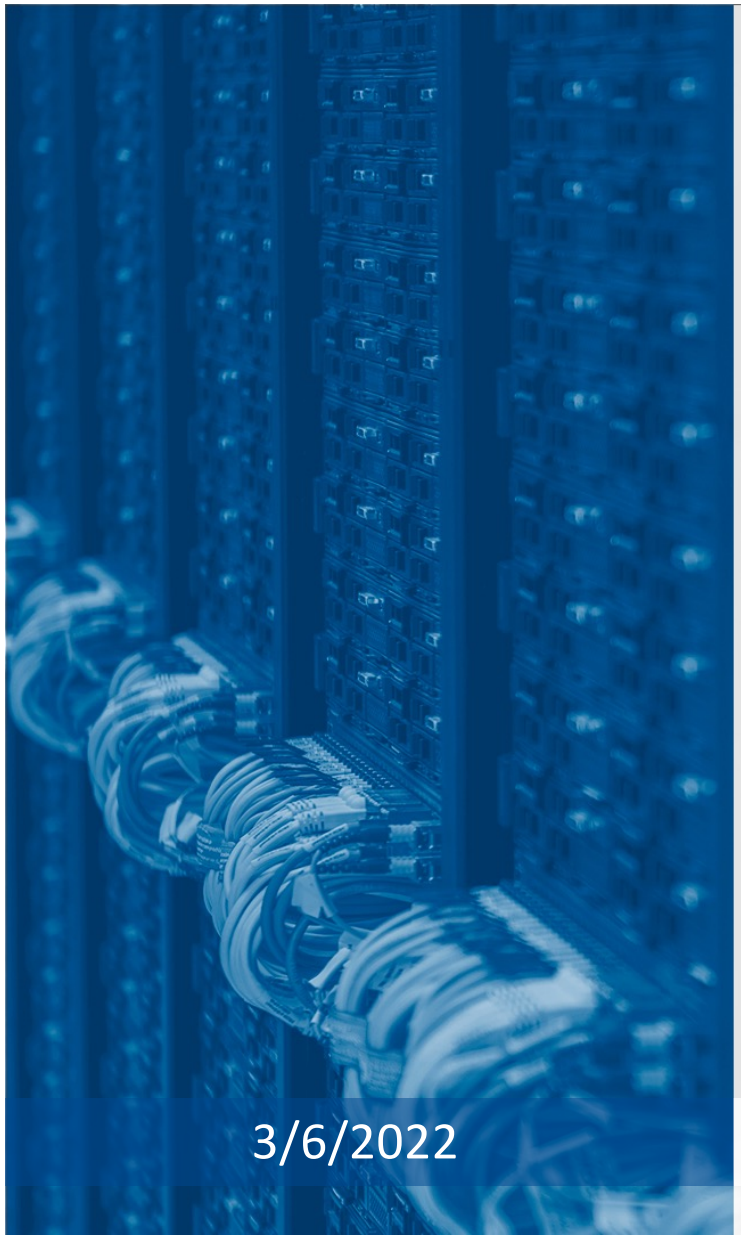


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 754337 (EuroEXA); it has been supported by the Spanish Ministry of Science and Technology (project TIN2015-65316-P), Generalitat de Catalunya (contracts 2014-SGR-1051 and 2014-SGR-1272) and the Severo Ochoa Programme (SEV-2015-0493).

Composability in HPC

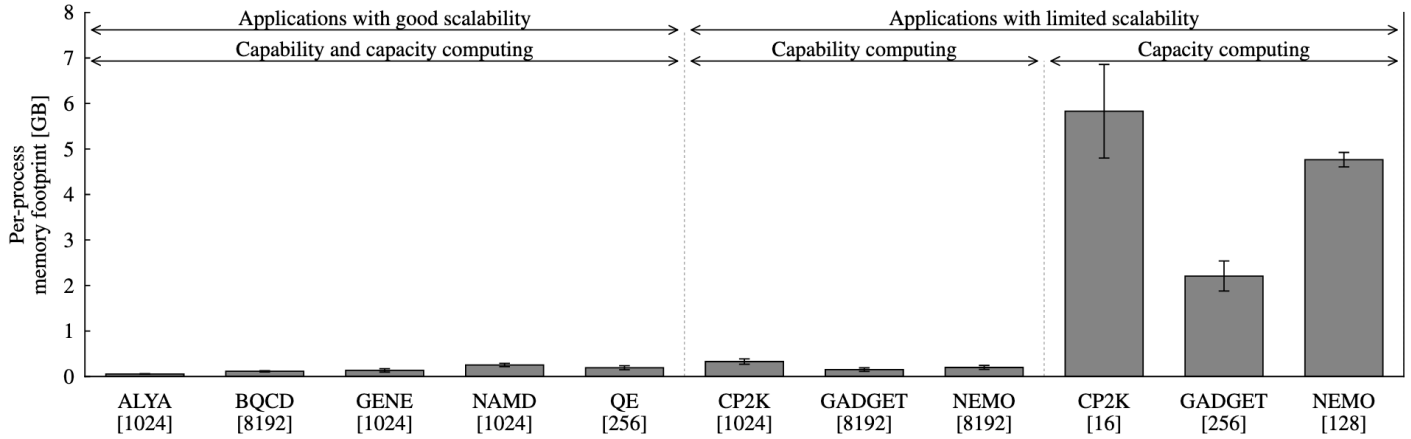
Paul Carpenter
Senior Researcher

3/6/2022

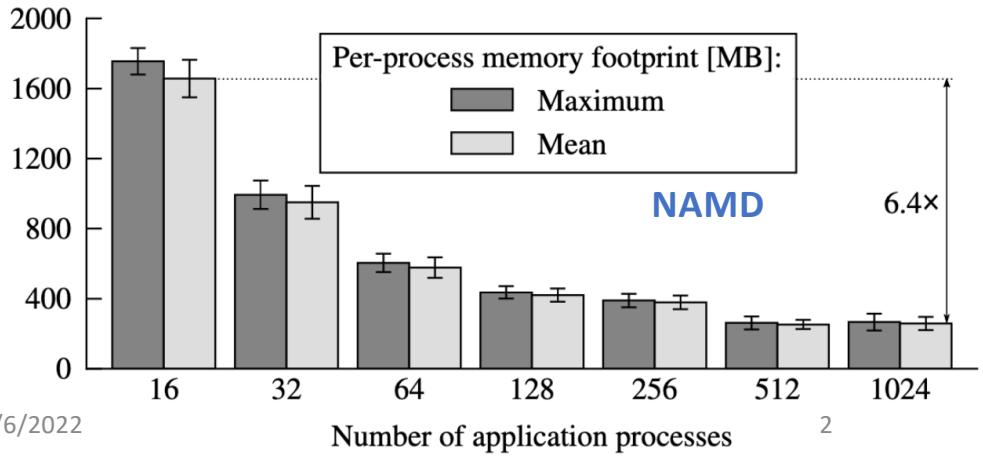
COMPSYS Workshop, IPDPS 2022

HPC application memory capacity requirements vary dramatically (apps)

