SFU SIMON FRASER UNIVERSITY
THINKING OF THE WORLD

Optimizing Shared Resource Contention in HPC Clusters

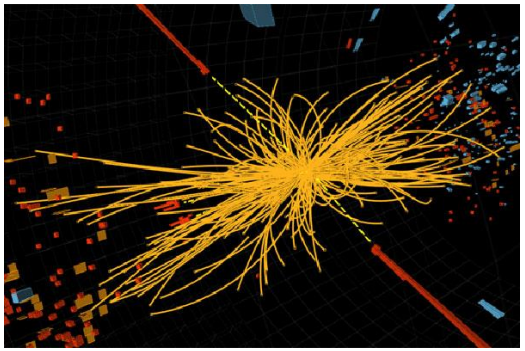
Electronic demo

Sergey Blagodurov

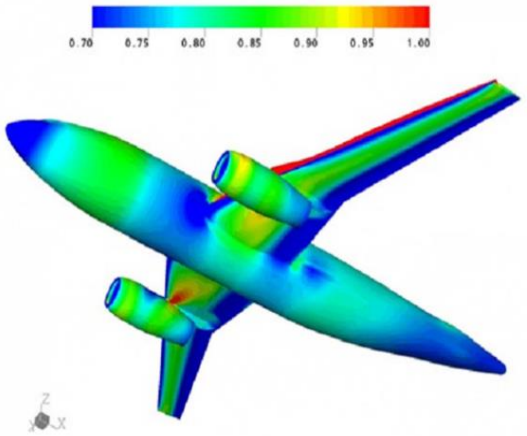
http://www.sfu.ca/~sba70/

SuperComputing
Fall 2013

Increasing demand for supercomputers



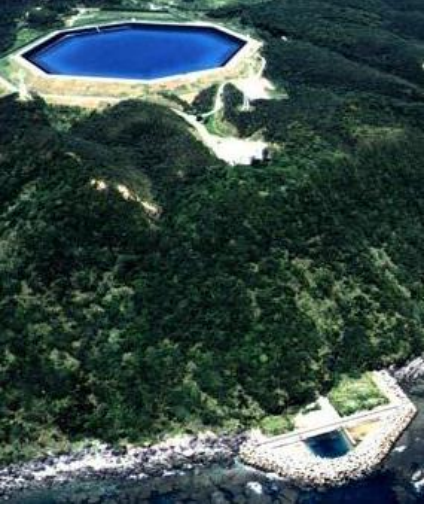
The biggest scientific discoveries



Tremendous cost savings

Medical innovations

Why datacenters are important?



Seawater hydro-electric storage on Okinawa, Japan

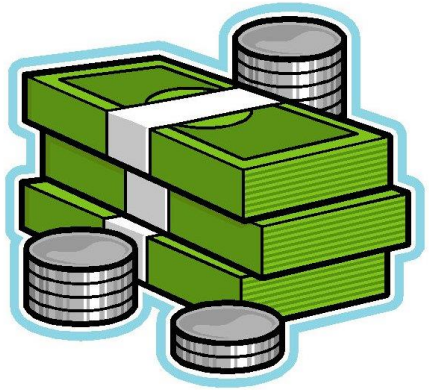
Datacenters use lots of energy:

- Consumption rose by 60% in the last five years
- More than the entire country of Mexico!
- now ~1-2% of world electricity

Datacenters consume lots of energy and its getting worse!

Typical electricity costs per year:

- Google (>500K servers, ~72MW): $38M
- Microsoft (>200K servers, ~68MW): $36M
- Sequoia (~100K nodes, 8MW): $7M



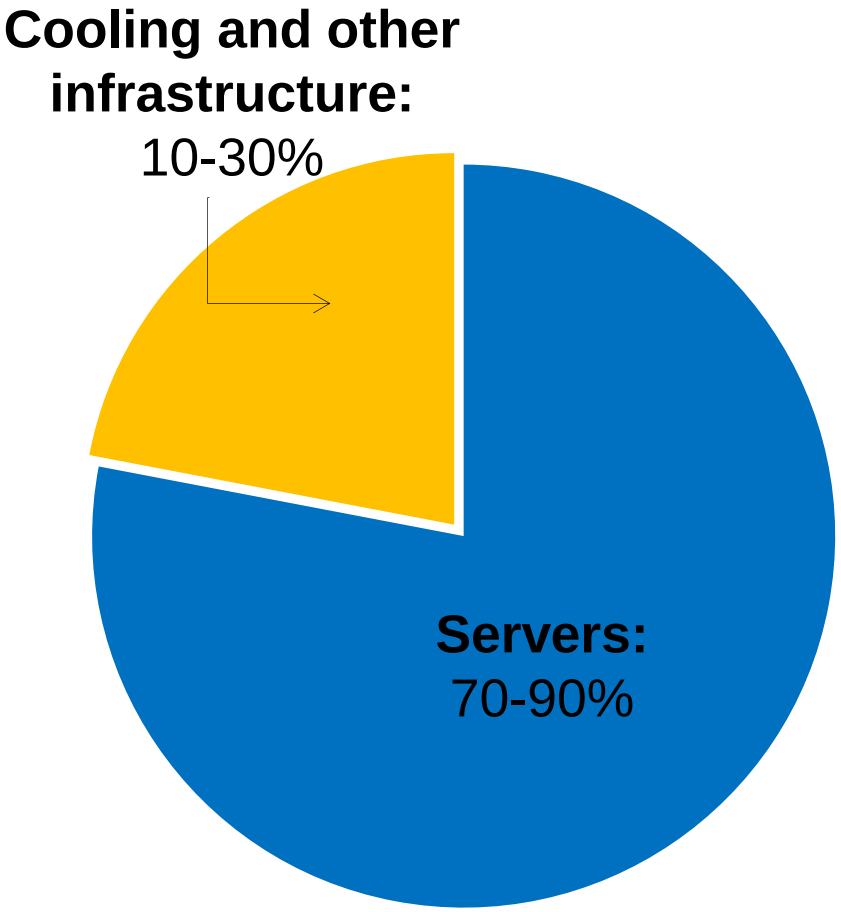
Why doing research in datacenters?

20 MW 24/7 datacenter that is on for 1 year is equivalent to:

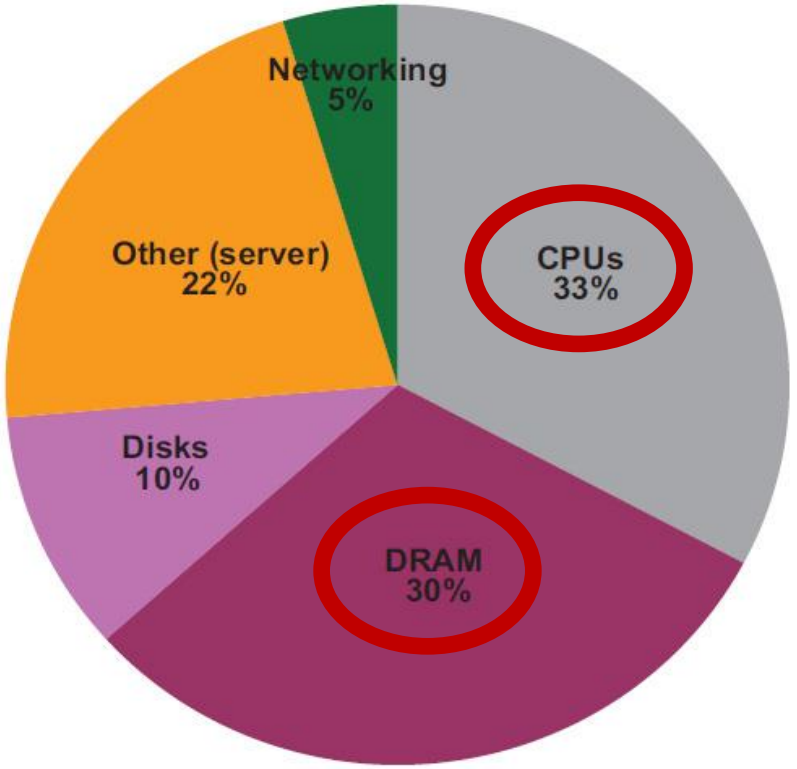
- 23k cars in annual greenhouse gas emissions
- CO_2 emissions from the electricity use of 15k homes for one year

A single datacenter generates as much greenhouse gas as a small city!

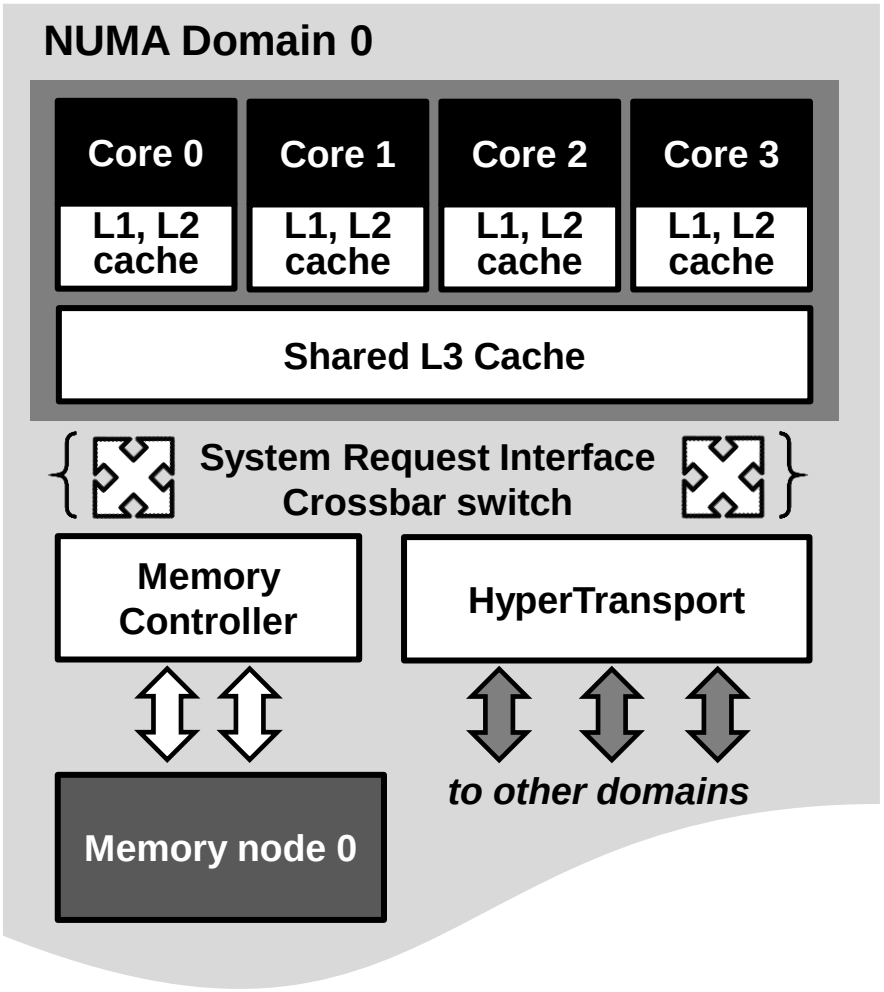
Why doing research in datacenters?



