



Cluster management at Google with Borg - coping with scale

2015-09 HPTS

john wilkes / johnwilkes@google.com
Principal Software Engineer

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



Google Cloud Platform

Cluster management



at Google with ^{the system we internally call} Borg -
coping with scale

2015-09 HPTS

john wilkes / johnwilkes@google.com
Principal Software Engineer

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

Borg contributors

Core: Abhishek Rai, Abhishek Verma, Andy Zheng, Ashwin Kumar, Ben Smith, Beng-Hong Lim, Bin Zhang, Bolu Szewczyk, Brad Strand, Brian Budge, Brian Grant, Brian Wickman, Chengdu Huang, Chris Colohan, Cliff Stein, Cynthia Wong, Daniel Smith, Dave Bort, David Oppenheimer, David Wall, Divyesh Shah, Dawn Chen, Eric Haugen, Eric Tune, Eric Wilcox, Ethan Solomita, Gaurav Dhiman, Geeta Chaudhry, Greg Roelofs, Grzegorz Czajkowski, James Eady, Jarek Kusmieriek, Jaroslaw Przybylowicz, Jason Hickey, Javier Kohen, Jeff Dean, Jeremy Dion, Jeremy Lau, Jerzy Szczepkowski, Joe Hellerstein, John Wilkes, Jonathan Wilson, Joso Eterovic, Jutta Degener, Kai Backman, Kamil Yurtsever, Ken Ashcraft, Kenji Kaneda, Kevan Miller, Kurt Steinkraus, Leo Landa, Liza Fireman, Madhukar Korupolu, Maricia Scott, Mark Logan, Mark Vandevoorde, Markus Gutschke, Matt Sparks, Maya Haridasan, Michael Abd-El-Malek, Michael Kenniston, Ming-Yee Iu, Monika Henzinger, Mukesh Kumar, Nate Calvin, Onufry Wojtaszczyk, Olcan Sercinoglu, Paul Menage, Patrick Johnson, Pavanish Nirula, Pedro Valenzuela, Percy Liang, Piotr Witusowski, Praveen Kallakuri, Rafal Sokolowski, Rajmohan Rajaraman, Richard Gooch, Rishi Gosalia, Rob Radez, Robert Hagmann, Robert Jardine, Robert Kennedy, Rohit Jnagal, Roy Bryant, Rune Dahl, Scott Garriss, Scott Johnson, Sean Howarth, Sheena Madan, Smeeta Jalan, Stan Chesnutt, Temo Arobelidze, Tim Hockin, Todd Wang, Tomasz Blaszczyk, Tomasz Wozniak, Tomek Zielonka, Victor Marmol, Vish Kannan, Vrigo Gokhale, Walfredo Cirne, Walt Drummond, Weiran Liu, Xiaopan Zhang, Xiao Zhang, Ye Zhao, and Zohaib Maya.

SRE: Adam Rogoyski, Alex Milivojevic, Anil Das, Cody Smith, Cooper Bethea, Folke Behrens, Matt Liggett, James Sanford, John Millikin, Matt Brown, Miki Habryn, Peter Dahl, Robert van Gent, Seppi Wilhelmi, Seth Hettich, Torsten Marek, and Viraj Alankar.

BCL and borgcfg: Marcel van Lohuizen and Robert Griesemer.

Reviewers: Christos Kozyrakis, Eric Brewer, Malte Schwarzkopf, and Tom Rodeheffer.