- PRACE UEABS: Benchmark suite of applications for procurement of largest HPC systems in Europe



- Strong scalability (same problem size on varying #nodes)
  - Problem size per node varies dramatically

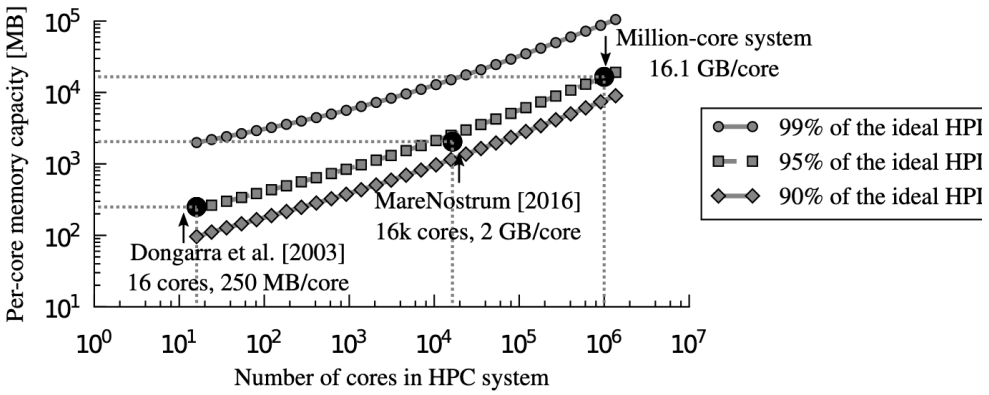


*** Zivanovic et al, Main Memory in HPC: Do We Need More, or Could We Live with Less? ACM Transactions on Architecture and Code Optimization, March 2017**

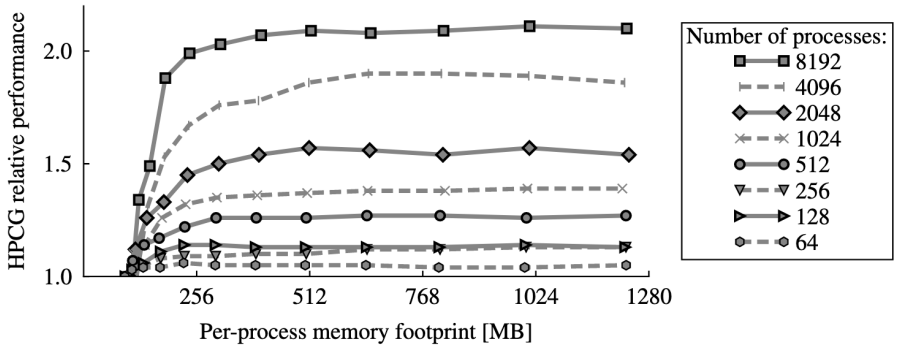
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

HPC application memory capacity requirements vary dramatically (benchmarks)