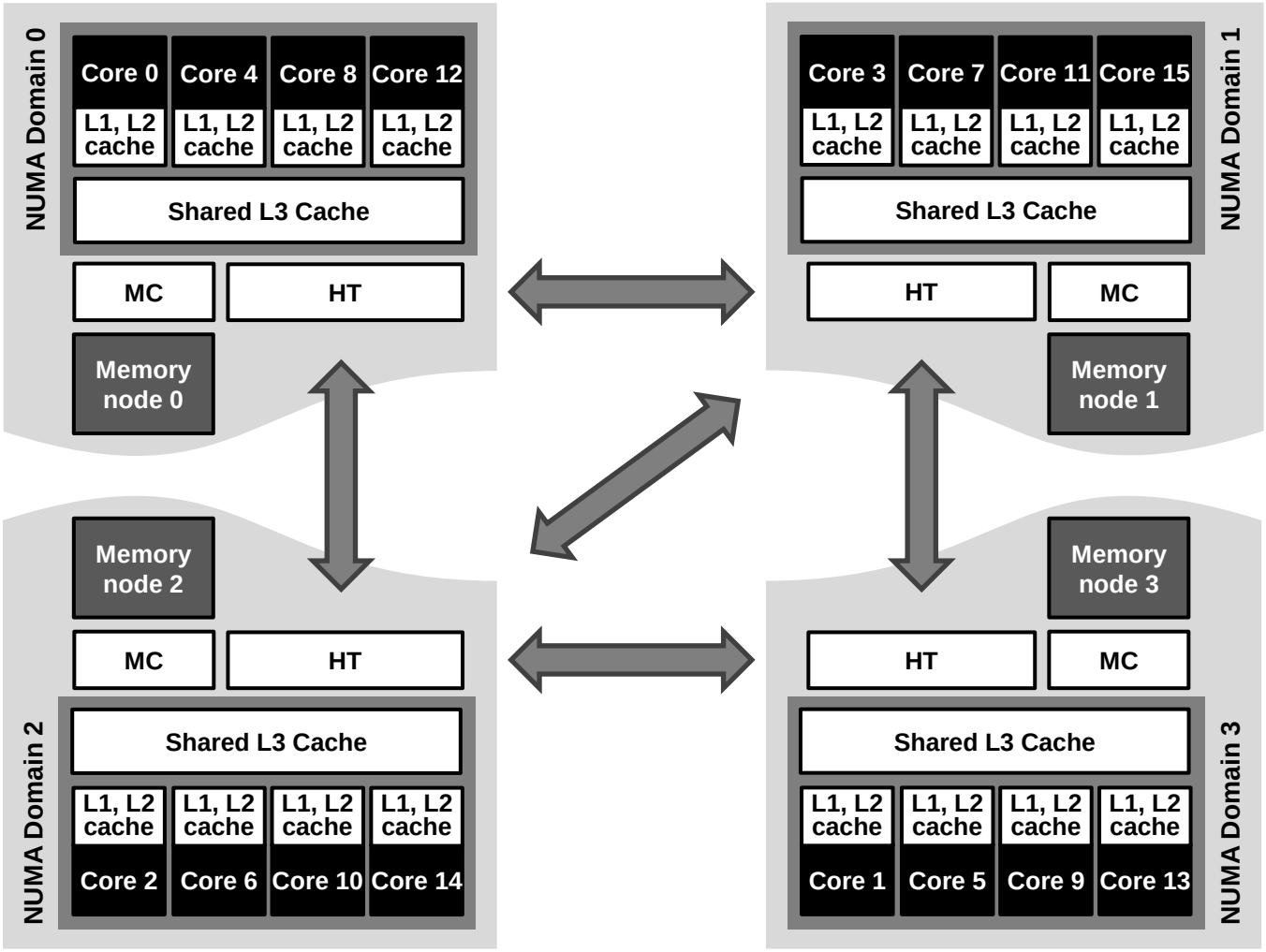
CPU and Memory are the biggest consumers



Where do datacenters spend energy?

