Towards understanding today's and tomorrow's scheduling challenges in HPC systems

Gonzalo P. Rodrigo - gonzalo@cs.umu.se
Erik Elmroth – elmroth@cs.umu.se
Lavanya Ramakrishnan – lramakrishnan@lbl.gov
P-O Östberg – p-o@cs.umu.se

Distributed Systems Group – Umeå University, Sweden
Data Science & Technology – Lawrence Berkeley National Lab


UMEÅ UNIVERSITET
BERKELEY LAB

Outline

- Batch schedulers: Some basics
- Challenges:
  "Exascale initiative" and "Data Explosion"
- Are schedulers ready?
- Takeaways

Disclaimer:
This talk is about single site HPC scheduling!

Is not Scheduling a "solved problem"?

[1] "And you have to realize that there are not very many things that have aged as well as the scheduler. Which is just another proof that scheduling is easy."

Censored, 2001

[1] End of Dennard scaling = scheduler with an incredibly complex implementation:

- Non-uniform memory access latencies (NUMA
- High costs of cache coherency and synchro
- Diverging CPU and memory latencies.

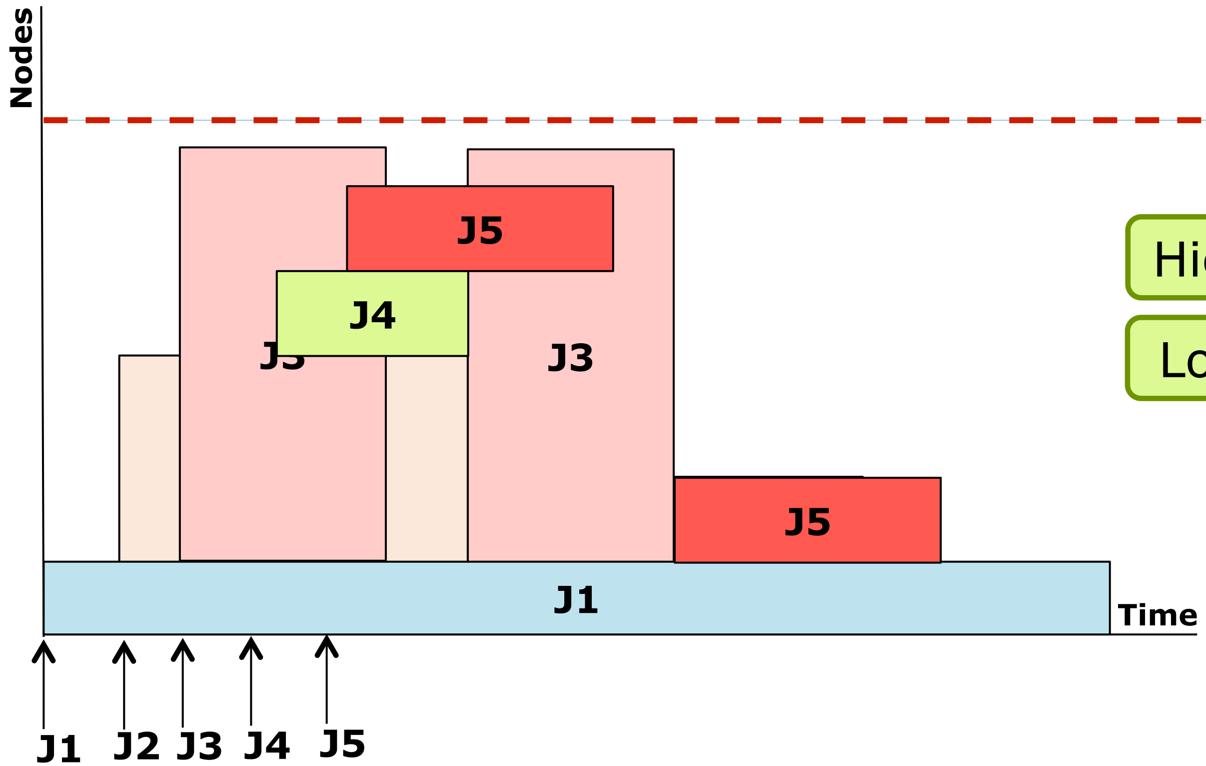
Non optimal schedulers

[1] Lozi, Jean-Pierre, et al. "The Linux scheduler: a decade of wasted cores." Proceedings of the Eleventh European Conference on Computer Systems. ACM, 2016.

Batch Schedulers: FCFS and Back-filling

FCFS: Jobs execute in arrival order

Back-filling: Job can start if it does not delay previous jobs.



High Utilization

Low Wait Time

Batch Schedulers: Fairness and prioritization

Fairness	Don't starve jobs or users
Priority	Run more important jobs first
Placement?	Actually not so important(?)

Episode 1: Upcoming challenges

Exascale

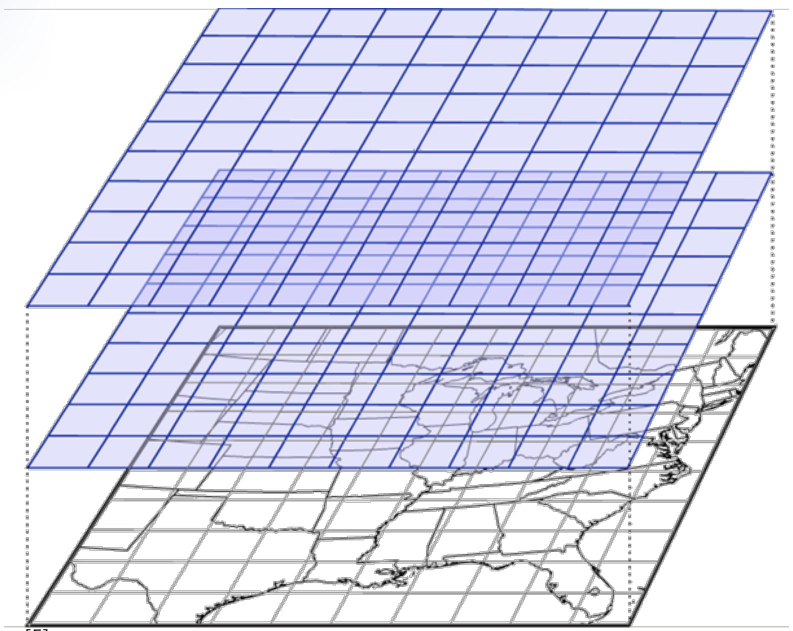
Data Explosion

Why?

Science is fueled by computation

Certain problems require better resolution

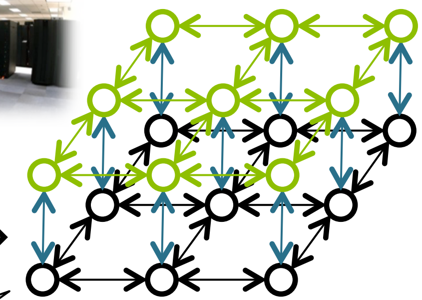
Understanding Large Parallel Tightly-Coupled Jobs



[5]

[6]

Map one cell per thread



One iteration per time step

1. Wait for neighbors' data
2. Simulate my "piece of atmosphere"
3. Send my data to neighbors
4. Repeat

[5] https://www.e-education.psu.edu/worldofweather/sites/www.e-education.psu.edu.worldofweather/files/image/Section2/Three_Dimensional_grid%20(Medium).PNG

[6] NOAA Stratus and Cirrus NOAA supercomputers 2009, http://www.noaanews.noaa.gov/stories2009/20090908_computer.html

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Exascale: Achieve One Exaflop in 2020

It’s all about power and cost

Tianhe-2
33.86 PFLOPS
US$390M
24 MW [7]

X 33 →

1 Exaflop
US$12 870M
792 MW

~Operative Income Ericsson 2014 [8]

~Average Swedish Nuclear reactor [9]

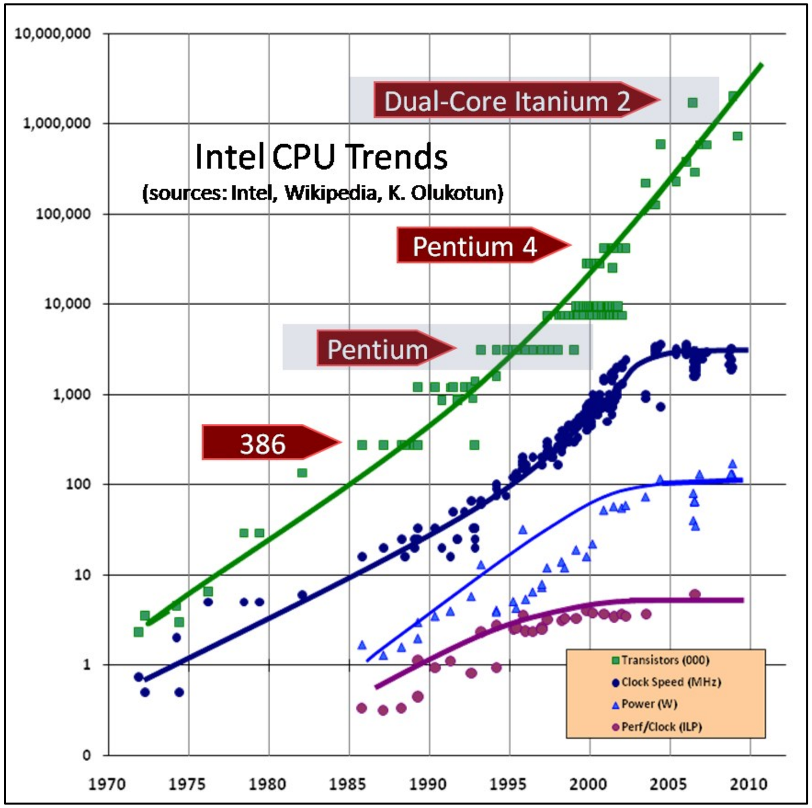
[7] https://en.wikipedia.org/wiki/Tianhe-2
[8] Fourth quarter and full-year report 2014 - Ericsson
[9] http://world-nuclear.org/information-library/country-profiles/countries-o-s/sweden.aspx

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Exascale: Achieve One Exaflop in 2020

It's all about power and cost

Break down of Dennard scaling



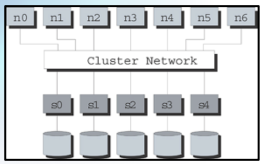
[10]

Extreme parallelization

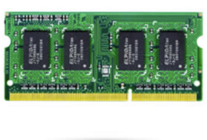
[10] http://www.extremetech.com/computing/116561-the-death-of-cpu-scaling-from-one-core-to-many-and-why-were-still-stuck

Exascale: Extreme paralellization

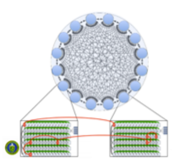
Raw Exaflops are possible but…



I/O	Only scalable in parallel! Not so good optimizations!



RAM	Power hungry!



Interconnect	More parallelism => More complexity **Less uniform latency**

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Exascale strategy: Paradox

Per Thread

(But so many of them!)

