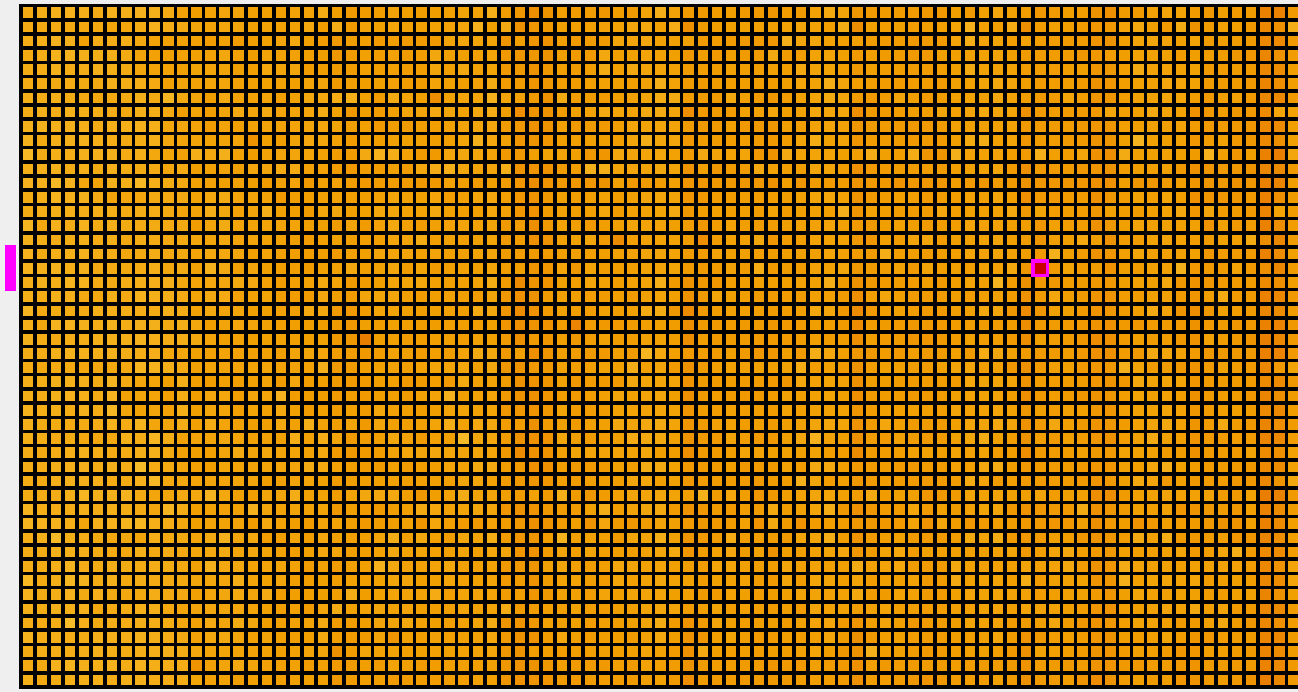


EXASCALE POTHOLES FOR HPC

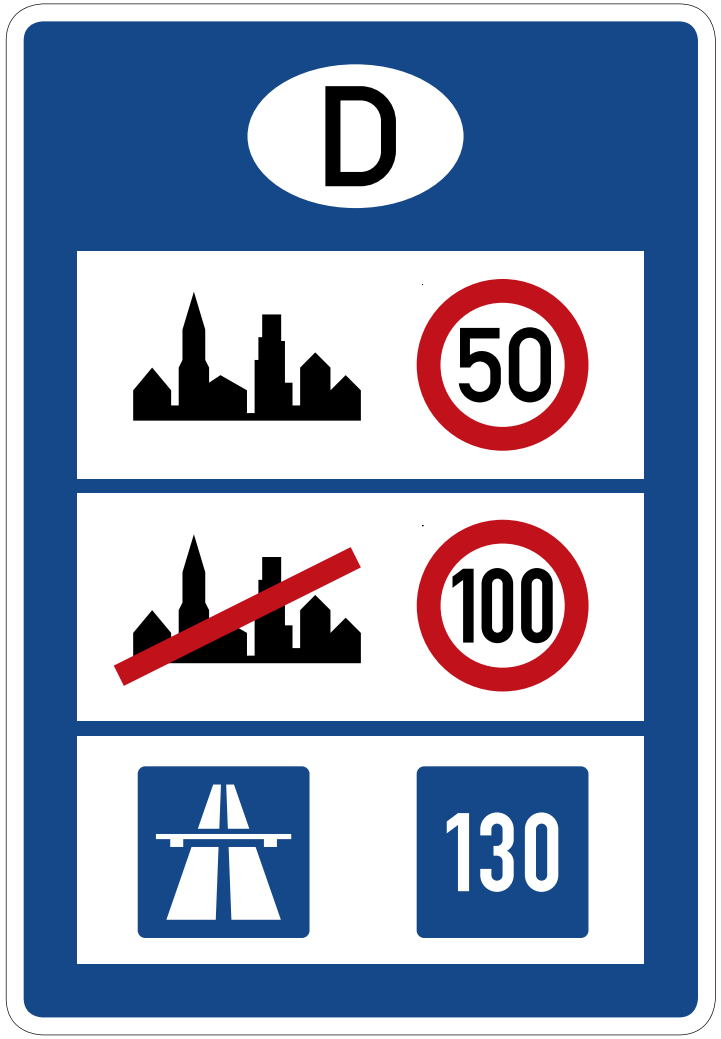
Execution performance and variability analysis of the flagship application code HemeLB

2020/11/12 I BRIAN WYLIE (B.WYLIE@FZ-JUELICH.DE)

JÜLICH
Forschungszentrum

DESIGNED & TUNED FOR HIGH PERFORMANCE!

No speed limit



YOUR MILEAGE MAY VARY

..it happens!

PERFORMANCE VARIATION

Typical causes (selection)

Application (user domain)

Generally intentional!

- changes of code versions, compilers & run-time libraries, optimization levels/flags
- non-determinism (Monte-Carlo, data races)

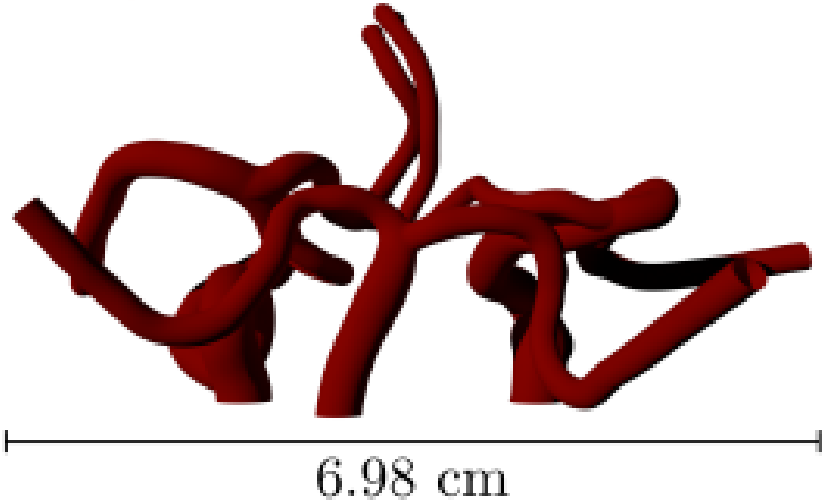
System (hardware/software)

- machine-hall power outages, power/energy management policies
- chip manufacturing variance / wear-and-tear
- error correction (cosmic rays)
- contention on network interconnect
- file I/O to shared filesystem
- (rogue) system daemons (NHC)
- ...

Unmodified jobscript & application executable still runs correctly, but takes (much) longer and not under user control!