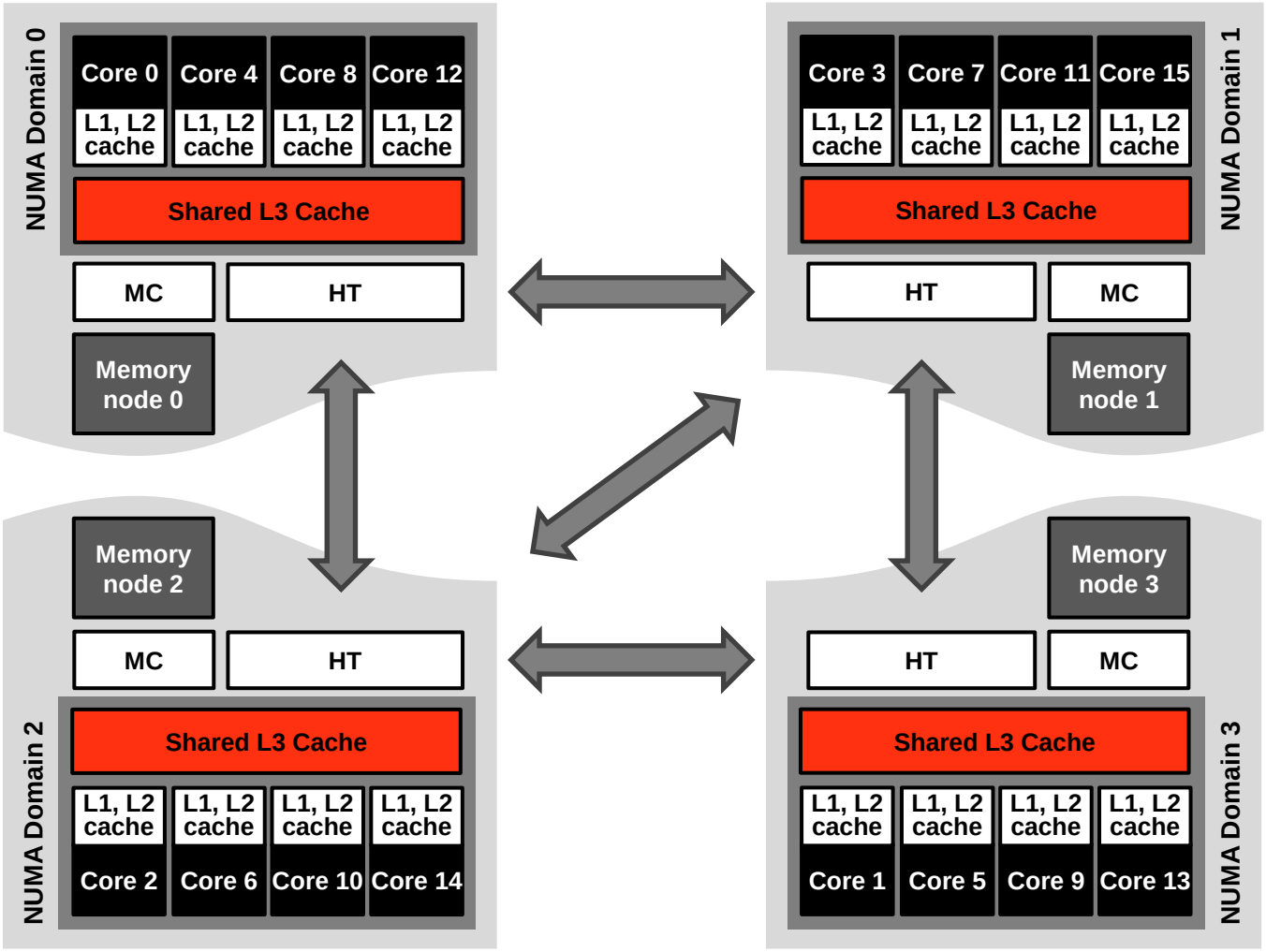


An AMD Opteron 8356 Barcelona domain

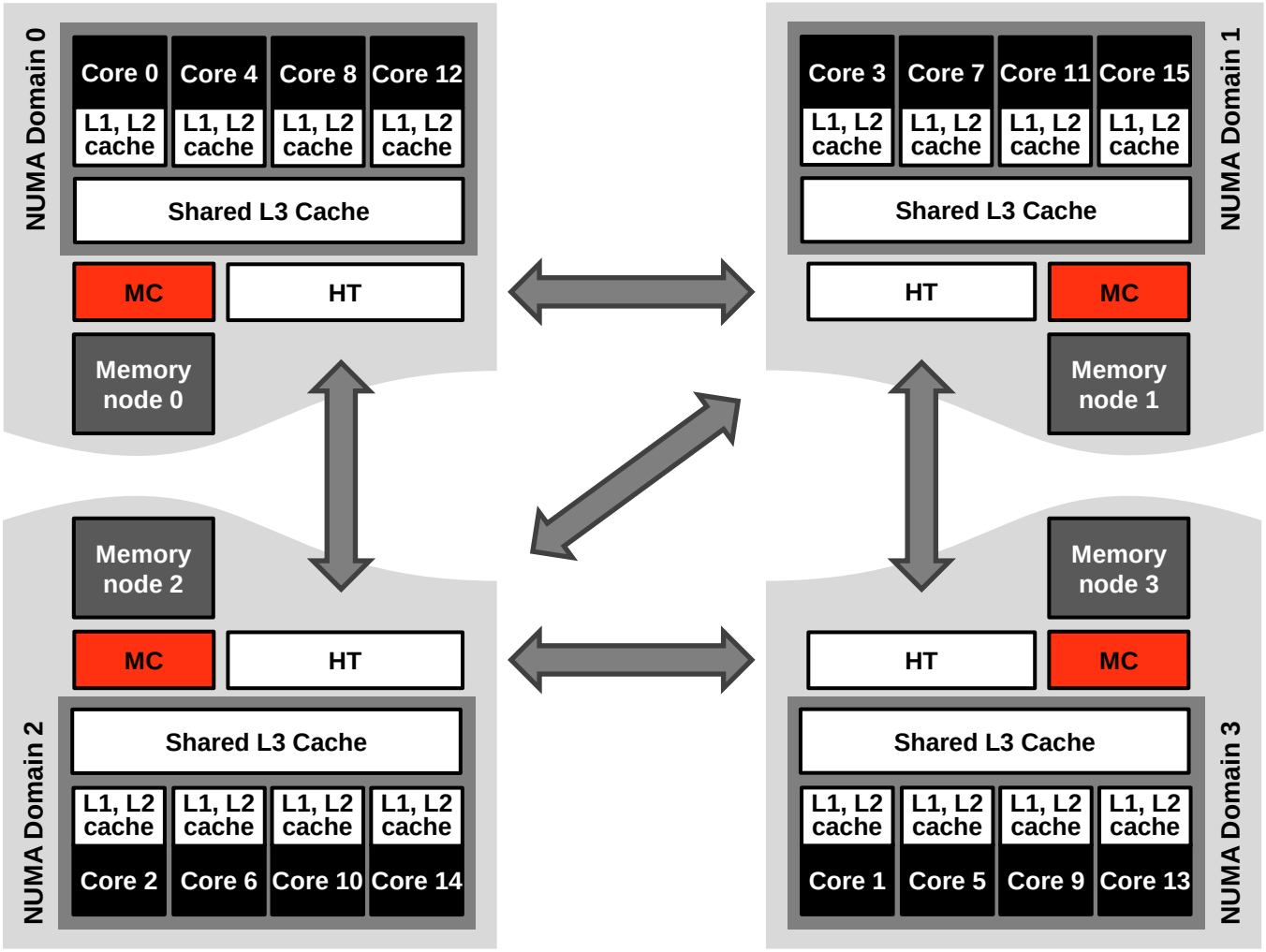


An AMD Opteron system with 4 domains

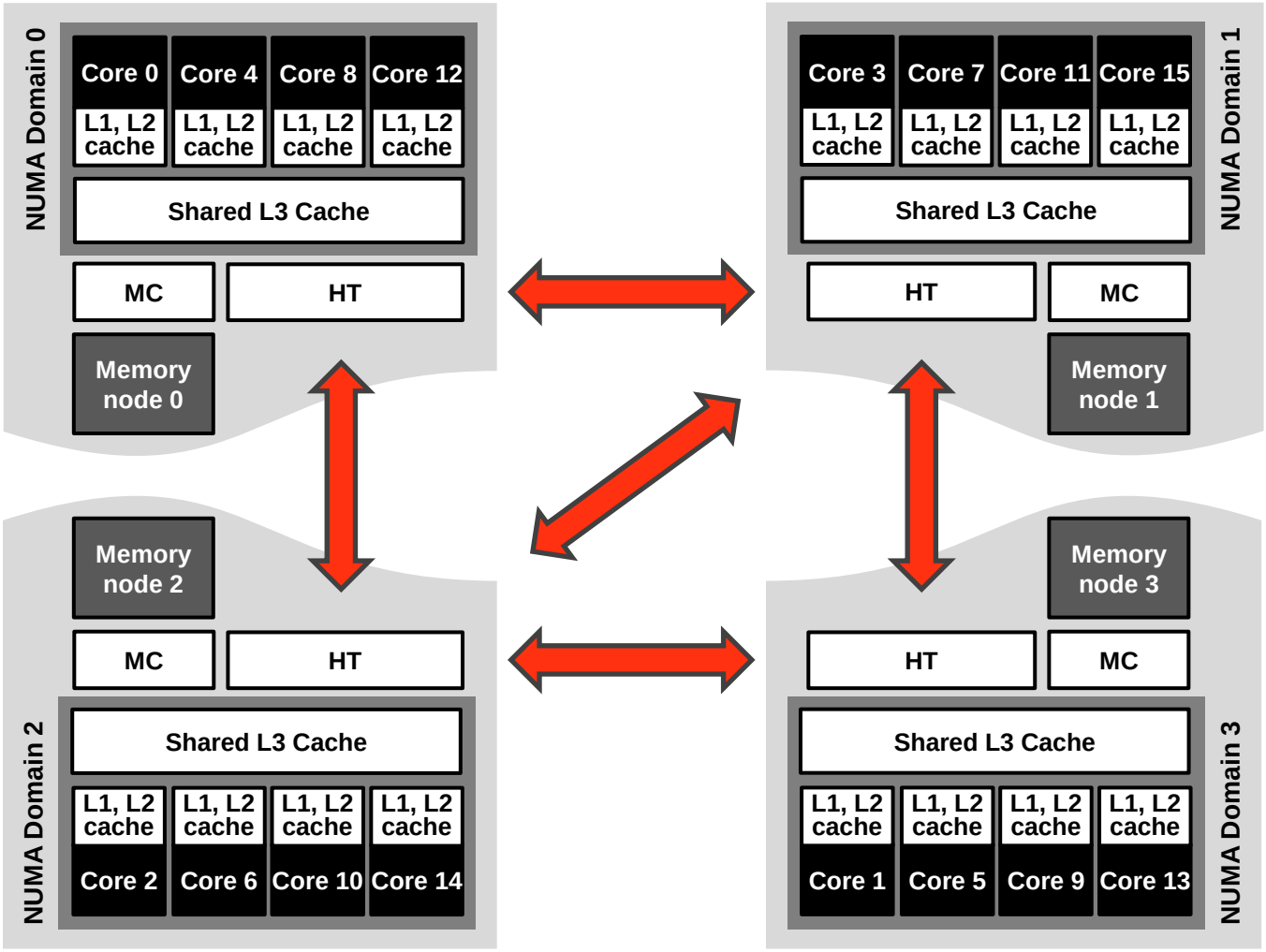
Talk by Sergey Blagodurov
SC'13 Electronic demo



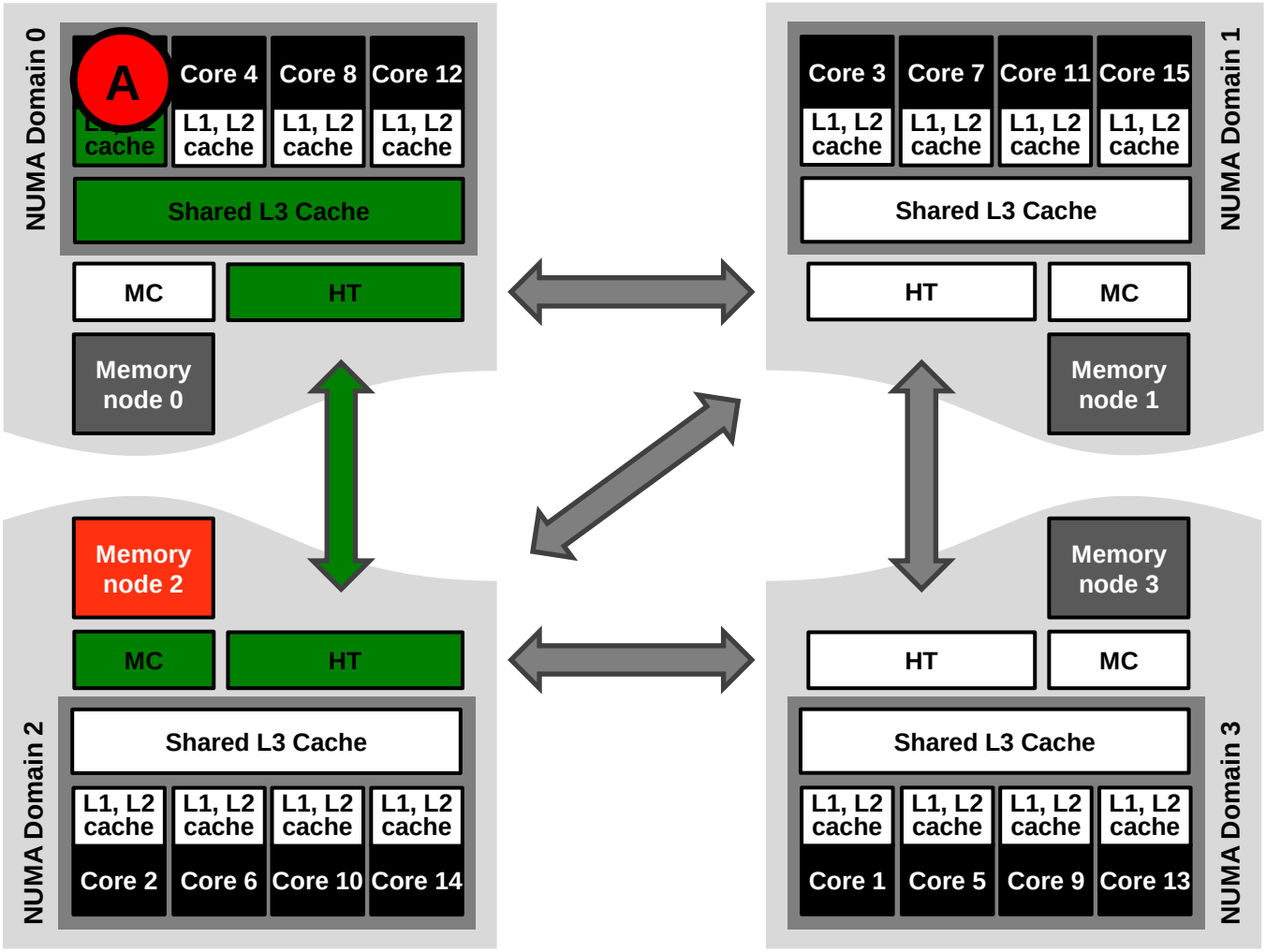
Contention for the shared last-level cache (CA)



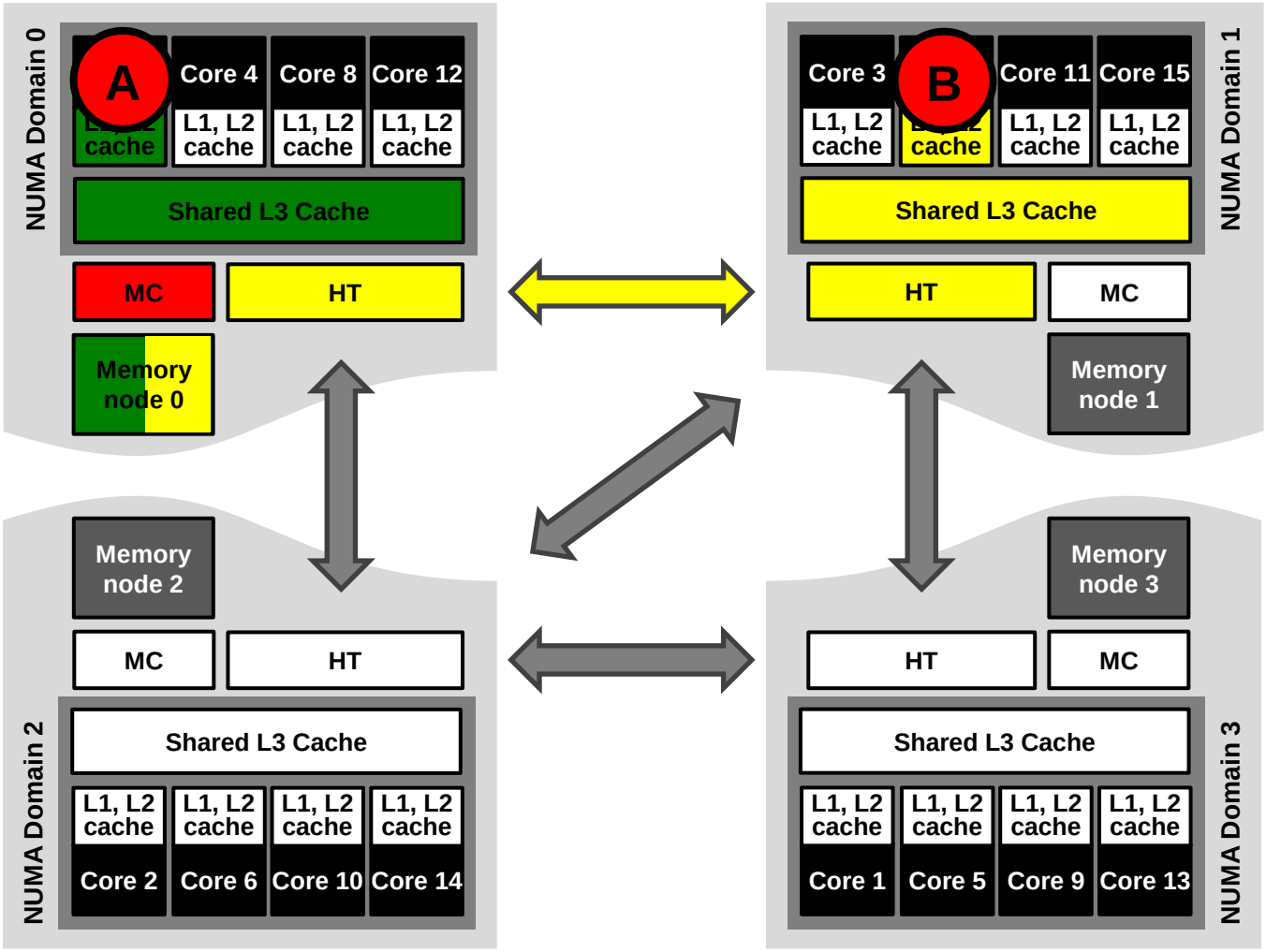
Contention for the memory controller (MC)



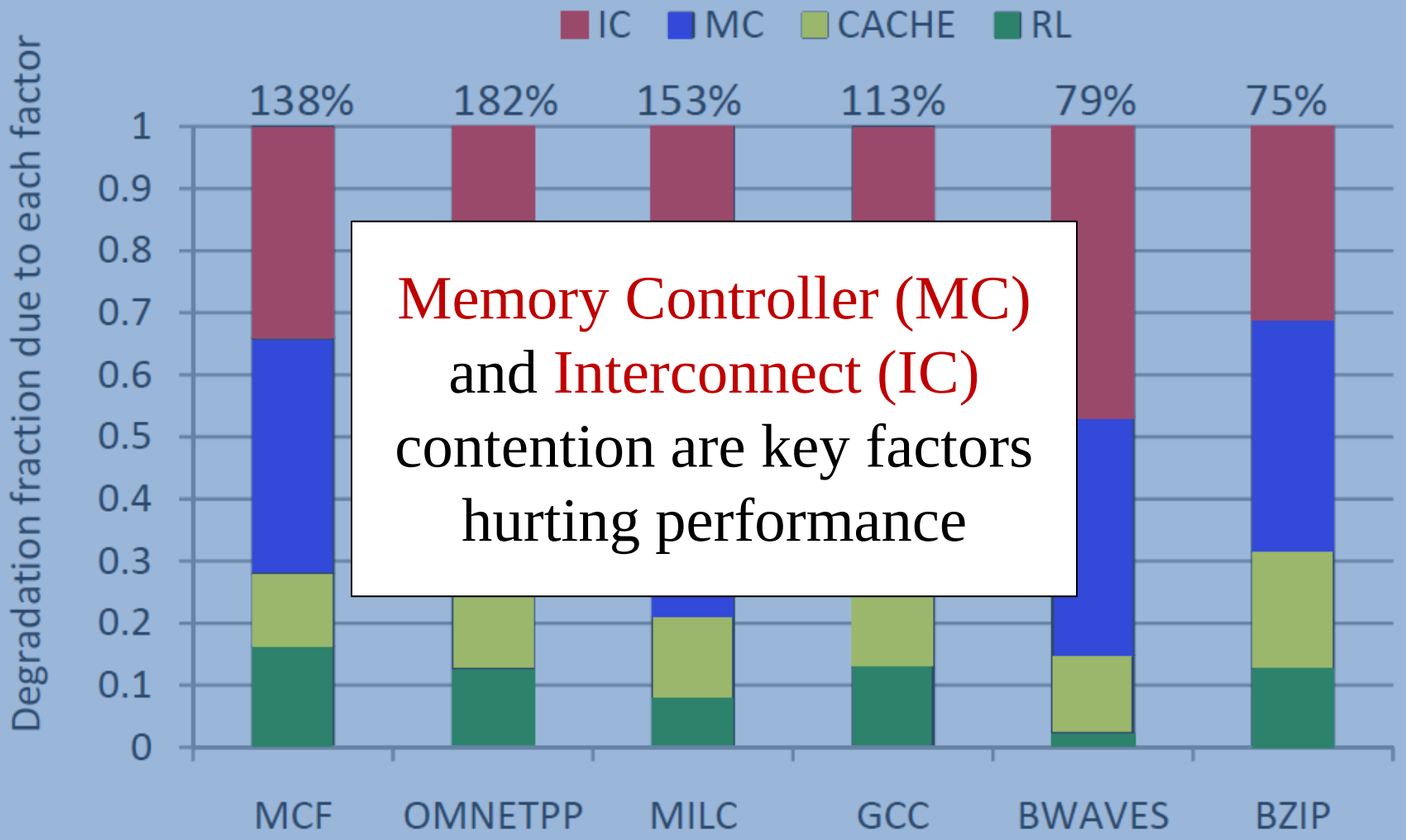
Contention for the inter-domain interconnect (IC)