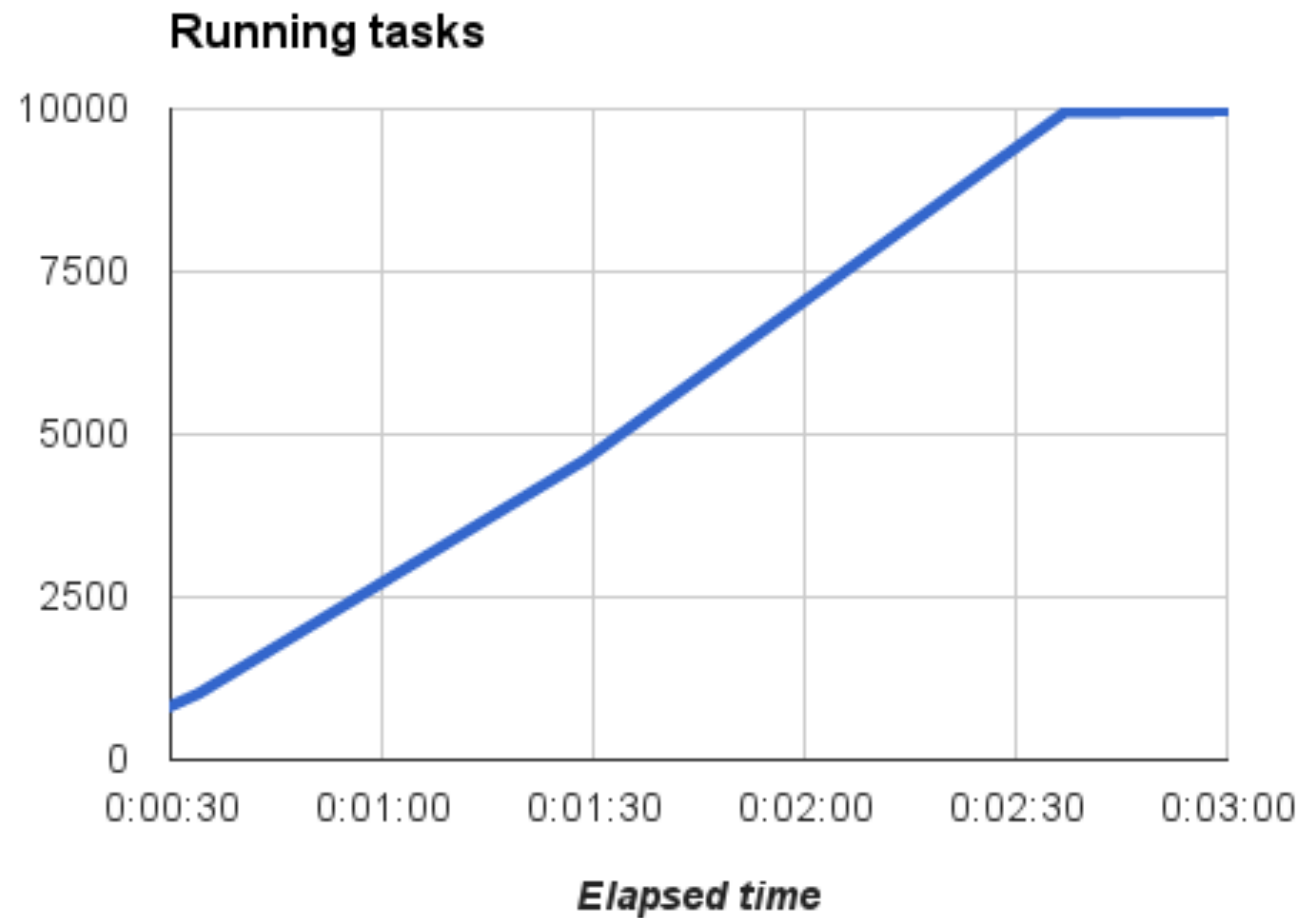




User view

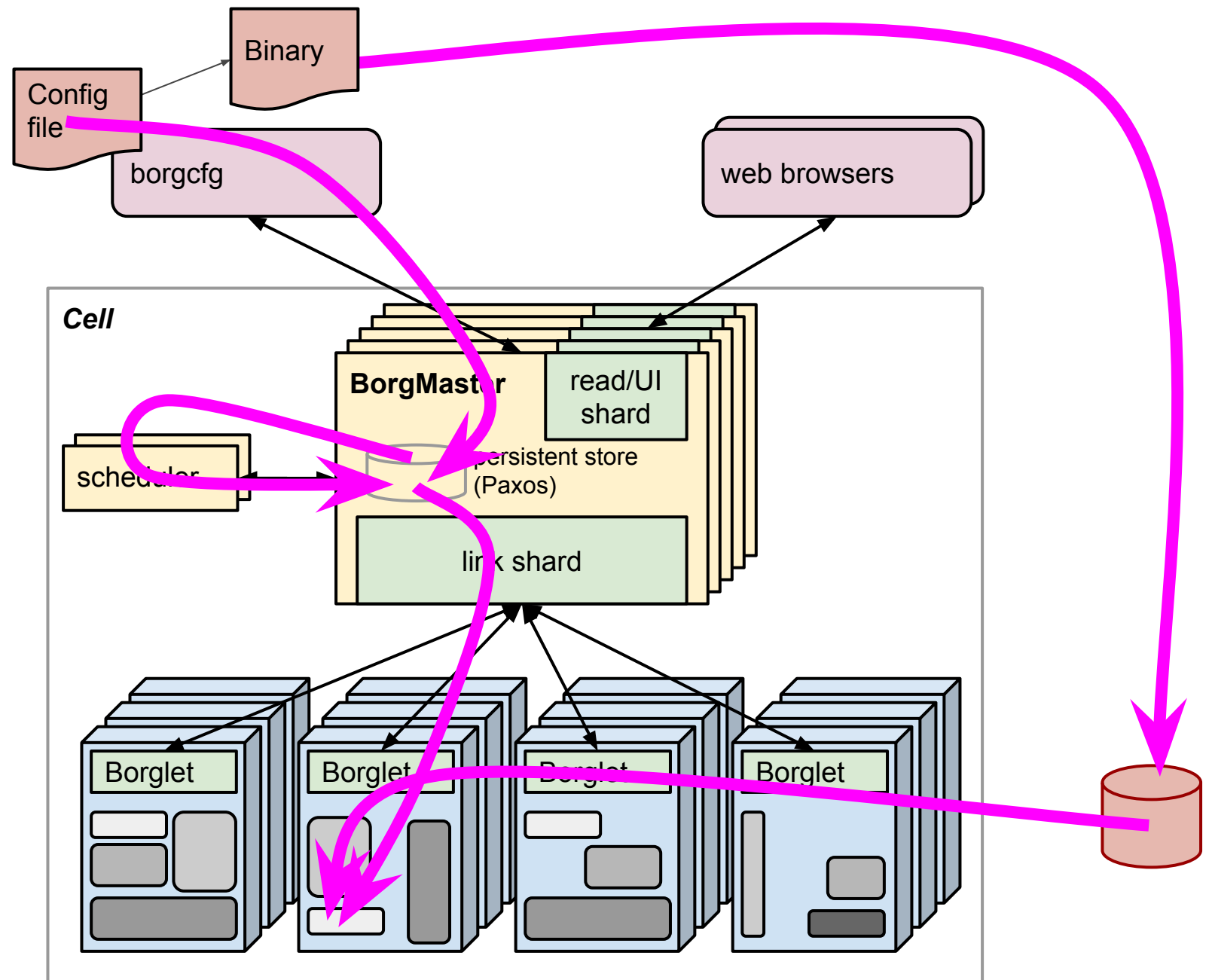
```
job hello_world = {  
    runtime = { cell = 'ic' }           // Cell (cluster) to run in  
    binary = '../hello_world_webserver' // Program to run  
    args = { port = '%port%' }         // Command line parameters  
    requirements = {                   // Resource requirements (optional)  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

User view

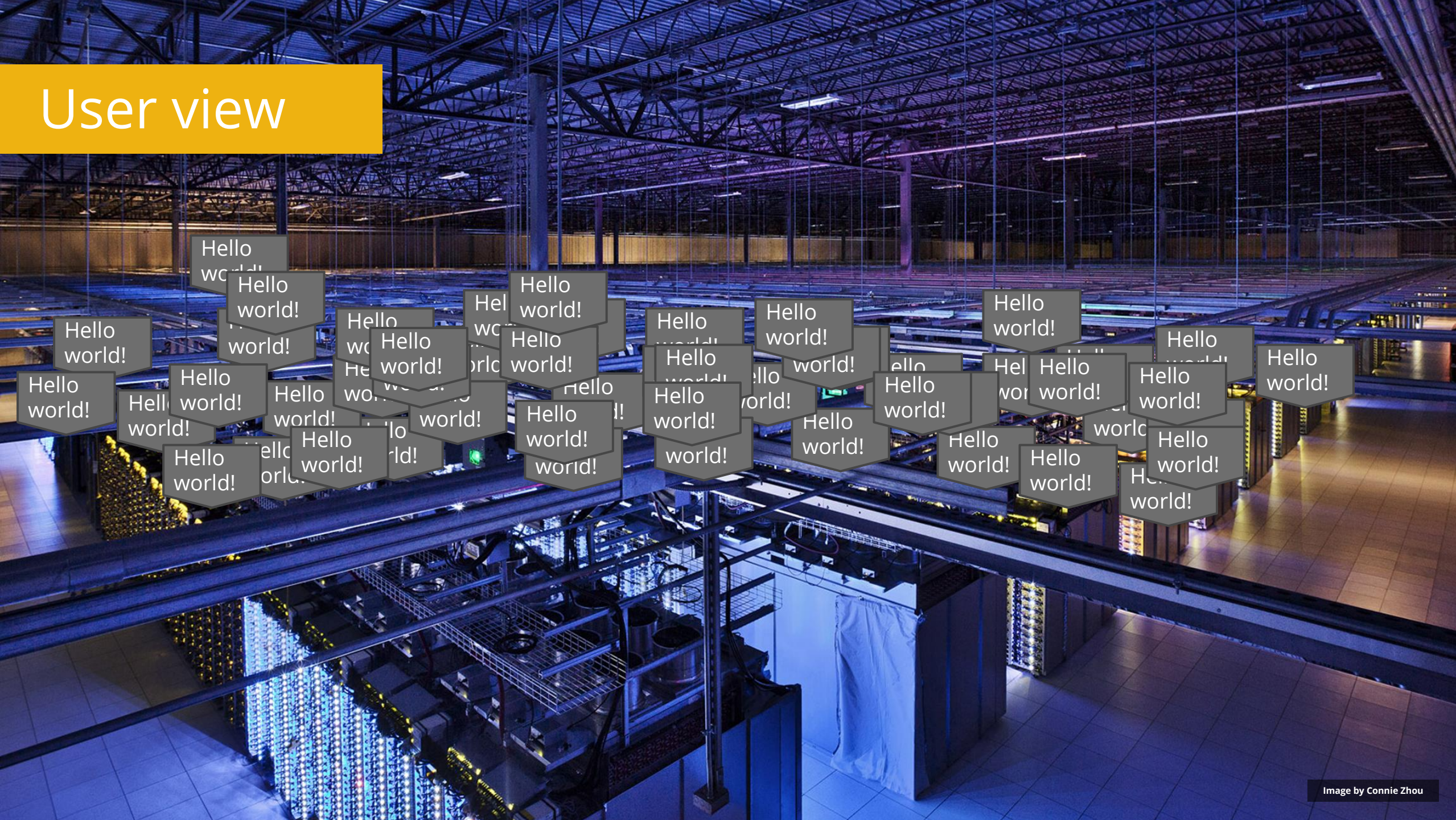


User view