Very little

Compute Power

RAM

PFS I/O BW

Network BW

Electric Power

Produces

Huge Compute Capacity

The Exascale paradox

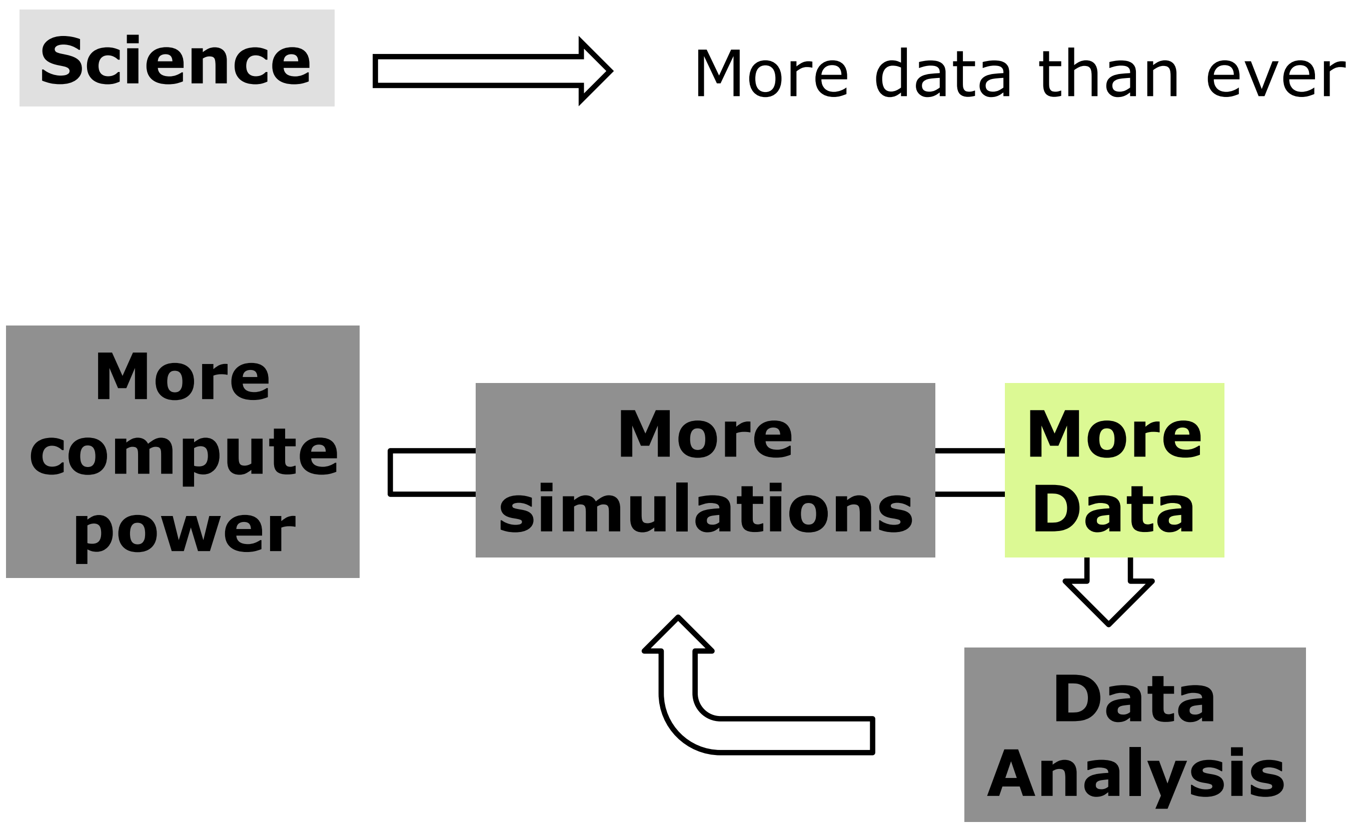
Per Thread

Very little

Compute Power	→	More coordination, more stages, heterogeneity. **Workflows!**
RAM	→	
PFS I/O BW	→	Complex I/O Hierarchy
Network BW	→	More in-chip comms: **OpenMP**
Electric Power	→	Reduced Resilience!

Increase Compute Gap Vs…

Data Explosion Challenge: 4th paradigm of Science [11]



[11] Tansley, Stewart, and Kristin Michele Tolle, eds. The fourth paradigm: data-intensive scientific discovery. Vol. 1. Redmond, WA: Microsoft research, 2009.

Data Explosion consequences

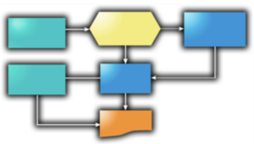
Data explosion

Data management

I/O Gap importance

Resource Heterogeneity

Temporary Data



Workflows

Episode 2: challenges vs. Schedulers

Are schedulers ready for current workloads?

Are other scheduling models possible?

Can we schedule workflows better?

Performance?

Are schedulers ready for current workloads?

Understanding how workloads have evolved in the past

Detailed analysis of current workloads

Observations on the performance

Workloads we studied

Supercomputers		Cluster
Hopper	**Edison**	**Carver**
Deployed January 2010	Deployed January 2014	Deployed 2010
Cray XE6	Cray XC30	IBM iDataPlex
Gemini Network	Aries Network	Infiniband (fat-tree)
6,384 Nodes, 24 cores/node **154,216 cores**	5,576 Nodes, 24 cores/node **133,824 cores**	1,120 Nodes, 8/12/32 cores/node, **9,984 cores**
1.28 Pflops/s	2.57 Pflops/s	106.5 Tflops
Torque + Moab	Torque + Moab	Torque + Moab

First step: System’s lifetime workload evolution

Hypothesis: Job geometry has changed during the system’s lifespan.

Method: Workload analysis

Job variables

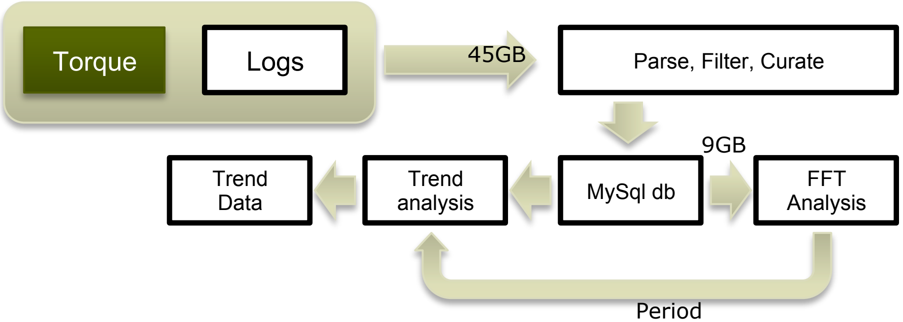
- Wall clock, number of cores (allocated), compute time, wait time, and wall clock time estimation.

Dataset

- 2010 – 2014: Torque logs
- 4.5M (Hopper) and 9.3M (Carver) Jobs
- Raw data 45 GB. Filtered data 9.3GB

Analysis

- Period slicing
- Period analysis
- Comparison



Rodrigo Álvarez, G. P., Östberg, P. O., Elmroth, E., Antypas, K., Gerber, R., & Ramakrishnan, L. (2015, June). HPC System Lifetime Story: Workload Characterization and Evolutionary Analyses on NERSC Systems. In Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (pp. 57-60)

First step: System's lifetime workload evolution

Two machines with very different starting workloads, become more similar towards the end.

Most jobs are not very long and very parallel

Systems get "more loaded" in time

Users' estimations are really inaccurate.

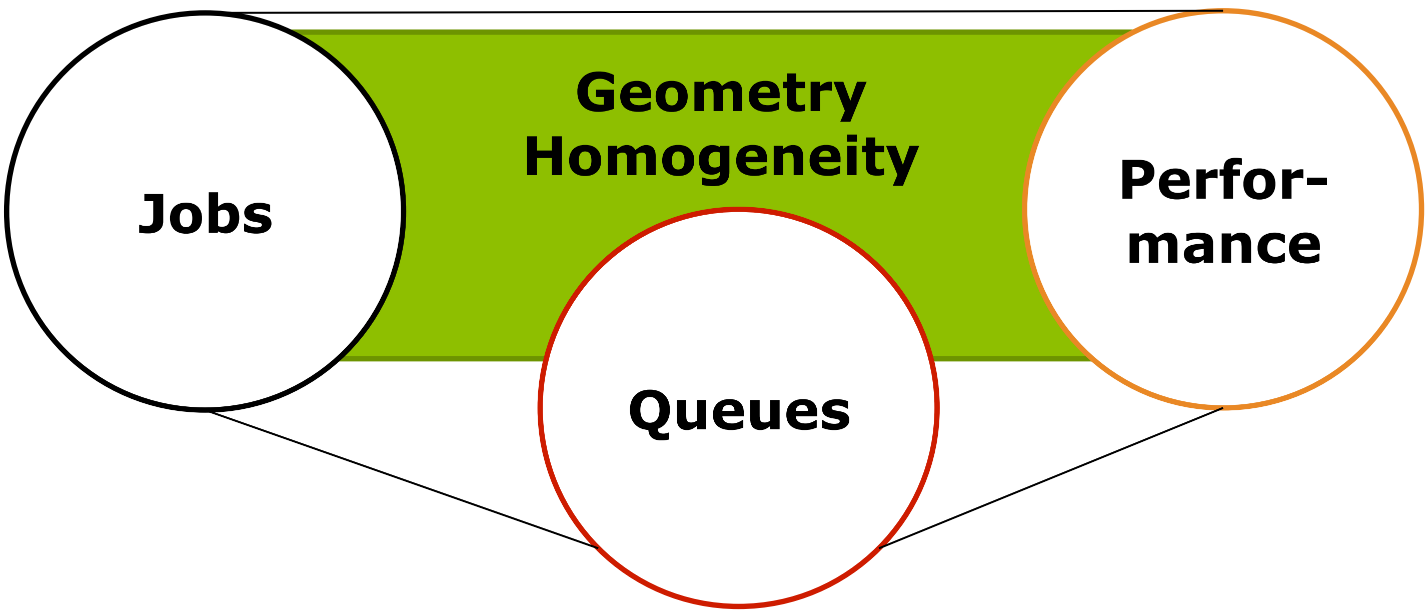
(medians)	2010 Hopper	2010 Carver	2014 Hopper	2014 Carver
Wall Clock	< 1 min	20 min	12 min	6 min
Number of Cores	100 cores	5 cores	30 cores	1 core
Core Hours	4 c.h.	0.9 c.h.	11 c.h.	0.09 c.h.
Wait time	100 s	10 min	20 min	20 min
Wall clock accuracy	0.2	0.25	0.21	< 0.1

Second step: Job Heterogeneity

Hypothesis: Job heterogeneity affects the scheduler performance.

Method: Detailed workload analysis of a year Dataset

- 2014 Torque logs
- Hopper, Edison, and Carver Jobs
- Define a method heterogeneity analysis



G. Rodrigo, P-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan. Towards Understanding Job Heterogeneity in HPC: A NERSC Case Study. CCGrid 2016 - The 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Accepted, 2016.

