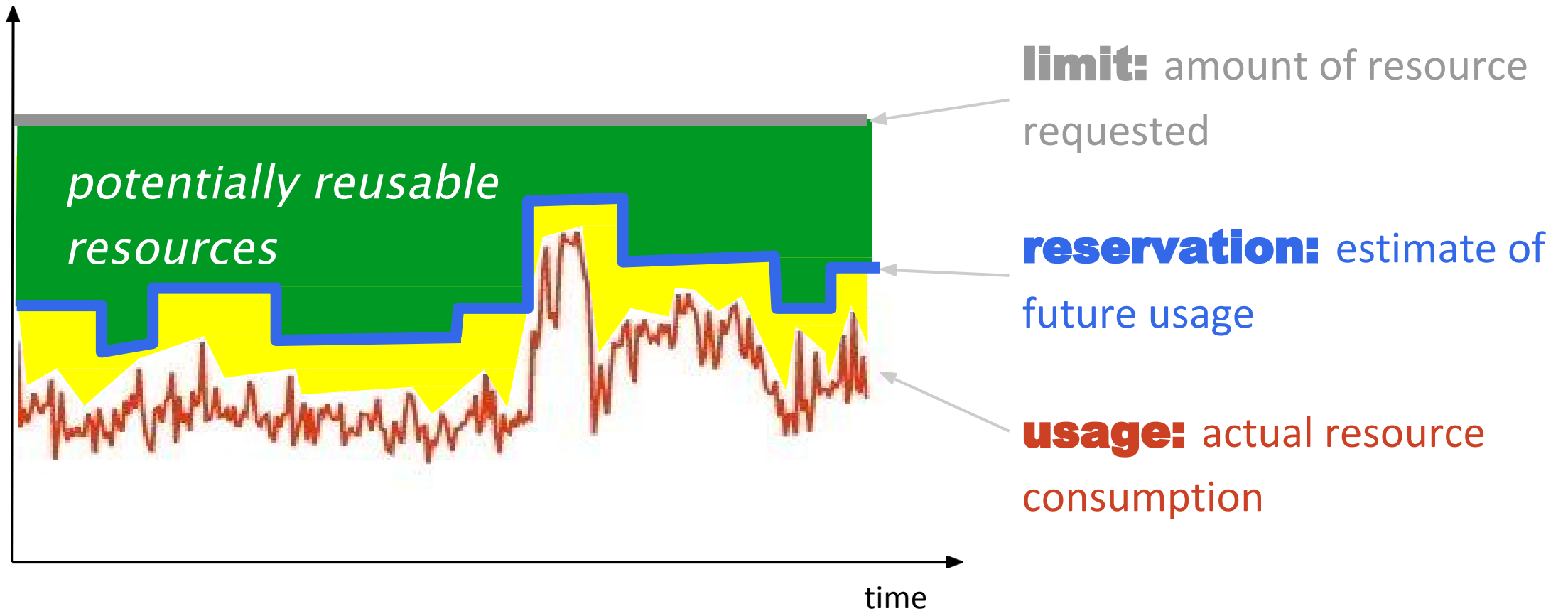
There are no  
“obvious”  
resource-bucket  
sizes