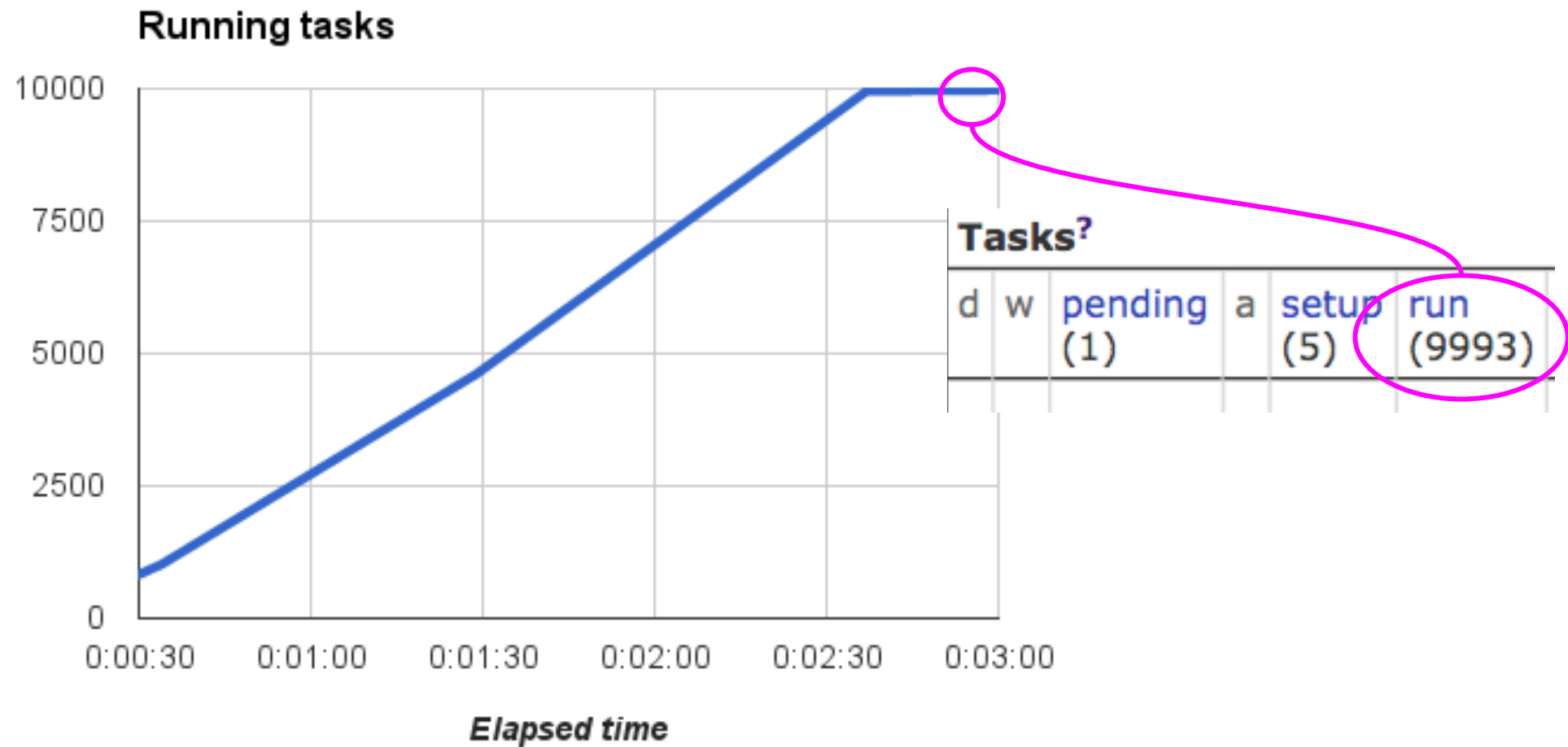
What just happened?



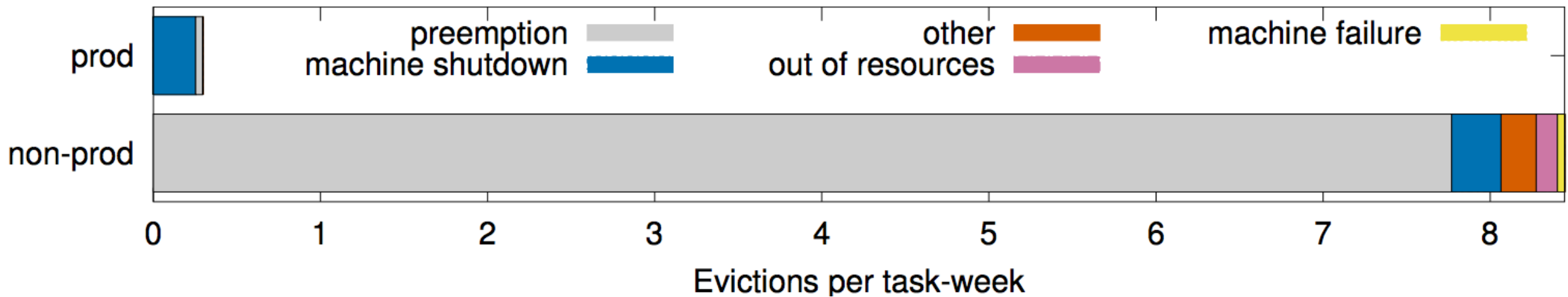
User view



User view



Failures



task-eviction rates
and causes

Failures

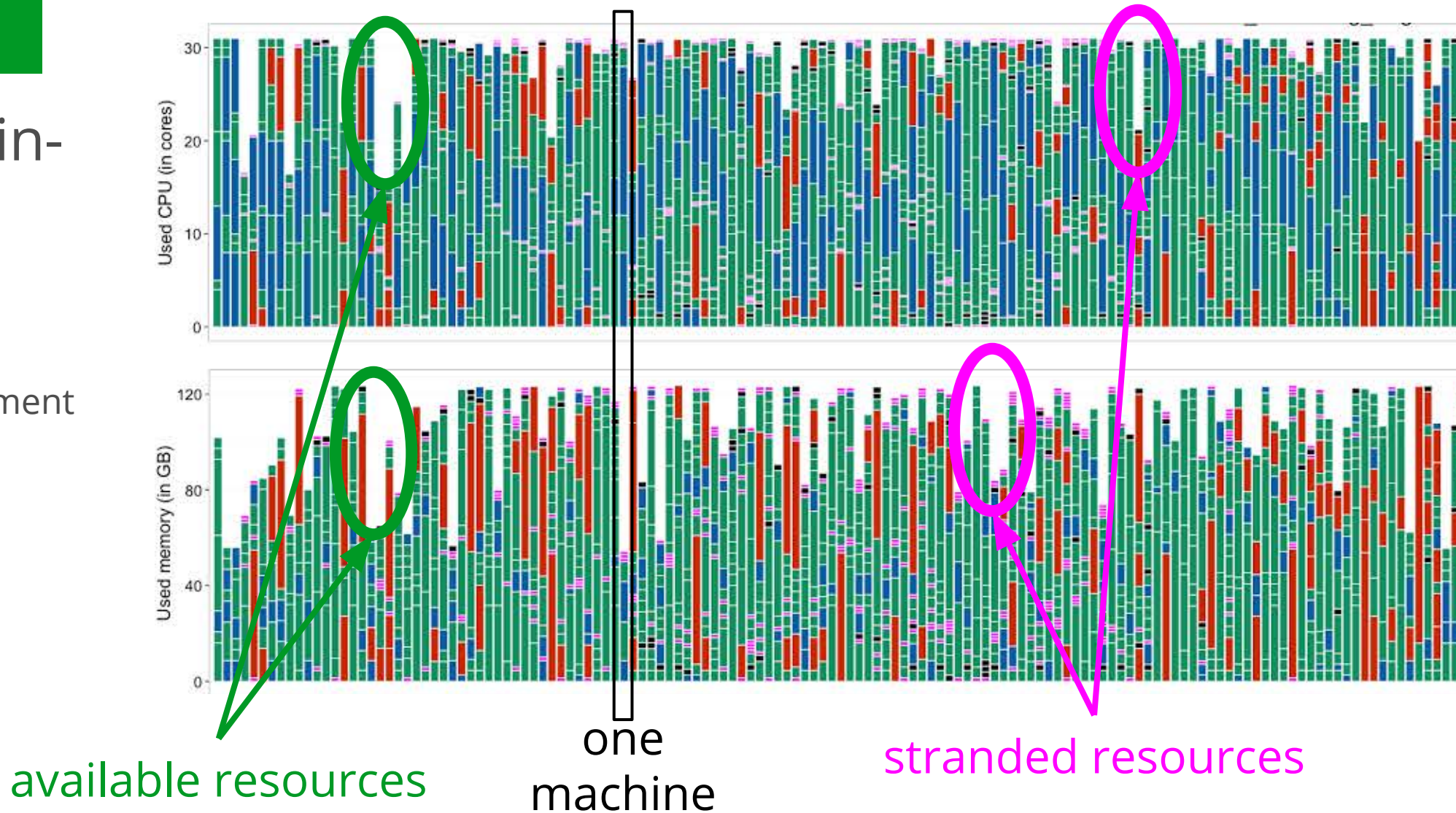
A photograph of a server room aisle. On the left, there are rows of server racks filled with various electronic components and a dense network of colorful cables (yellow, orange, blue, green). The racks extend into the distance. On the right, there are more racks, also filled with equipment and