Job geometry + Job priority + Job Wait time

Job Geometry	Bigger = Longer Wait
Job Priority	Higher = Shorter Wait
Queue busy	Higher = Longer Wait
Queue Homog.	Low = Predictable?

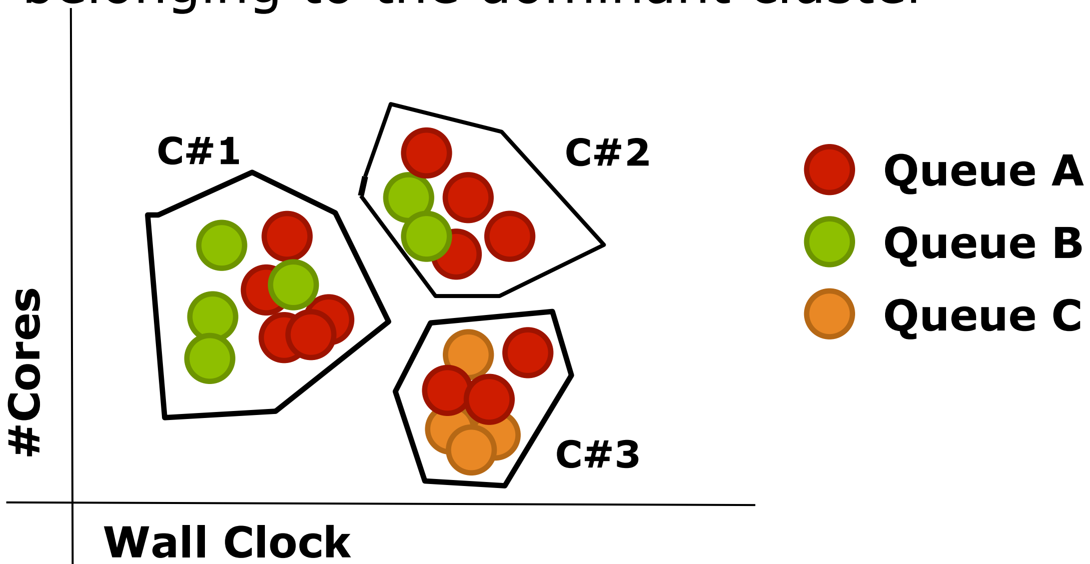
Do wait time expectation
hold in
Heterogeneous Queues?

Job and Queue homogeneity: Cluster mapping

Machine learning technique to detect clusters (k-means)
Wall clock time + allocated cores

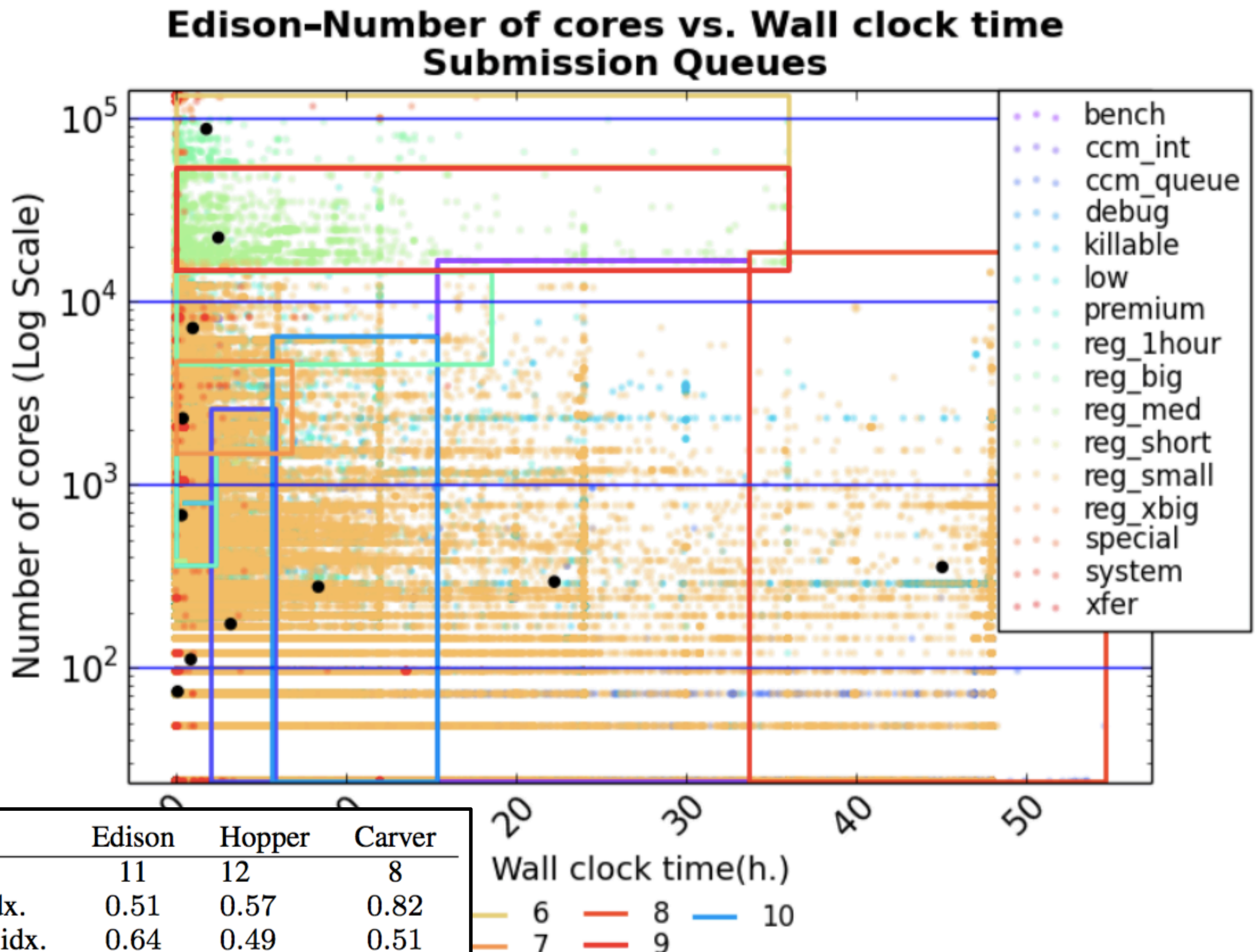
Dominant Cluster
  Cluster to which most queue jobs belong
Queue homogeneity index
  % of jobs belonging to the dominant cluster

Q	Dom. C	Hom. Idx
A	1	41%
B	1	71%
C	3	100%



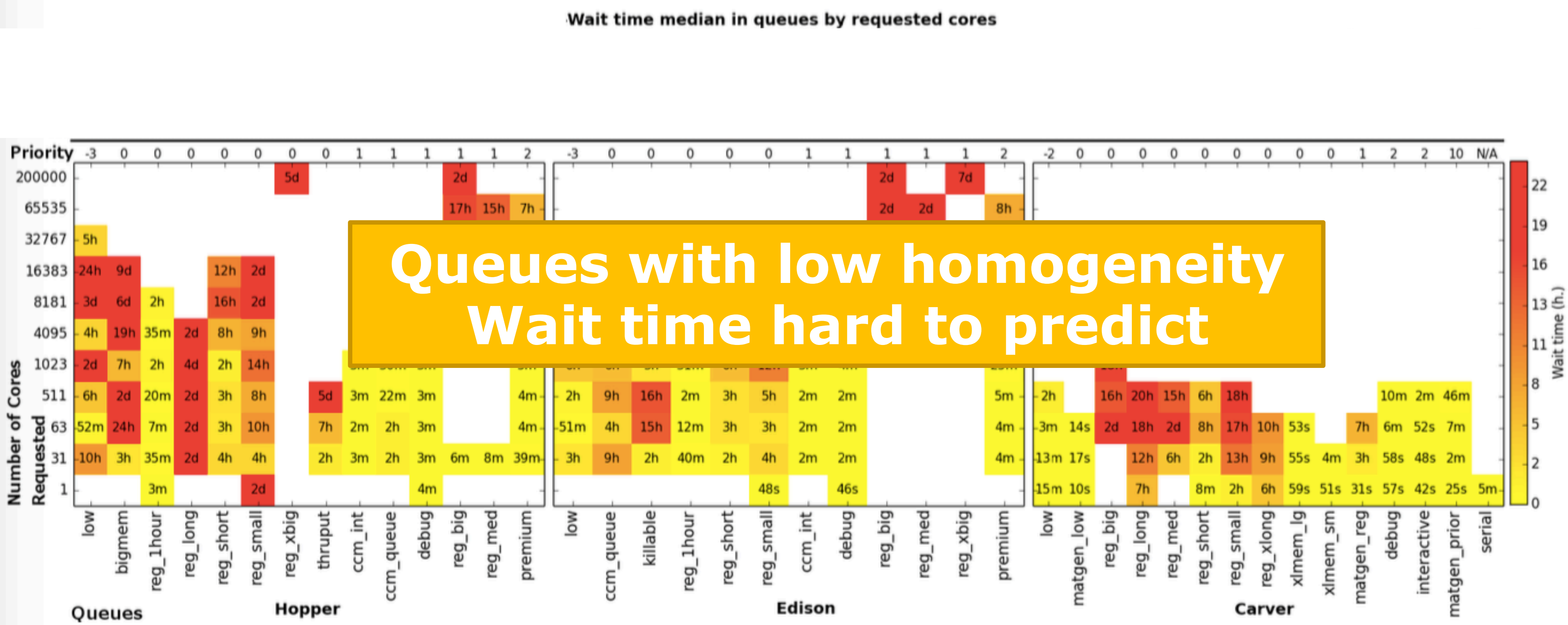
Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Queue homogeneity: Cluster mapping



	Edison	Hopper	Carver
Clusters	11	12	8
Job homogeneity idx.	0.51	0.57	0.82
Time homogeneity idx.	0.64	0.49	0.51

Performance + Queues + Homogeneity



Queues with low homogeneity
Wait time hard to predict

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Conclusions

(1) job geometries were fairly diverse, including a significant number of sma

Job diversity is high

The low per queue homogeneity indexes, show that (2) single priority policies are affecting jobs with a fairly diverse geome

Deal with it, or your system wait time might be hard to predict

The wa
low homogeneity indexes present poor correlation between job's wait time and geometry.

Finall
clock
backfilling functions) is very low.

Maybe queues should be re-ordered

Let's do something about run time prediction

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

So..

Are schedulers ready for the current (and future) workload?

Other challenges?