CASE STUDY

POP performance assessment of HemeLB on SuperMUC-NG



HemeLB open-source code and test case: www.**hemelb.org**

- CoE flagship code developed by UC London (UK)
- C++ parallelised with MPI
- built with Intel Studio compiler & MPI library
- 5,000 time-step (500μs) simulation of cerebrovascular “circle of Willis” geometry
  - 6.4μm lattice resolution (21.15 GiB): 10,154,448,512 lattice sites

SuperMUC-NG Lenovo ThinkSystem SD650 at LRZ

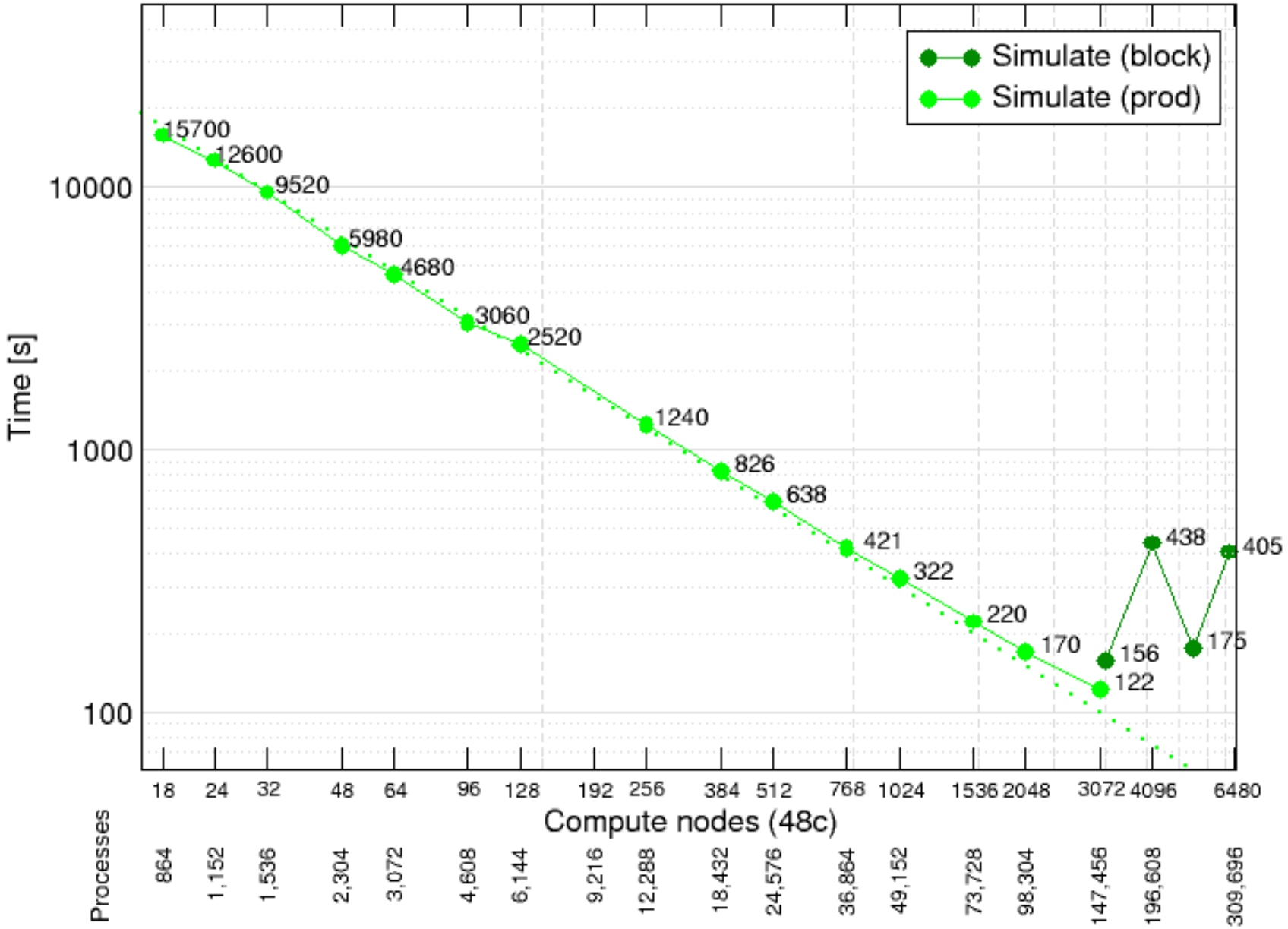
- dual 24-core Intel Xeon Platinum 8174 (‘Skylake’) @ 3.1GHz compute nodes
- 48 MPI processes per node, 6,452 (of 6,480) compute nodes: 309,696 MPI processes
- 144 ‘fat’ [768 GiB] + 6,336 ‘thin’ [96 GiB] compute nodes

INITIAL RESULTS

Application strong scaling

up to 6,250 compute nodes (dual 24-core)

- 96.5% of entire system (6,480 compute nodes)
- 300,000 MPI processes (one per core)
- 80% scaling efficiency maintained to over 100,000 processes in production
- but ‘poor’ performance for jobs using the entire system in dedicated block operation – also erratic