Remote access latency (RL)



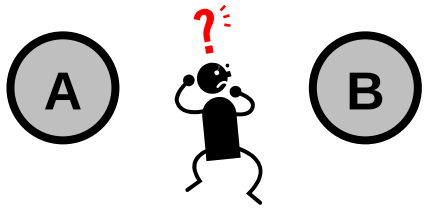
Isolating Memory controller contention (MC)



Dominant degradation factors

Characterization method

- Given two threads, decide if they will hurt each other's performance if co-scheduled



Scheduling algorithm

- Separate threads that are expected to interfere

Contention-Aware Scheduling

Limited observability

- We do not know for sure if threads compete and how severely!

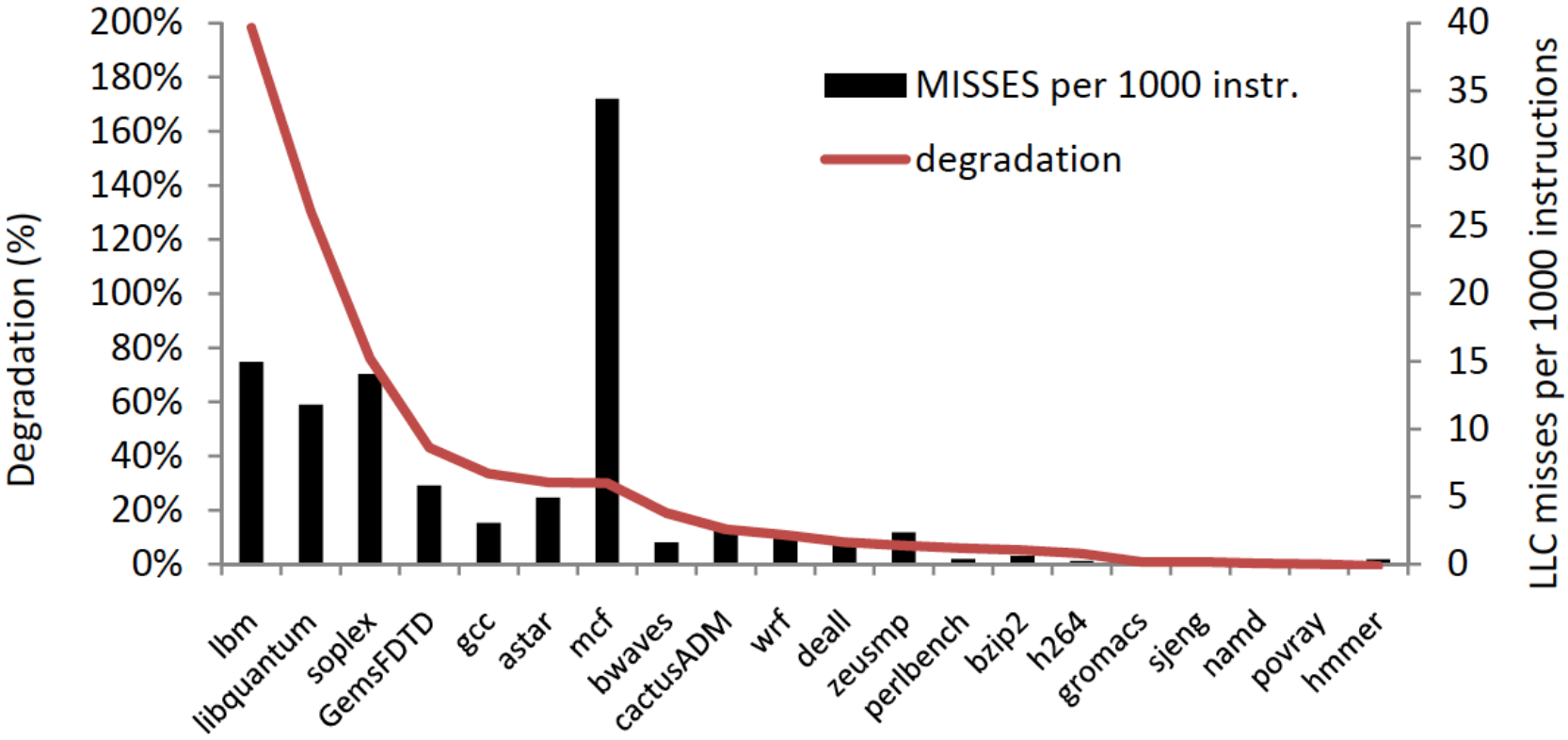
Trial and error infeasible on large systems

- Can't try all possible combinations
- Even sampling becomes difficult

A good trade-off: measure LLC Miss rate!

- Threads interfere if they have high miss rates
- No account for cache contention impact

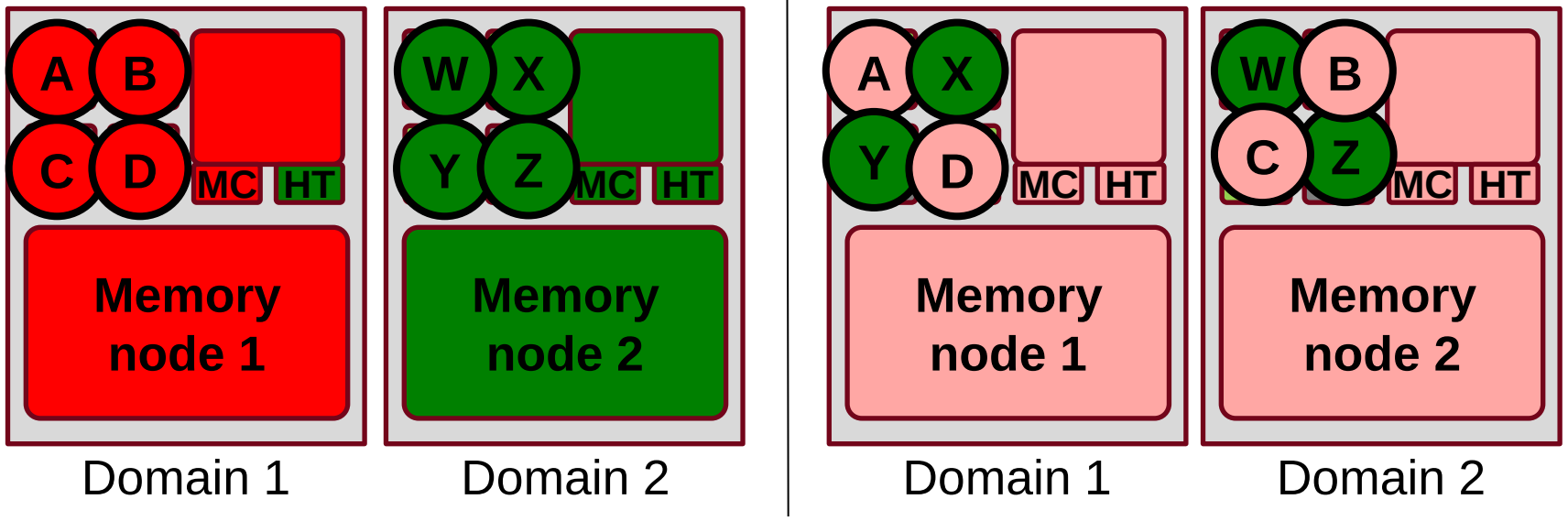
Characterization Method



Miss rate as a predictor for contention penalty

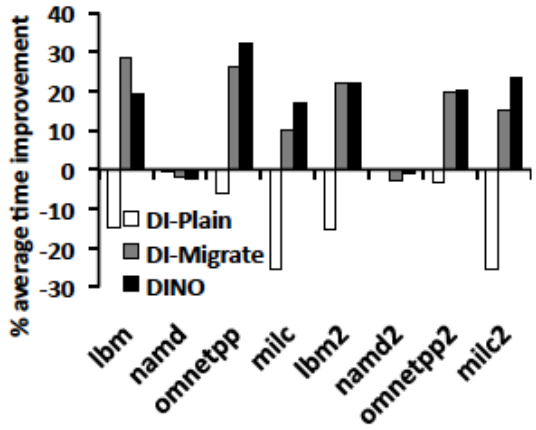
Sort threads by LLC missrate: A B C D W X Y Z

Goal: isolate threads that compete for shared resources
and pull the memory to the local node upon migration