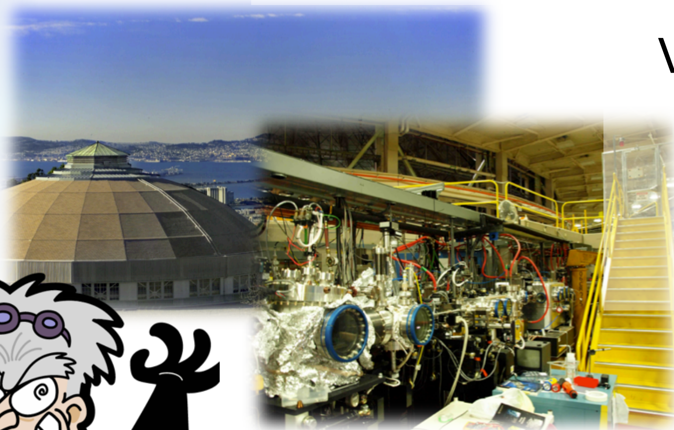
Game changers vs. Schedulers

Game changers		Schedulers
Job diversity		Less predictability
Application diversity		Different requirements
Exascale	→	New challenges
Data intensive Applications		Malleable
Real time applications		Low latency allocation

Rodrigo Álvarez, G. P., Östberg, P. O., Elmroth, E., & Ramakrishnan, L. (2015, June). A2L2: An Application Aware Flexible HPC Scheduling Model for Low-Latency Allocation. In Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing (pp. 11-19)

Game changer:
Live experiments data processing **(stream)**

Advance **L**ight **S**ource



Video recording (data)

3D Scanner of materials

Carver

(IBM iDataPlex)

Post processed Data (one day later)

- Live experiment
- Produces data (large amounts)
- Required to be processed on a super computer
- Processed results one day later

- Experiment would benefit of live feedback!
- Reservations are hard to align to reality!

Looking for inspiration… in the clouds.

Cloud infrastructures have faced similar challenges…

Hypothesis: Cloud scheduling techniques can be applied to tackled new HPC challenges.
Method: Compared study on techniques and application circumstances (Survey)

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Similarities

Applications

Cloud	HPC
Heterogeneous Workload	**Heterogeneous Workload**
Batch Jobs	Many non tightly coupled
Data is Key	Non-classical HPC
Wait Time is important	Response time
SSDs on Nodes	**Burst Buffer**
Distributed Filesystems	
Heterogeneous resources	**Accelerator HW**, **BB nodes**, **Compute nodes**

Infrastructure

A2L2

Application aware scheduling: Aware of characteristics, performance models, different rules for different types of job.

Dynamically malleable management: runtime re-scaling of jobs, performance based allocation.

Flexible backfilling: for better utilization

Low latency allocation: To allow allocation of jobs a short time after submission (stream job)

Position Paper

Scheduler model

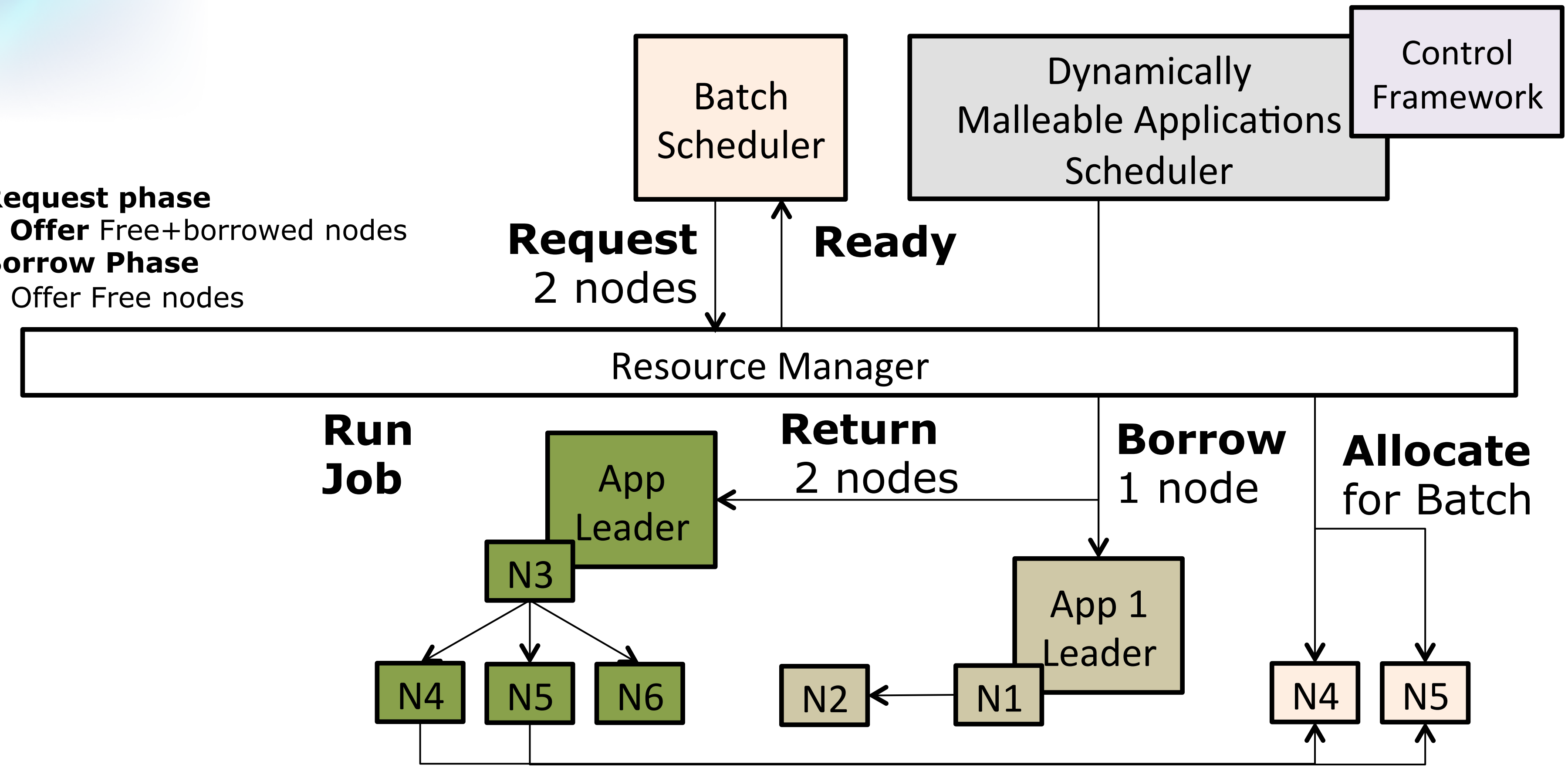
Cloud borrowed solution: **Two level scheduling**

One scheduler per application + smart RM

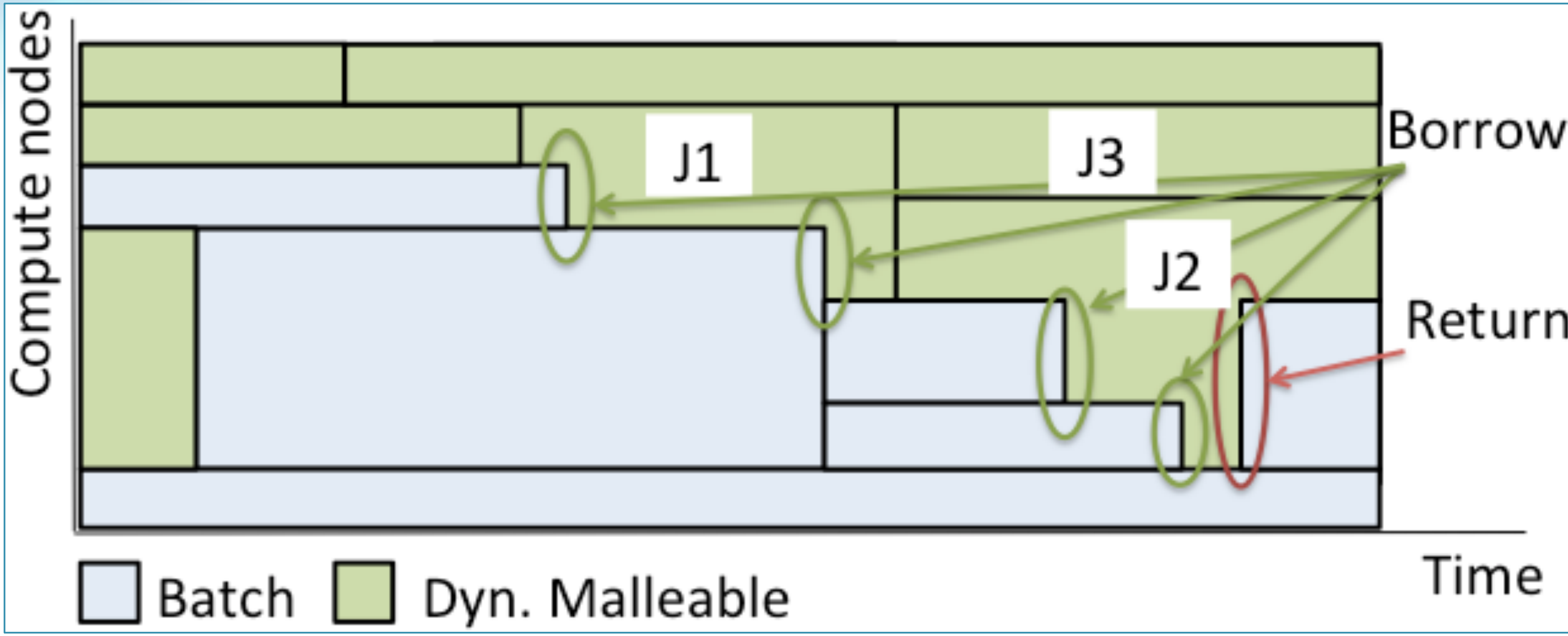
Malleable Applications: Dynamic allocation