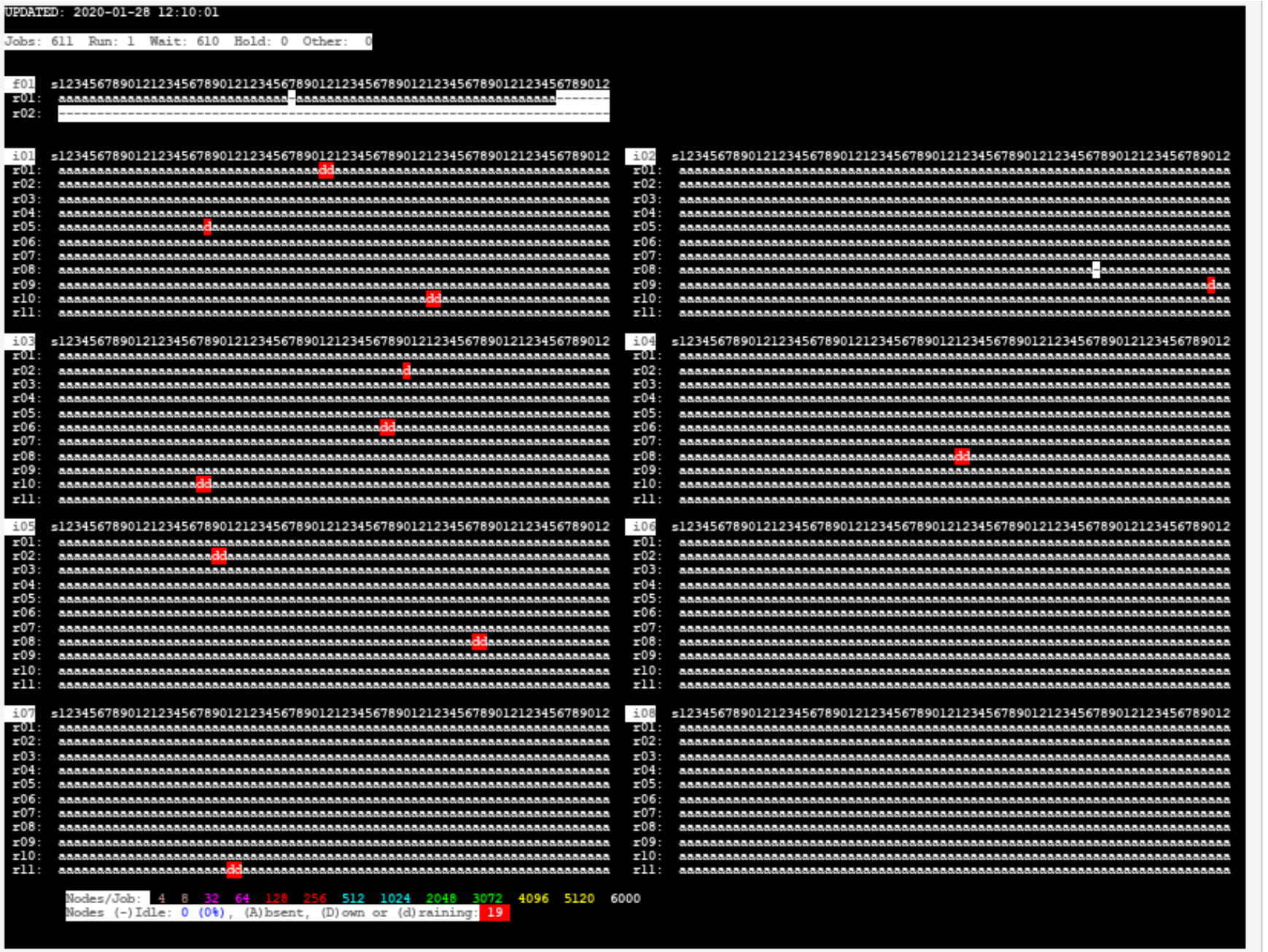


SYSTEM VIEW

HemeLB application execution

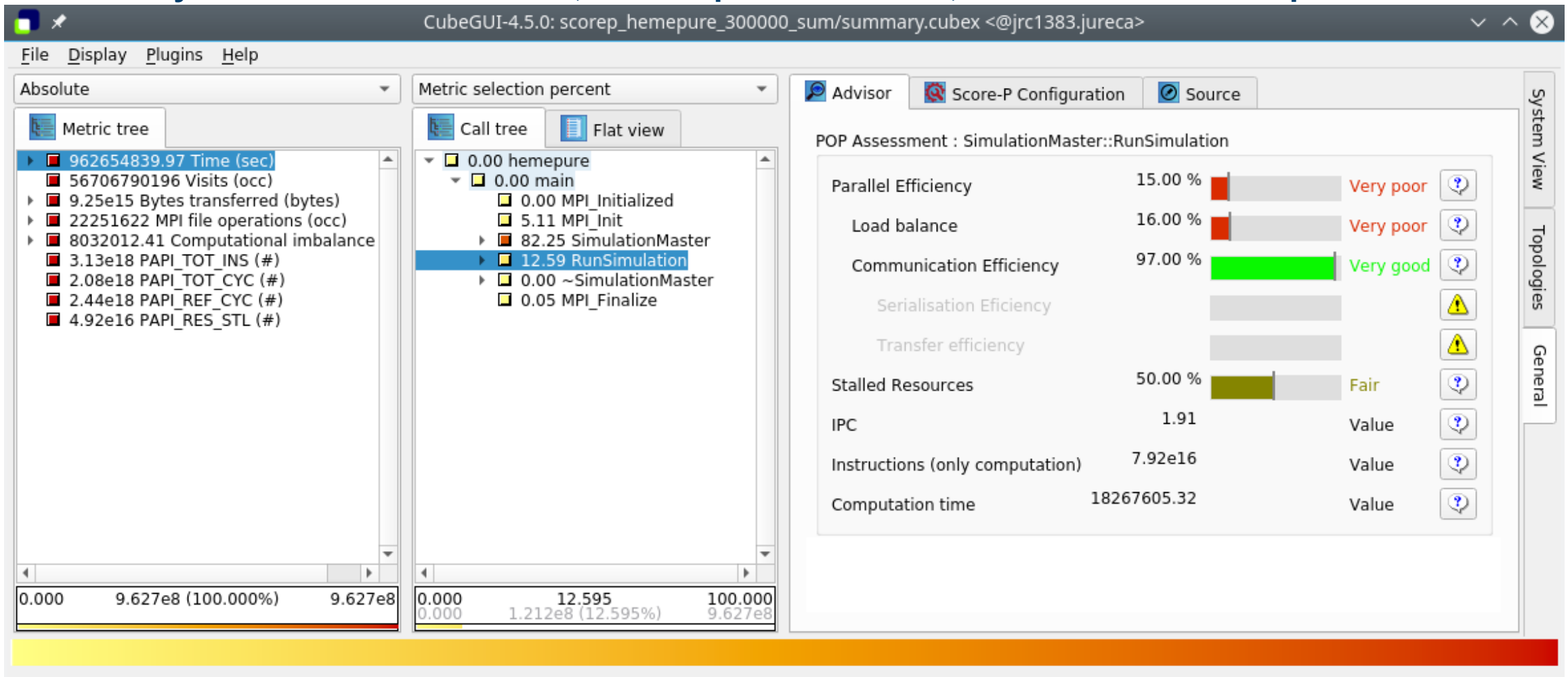
One of the first (non-HPL) application executions on the entire virgin compute system

- 300,000 MPI processes on 6,250 compute nodes
- spread over all 9 ‘islands’
  - production jobs limited to only half of system (3168 compute nodes)
- avoiding 20 off-line nodes, and additional 2 which failed to boot in first attempt



LOAD BALANCE?

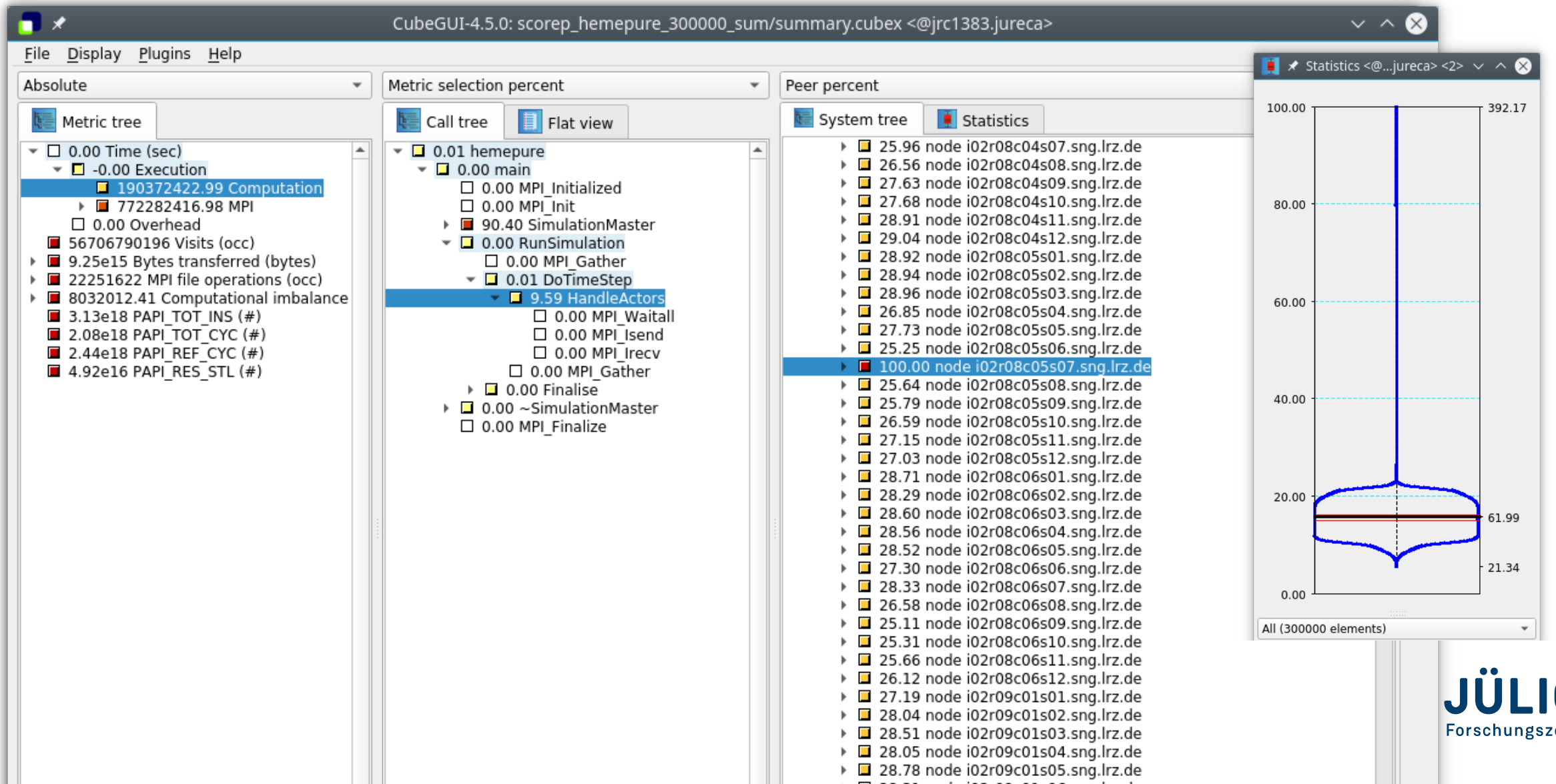
Scalasca analysis of execution with 300,000 MPI processes on 6,250 dual 24-core compute nodes



Focus of Analysis (FOA) = SimulationMaster::RunSimulation

LOAD BALANCE?

Scalasca analysis of execution with 300,000 MPI processes on 6,250 dual 24-core compute nodes



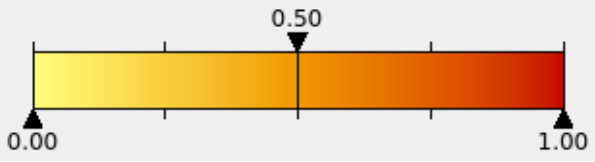
JÜLICH
Forschungszentrum

COMP TIME

Process topology (folded)

300,000 MPI processes as 625x480 topology

- 10 compute nodes / column
- computation time peer %
  - highest process value shown red and sets colour scale of values
- looks fairly homogeneous = well balanced?

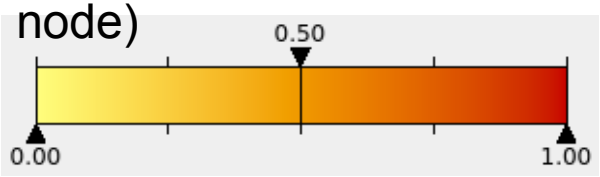


HOTSPOT?