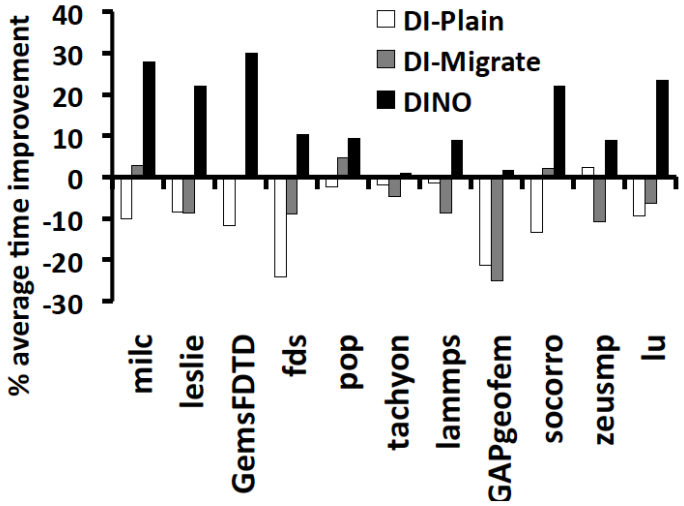


Migrate competing threads along with memory to different domains

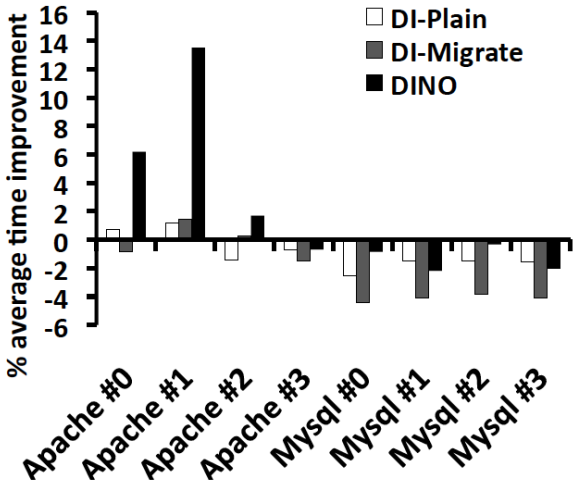
Server-level scheduling



SPEC CPU 2006



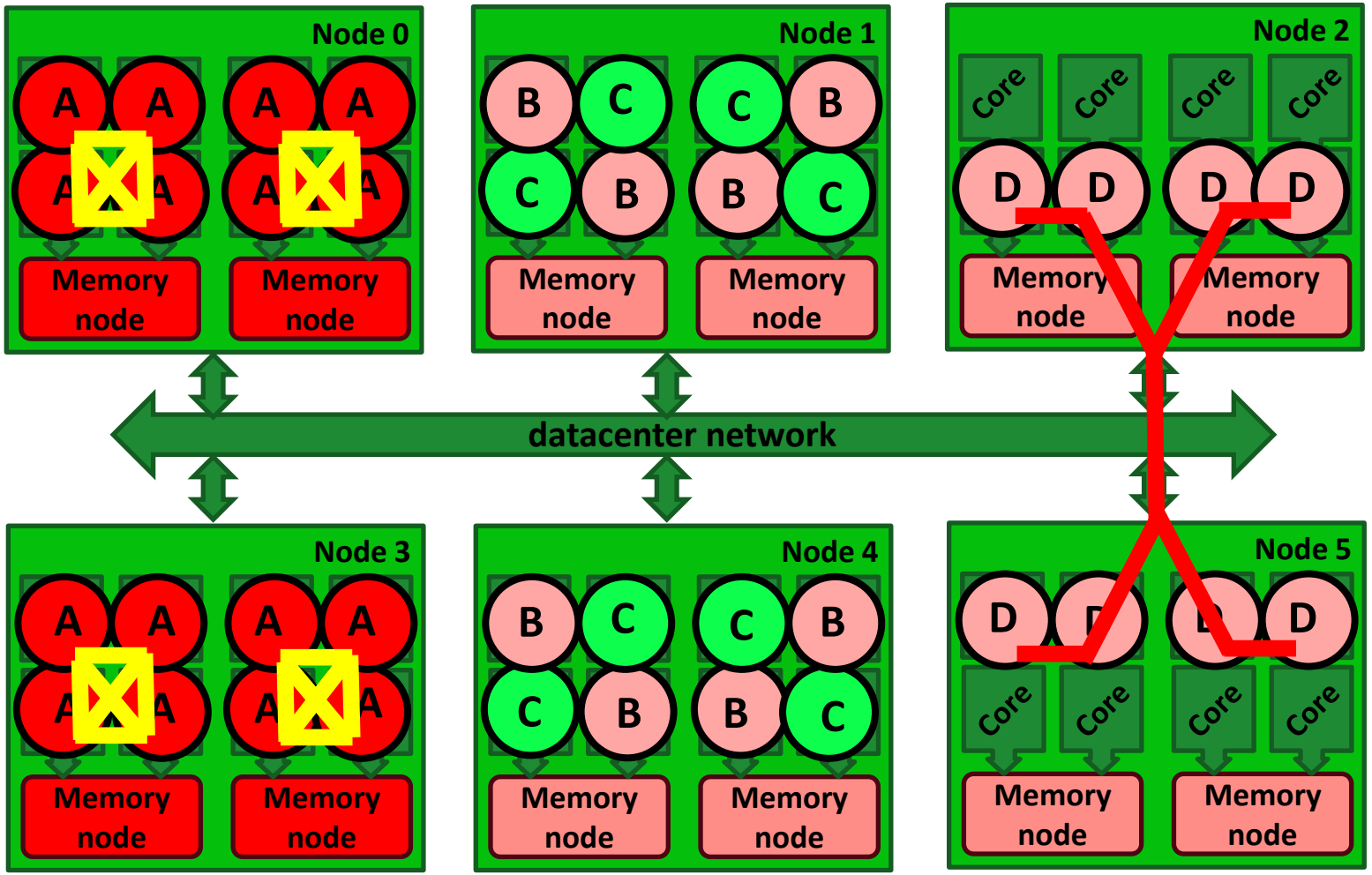
SPEC MPI 2007



LAMP

Server-level results

Talk by Sergey Blagodurov
SC'13 Electronic demo



Possibilities of datacenter-wide scheduling

Contention-aware cluster scheduling:

See: **We monitor job processes on-the-fly and classify them with 2 parameters:**

a) a process is a *devil* if it is memory intensive, has high last-level cache missrate, otherwise - a *turtle*.

b) if a given process is communicating with other processes.

Think: **We develop a multi-objective scheduling algorithm Clavis-Cluster that simultaneously:**

a) minimizes the number of devils on each node;

b) maximizes the number of communicating processes on each node;

c) minimizes the number of powered up nodes in the cluster.

***Do:* After the new schedule is found, we enforce it by introducing a low-overhead live migration into cluster:**

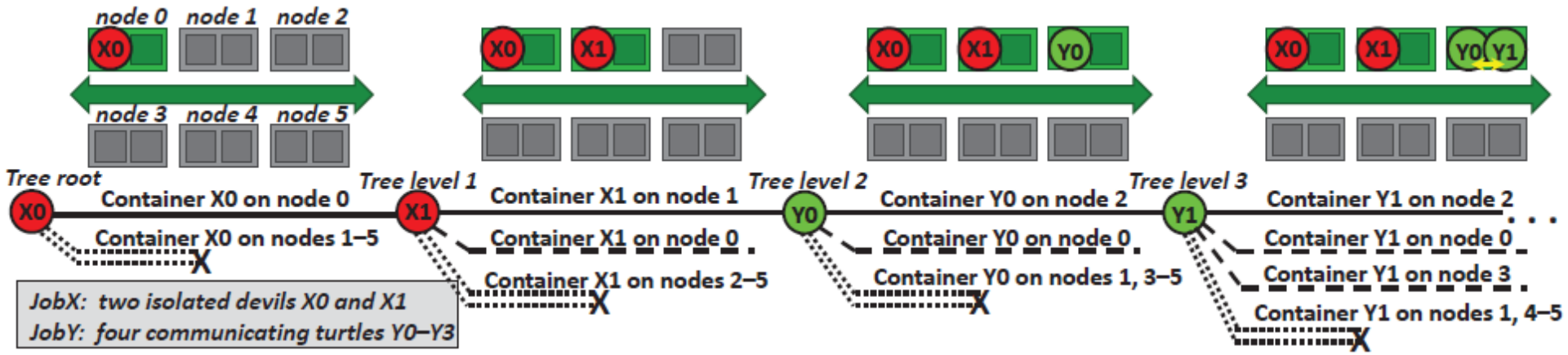
the job scheduler places processes into OpenVZ containers, Clavis-Cluster migrates containers.

Finding an optimal schedule:

- an implementation using Choco solver
- minimizes weighted sum:

$$F = \alpha \times D + \beta \times P - \gamma \times C$$

Branch-and-Bound enumeration search tree:

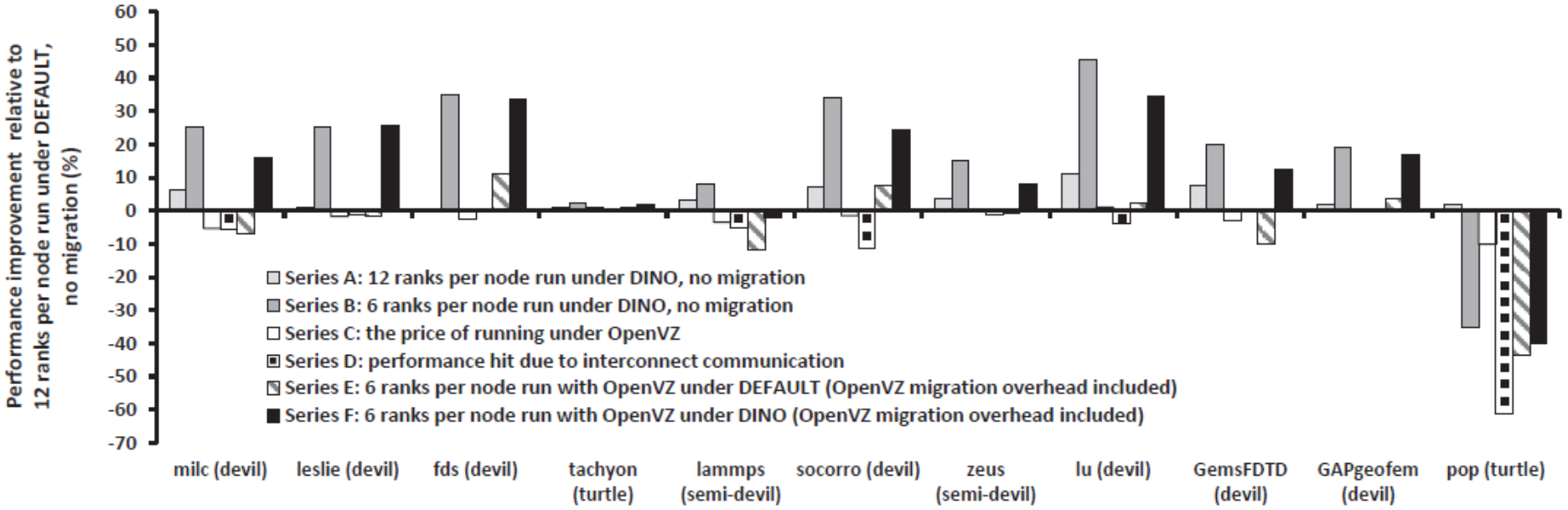


Enumeration tree search

Solver evaluation
(custom branching strategy)

Row	Features enabled:							Performance (up to 128 containers per job):			
	Best Random	Custom branching?	Naïve bounding function?	Elaborate bounding function?	Elaborate fathoming?	Prune top % branches?	Time to solve, sec	Weighted sum relative to the best one	Average tree nodes evaluated (solutions tried for random)	Average fraction of tree nodes pruned	Average fraction of tree levels explored
0	yes						60	3.14x	505078		
1		no	no	no	no	no	60	5.18x	338789	0.00	0.00
2		yes	no	no	no	no	60	1.29x	663692	0.00	0.01
3		yes	yes			no	60	1.29x	3600690	0.41	0.04
4		yes		yes	no	no	60	1.23x	2267173	0.81	0.67
5		yes		yes	yes	no	60	1x	376533	0.80	0.93
6		yes		yes	yes	yes	60	1.03x	58055	0.88	0.96
7		yes		yes	yes	yes	600	0.99x	6934349	0.89	0.97
8		yes		yes	yes	yes	6000	0.99x	71722876	0.90	0.97

Price of running under OpenVZ

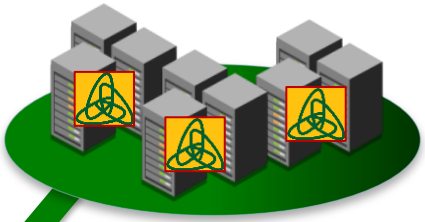


Clavis-HPC virtualization overhead

Clients (tablet, laptop, desktop, etc)

1). User connects to the HPC cluster via client and submits a job with a PBS script. The user can characterize the job with a contention metric (devil, comm-devil).

2). Resource Manager (RM) on the head node receives the submission request and passes it to the Job Scheduler (JS).
3). JS determines what jobs execute on what containers and passes the scheduling decision to RM.
4). RM starts/stops the jobs on the given containers.
6). RM generates a contention-aware report about resource usage in the cluster during the last scheduling interval.
10). RM passes the contention-aware resource usage report to JS.



Compute nodes
contention monitors (Clavis)
OpenVZ containers
libraries (OpenMPI, etc)
RM daemons (pbs_mom)

5). The virtualized jobs execute on the containers under the contention aware user-level scheduler (Clavis-DINO). They access cluster storage to get their input files and store the results.

Head node
RM, JS, Clavis-HPC

Cluster network
(Ethernet, InfiniBand)

Monitoring (JS GUI), control (IPMI, iLO3, etc)

7). Users or sysadmins analyze the contention-aware resource usage report.
8). Users can checkpoint their jobs (OpenVZ snapshots).
9). Sysadmins can perform automated job migration across the nodes through OpenVZ live migration and are able to dynamically consolidate workload on fewer nodes , turn the rest off to save power.