Low latency allocation

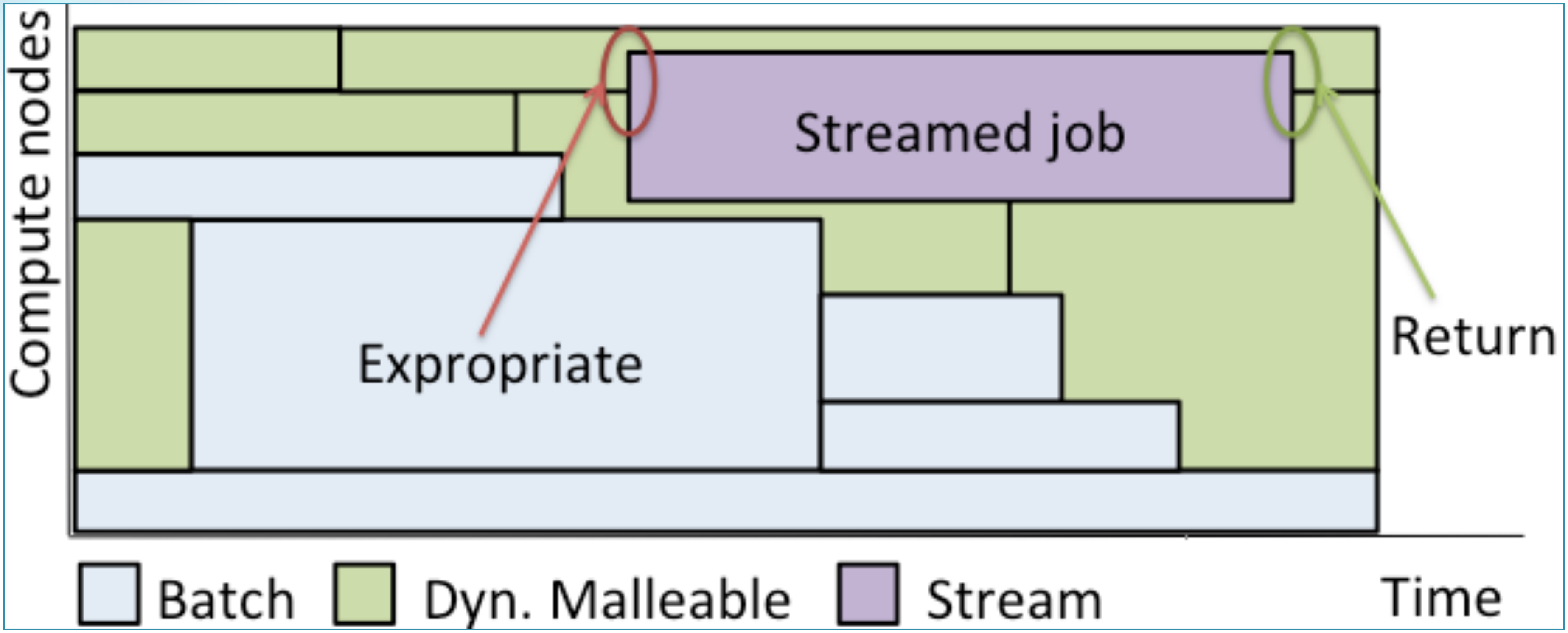
Request phase
  Offer Free+borrowed nodes
Borrow Phase
  Offer Free nodes



Flexible backfilling



Resource Expropriation: Low latency allocation



Gonzalo P. Rodrigo – gonzalo@cs.umu.se

A2L2: Conclusions

Application heterogeneity ar… …it of both cloud and HPC …

Application Aware

Flexi… …e applications can be …aybe enough malleable …d to make be useful)

Experimentation in progress

Application Management

Better utilization

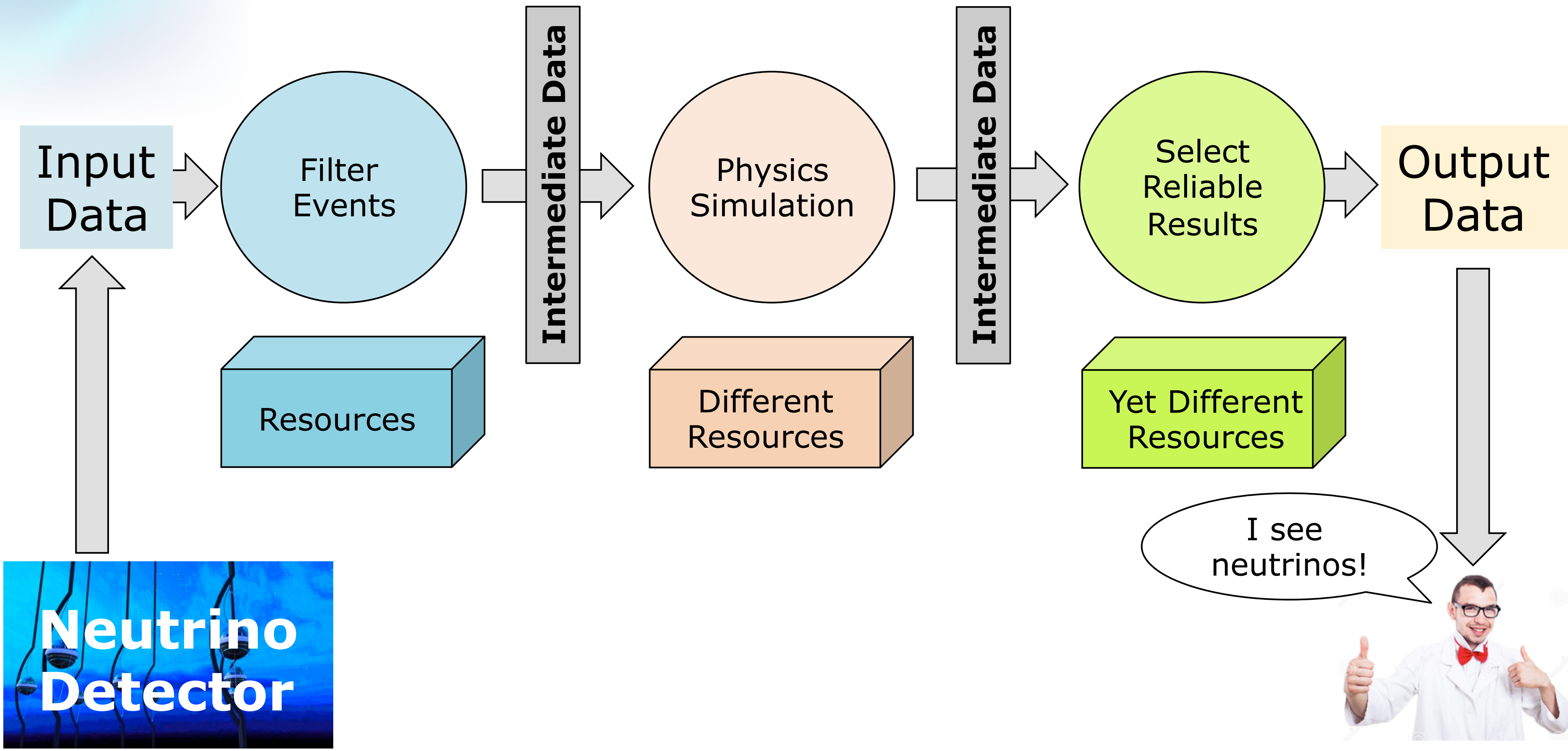
Stream job allocation

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

But..

How do scheduler deal with **Workflows**?

But before… What is a workflow?

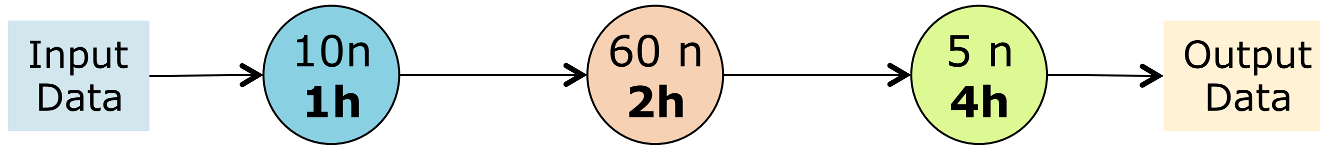


But before… What is the problem?

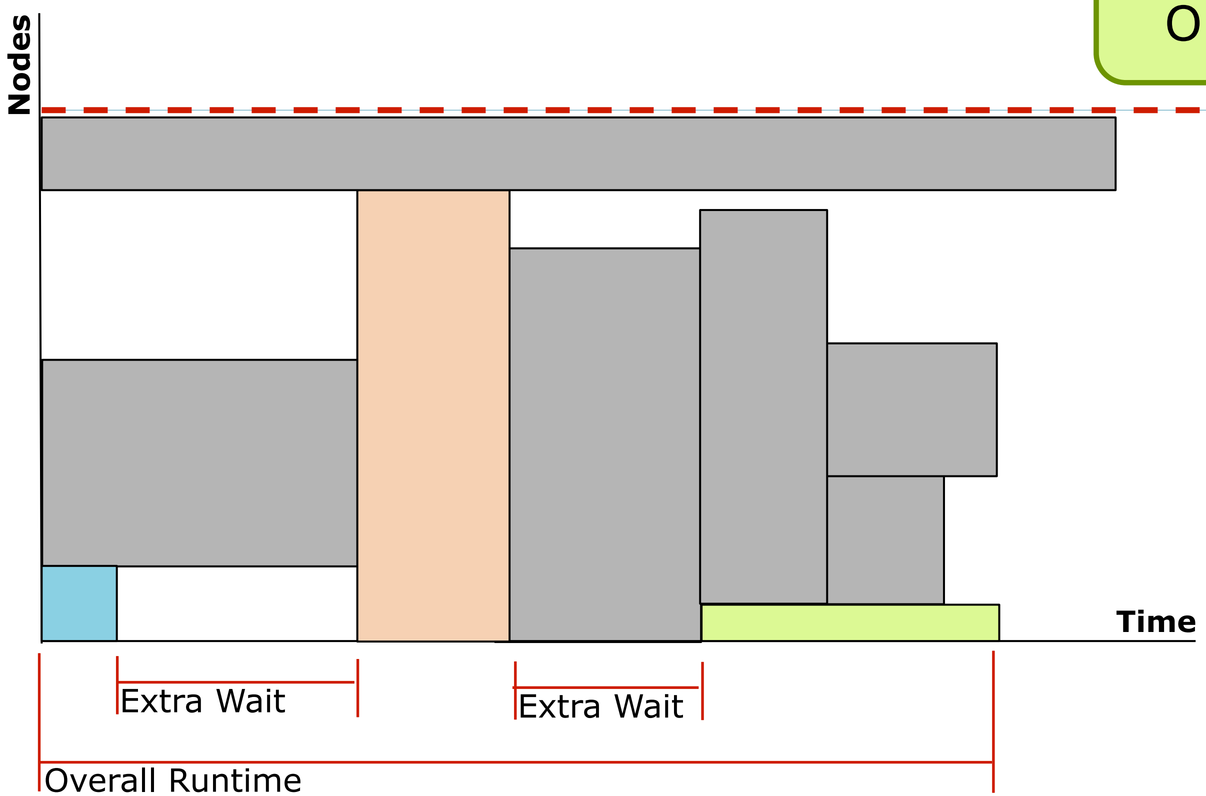
Classical schedulers are not optimized to manage workflows within the cluster.

Is that so bad?