- Benchmarking runs need large problem sizes to stress the machine

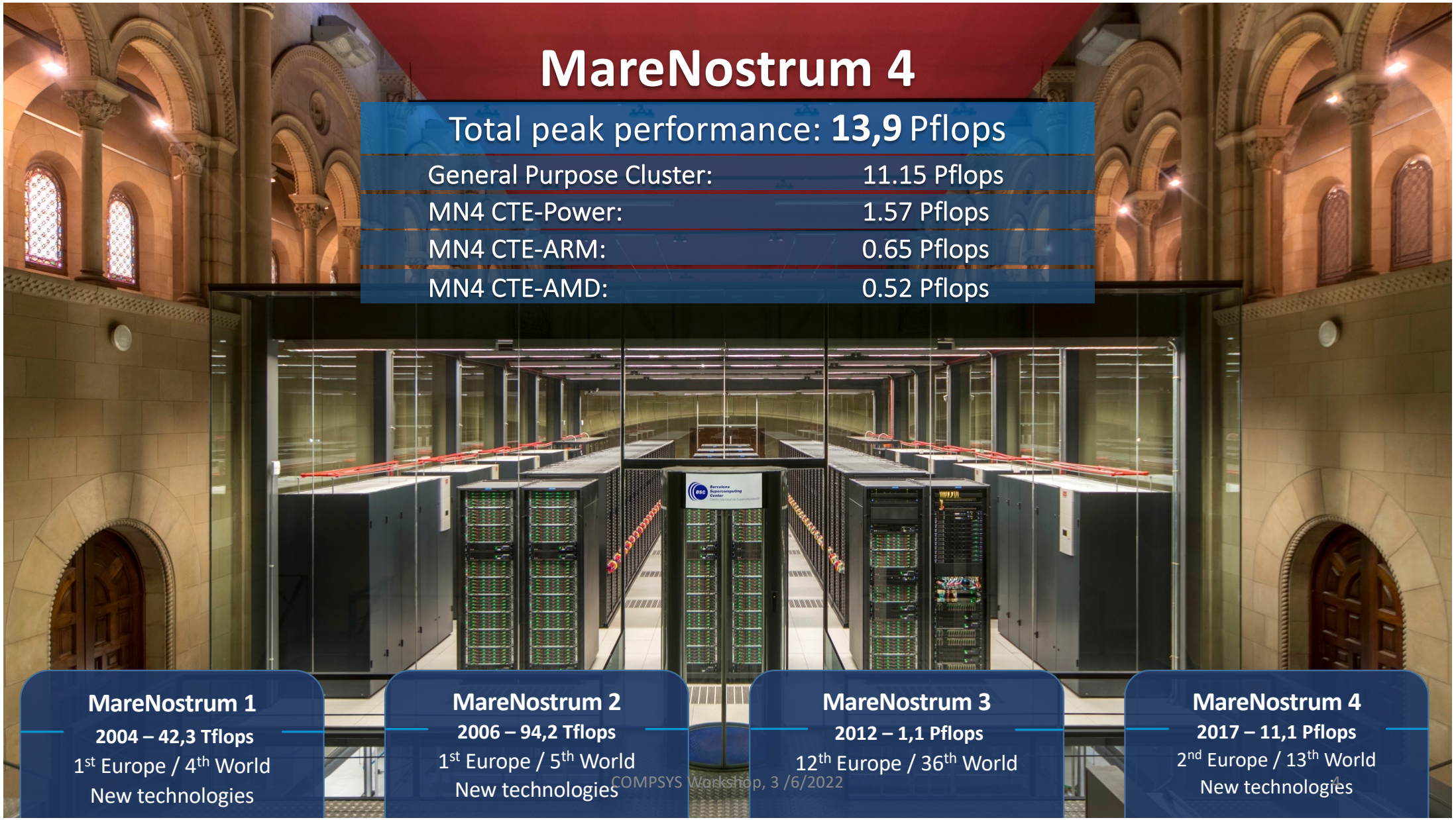


HPL scaling (model)



HPCG scaling (model)

* Zivanovic et al, Main Memory in HPC: Do We Need More, or Could We Live with Less? ACM Transactions on Architecture and Code Optimization, March 2017 Article No. 3


MareNostrum 4
Total peak performance: 13,9 Pflops
General Purpose Cluster: 11.15 Pflops
MN4 CTE-Power: 1.57 Pflops
MN4 CTE-ARM: 0.65 Pflops
MN4 CTE-AMD: 0.52 Pflops
MareNostrum 1
2004 – 42,3 Tflops
1st Europe / 4th World
New technologies
MareNostrum 2
2006 – 94,2 Tflops
1st Europe / 5th World
New technologies
MareNostrum 3
2012 – 1,1 Pflops
12th Europe / 36th World
MareNostrum 4
2017 – 11,1 Pflops
2nd Europe / 13th World
New technologies

Handling varying memory demands: state of the art: Large and small memory nodes in MareNostrum 4

System Overview ↩

MareNostrum4 is a supercomputer based on Intel Xeon Platinum processors from the Skylake generation. It is a Lenovo system composed of SD530 Compute Racks, an Intel Omni-Path high performance network interconnect and running SuSE Linux Enterprise Server as operating system. Its current Linpack Rmax Performance is 6.2272 Petaflops.

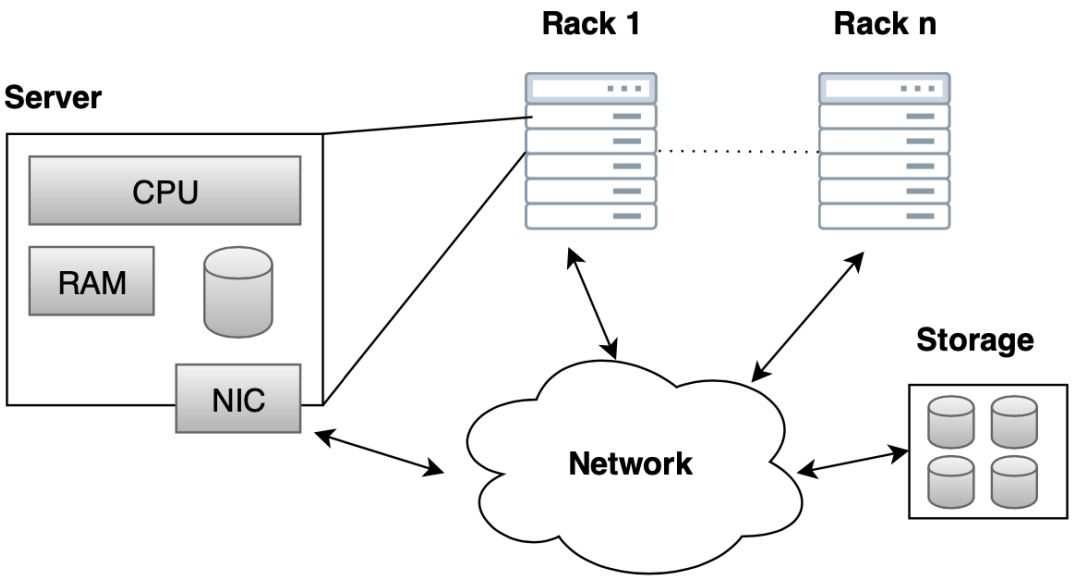
This general-purpose block consists of 48 racks housing 3456 nodes with a grand total of 165,888 processor cores and 390 Terabytes of main memory. Compute nodes are equipped with:

- 2 sockets Intel Xeon Platinum 8160 CPU with 24 cores each @ 2.10GHz for a total of **48 cores per node**
- L1d 32K; L1i cache 32K; L2 cache 1024K; L3 cache 33792K
- 96 GB of main memory **1.880 GB/core**, 12x 8GB 2667Mhz DIMM (216 nodes high memory, 10368 cores with 7.928 GB/core)
- 100 Gbit/s Intel Omni-Path HFI Silicon 100 Series PCI-E adapter
- 10 Gbit Ethernet
- 200 GB local SSD available as temporary storage during jobs ($TMPDIR=/scratch/tmp/[jobid])

https://www.bsc.es/user-support/mn4.php

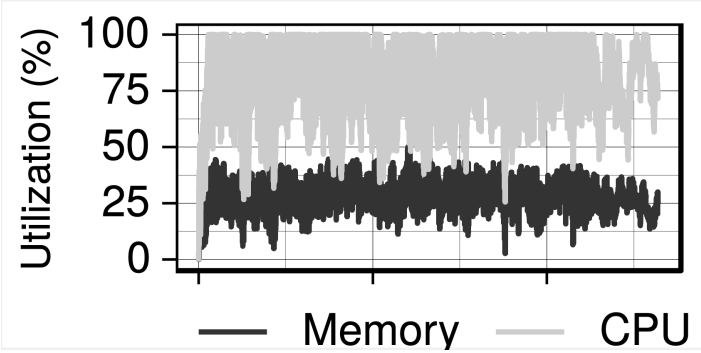
Stranded resources in cluster architecture