Centralized cluster storage (NFS, Lustre)

Clavis-HPC framework

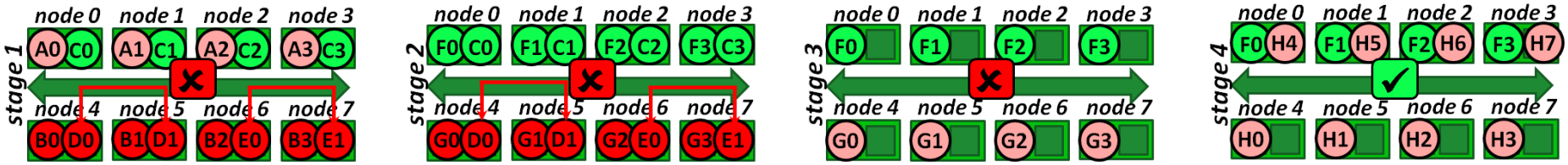
How Clavis-HPC works
(a video demonstration):

http://www.youtube.com/watch?feature=player_embedded&v=h7SFkmbv7-M

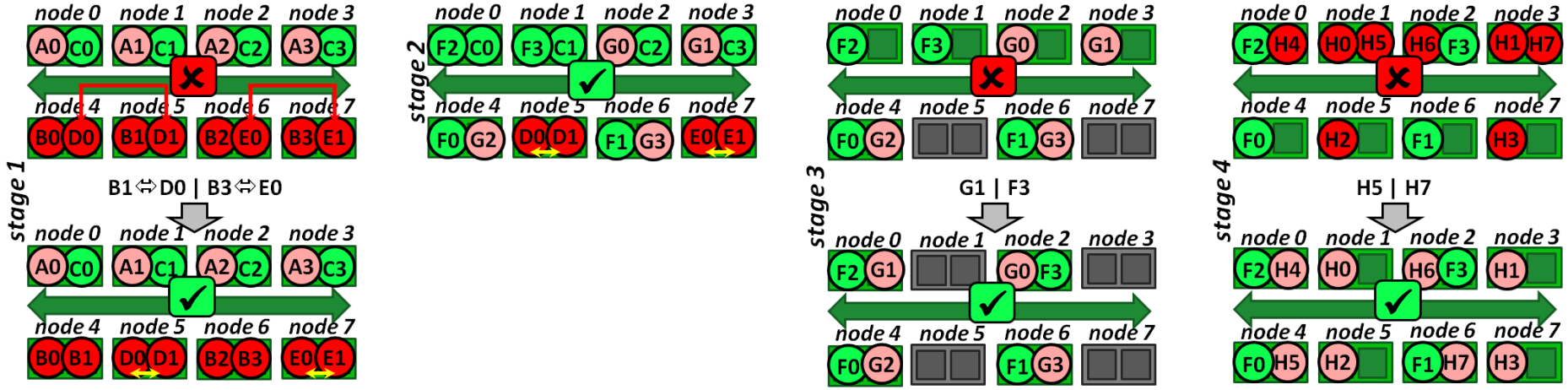
http://www.youtube.com/watch?feature=player_embedded&v=7dUTq6yuMzg

Videos

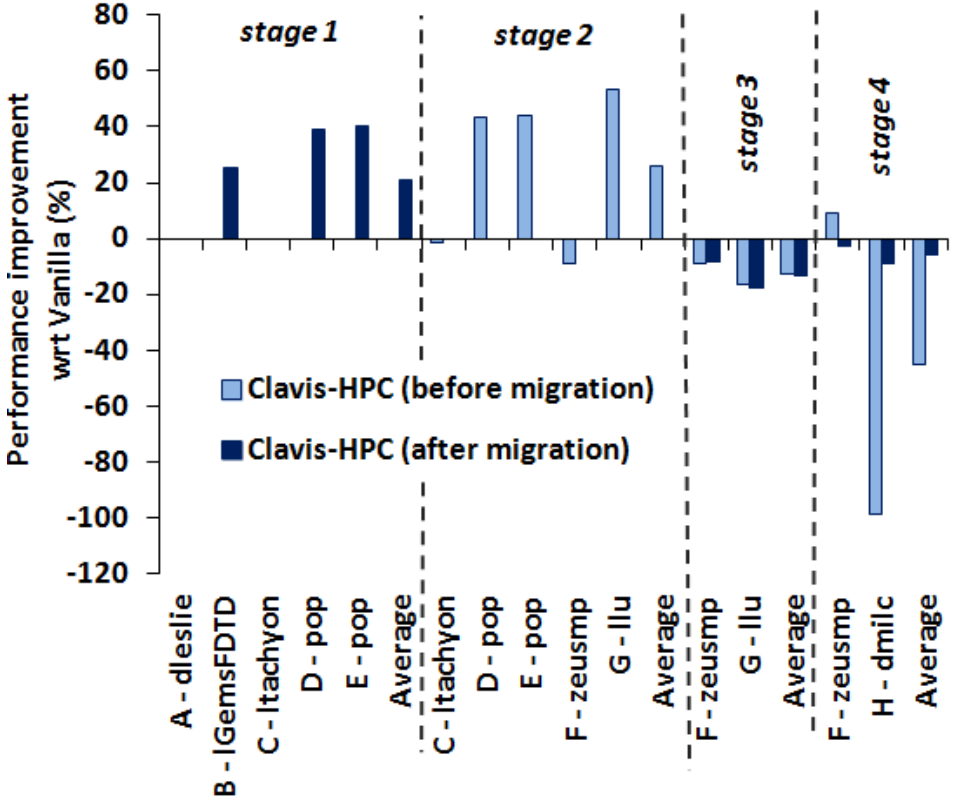
Vanilla HPC framework:

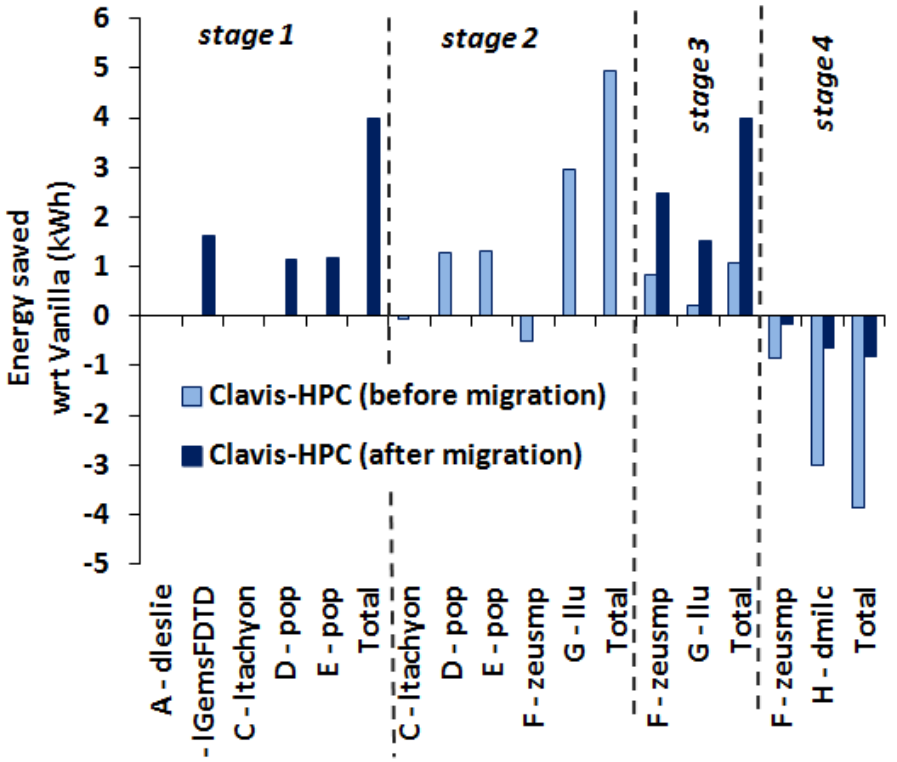


Clavis-HPC:



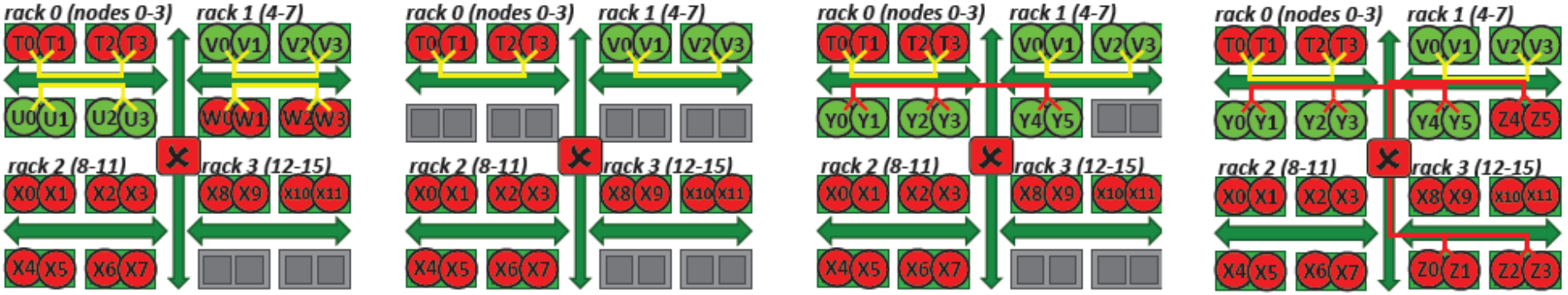
Cluster-wide scheduling (a case for HPC)


Performance improvement wrt Vanilla (%)
80
60
40
20
0
-20
-40
-60
-80
-100
-120
stage 1
stage 2
stage 3
stage 4
Clavis-HPC (before migration)
Clavis-HPC (after migration)
A - dleslie
B - lGemsFDTD
C - ltachyon
D - pop
E - pop
Average
C - ltachyon
D - pop
E - pop
F - zeusmp
G - llu
Average
F - zeusmp
G - llu
Average
F - zeusmp
H - dmilc
Average

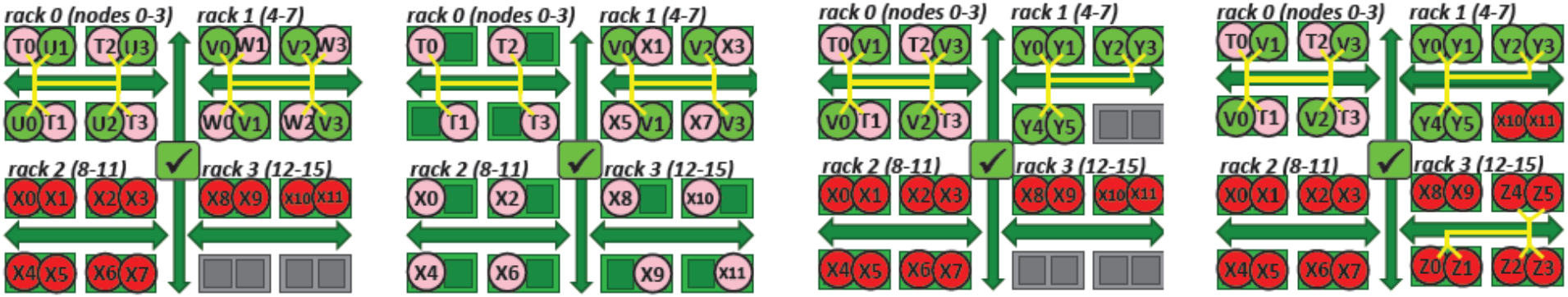

Energy saved wrt Vanilla (kWh)
6
5
4
3
2
1
0
-1
-2
-3
-4
-5
stage 1
stage 2
stage 3
stage 4
Clavis-HPC (before migration)
Clavis-HPC (after migration)
A - dleslie
B - lGemsFDTD
C - ltachyon
D - pop
E - pop
Total
C - ltachyon
D - pop
E - pop
F - zeusmp
G - llu
Total
F - zeusmp
G - llu
Total
F - zeusmp
H - dmilc
Total

Results

Vanilla HPC framework:

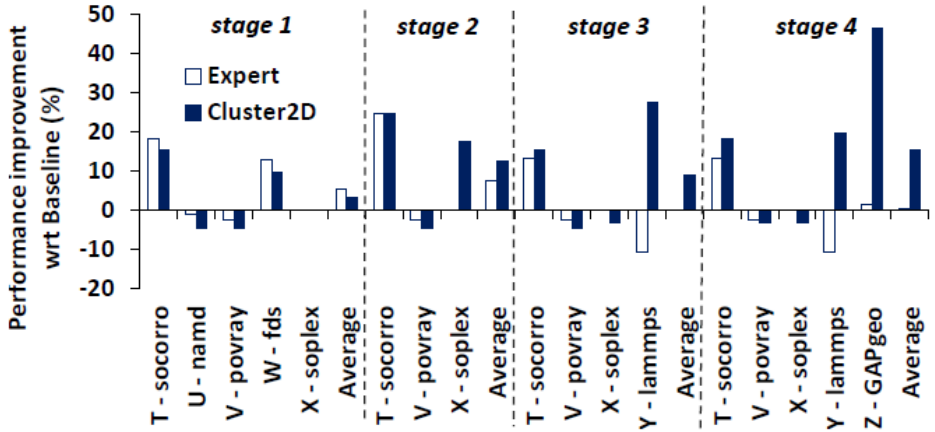


Clavis-HPC:

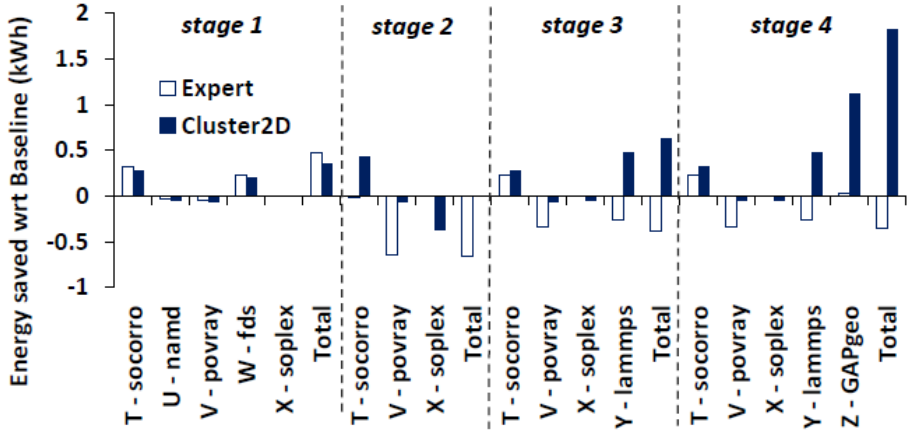


Cluster-wide scheduling (a case for HPC) #2

Talk by Sergey Blagodurov
SC'13 Electronic demo



(a) The performance results of India cluster runs.



(b) The energy measurements from India cluster runs.

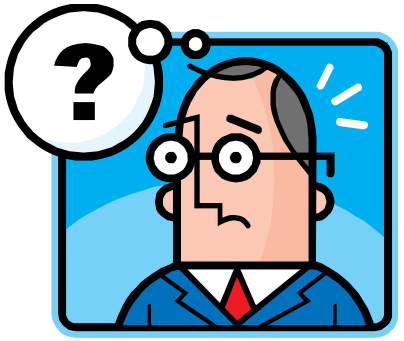
Results #2

Talk by Sergey Blagodurov
SC'13 Electronic demo

In a nutshell:

- Datacenters is the platform of choice
- Datacenter servers are major energy consumers
- The energy is wasted because of resource contention
- I address the resource contention automatically and on-the-fly

Any [time for] questions?

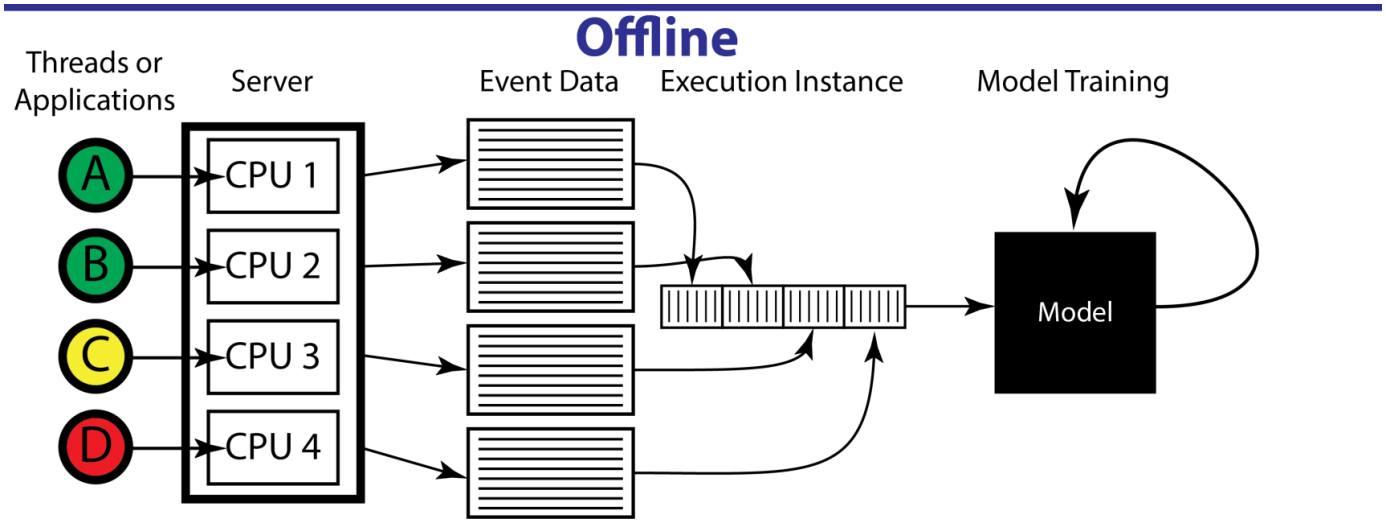


Bonus: LLC missrate works, but is not very accurate

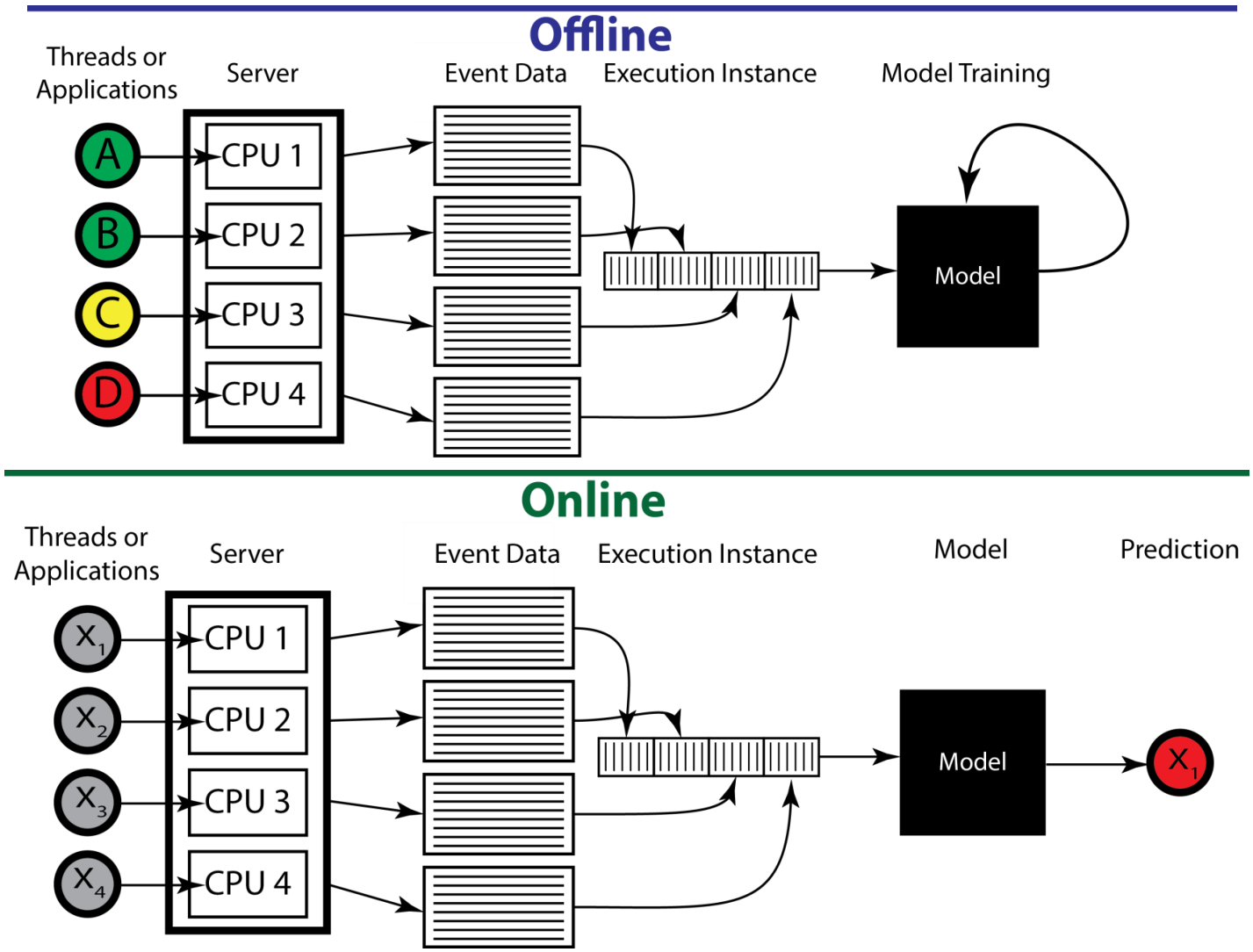
What if we want a metric that is more accurate?

- Then we need to profile many performance counters simultaneously
- … and we need to build a model that predicts the degradation.
- We would have to train the model beforehand on a representative workload.

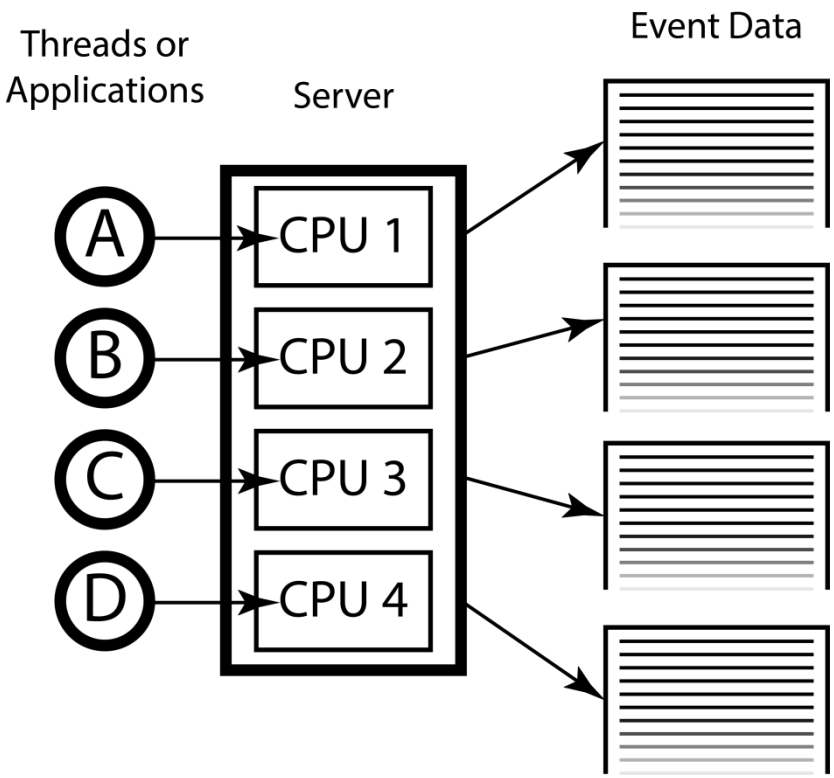
The need of training the model is the price of higher accuracy!