Submitting a workflow: Wait! (approach)

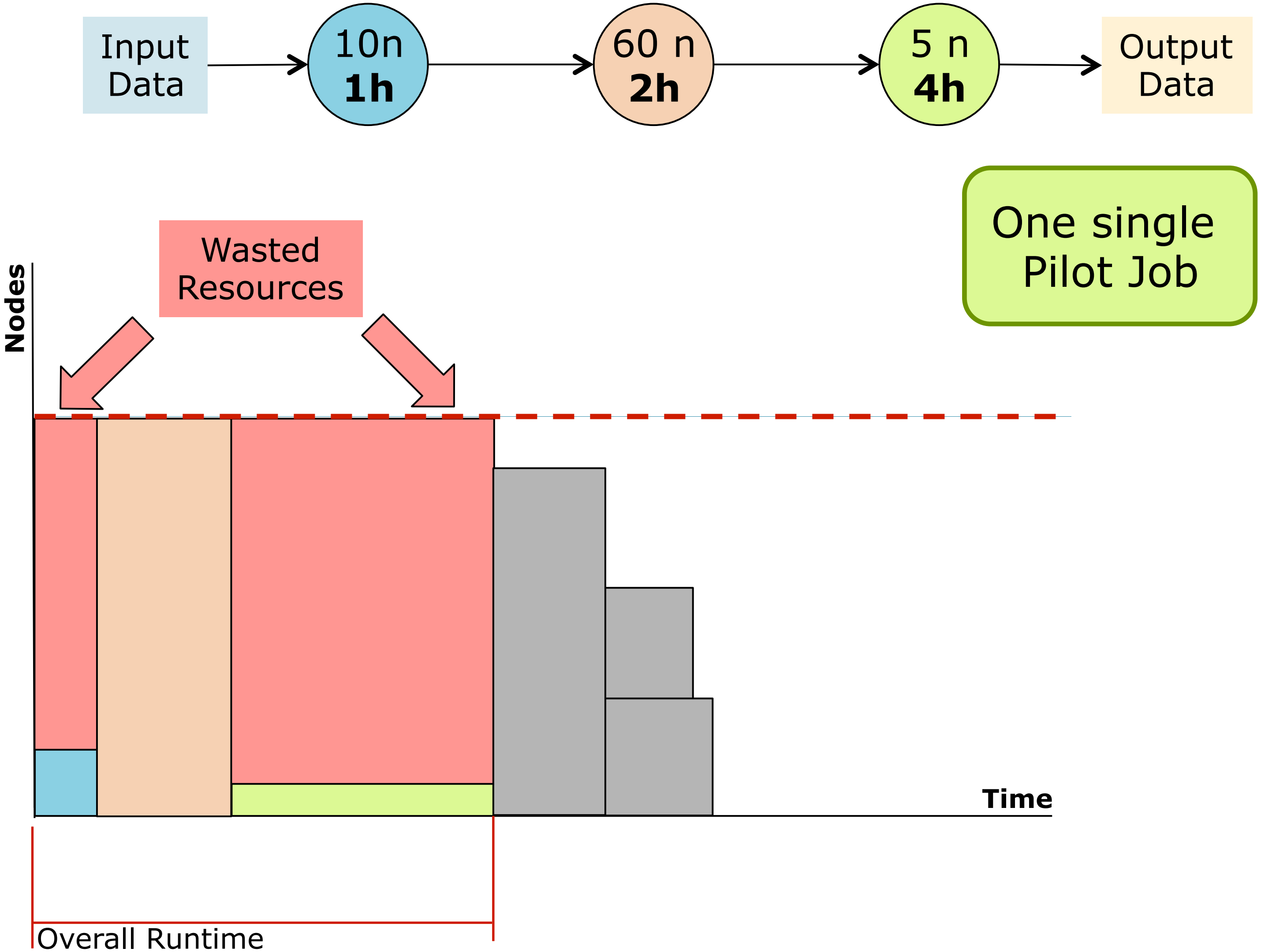


One stage
One Job



Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Submitting a workflow: Waste! (approach)



The reality of current workflow scheduling

Users either waste resources…
...or wait long time.

Something in between could be done!

(...To be published next fall)

A final note on scale and performance

Exascale = More parallel jobs

How many flops will need the scheduler alone?
A mini cluster to manage the cluster?

Distributed Scheduler

Multiple schedulers | Partitions | Smart RMs

New programming models?

Takeaways!

Workloads have **changed**

Observations of **job heterogeneity** possibly affecting schedulers performance

Alternate models of scheduling should be explored to address new challenges

Workflows are more important than ever: Scheduler should address them accordingly.

Big systems: more scheduling load… **performance**!

Thanks for time… questions?

Gonzalo P. Rodrigo – gonzalo@cs.umu.se

To know more….

Contact:
gonzalo@cs.umu.se - gprodrigoalvarez@lbl.gov

G. Rodrigo, P-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan. Towards Understanding Job Heterogeneity in HPC: A NERSC Case Study. CCGrid 2016 - The 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Accepted, 2016.

Rodrigo Álvarez, G. P., Östberg, P. O., Elmroth, E., Antypas, K., Gerber, R., & Ramakrishnan, L. (2015, June). HPC System Lifetime Story: Workload Characterization and Evolutionary Analyses on NERSC Systems. In Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing (pp. 57-60). ACM. Citation

Rodrigo Álvarez, G. P., Östberg, P. O., Elmroth, E., & Ramakrishnan, L. (2015, June). A2L2: An Application Aware Flexible HPC Scheduling Model for Low-Latency Allocation. In Proceedings of the 8th International Workshop on Virtualization Technologies in Distributed Computing (pp. 11-19). ACM. Citation

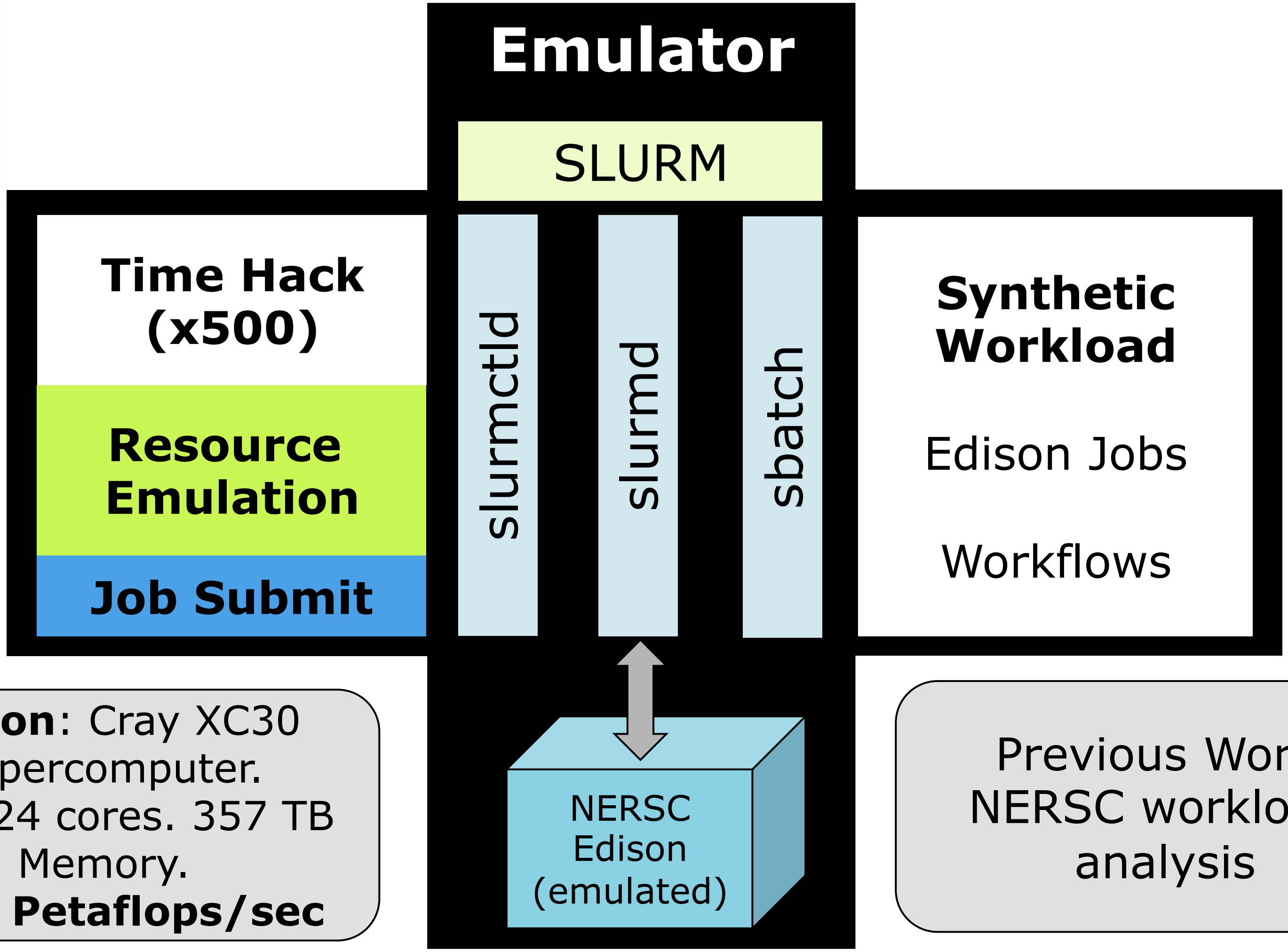
Rodrigo, G. P., Östberg, P-O. & Elmroth, E. (2014).Priority Operators for Fairshare Scheduling. 18th Workshops on Job Scheduling Strategies for Parallel Processing (JSSPP 2014) hosted at the IPDPS-2014 conference. Full Text

Rodrigo, G. P. Establishing the equivalence between operators: theorem to establish a sufficient condition for two operators to produce the same ordering in a Faishare prioritization system. January 2014. Full Text

Rodrigo, G. P. Proof of compliance for the relative operator on the proportional distribution of unused share in an ordering fairshare system. January 2014. Full Text

Slurm as a scheduling research platform

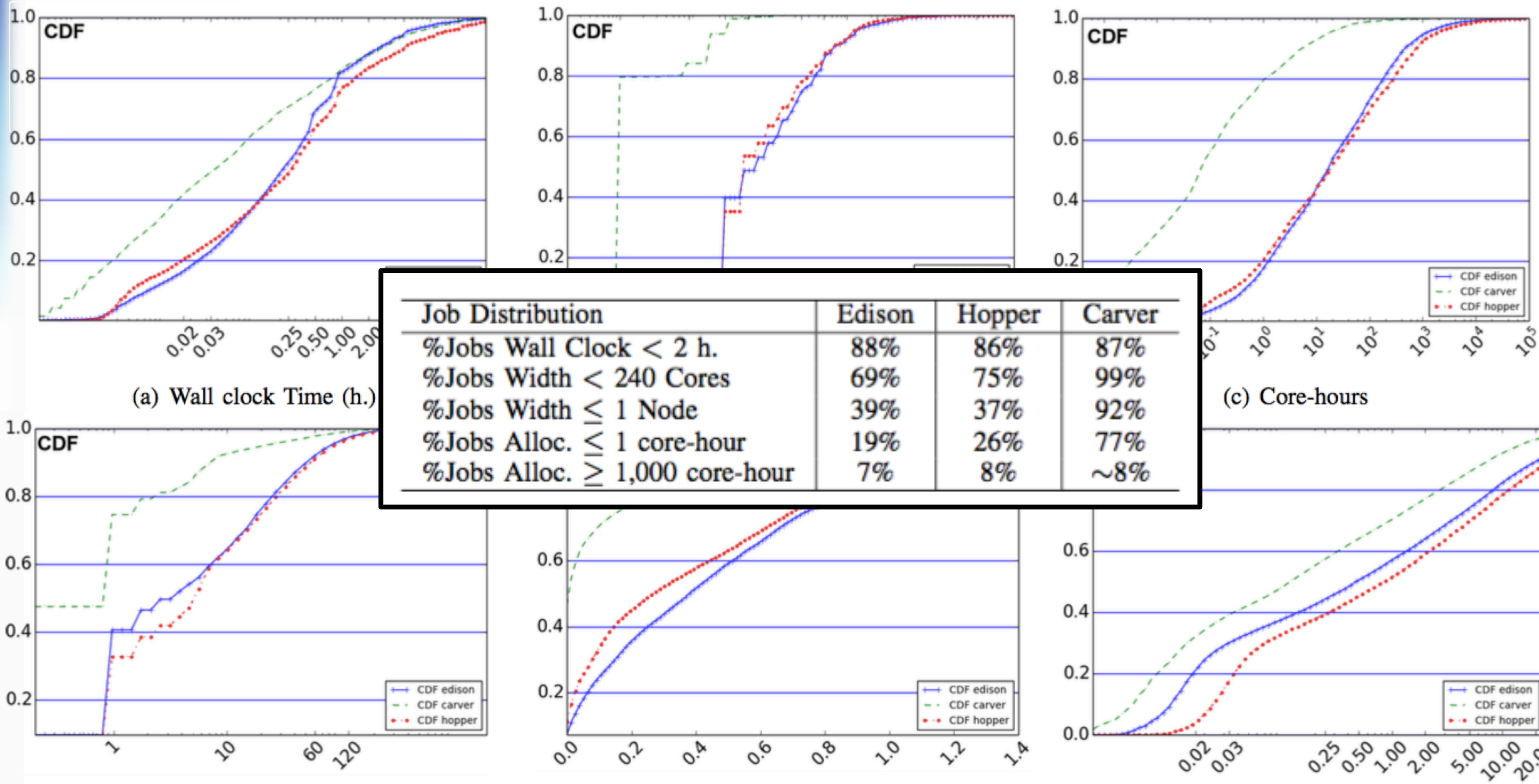
Emulator based work on Barcelona Supercomputing Center (BSC) and Swiss National Supercomputing Center (CSCS)... **but our own timing routines**