- >90% of TOP500 HPC systems are clusters
- Cluster is large number of servers interconnected by network
- Each server has separate memory hierarchy
  - Memory capacity unused by the CPUs on a node cannot be used by CPUs on another node

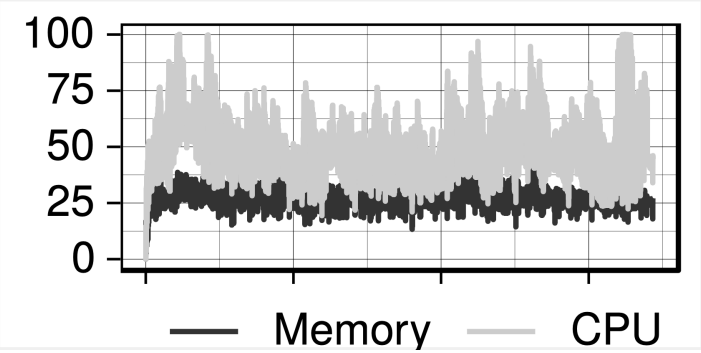


Stranded resources (simulation)

- Large and small memory nodes work well when system matches jobs
  - Left-hand plot: in this example bottlenecked by CPU availability
- But not so well when system does not match job mix
  - Right–hand plot: system memory and CPU utilization **both** low
  - => Resources exist in system but are “stranded”
- Methodology later in presentation



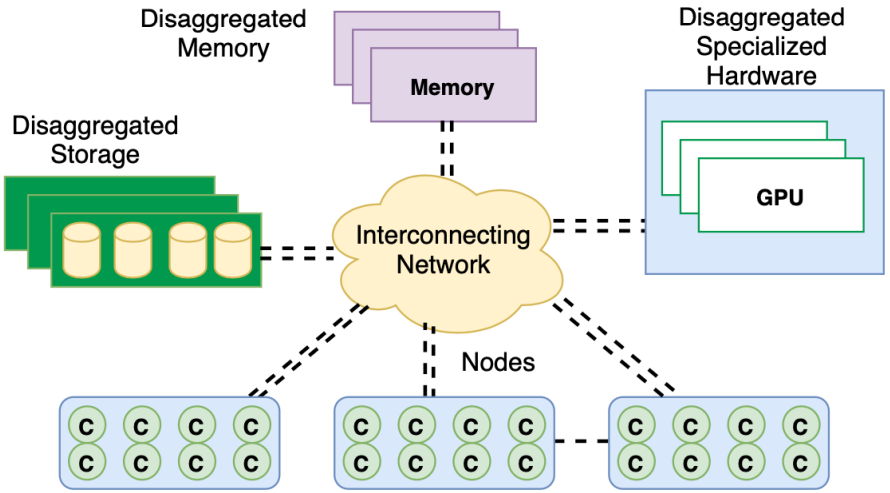
System matches job mix
(25% large nodes, 25% large jobs)



System mismatches job mix
(25% large nodes, 50% large jobs)

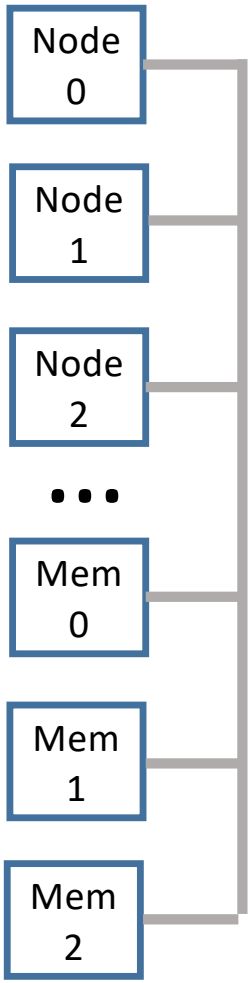
Disaggregated architecture

- Share memory capacity, storage, accelerators among nodes
- Under investigation in academia and industry
- (1) Addresses stranded resources problem
- (2) Addresses coarse granularity of DIMM provisioning
  - Power of two sizing, balanced over memory controller channels



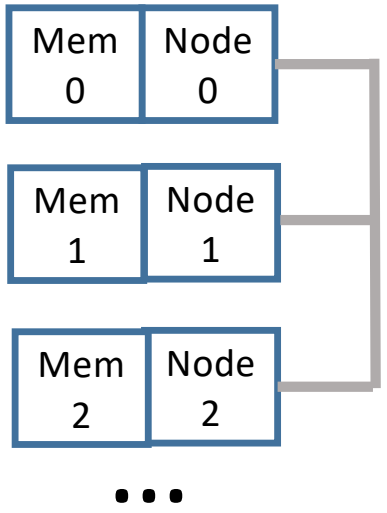
Approach 1: Disaggregated memory (aka centralized)

- Compute and memory completely separate
  - Independent scaling of compute and far memory
  - Independent upgrading (different rates of technology improvement)
  - More “far” memory than single machine
  - Separate failure domains for compute nodes and disaggregated memory
  - Better shareability among processors (equidistant)
- Examples
  - Memory API: The Machine (HPE), OpenFAM [LNCS2019], RAM Area Network (ANL)
  - Paging device: [ISCA2009] [HPCA 2012]
  - Persistent key–value store: FlatStore [ASPLOS 2020], Clover [ATC2020]
  - Persistent memory file system: Octopus [ATC2017], Orion [FAST 2019]
  - Database storage engine: NAM [VLDB 2016]
  - Recoverable persistent data structures: AsymNVM [ASPLOS 2020]
  - OS: LegoOS [OSDI 2018]
  - Full stack prototype: dRedBox [DATE 2018]