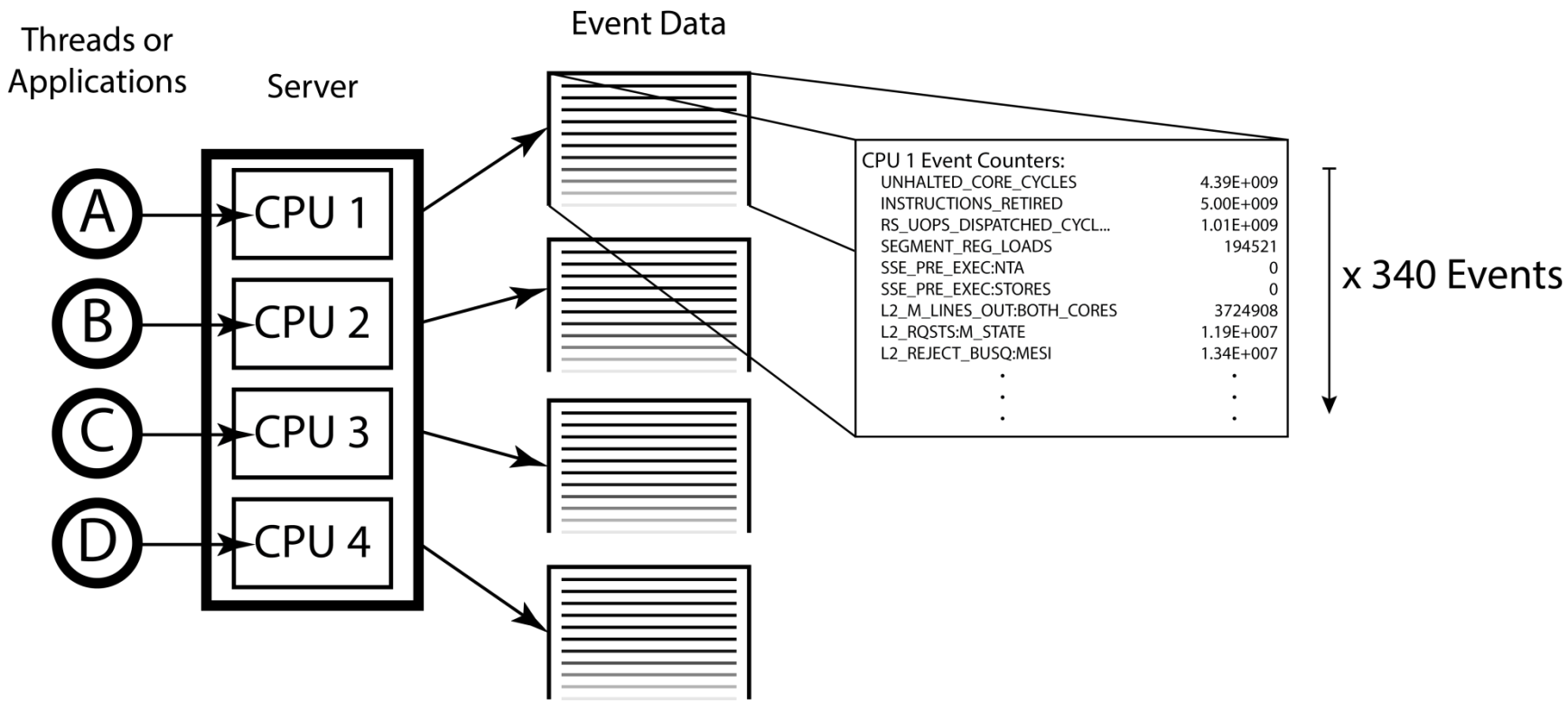
Devising an accurate metric (outline)



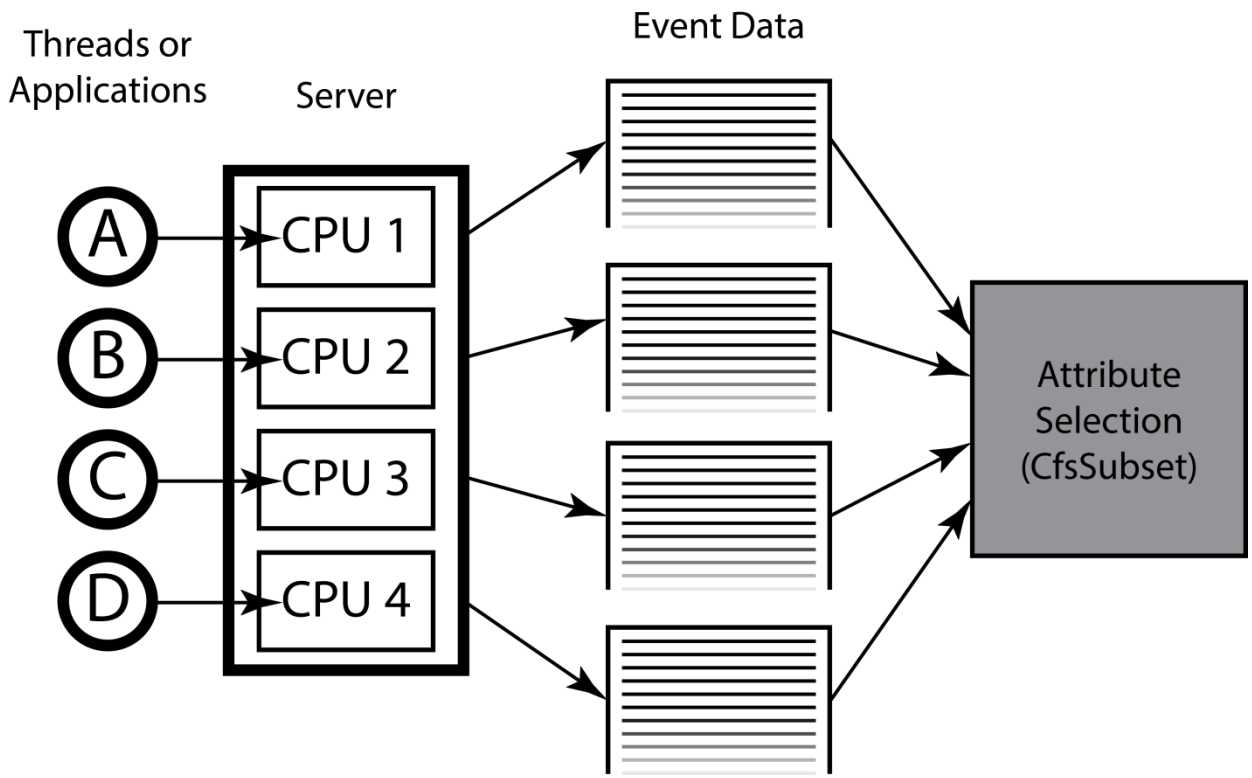
Devising an accurate metric (outline)



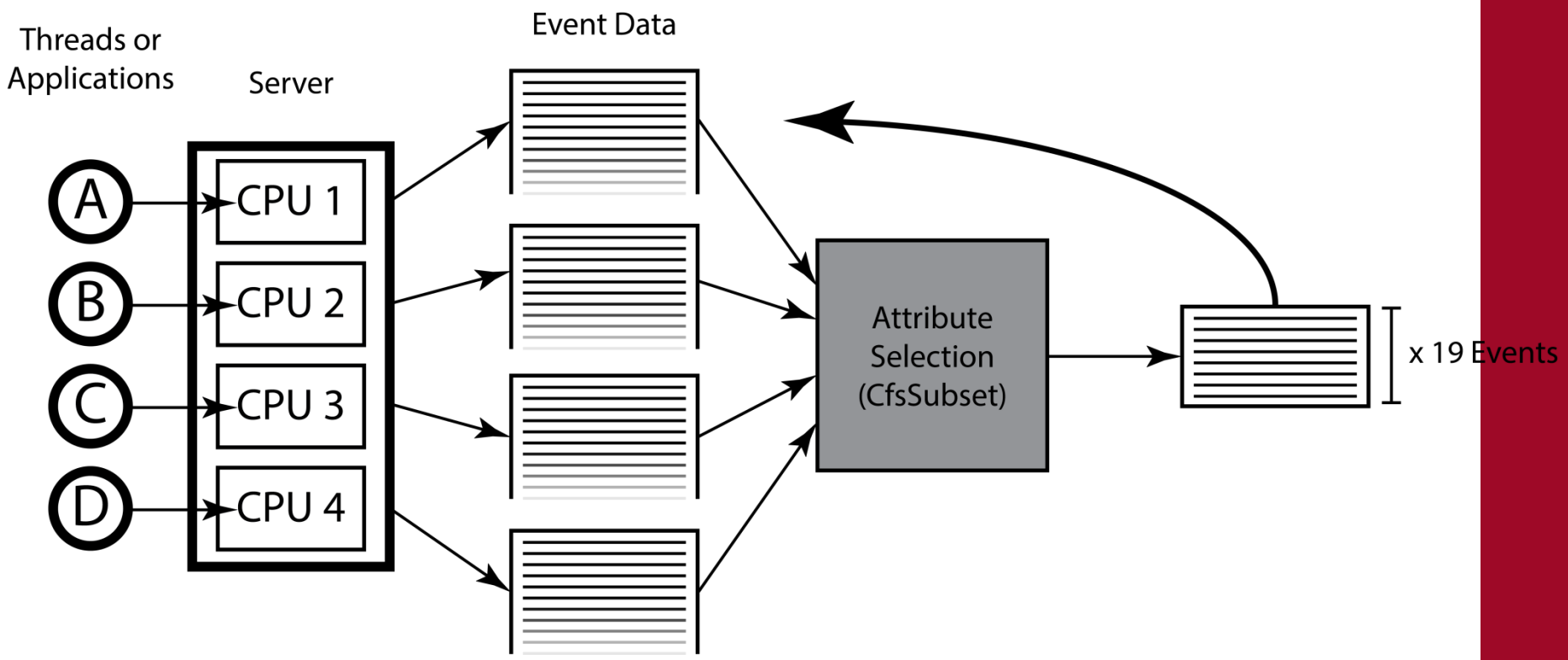
Devising an accurate metric (methodology)



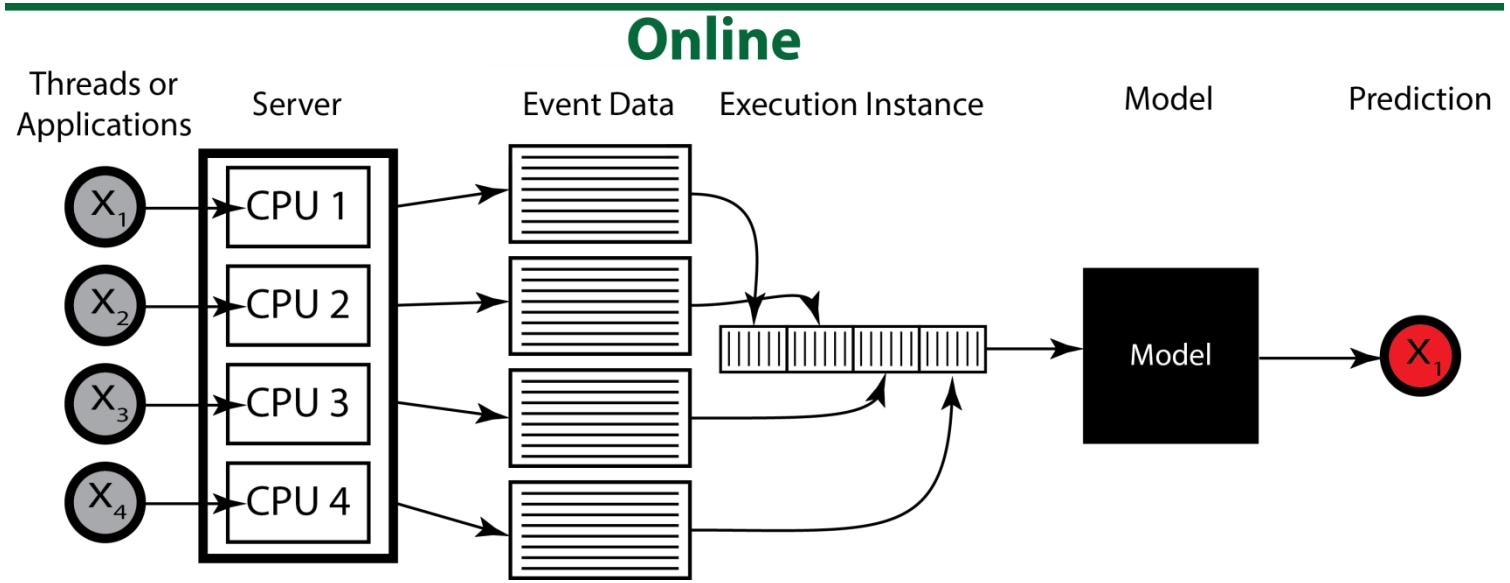
Devising an accurate metric (methodology)



Devising an accurate metric (methodology)



Devising an accurate metric (methodology)



REPTree module in Weka:

- creates a tree with each attribute placed in a tree node
- branches of the tree are values that this attribute takes
- The leaf stores degradation (obtained on the training stage)

Devising an accurate metric (model)

AMD

Events:

- 208 Recordable Core events, 223 Recordable Chip Events
- **32 Core events selected, 8 Chip events selected**

Average Prediction Error: 13%

Intel

Events:

- 340 Recordable core events, **19 Core events selected**

Average Prediction Error: 16%

Devising an accurate metric (results)