Node i02r08c05s07 (odds)

300,000 MPI processes as 625x480 topology map

- 10 compute nodes / column
- computation time peer %
  - highest process value shown red and sets colour scale of values
- 24 processes on one node take much longer than any others (even on the same node)



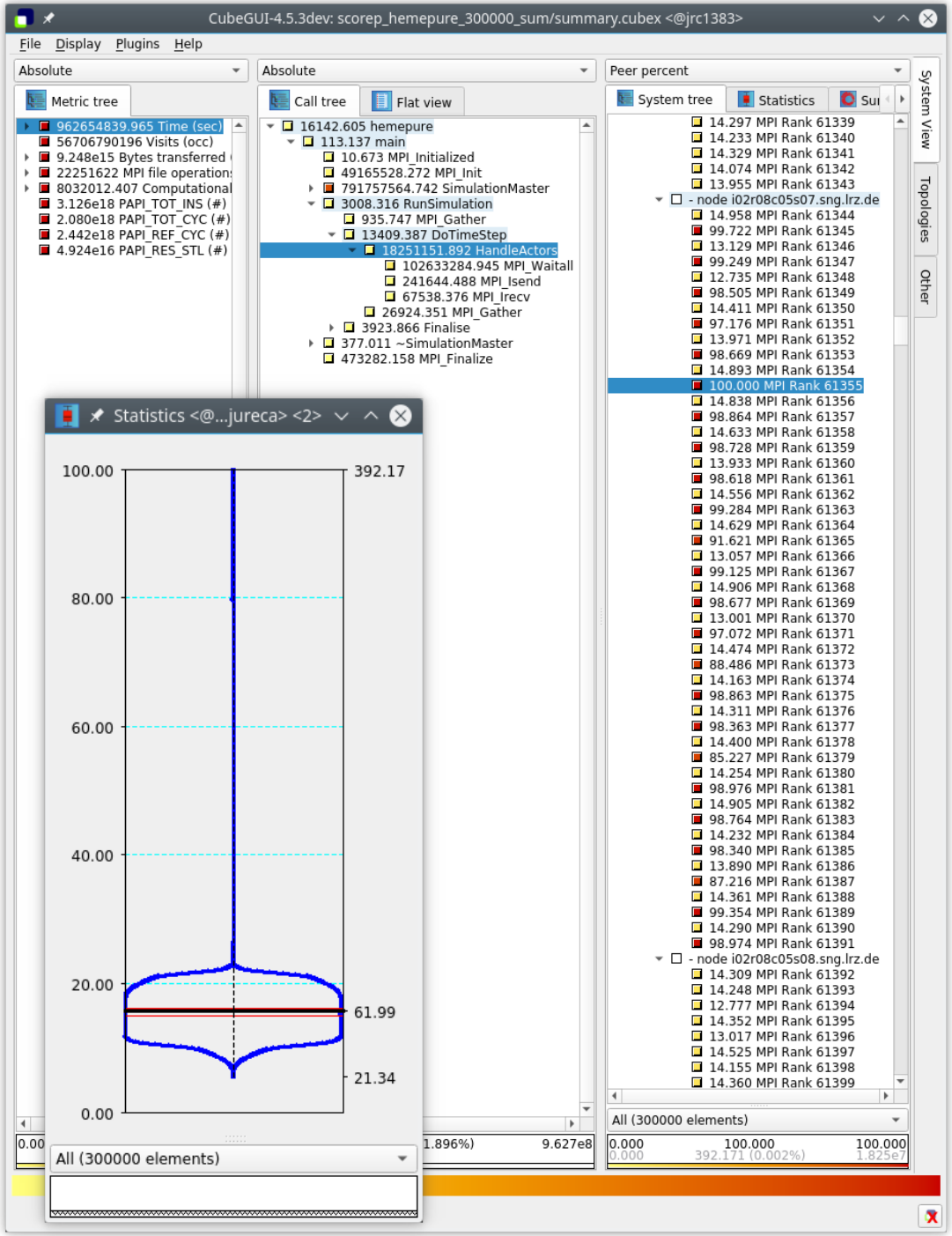
Process (size 300000)	61355
Thread (size 1)	0
Node:	node i02r08c05s07.sng.lrz.de
Name:	MPI Rank 61355
MPI rank:	61355
Thread id:	0
Value:	100.00000000 (100.000%)
Absolute:	392.17060900 (100.000%)
Number of elements:	1

PEER VALUES?

Node i02r08c05s07 (odds)

300,000 MPI processes in system tree

- 6,250 compute nodes each with 48 MPI processes
- computation time peer %
- 24 odd-numbered processes on node i02r08c05s07 take ***more than 6x longer*** than any of their peers
  - who must then wait in their subsequent point-to-point neighbour communication

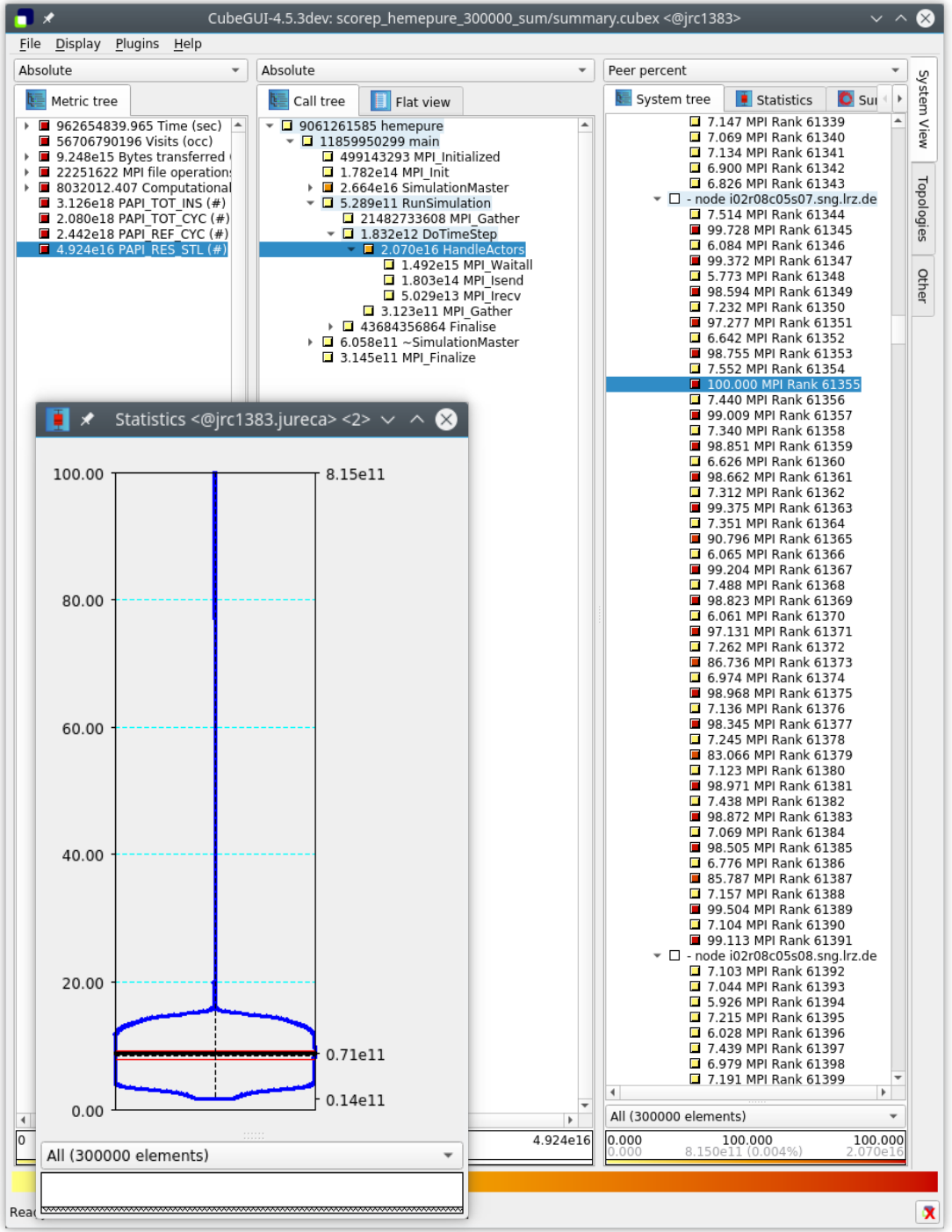


WHY?

Node i02r08c05s07 (odds)

300,000 MPI processes in system tree

- 6,250 compute nodes each with 48 MPI processes
- computation time peer %
- 24 odd-numbered processes on node i02r08c05s07 take ***more than 6x longer*** than any of their peers
  - executing roughly the same number of instructions
  - but suffering ***14x the CPU resource stall cycles***
  - ultimately due to *memory access stalls*
- node subsequently taken off-line for diagnostic checks which immediately identified that it had a ***faulty DIMM***

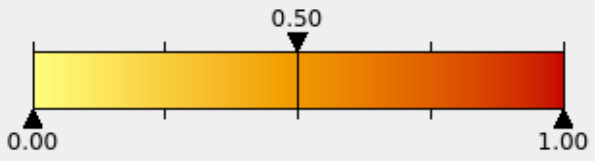


ONE-OFF?