*cf.* cloud VMs



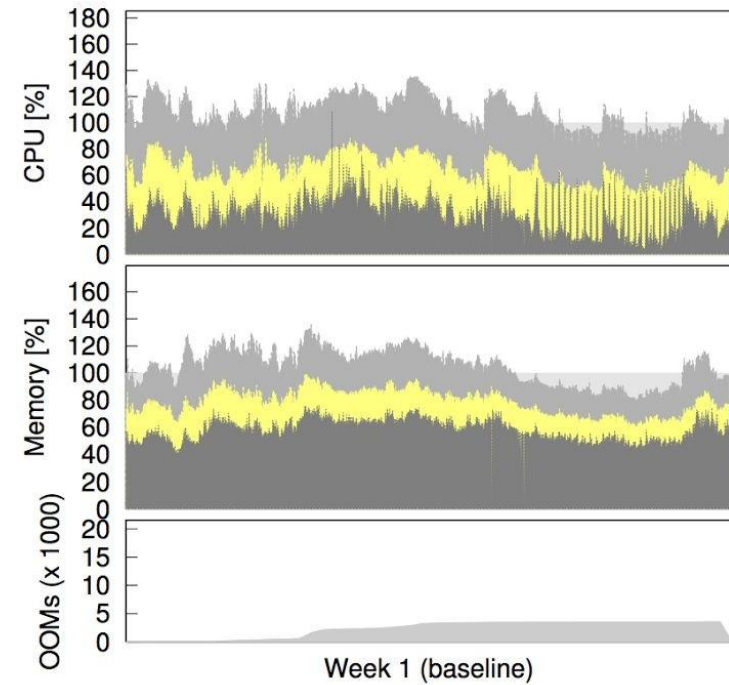
# Efficiency

## Resource reclamation



# Efficiency

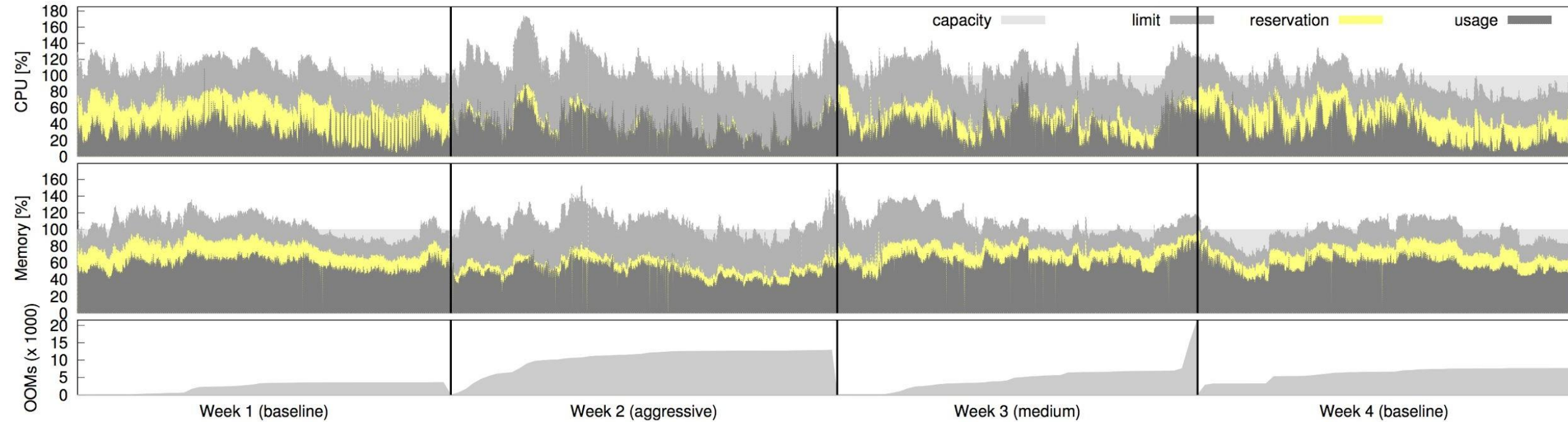
Resource reclamation could be more aggressive