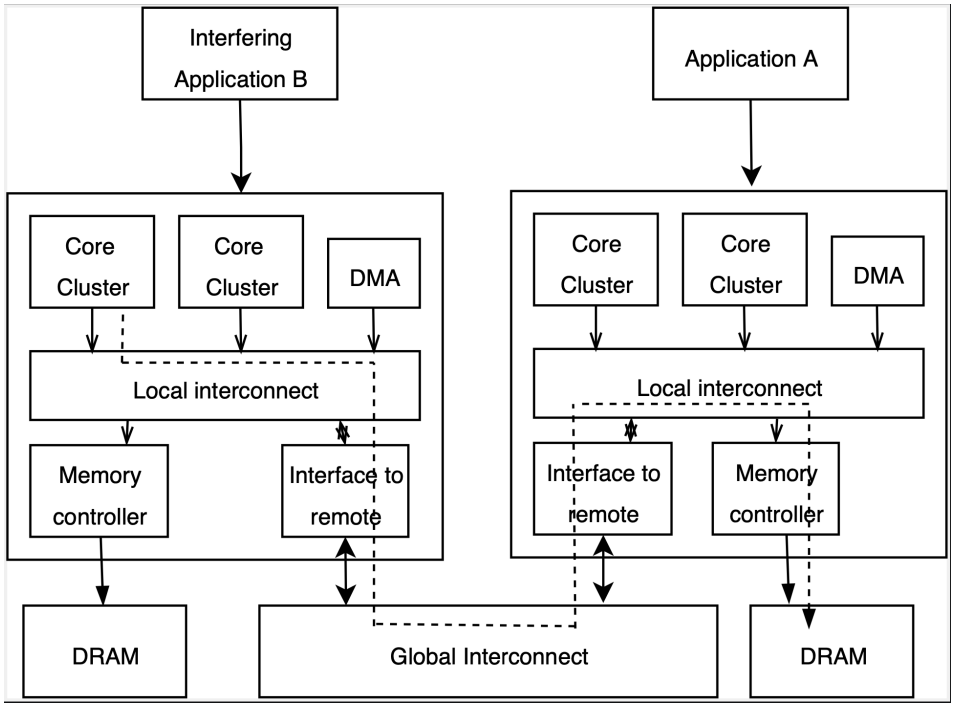
Approach 2: “Remote memory” (aka distributed)

- Nodes can “borrow” unused memory from other machines
  - Main advantage: tight coupling between CPU and its memory
  - No cost/small cost if not using memory disaggregation
- Examples
  - Paging and prefetching: Infiniswap [NSDI 2017], Fastswap [EuroSys 2020], Leap [ATC 2020]
  - Coherence-based memory: Kona [ASPLOS 2021]
  - PGAS: OpenSHMEM
  - Distributed shared memory: FaRM [NSDI 2014], Grappa [ATC 2015], HotPot [SOCC 2017], GAM [VLDB 2018], Concordia [FAST 2021]
  - Memory mapped files: Remote Regions [ATC 2018]
  - Offloaded kernels: StRom [EuroSys 2020]
  - Full stack prototype: ExaNoDe/EuroEXA, ThymesisFlow [MICRO 2020]
- Distinction between local and remote memory
- We use this approach



Model disaggregated memory hierarchy

- Inspired by UNIMEM* from Euroserver, ExaNoDe and EuroEXA projects
- Remote memory access performed through a common Address Space
- We consider capacity sharing among pairs, quads, up to whole system



EURO SERVER

FP7 grant number 610456

ExaNoDe

H2020 grant number 671578

EURO EXA

H2020 grant number 754337

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

- Yves Durand et al. "Euroserver: Energy efficient node for european micro-servers". In: 2014 17th Euromicro Conference on Digital System Design. IEEE. 2014, pp. 206–213

Software is the issue!

- System software
  - OS support, MPI and PGAS communication, filesystems
  - Dynamic changes and migration to avoid memory fragmentation (topic of ”malleability” still research)
- Resource management
  - Batch scheduler: combinations of resources: memory types, memory capacities, accelerators, etc.
- Hardware abstraction layer
- Isolation among applications
  - HPC used to exclusive access and non-blocking networks
  - Memory bandwidth interference on one node may affect whole job, possibly with hundreds or thousands of waiting nodes
- Programming models
  - Must be open and vendor independent
  - Does application distinguish near and far memory? (when allocating, when using)
- Applications
  - How to tolerate increase in memory latency (base and due to contention): Little’s law
  - Applications have lifetimes of decades, domain knowledge embedded in source code
  - Major effort to adopt new technologies

Other aspects of disaggregated memory

- User practices
  - Characterization of jobs: knowing what requirements are (how much memory)
  - Possible change to peer review process to allocate memory–hours, as well as CPU–hours (and energy?)
- Resilience
  - MareNostrum 4 had 67 UEs in two years of operation
  - Estimated loss of 40k node–hours due to aborted jobs (<0.1% of operation) [*]
  - Disaggregated memory may make it worse though.
- Security
  - How strong are the security guarantees?

[*] Isaac Boixaderas, Javier Bartolome, Paul M. Carpenter, Darko Zivanovic, David Vicente, Petar Radojkovic, Sergi More, Marc Casas, and Eduard Ayguade. Cost-Aware Prediction of Uncorrected DRAM Errors in the Field. *International Conference for High Performance Computing, Networking, Storage, and Analysis, SC'20*

Scheduling and allocation problem

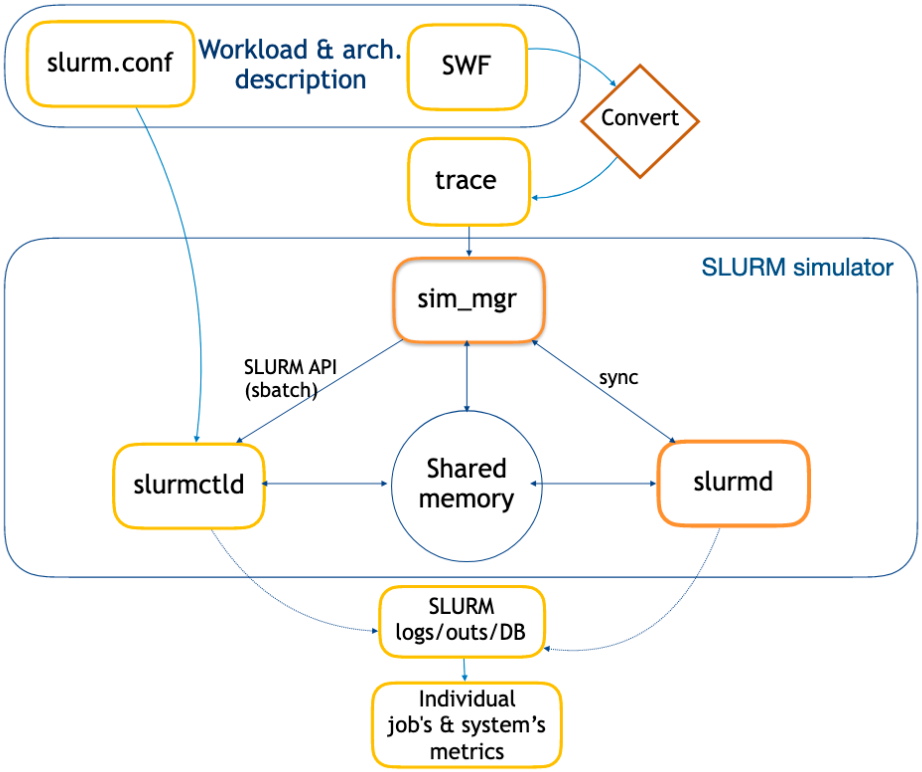
- Develop disaggregated memory aware Slurm job scheduler
  - Only new information is job memory demands at submission time
- Evaluate policy and architectural tradeoffs using simulation