Edison: Cray XC30 Supercomputer. 133,824 cores. 357 TB Memory. **2.57 Petaflops/sec**

Previous Work NERSC workload analysis

Second step: Current Jobs



(a) Wall clock Time (h.)

(c) Core-hours

(a) Inter-arrival time (s.)

(b) Wall clock time accuracy

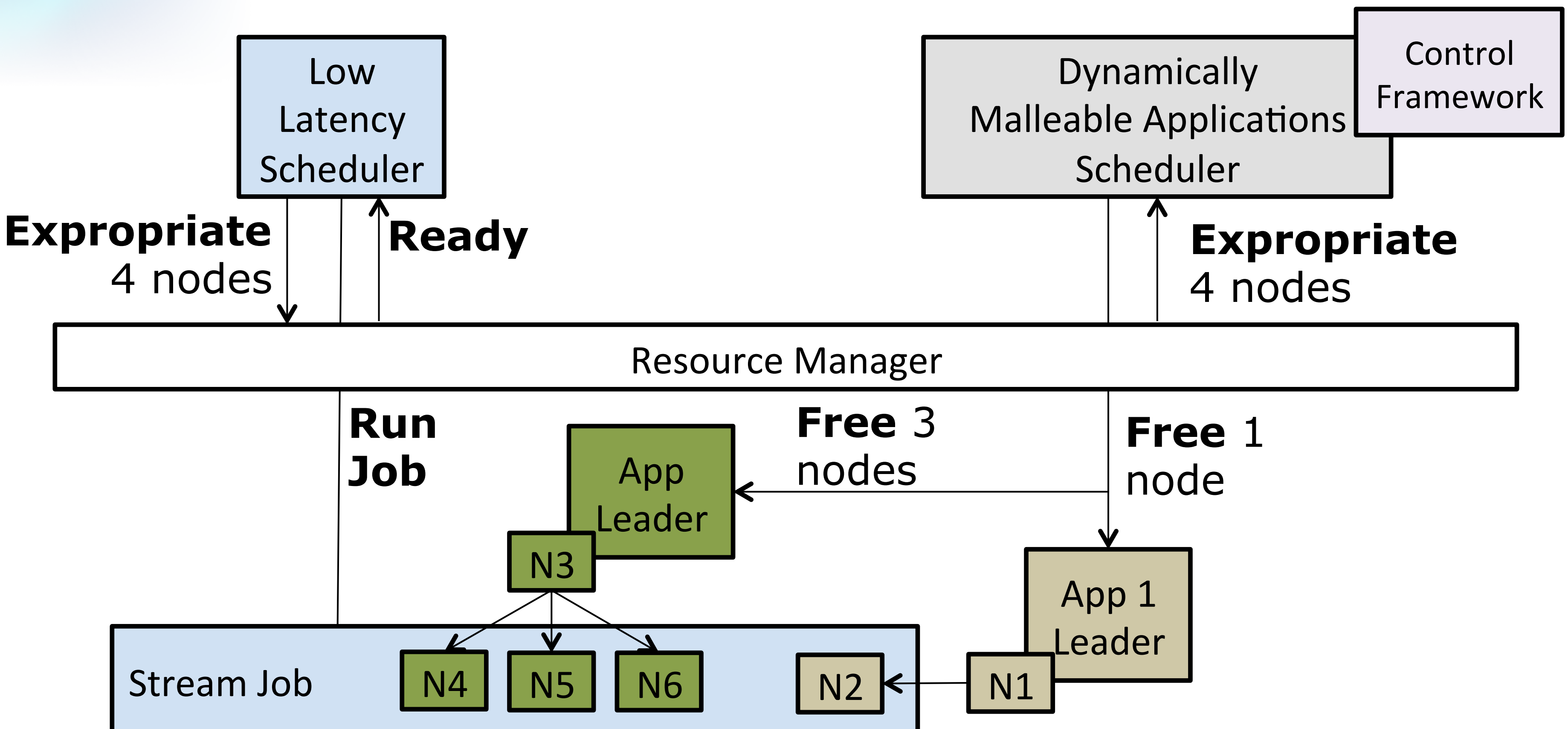
(c) Wait time (h.)

Job Distribution	Edison	Hopper	Carver
%Jobs Wall Clock < 2 h.	88%	86%	87%
%Jobs Width < 240 Cores	69%	75%	99%
%Jobs Width ≤ 1 Node	39%	37%	92%
%Jobs Alloc. ≤ 1 core-hour	19%	26%	77%
%Jobs Alloc. ≥ 1,000 core-hour	7%	8%	~8%

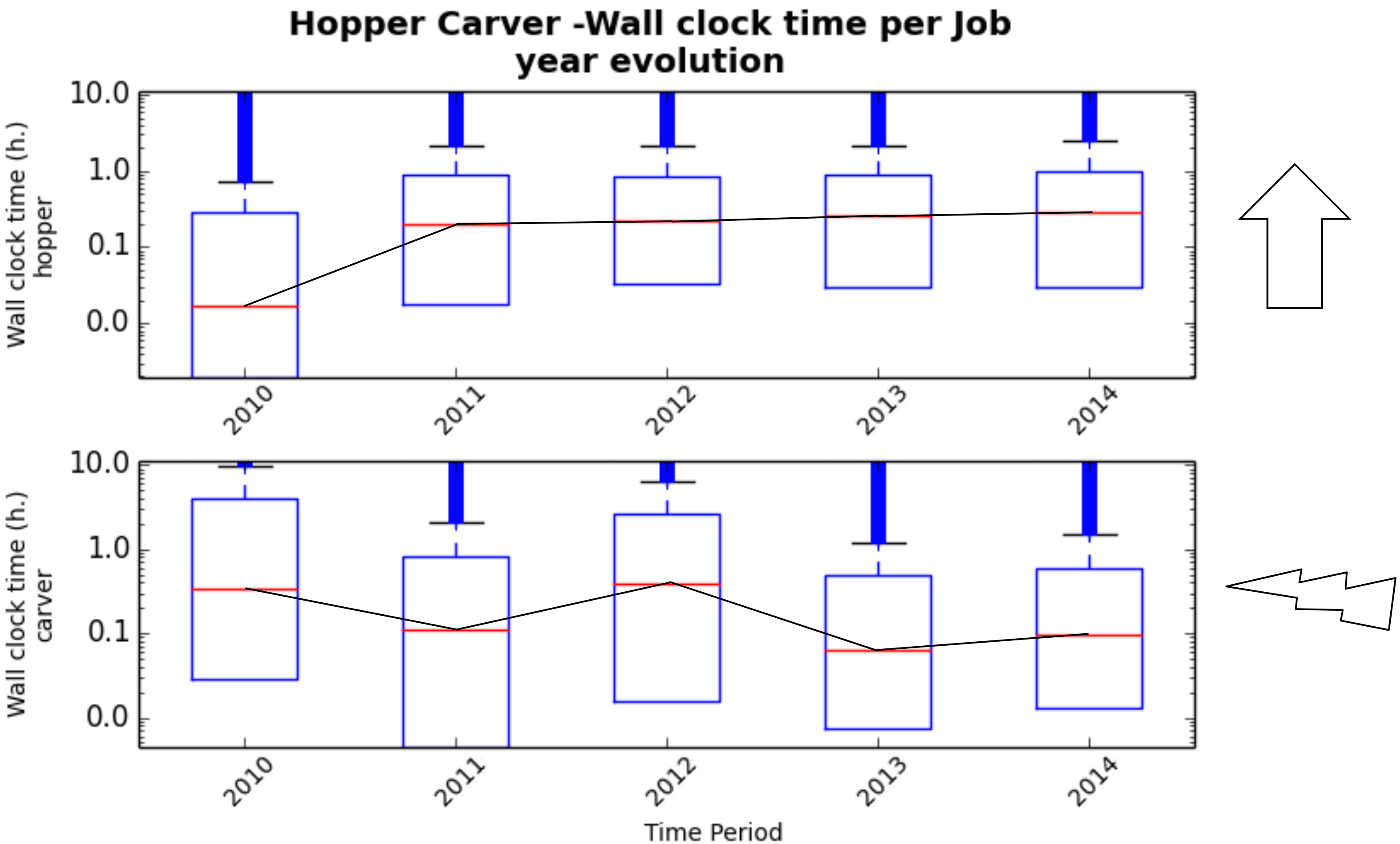
Resource Expropriation: Low latency allocation

Temporary "expropriation" of resources assigned assigned to dynamically malleable applications