Nov/Dec 2013

# Efficiency

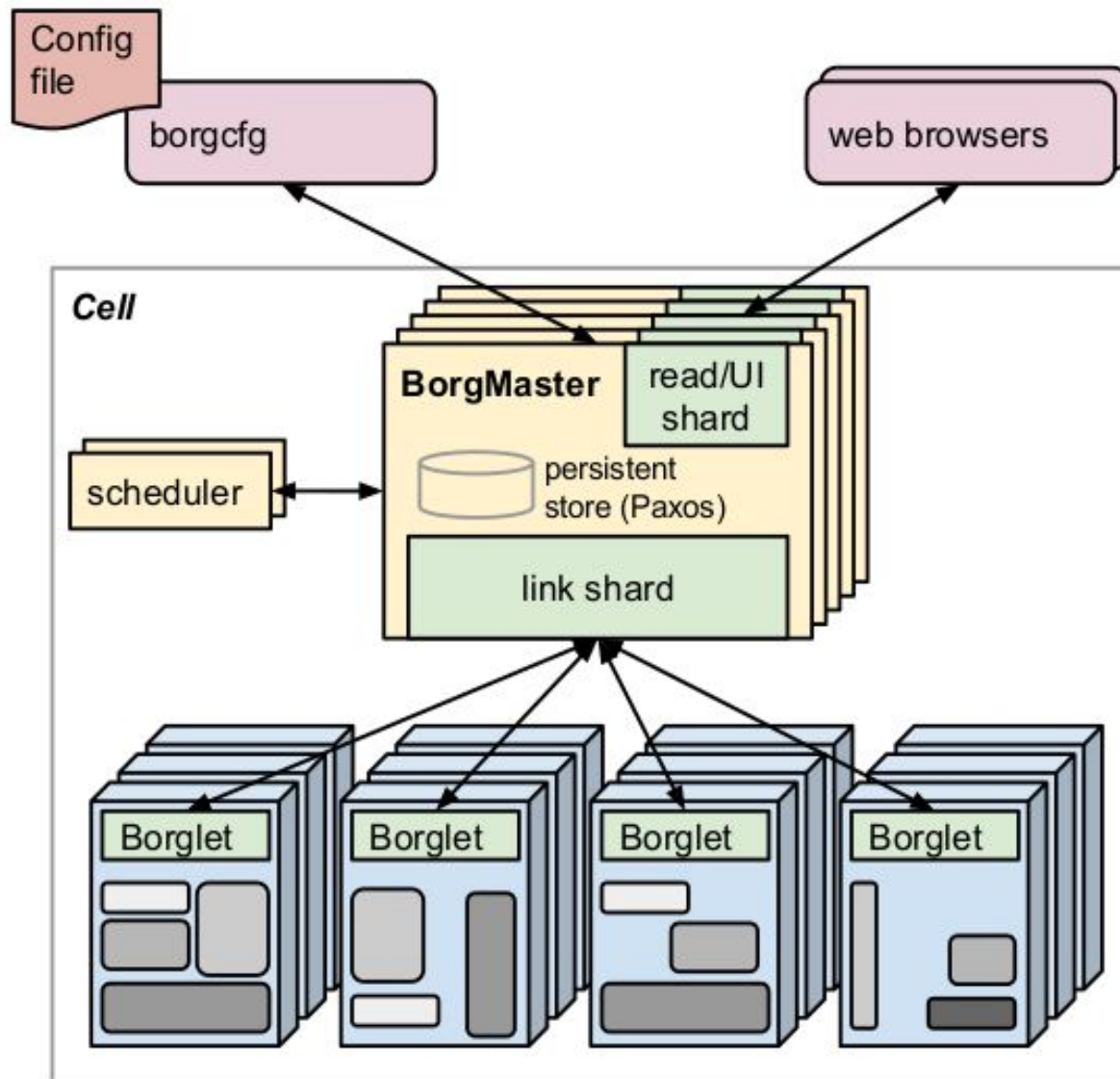
Resource reclamation could be more aggressive