cables. The floor is light-colored and reflective. The overall scene depicts a large-scale data center environment.

A 2000-machine service will
have >10 task exits per day
This is not a problem: it's normal

Efficiency

Advanced bin-packing algorithms

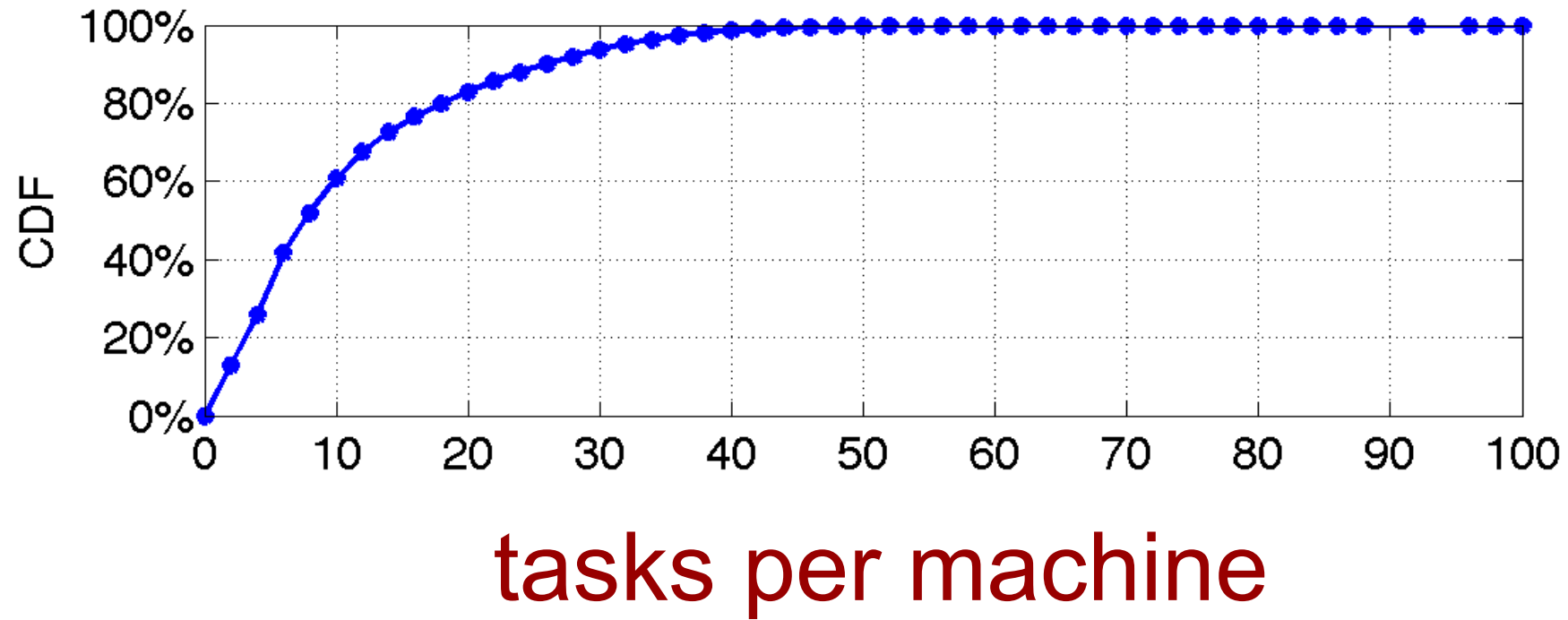
Experimental placement of production VM workload, July 2014



Efficiency

Multiple
applications
per machine

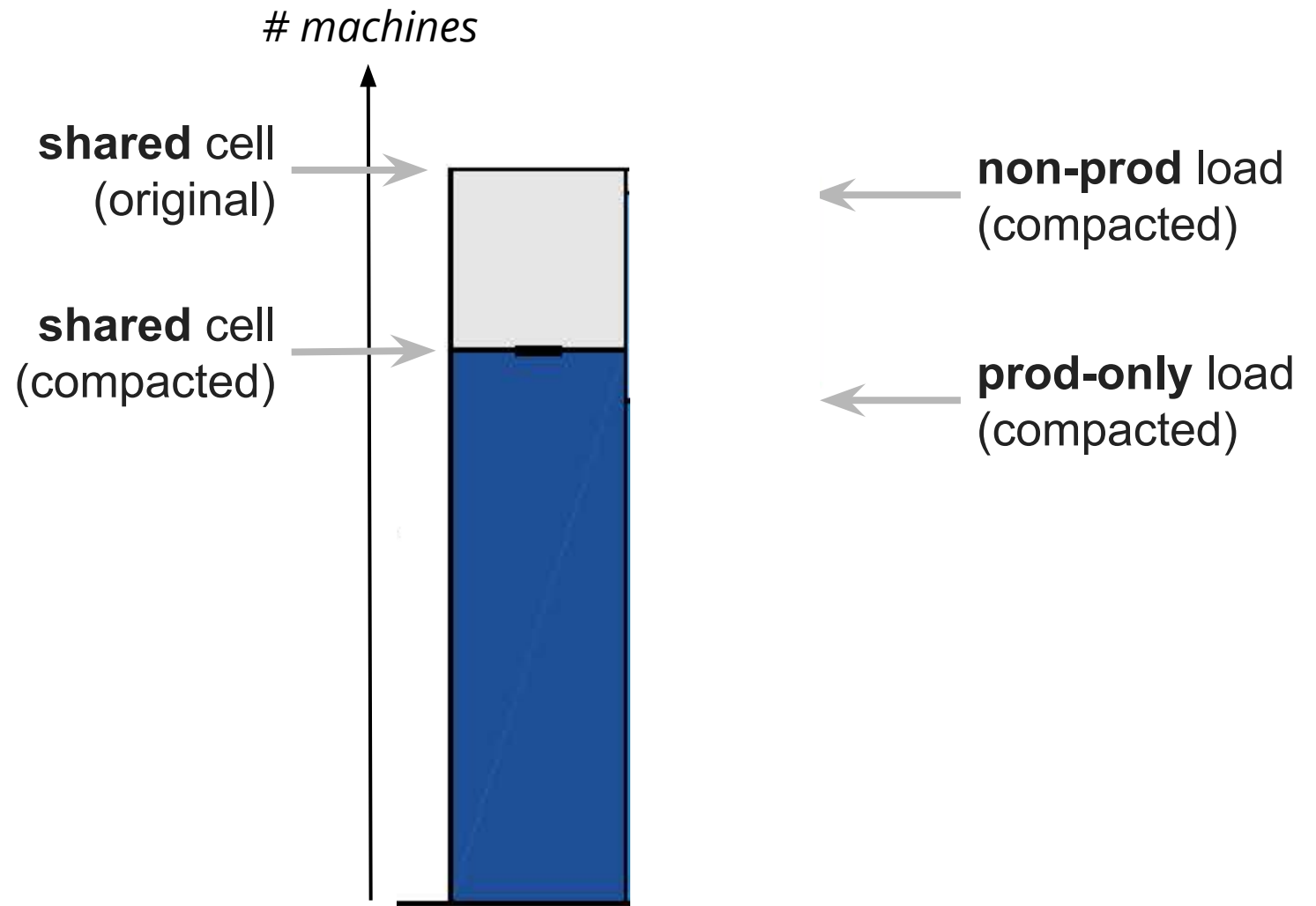
CPI^2 paper,
EuroSys 2013



Efficiency

Sharing clusters
between