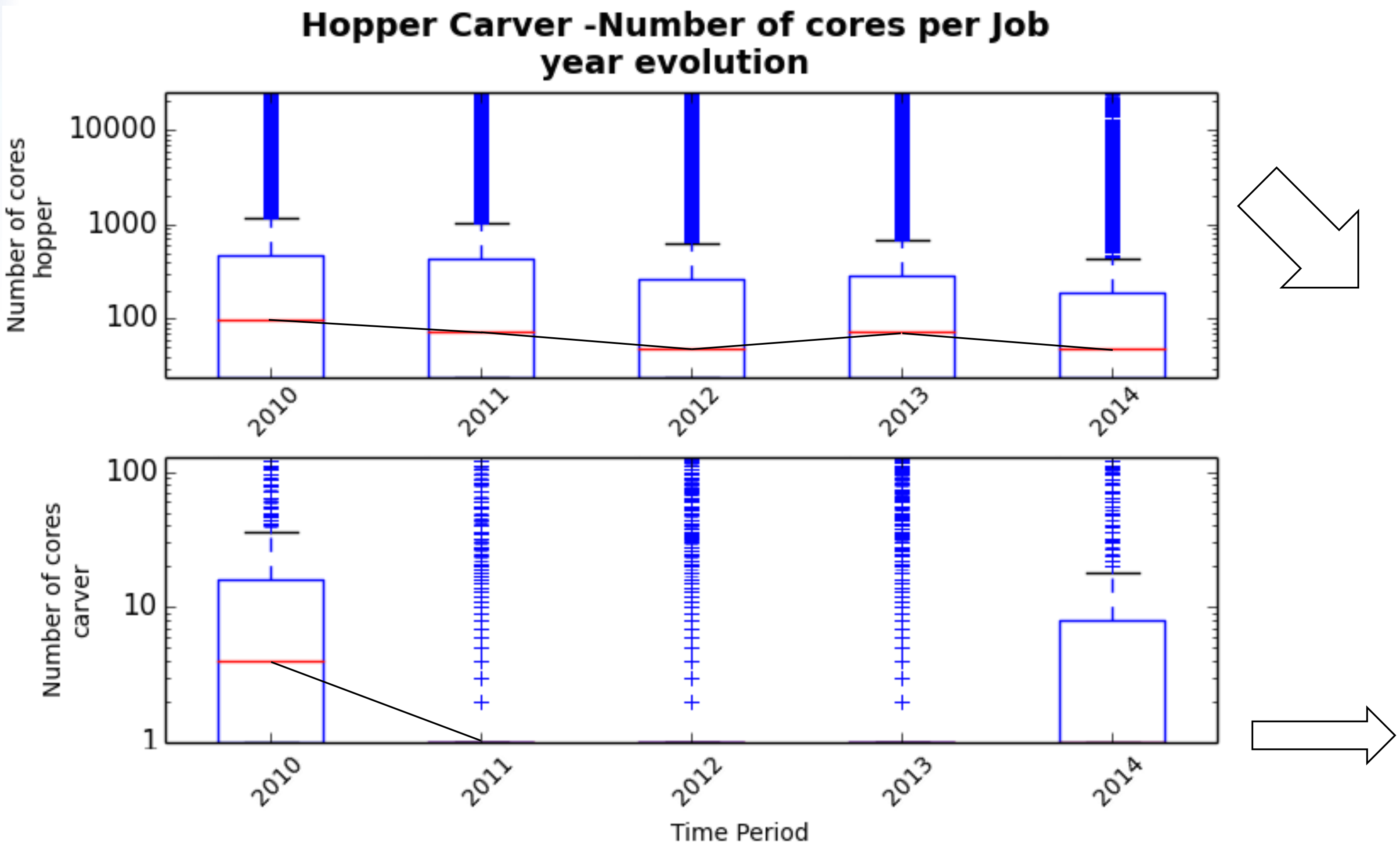
Expropriate and return actions



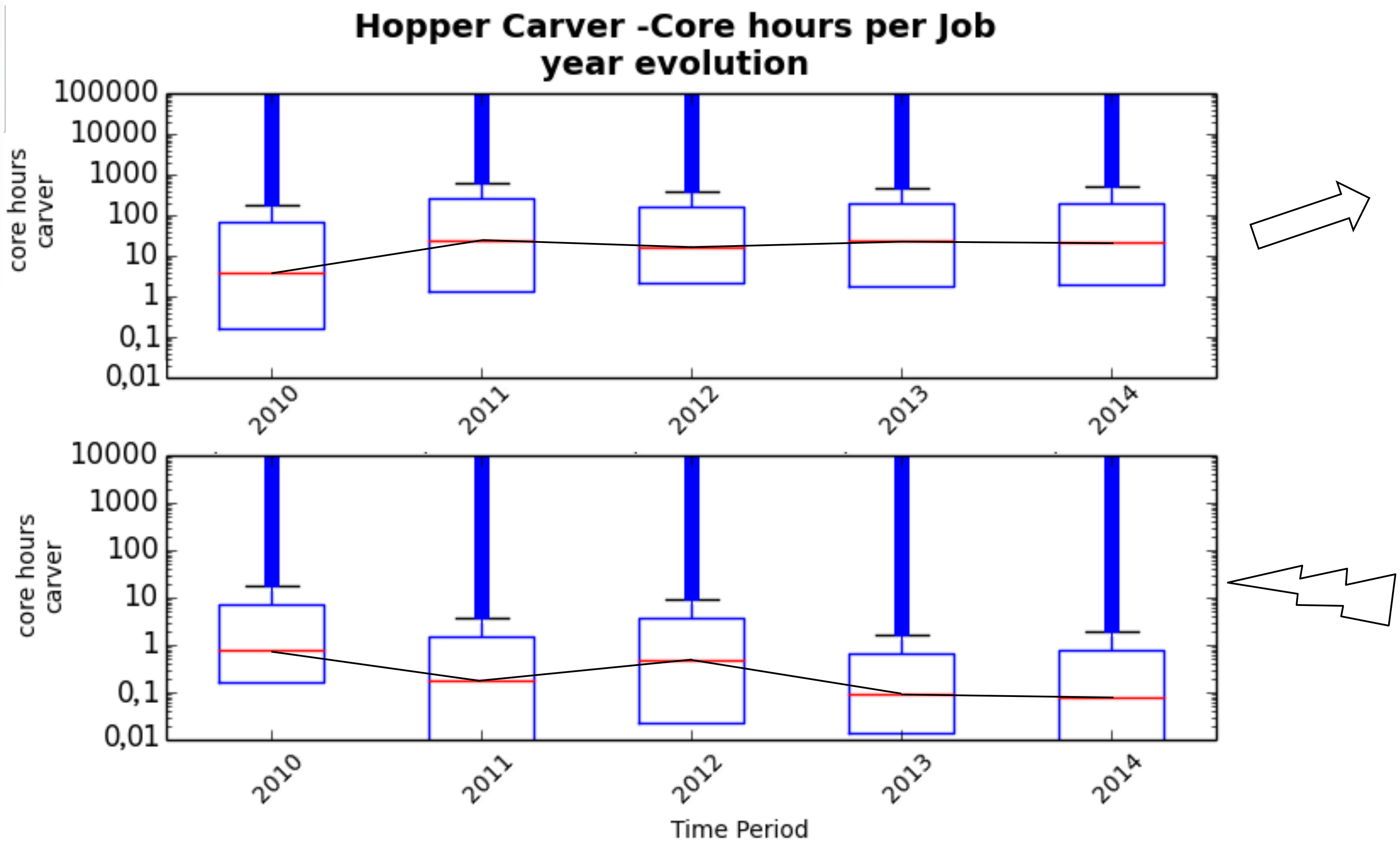
Wall clock time



Number of cores per job

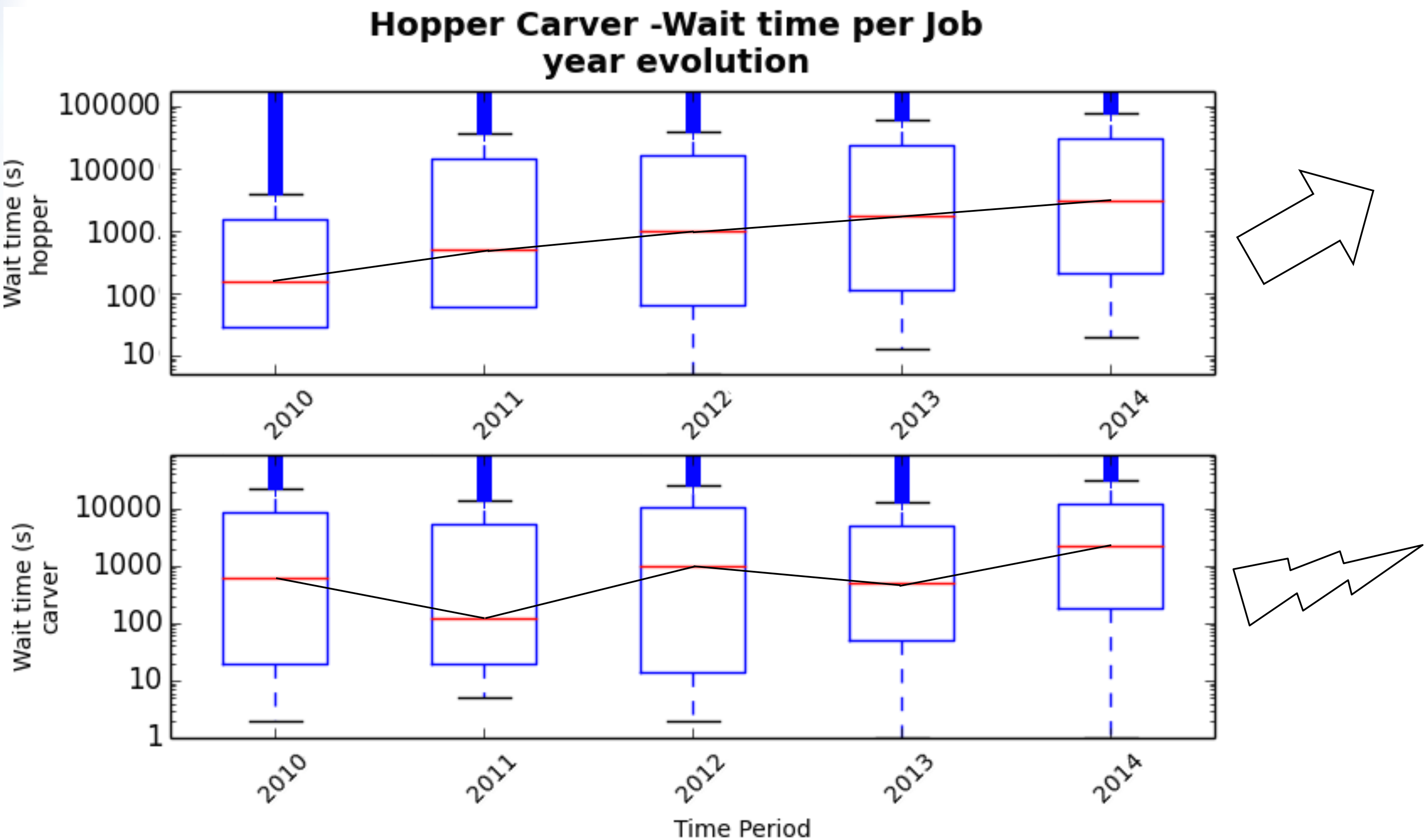


Core hours per job



Gonzalo P. Rodrigo – gonzalo@cs.umu.se

Wait time per job



Gonzalo P. Rodrigo – gonzalo@cs.umu.se