Folded process topology

300,000 MPI processes as 625x480 topology map

- 10 compute nodes / column
- computation time peer %
  - highest process value sets maximum of colour scale

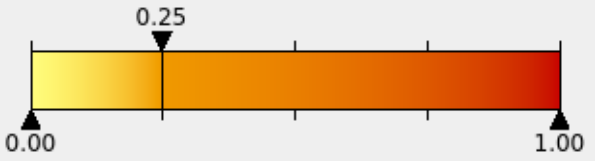


ONE-OFF?

Node i04r01c05s10 (evens)

300,000 MPI processes as 625x480 topology map

- 10 compute nodes / column
- resource stall cycles peer %
  - enhanced colour map
- 24 even-numbered processes on node i04r01c05s10 also take notably longer than their peers: diagnostics reported lacklustre performance

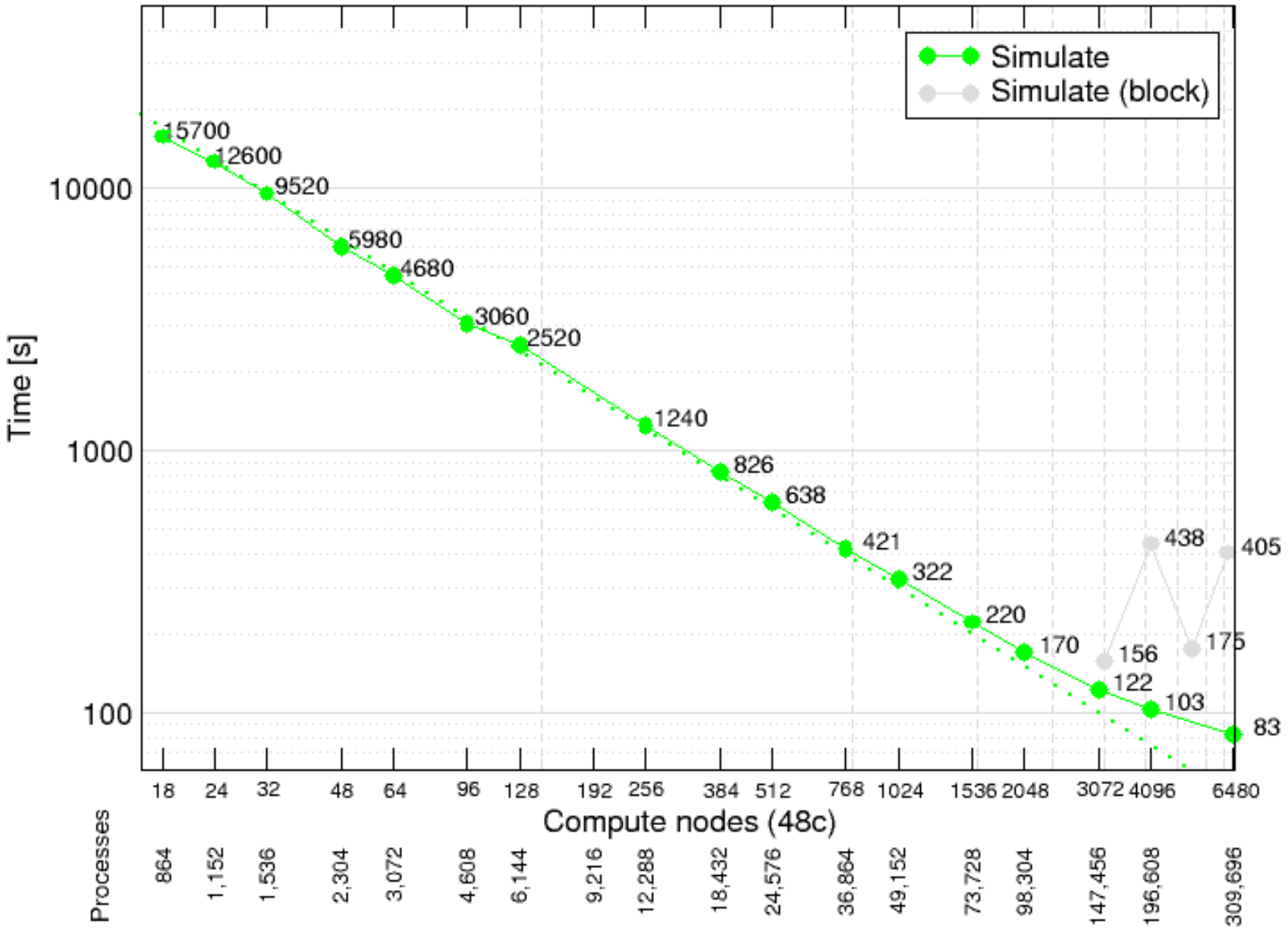


SUCCESS?

Application strong scaling

up to 6,452 compute nodes

- 99.6% of entire system
- 309,696 MPI processes (cores)
- 80% scaling efficiency maintained to over 100,000 processes
- ***190x speed-up*** compared to 18 ‘fat’ compute nodes with larger memory



STRONG SCALING

POP performance assessment

Compute nodes	24	32	48	64	96	128	192	256	384	512	768	1024	1536	2048	3072	4096	6452
Processes	**1152**	**1536**	**2304**	**3072**	**4608**	**6144**	**9216**	**12288**	**18432**	**24576**	**36864**	**49152**	**73728**	**98304**	**147456**	**196608**	**309696**
Global scaling efficiency	**0.79**	**0.79**	**0.84**	**0.80**	**0.82**	**0.75**		**0.73**	**0.72**	**0.73**	**0.74**	**0.68**	**0.68**	**0.65**	**0.62**	**0.57**	**0.45**
- Parallel efficiency	0.79	0.80	0.87	0.83	0.86	0.80		0.75	0.74	0.74	0.77	0.71	0.72	0.70	0.72	0.70	0.73
- - Load balance efficiency	0.79	0.80	0.88	0.84	0.86	0.80		0.75	0.74	0.75	0.78	0.72	0.74	0.72	0.74	0.73	0.80
- - Communication efficiency	1.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00	0.99	0.99	0.99	0.98	0.98	0.97	0.96	0.92
- Computation scaling	*1.00*	0.99	0.96	0.96	0.95	0.93		0.98	0.98	0.98	0.96	0.96	0.94	0.93	0.87	0.81	0.61
- - Instructions scaling	*1.00*	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	0.99	0.97	0.94	0.89	0.79	0.67	0.45
- - IPC scaling	*1.00*	0.99	0.96	0.96	0.95	0.93		0.98	0.98	0.99	0.98	0.99	1.00	1.04	1.11	1.21	1.36
IPC	1.411	1.395	1.353	1.355	1.342	1.316		1.377	1.387	1.396	1.383	1.390	1.417	1.473	1.566	1.704	1.919
											Key:	<0.65	<0.75	<0.85	<0.95	<1.00	>1.00

Global scaling efficiency fairly good around 80%, before generally degrading at larger scales