1. Disaggregated memory systems are immature
   - Not yet available (e.g. IBM POWER10)
   - System software immature (e.g. ExaNoDe and EuroEXA)
   - Typically high latencies due to FPGA emulation (e.g. ExaNoDe and EuroEXA, IBM ThymesisFlow)
2. At-scale research in job scheduling cannot use a production machine
   - Would negatively impact service delivered to users

- All results in this presentation were obtained using BSC Slurm simulator

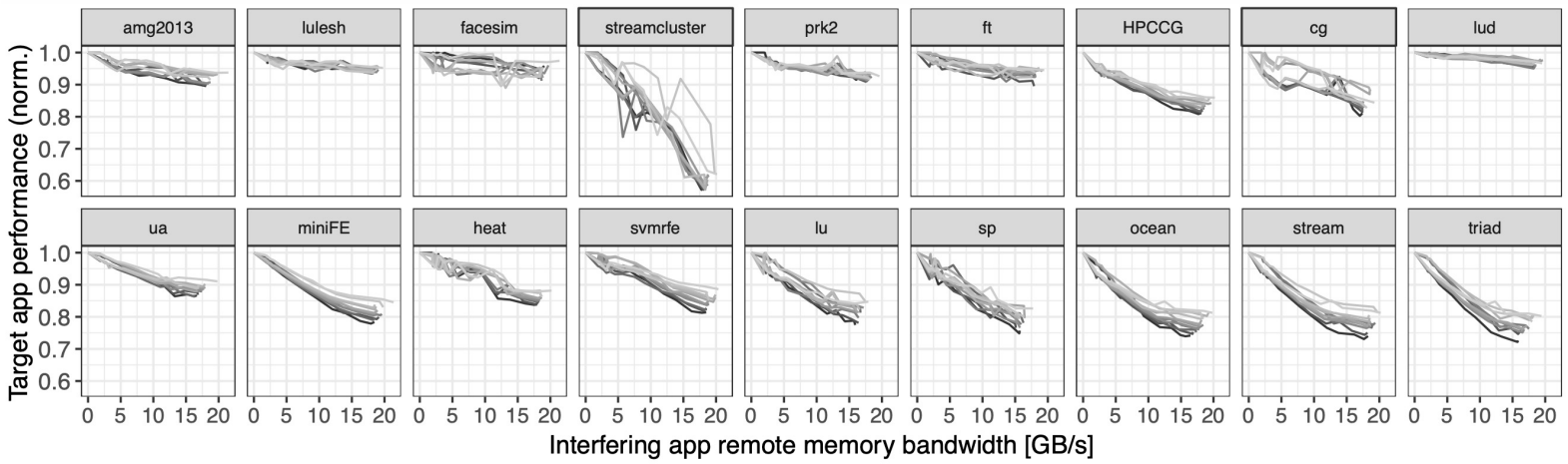
BSC Slurm simulator

- Uses **real Slurm daemons** to capture influence of all Slurm parameters
- Allows fast exploration and evaluation of batch scheduler algorithms
- Input is Standard Workload Format trace (SWF)
- Open-source: https://github.com/BSC-RM/slurm_simulator
- Developed by Marco D'Amico, Ana Jokanovic, Julita Corbalan at BSC



Performance impact of disaggregated memory

- Slurm simulator needs to estimate cost of disaggregated memory: network and memory contention
  - For evaluation, not scheduling policy
- Curve of performance depending on memory interface contention
  - *x*-axis is maximum interference across all processes
  - Curve depends on number of processes with interference close to maximum