prod/batch helps

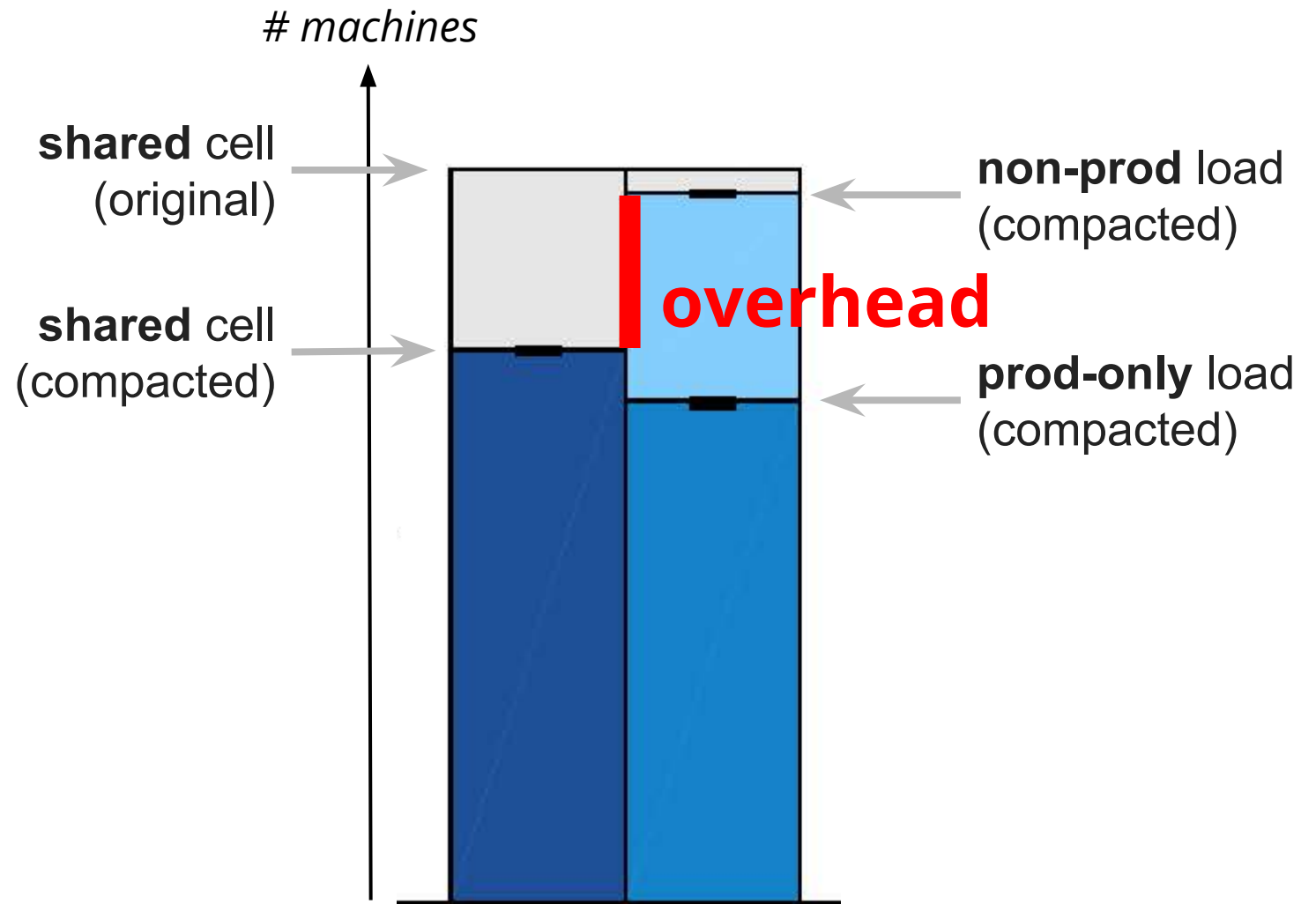
Segregating them would need
more machines



Efficiency

Sharing clusters
between
prod/batch helps

Segregating them would need
more machines

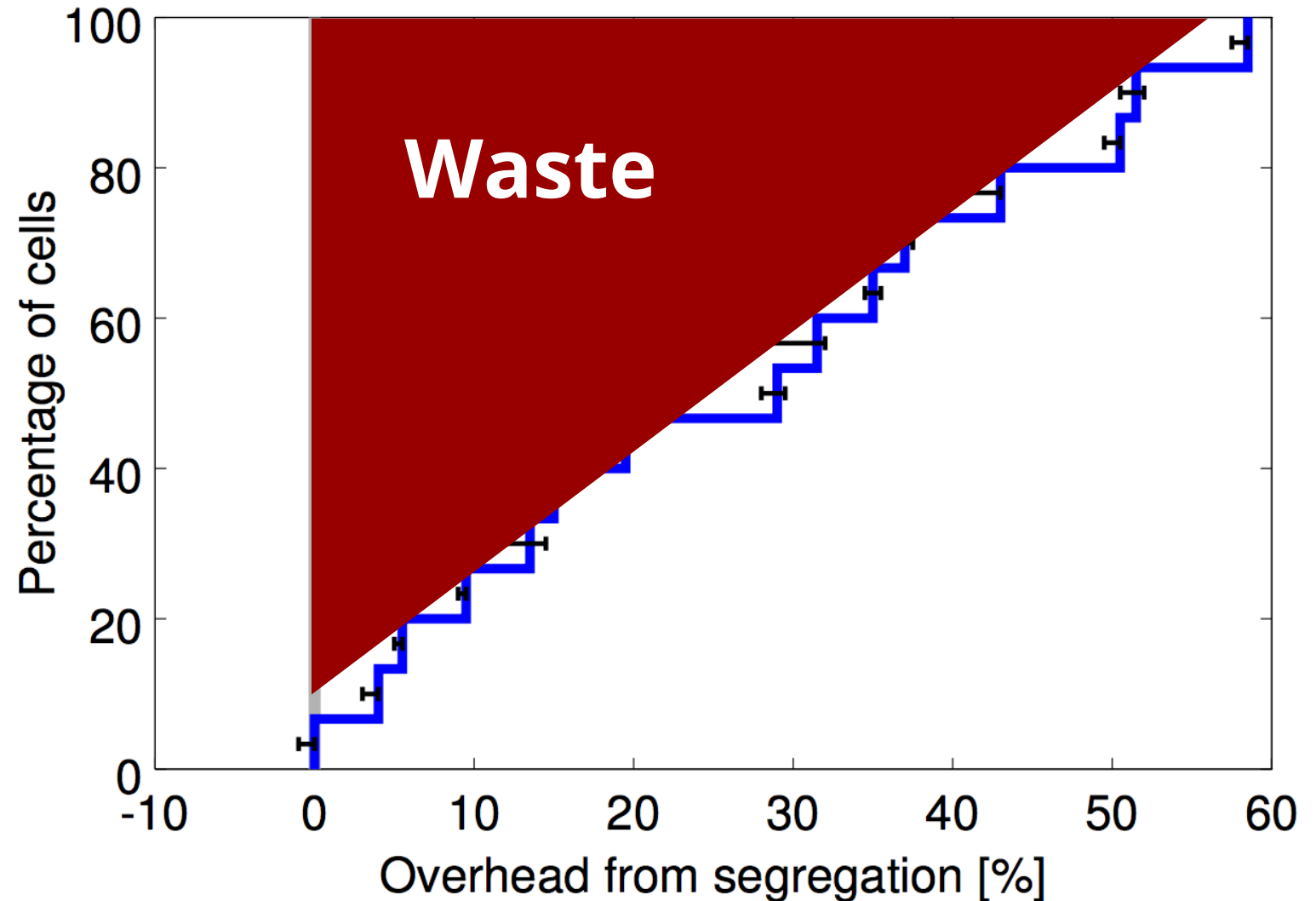


Efficiency

Sharing clusters
between
prod/batch helps

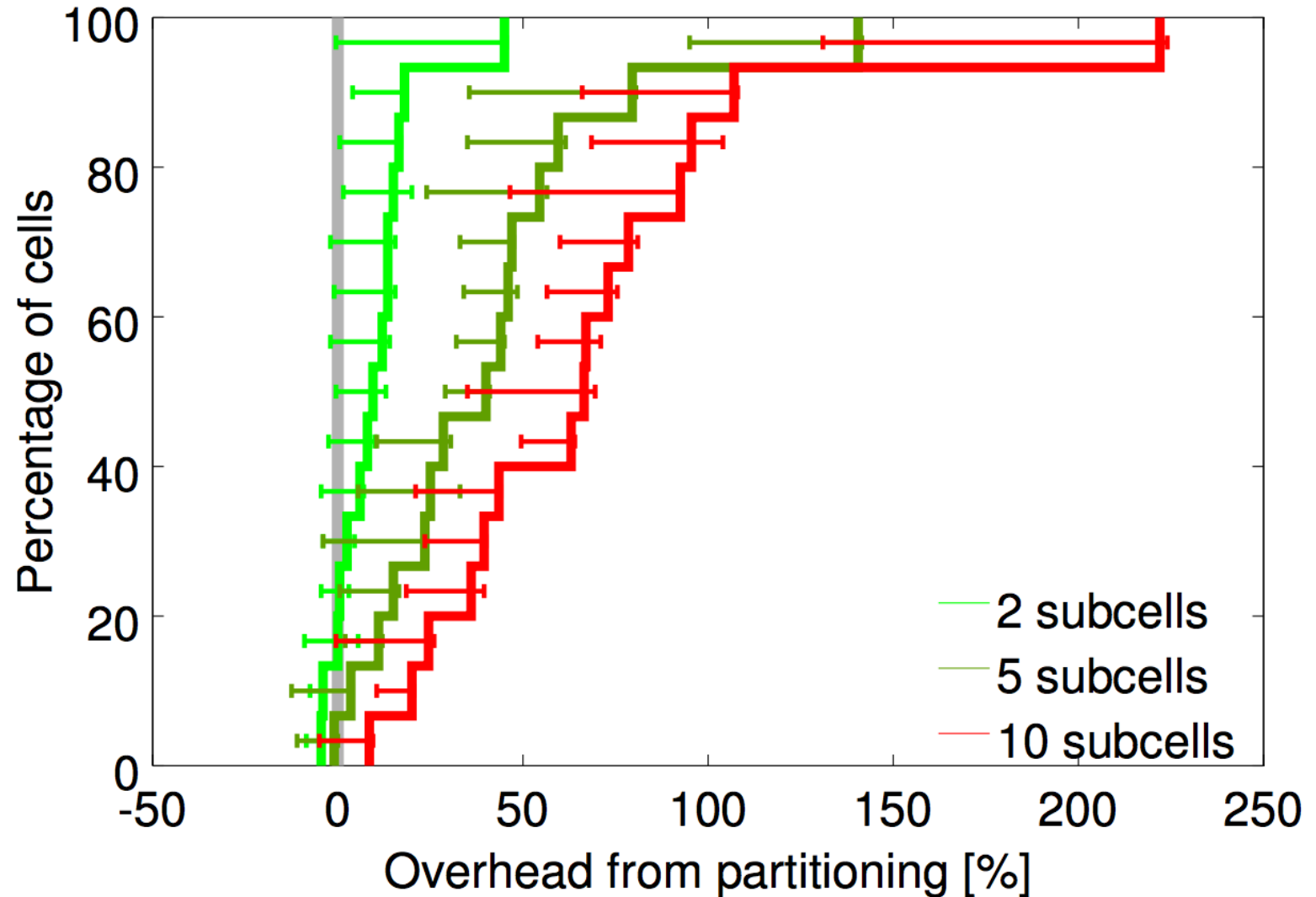
Segregating them would need
more machines

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



Efficiency

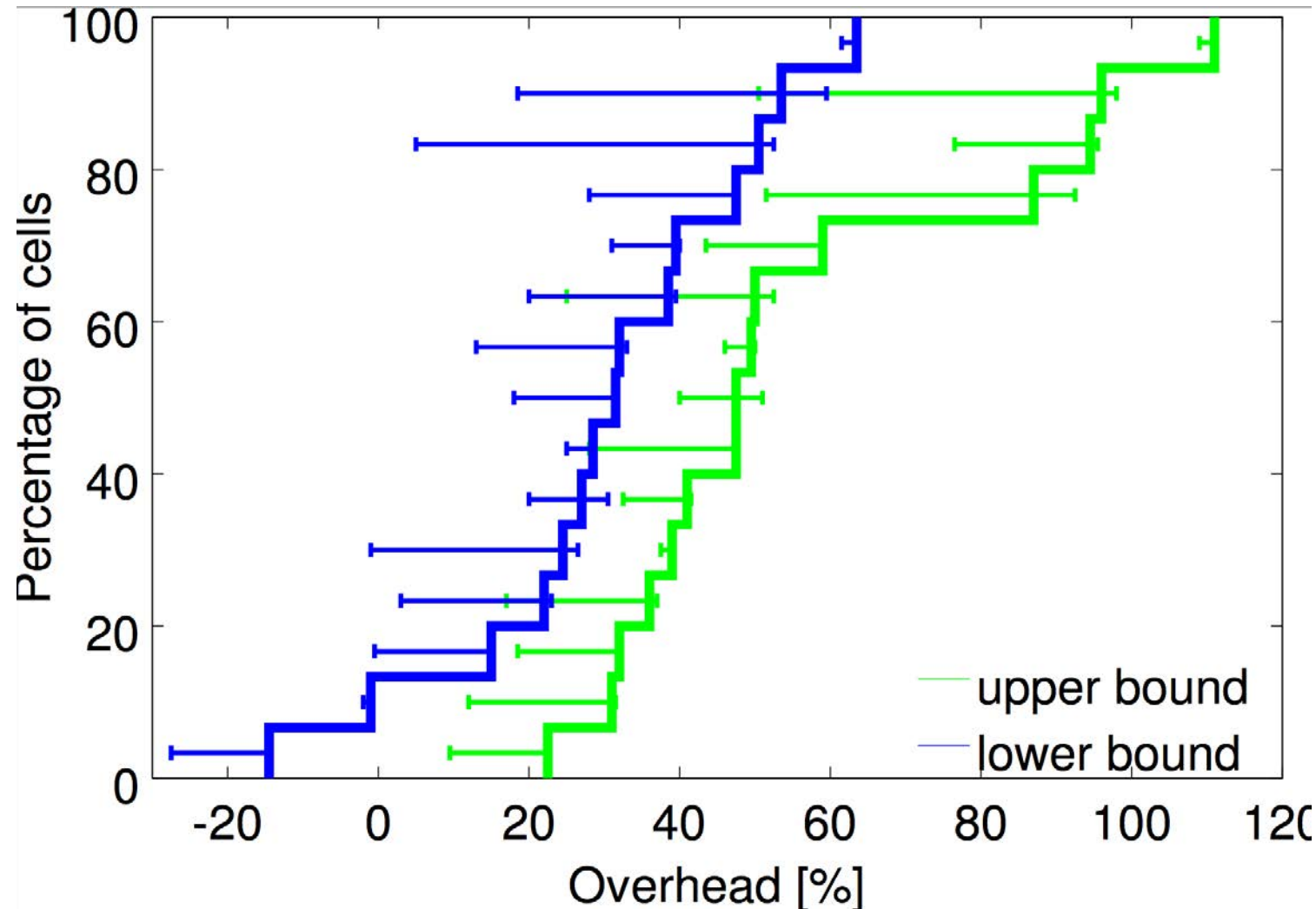
Smaller cells
would need more