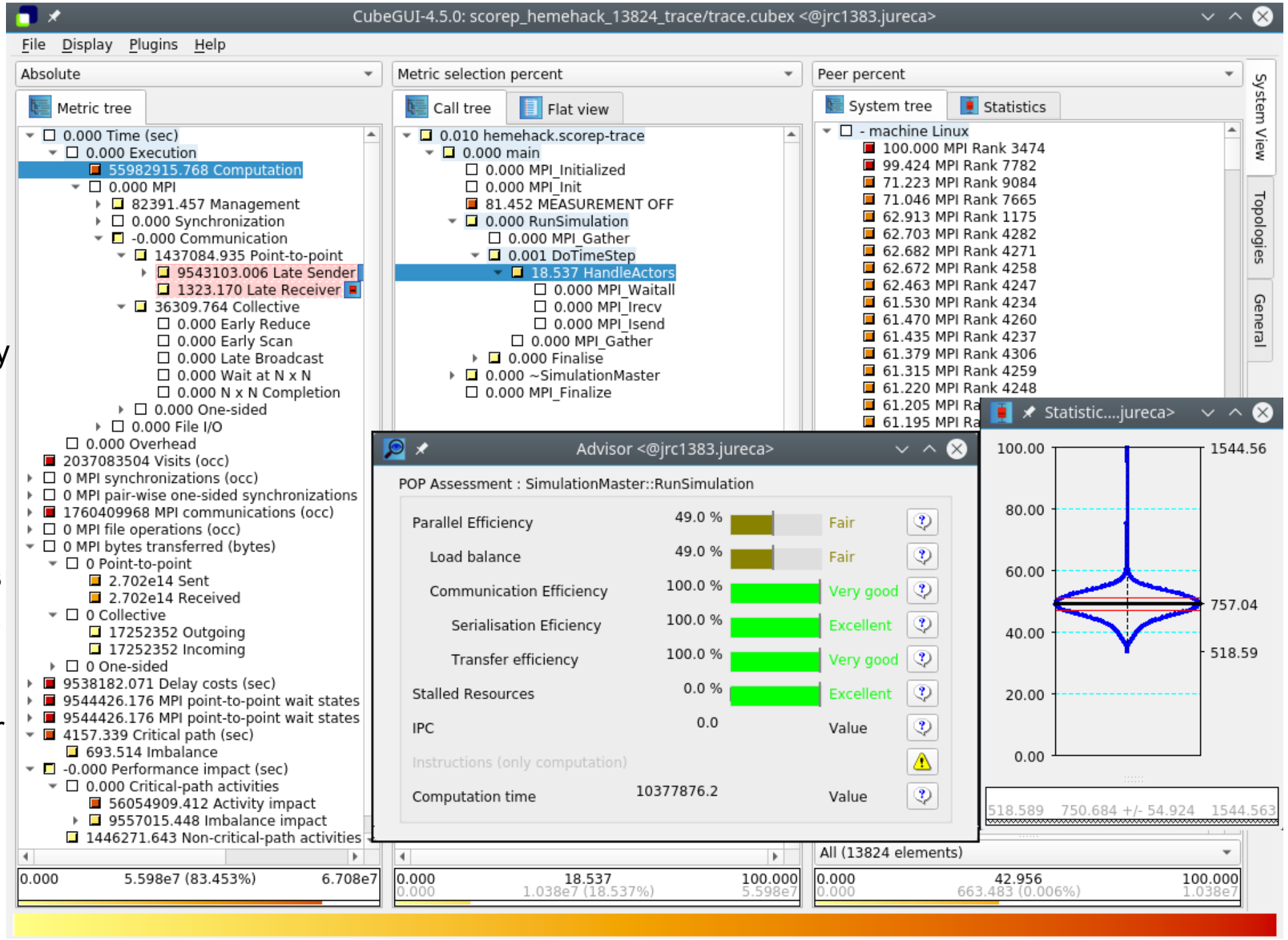
- Parallel efficiency deteriorating following Load balance efficiency; excellent Communication efficiency throughout
- Computation scaling (relative to 24 nodes) very good, except at largest scale
- Degradation of Instructions scaling partially compensated by improving IPC scaling

ALL DONE?

Look closer

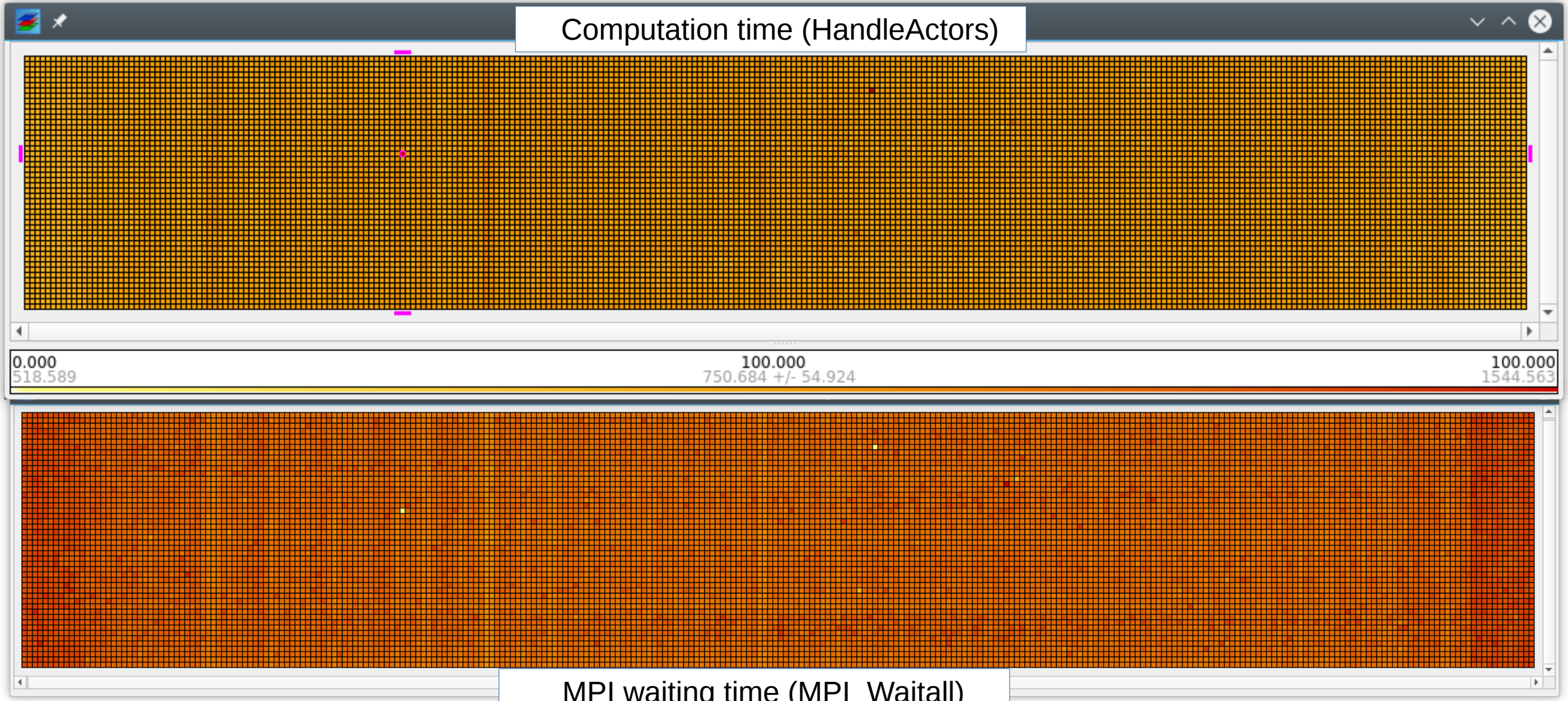
13,824 MPI processes
(on 288 compute nodes)

- communication efficiency is excellent (100%)
- load balance still only miserable 49%
- due to **2 MPI processes** 3474 & 7782 taking 30% longer than any others
  - so that they never have to wait



ROGUE PROCESSES?

MPI ranks 3474 & 7782 in topological representation (compute node vs. core)