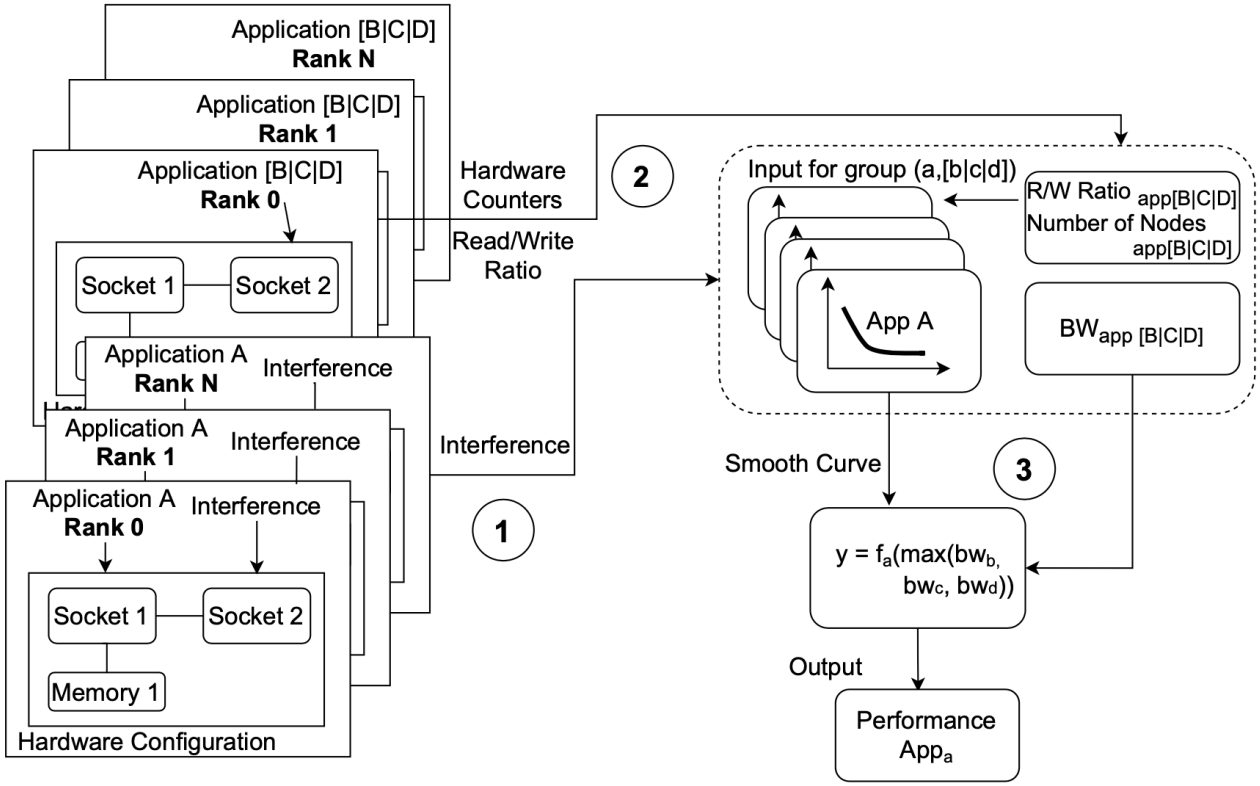


Slowdown curve: quantifies performance impact of interfering memory bandwidth

Model inputs and outputs

Value	Description
Application inputs:	
$f(bw, N)$	Slowdown curve: normalized performance as a function of interfering bandwidth and number of nodes
R/W Ratio	Percentage of memory operations that are reads
bw_{app}	Total memory bandwidth (bytes/s)
Execution inputs:	
bw	Interfering bandwidth
N	Number of interfering applications
Output:	
P_{est}	Estimated normalized performance, typically $0 \leq P_{\text{est}} \leq 1$

Multi-node (MPI program) slowdown methodology



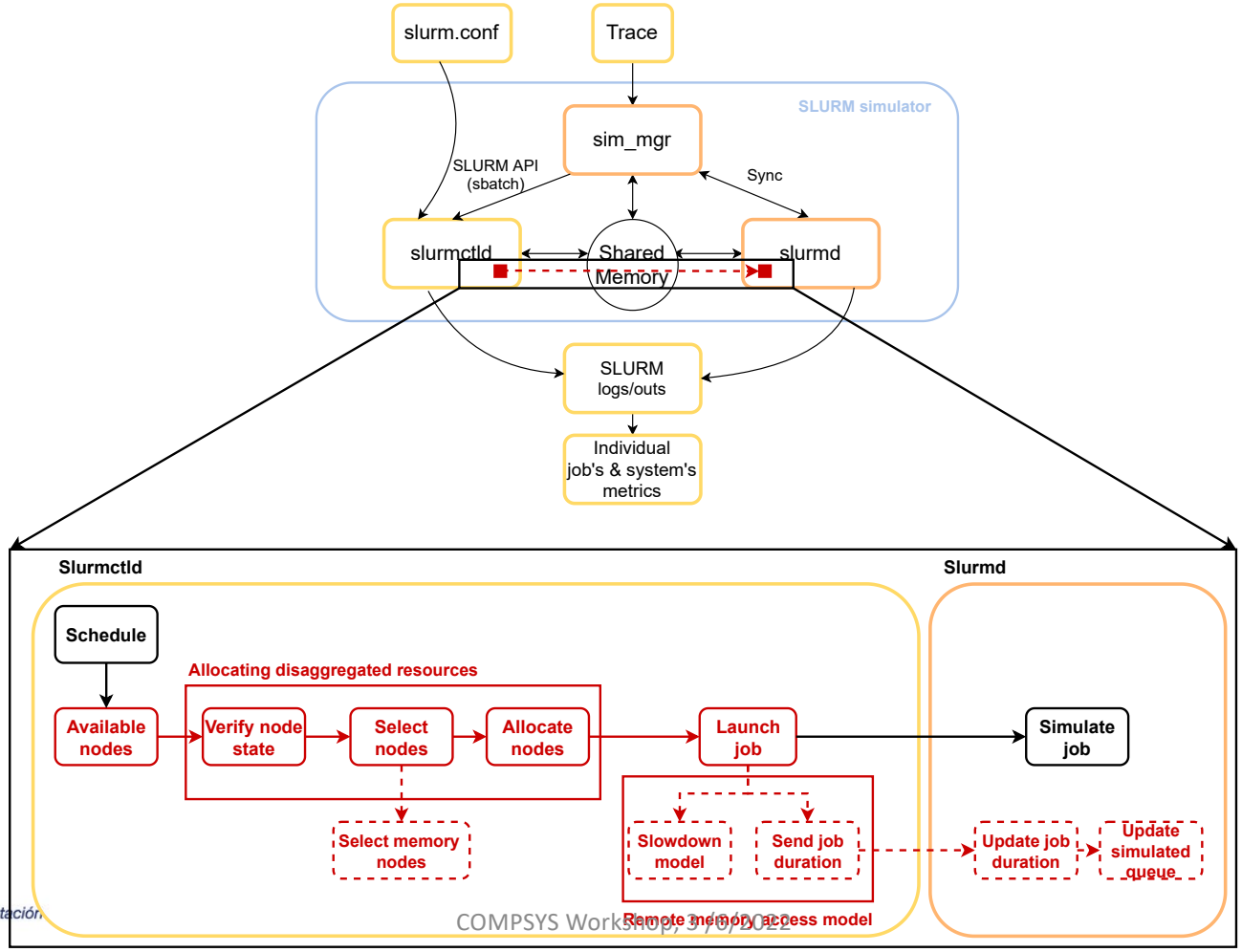
Extending Slurm simulator for disaggregated memory

- We extended Slurm's allocation policy to exploit disaggregation
- Remote memory accesses are model/ed after prior slowdown-based method
- Disaggregation-aware policy: Allocates nodes with enough local memory to satisfy the job's request, otherwise favours node with higher free memory available