machines



Efficiency

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

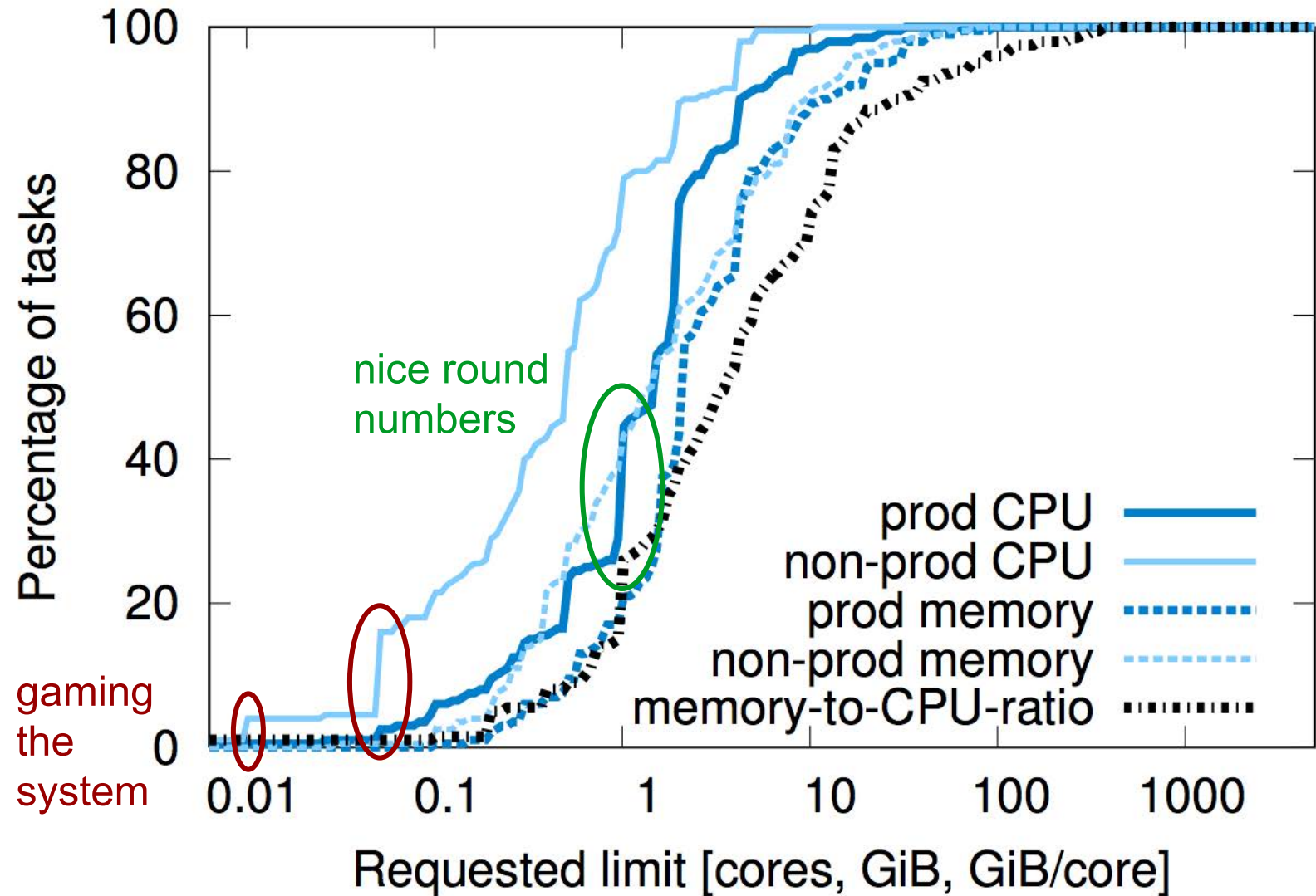
prod only,
starting from 0.5 cores, 0.5GiB



Efficiency

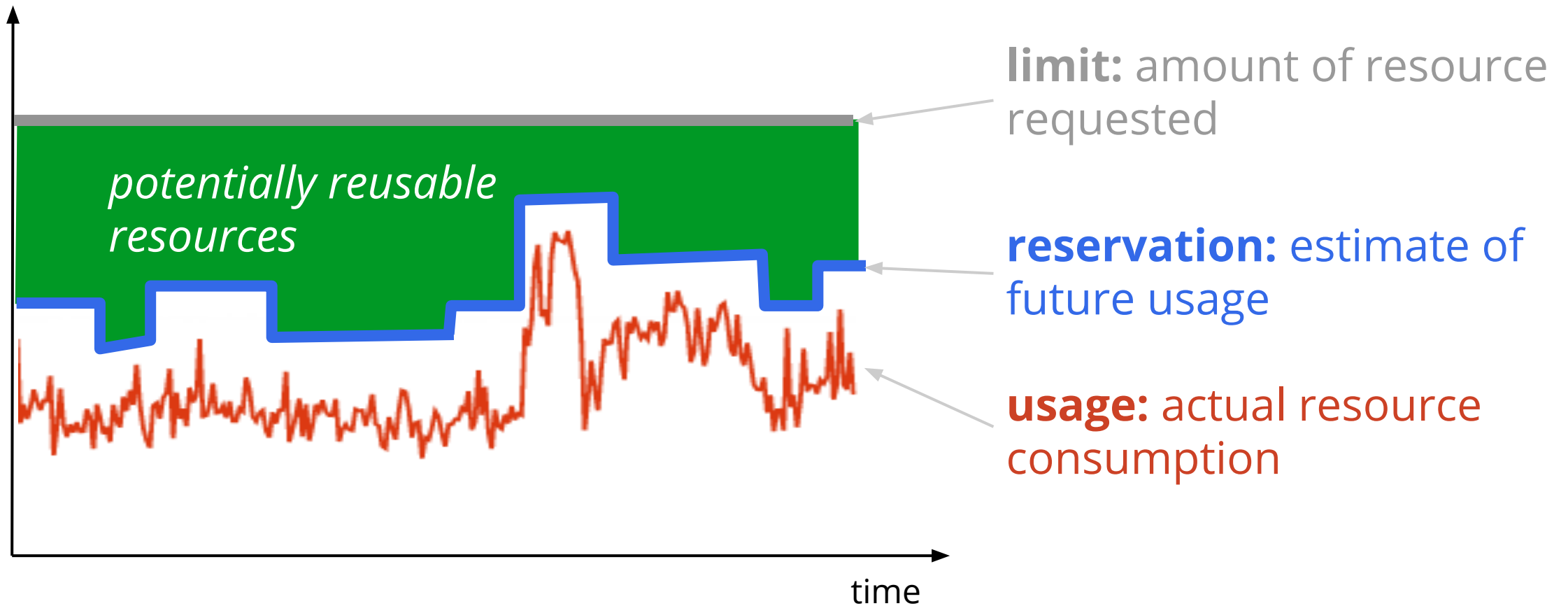
There are no obvious resource bucket sizes

cf. cloud VMs



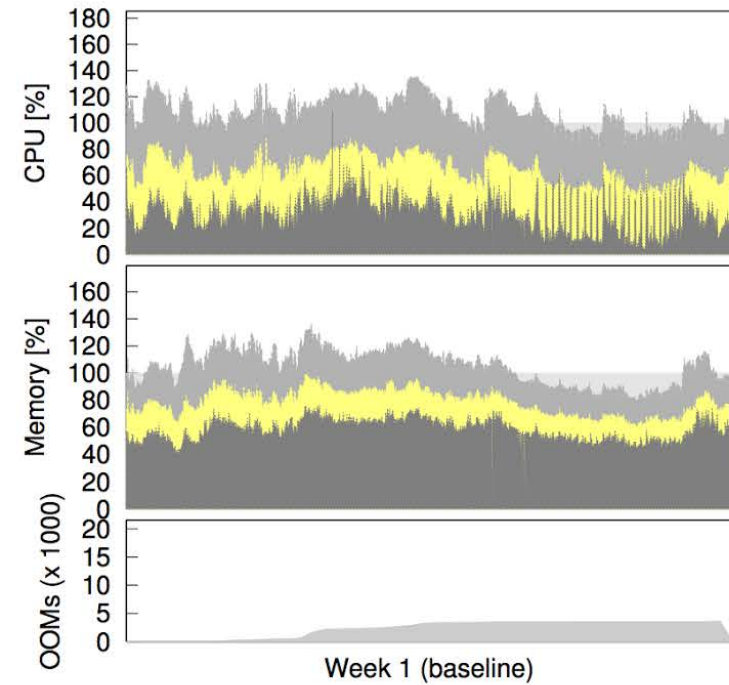
Efficiency

Resource reclamation



Efficiency

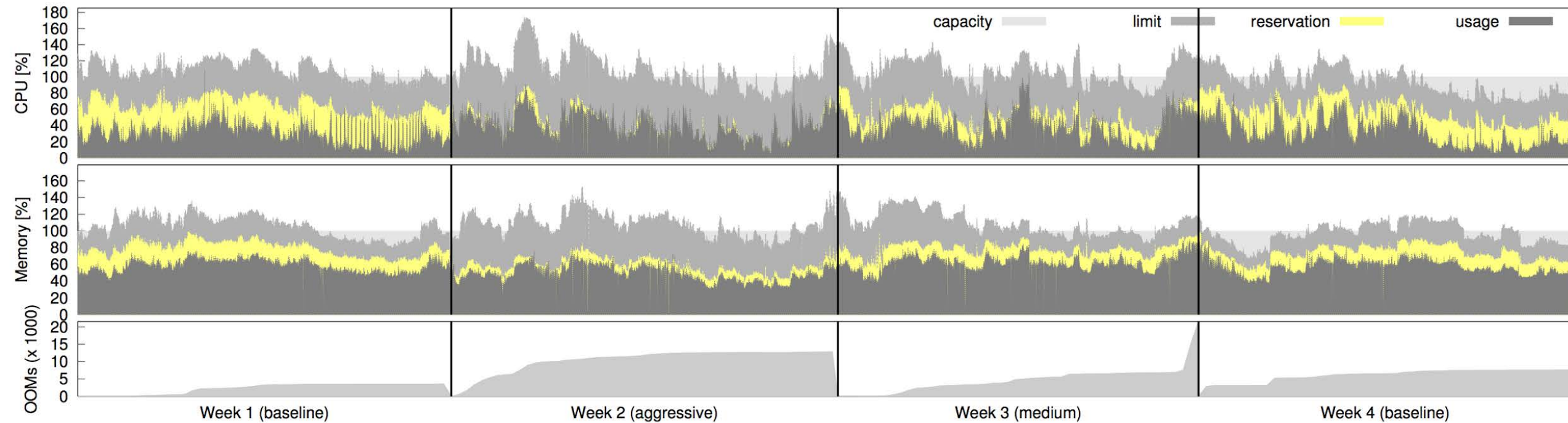
Resource reclamation could be more aggressive



Nov/Dec 2013

Efficiency

Resource reclamation could be more aggressive