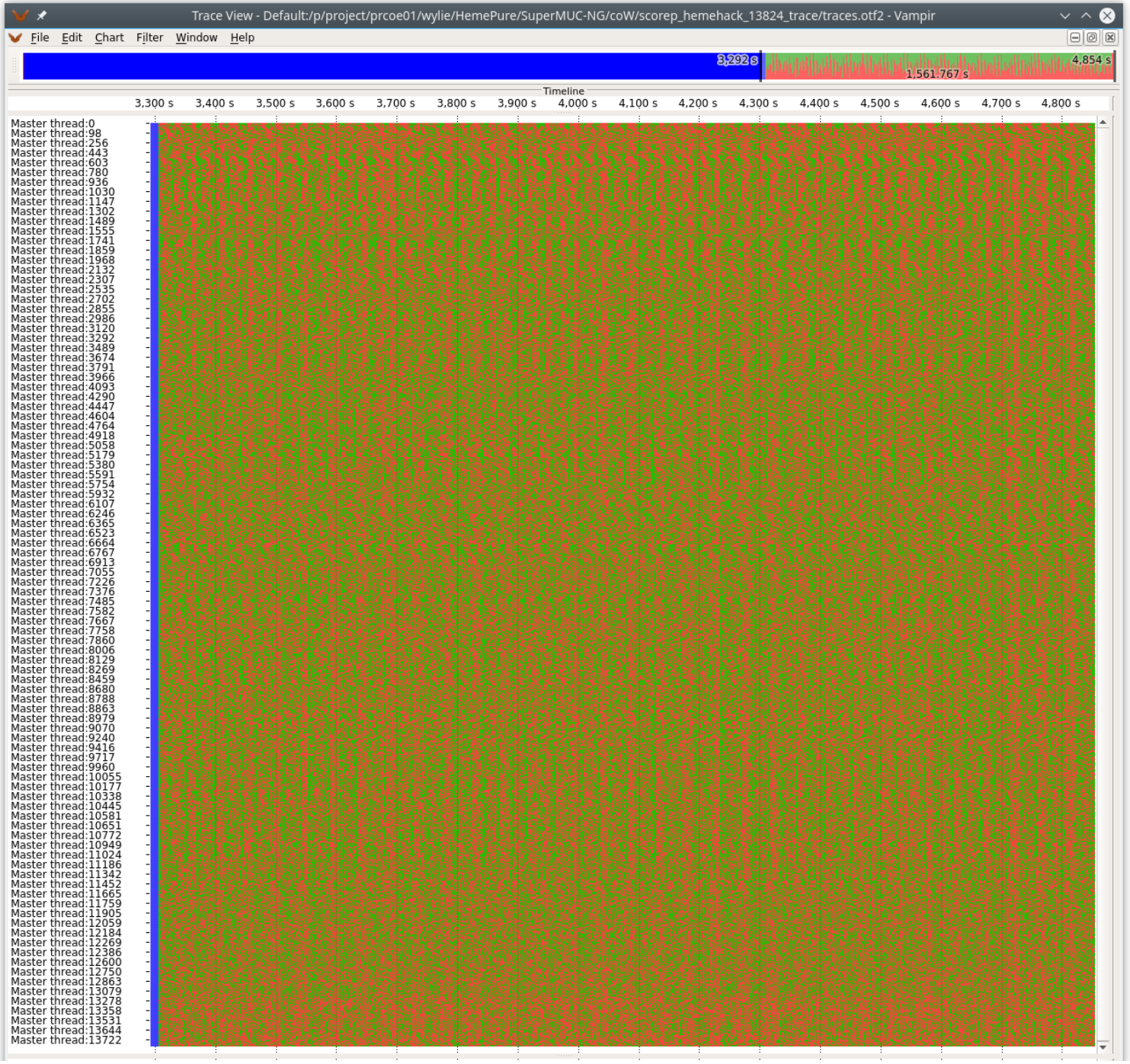


TRACE VISUALISATION

Vampir

Execution timeline view of 13,824 processes (on 288 compute nodes)

- zoom to RunSimulation phase of 5,000 simulation time-steps (duration 1500 seconds)
- lots of MPI waiting time [red] vs. computation [green]
  - slightly more for those MPI processes on first and last compute nodes
- each line of pixels for 11 processes!

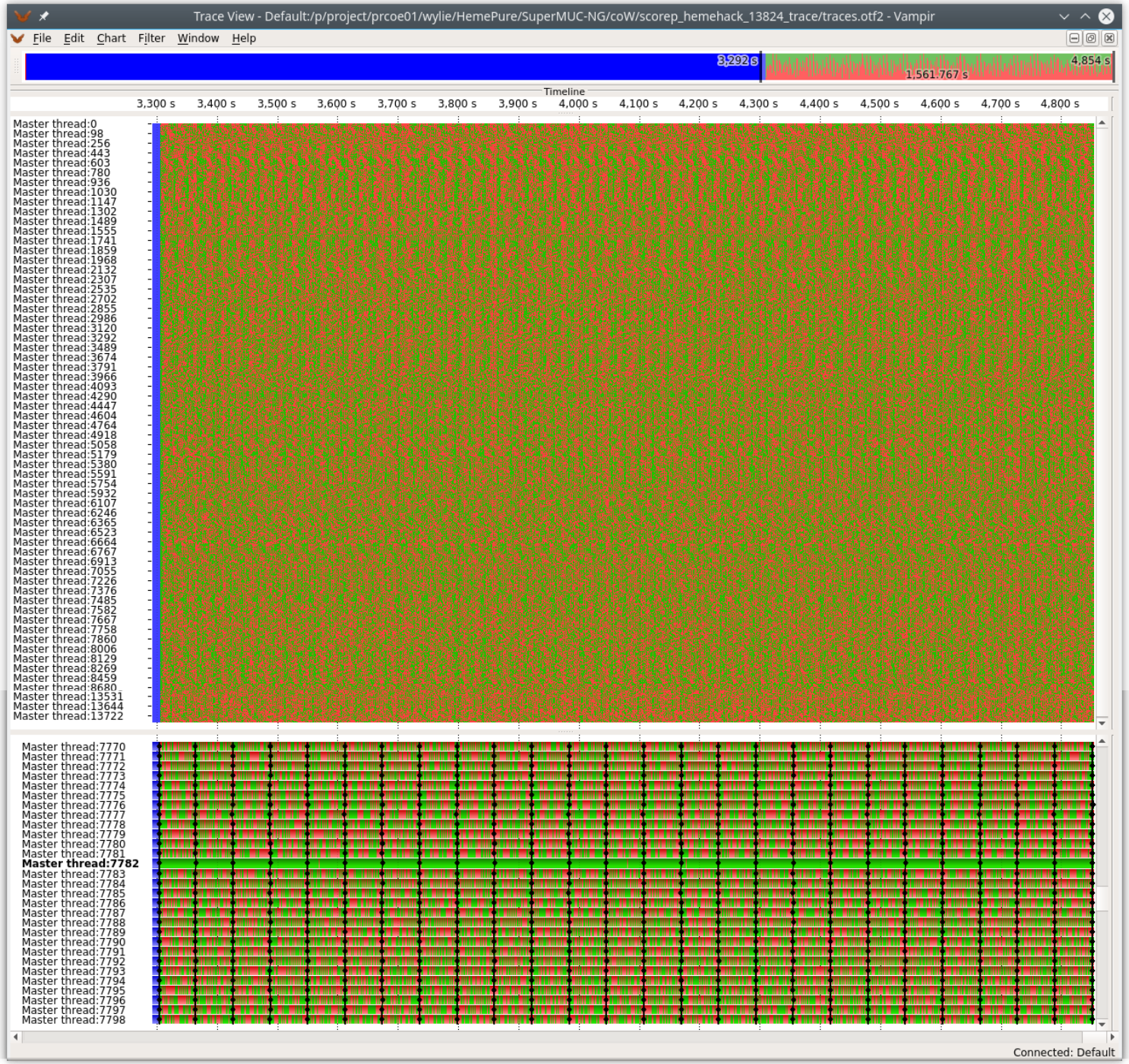


TRACE VISUALISATION

Vampir

Execution timeline view of 13,824 processes (on 288 compute nodes)

- zoom to RunSimulation phase of 5,000 simulation time-steps (duration 1500 seconds)
- lots of MPI waiting time [red] vs. computation [green]
- examining individual processes: MPI process 7782 takes much longer for its computation
  - it never has to wait
  - others must wait for it!