Nov/Dec 2013



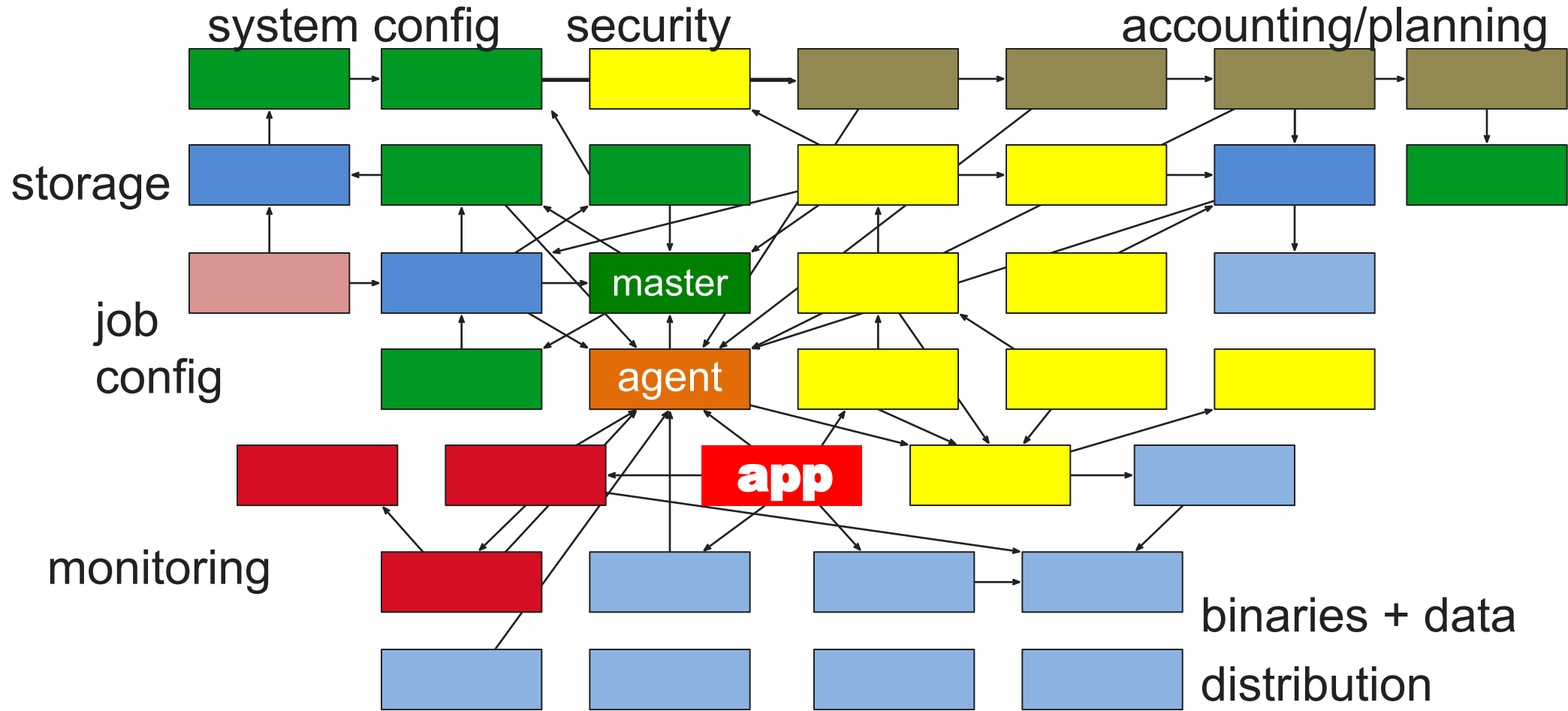
# A few other moving parts



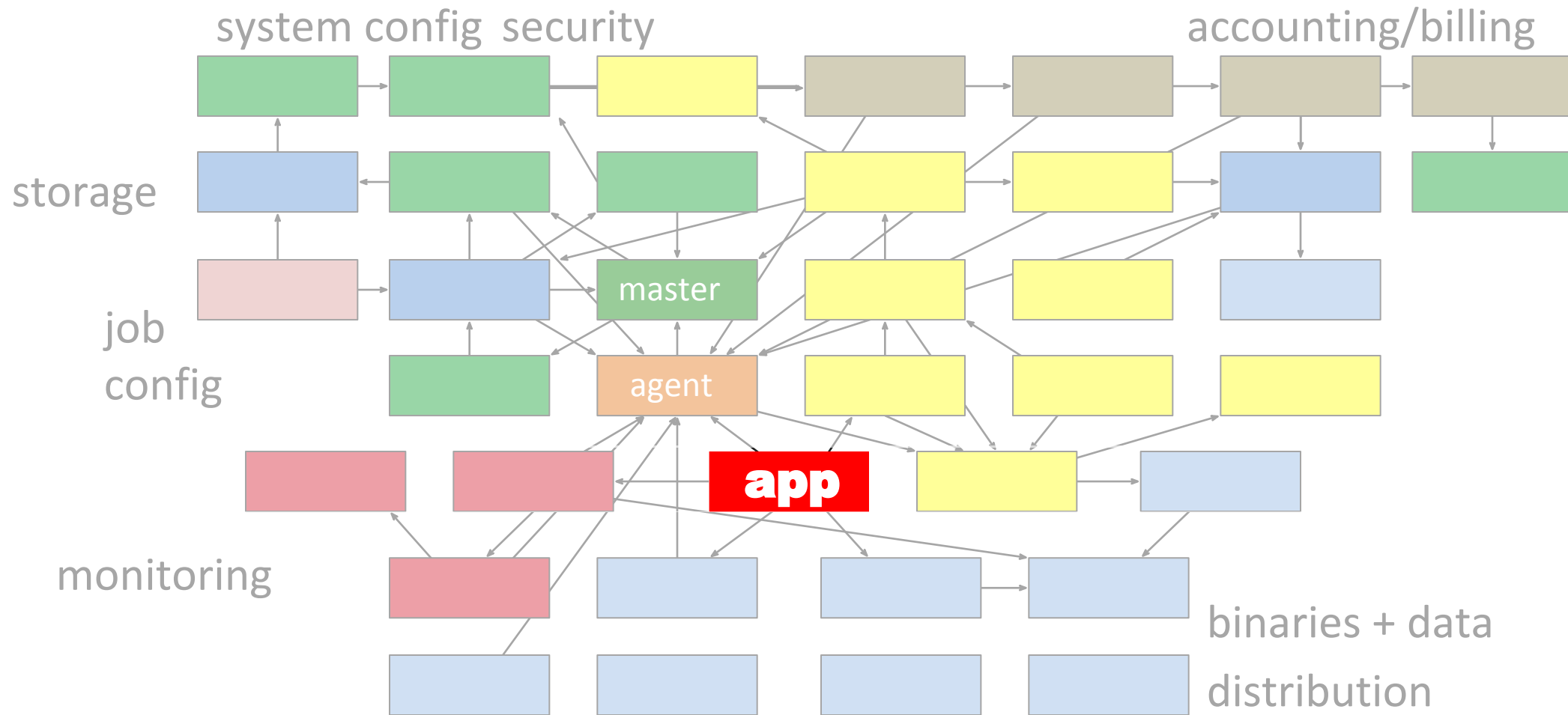
# A few other moving parts



# A few other moving parts



# A few other moving parts



## Observations:

1. **Resiliency** is achieved only by ruthless attention to detail
  - a. ubiquitous software fault tolerance
  - b. persistent, declarative specs
2. We get **efficiency** by:
  - a. sharing resources
  - b. reclaiming unused allocations
3. **Containers** make users more productive

[johnwilkes@google.com](mailto:johnwilkes@google.com)  
<http://goo.gl/1C4nuo> (Borg paper)

# Lessons

- **Bad**
  - 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
- **Good**
  - Allocs are useful
  - Cluster management is more than task management
  - Introspection is vital
  - The master is the kernel of a distributed system

Q & A

Thank you!  
Time for Q & A