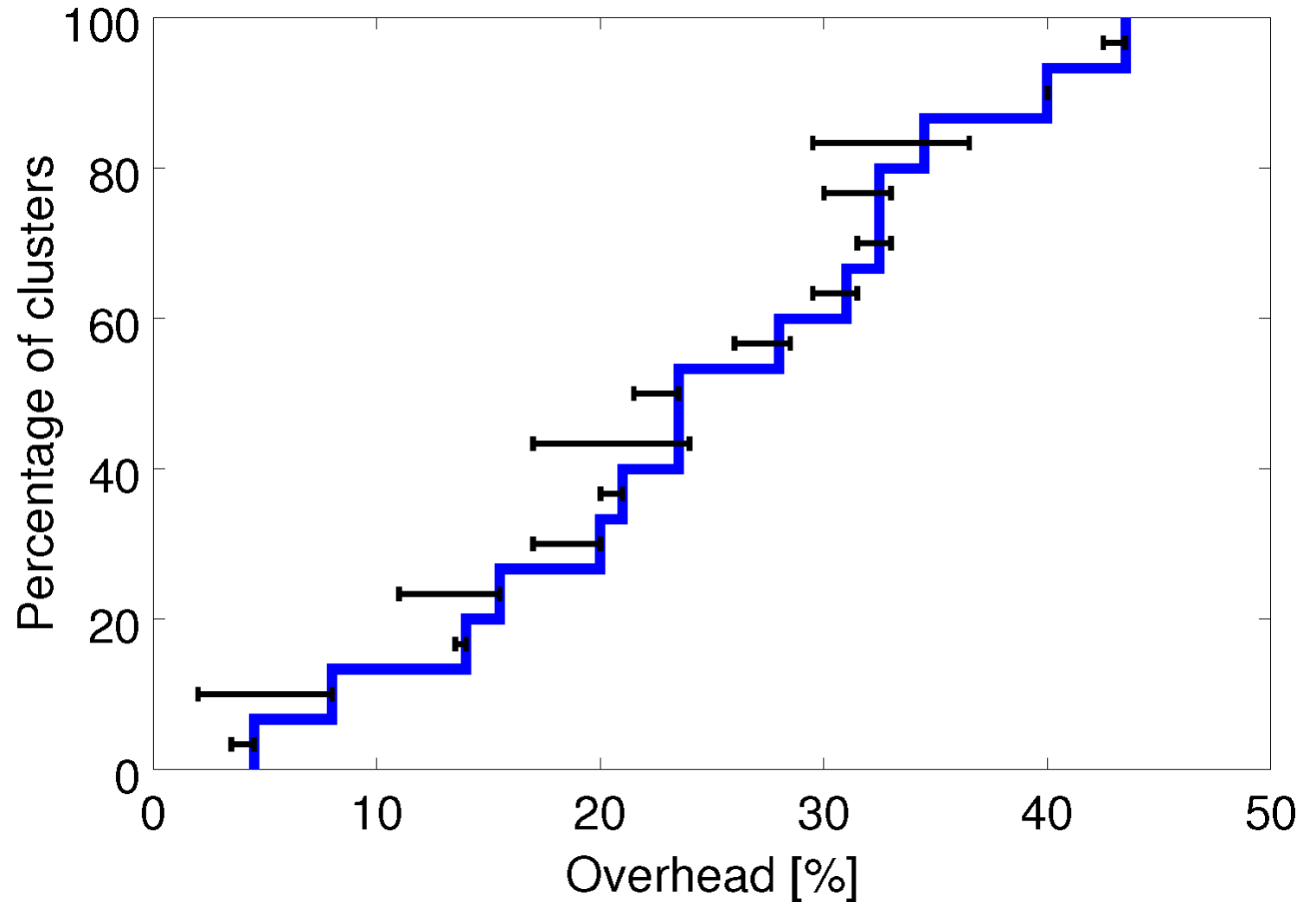


Nov/Dec 2013

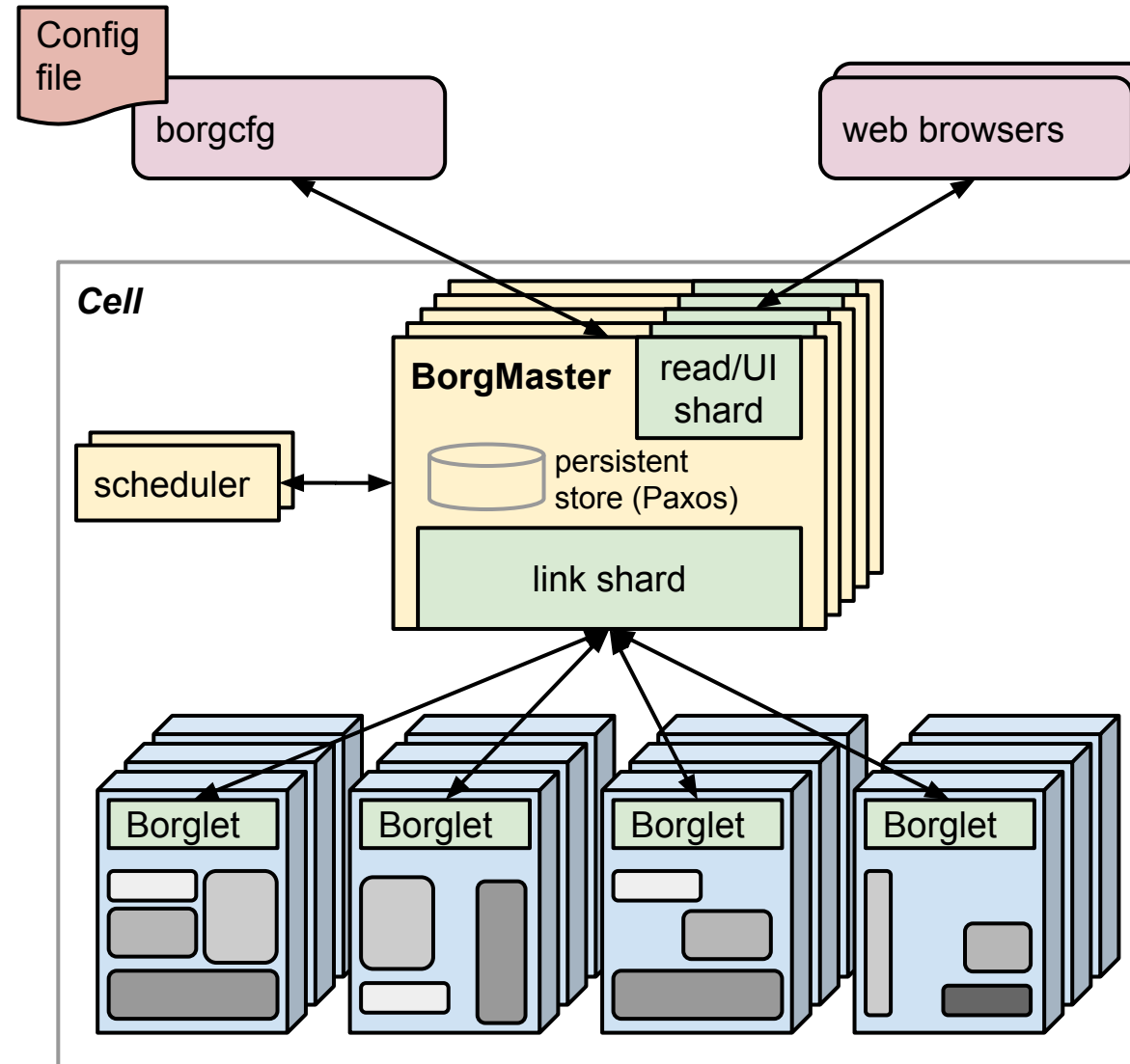
Efficiency

Resource reclamation

Turning off
resource
reclamation
would need
more
machines



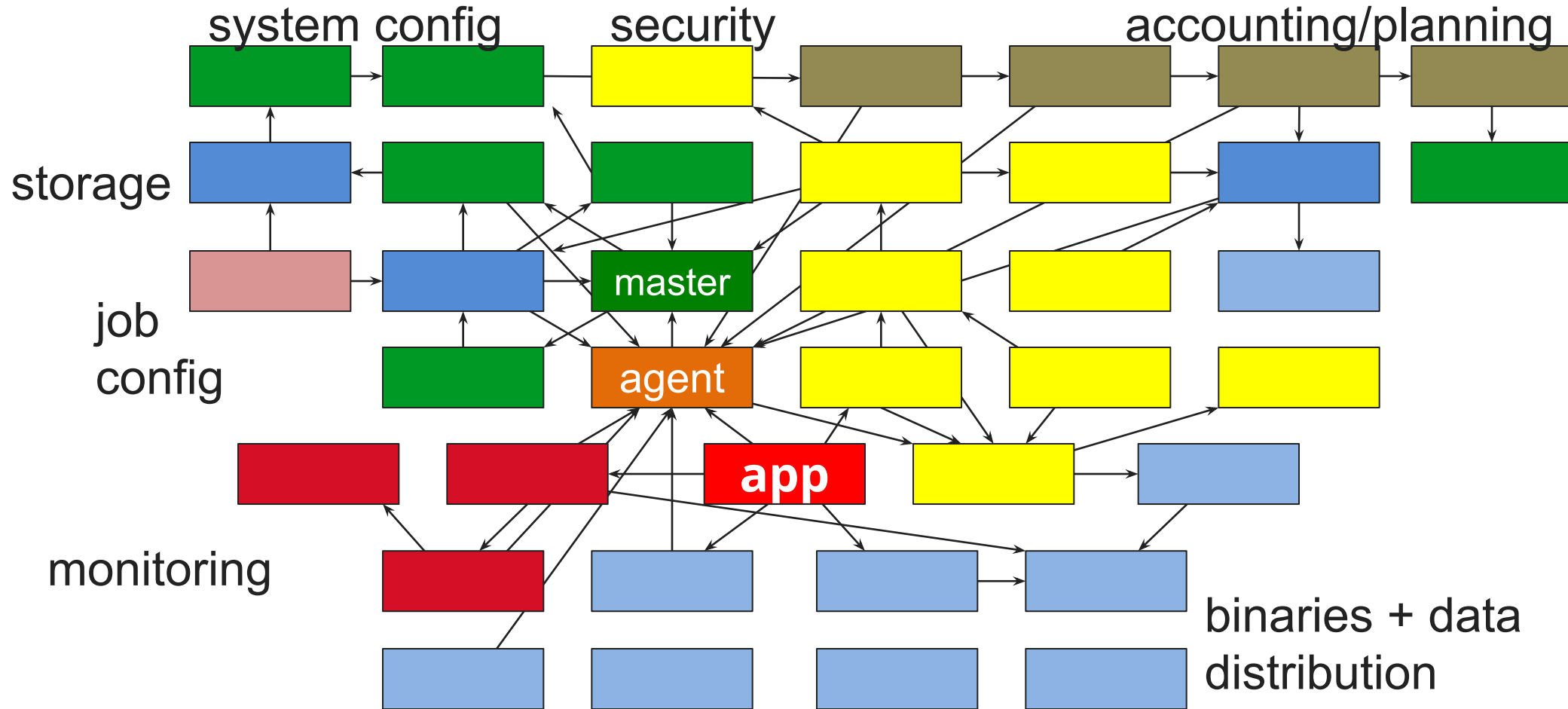
A few other moving parts



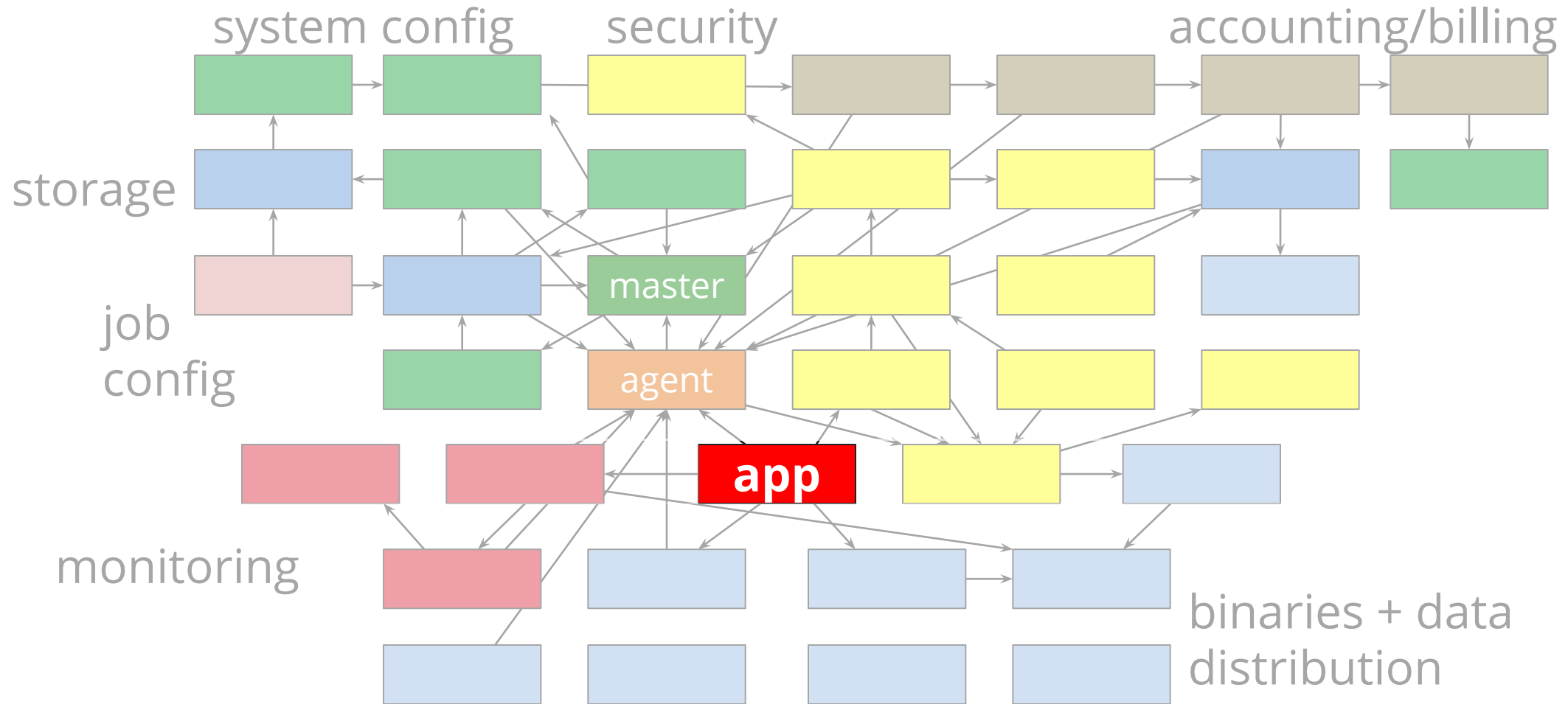
A few other moving parts



A few other moving parts



A few other moving parts





kubernetes

Home

Getting Started

Community

News

Events

Blog

Kubernetes

κυβερνήτης: *pilot*
or helmsman of a ship



kubernetes by Google

<http://kubernetes.io>

Kubernetes

Direct Borg analogues:

- Borg containers => **Docker containers**
- alloc (task group) => **pod (container group)**
- Borglet => **Kubelet**
- **persistent, declarative specs**
- **reconciliation loops**

Kubernetes

New / improved:

- **labels + label queries**
- **service abstraction**
- **composable microservices**
- **IP per pod**

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

<http://kubernetes.io>

<http://goo.gl/1C4nuo> (Borg paper)