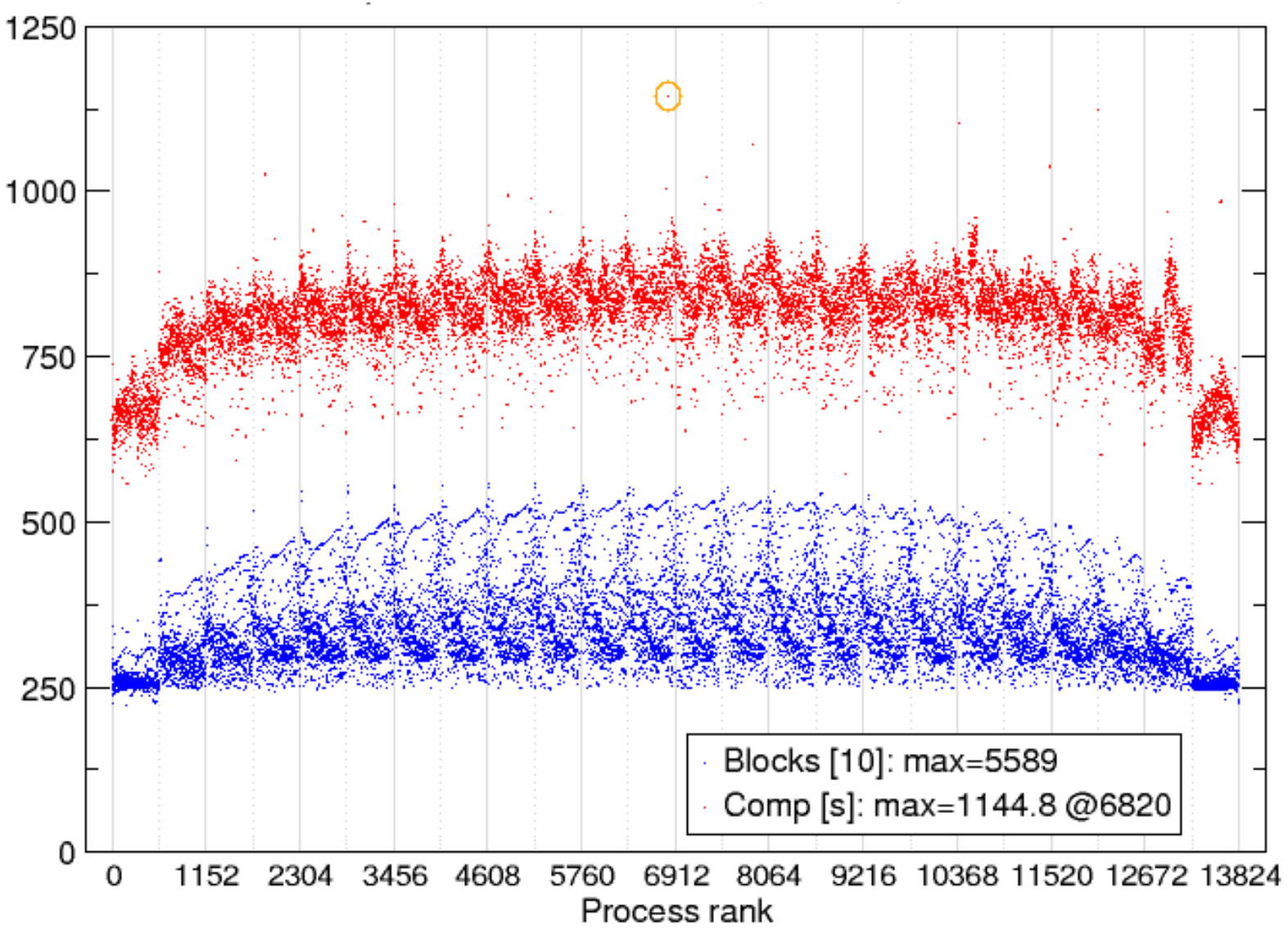


1ST RUN IN JOB

Computation time by MPI rank

13,824 processes (on 288 nodes)

- work distribution by process shown by number of blocks of lattice sites [blue, in tens]
  - somewhat less for first and last compute nodes
- resulting computation time for simulation by process [red, sec]
  - certainly imbalanced
  - but one serious outlier!

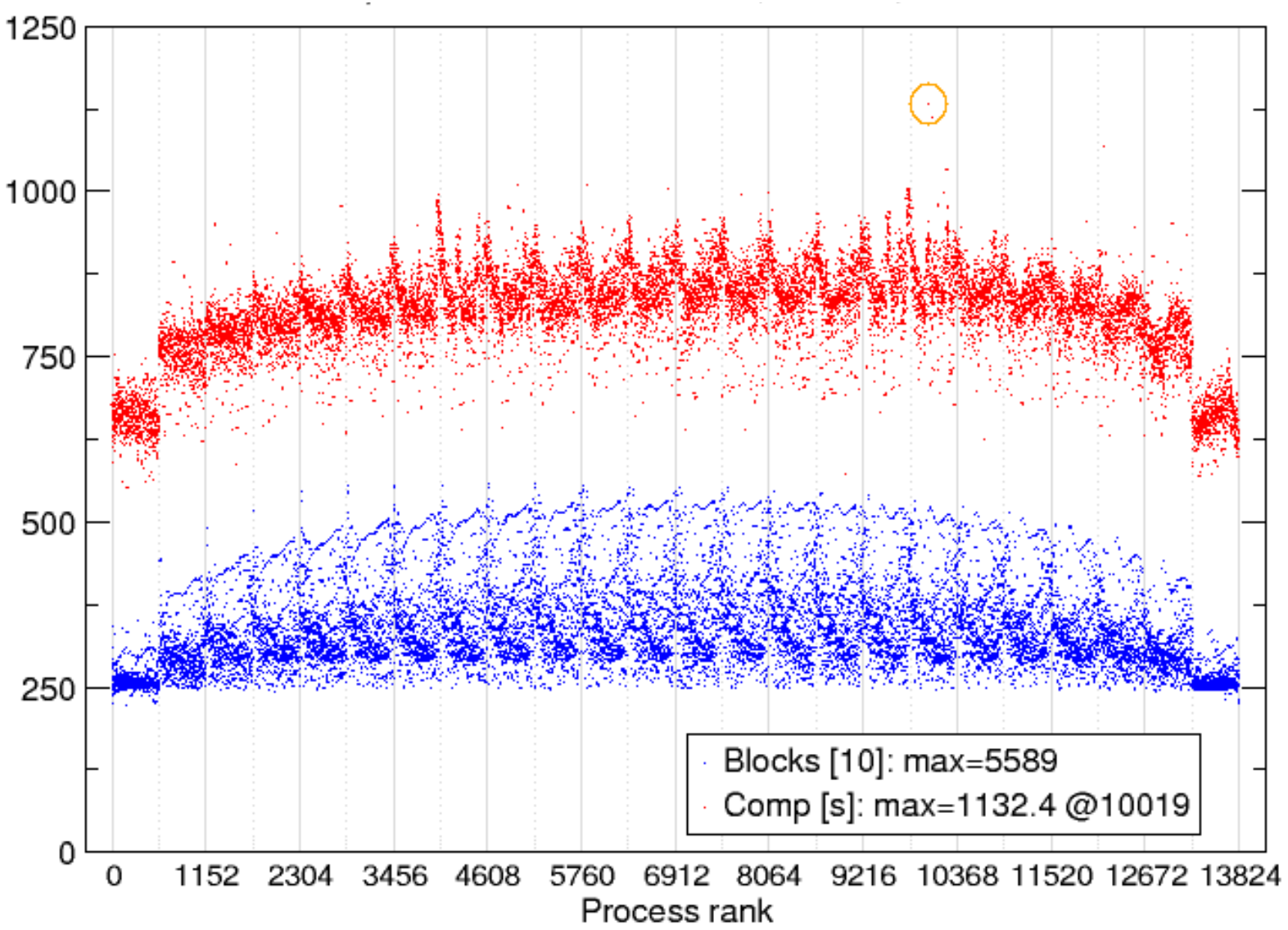


2ND RUN IN JOB

Computation time by MPI rank

13,824 processes (on 288 nodes)

- work distribution by process shown by number of blocks of lattice sites [blue, in tens]
  - identical to previous run (and for all runs)
- resulting computation time for simulation by process [red, sec]
  - imbalanced much as in previous run
  - a different serious outlier!



RÉSUMÉ

State of play

“Slow” memory on a few compute nodes affects all processes/threads on those nodes

- results in large performance degradation that’s entirely repeatable
- can be circumvented by excluding those nodes

Processes/threads bound to individual cores sometimes suffer more modest degradation

- also reflected in elevated stall cycles for memory accesses, throughout execution
- affects different cores in each execution (using the exact same compute nodes)
- see McCalpin@SC18 [DOI: 10.1109/SC.2018.00021]
- not predictable, therefore requires adaptive work redistribution

Neither unique to SuperMUC-NG: apparently ubiquitous!

METHODOLOGY & TOOLS

“pretty standard”

Performance Optimisation & Productivity methodology: www.**pop-coe.eu**

- hierarchical execution efficiency analysis
- scaling of FoA relative to smallest executable configuration
- investigation of core performance via top-down Intel hardware counter analysis

Scalasca/CUBE & Vampir profile & trace analysis tools: www.**scalasca.org** scalasca

- based on community-developed, open-source Score-P measurement infrastructure
- widely deployed and highly scalable
- (independently verified with Intel tools)

LESSONS?

Do try this yourself

Defective compute nodes don't always result in application failures

- apparently aren't always caught by system/node bring-up and pre/post-job diagnostics
- but some (sensitive) applications execute much, much slower
  - *over 6x slower* in the studied case of HemeLB
  - *one* lacklustre compute node resulting in 6,249 others busy waiting 5/6 of time
    - depleting CPU allocation, wasting energy and preventing other jobs from running
  - additional core-to-core variation (moving around) for each 'identical' run
    - *5% and more*, but application (working set) dependent
    - *multiple HPC systems affected*: different processors, interconnects, OS kernels, MPI libraries, compilers, ...

ADVICE TO USERS

Stay alert

Keep a vigilant look out for occurrences of performance degradation!

- provide detailed reports to system admins and performance analysts
  - facilitates more timely investigation & fixes
  - potentially refunding your lost CPU time
- be prepared to swerve?

… or turn up the volume and ignore the noise?

JÜLICH Forschungszentrum

Performance Optimisation and Productivity

Centre of Excellence in Computing Applications & HPC

- contact: Prof. Jesús Labarta
- mailto: pop@bsc.es
- https://www.**pop-coe.eu**

This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreements No 676553 & 824080.