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

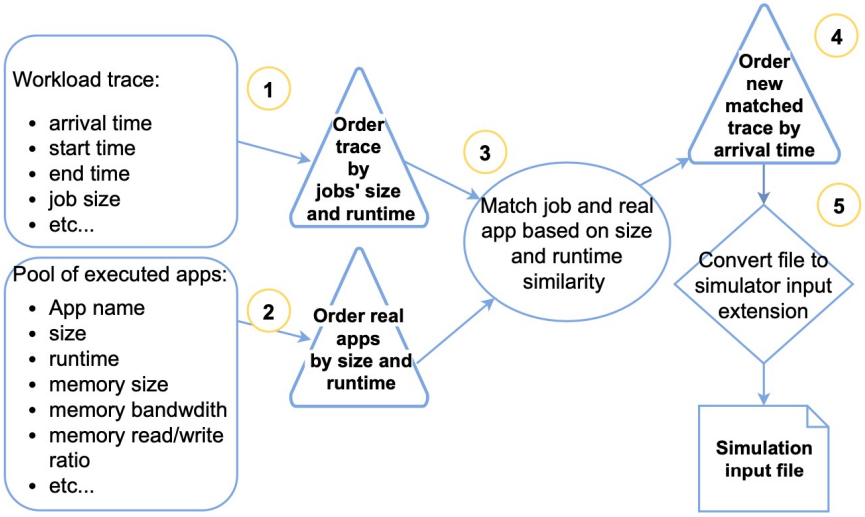
Felippe Vieira Zacarias, Rajiv Nishtala, and Paul Carpenter. "Contention-aware application performance prediction for disaggregated memory systems". In: Proceedings of the 17th CF. 2020, pp. 49–59

Extending Slurm simulator for disaggregated memory II



Experimental methodology

- We generated synthetic workloads using CIRNE Model
- And augmented it with real application data



- We set up different configurations to explore heterogeneity in job requirements and node capacities

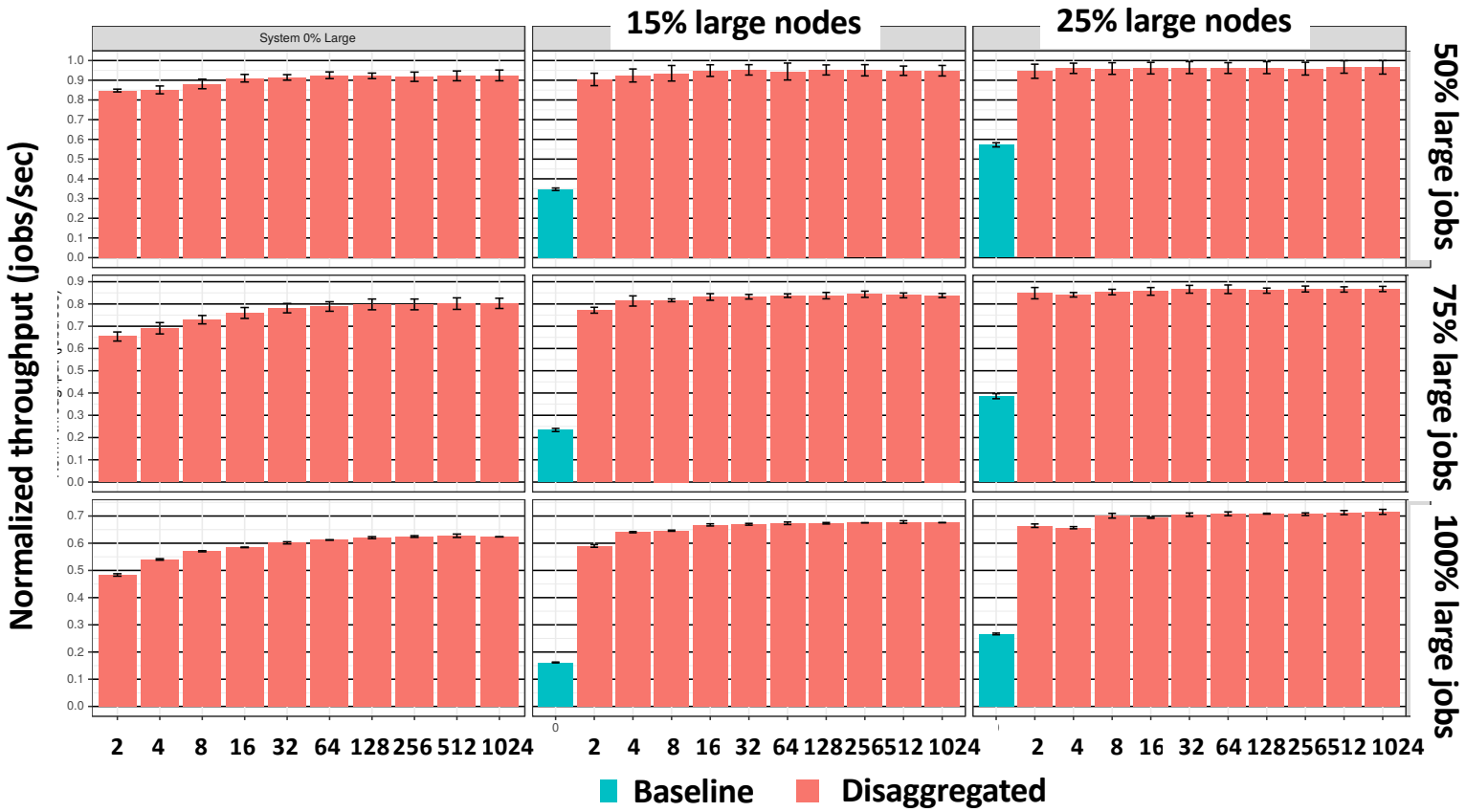
Configuration parameter	Value(s)
System size	1024 nodes
Number cores per node	32
Memory per node	32 GB, 64 GB
Allocation policy	Baseline, disaggregated
Scheduling policy	Backfill
Queue and Backfill size	100
Backfill and Scheduling interval	30 s
Heterogeneous system ratio: % Large nodes	0, 15, 25, 50, 75, 100

Large and small memory job characteristics

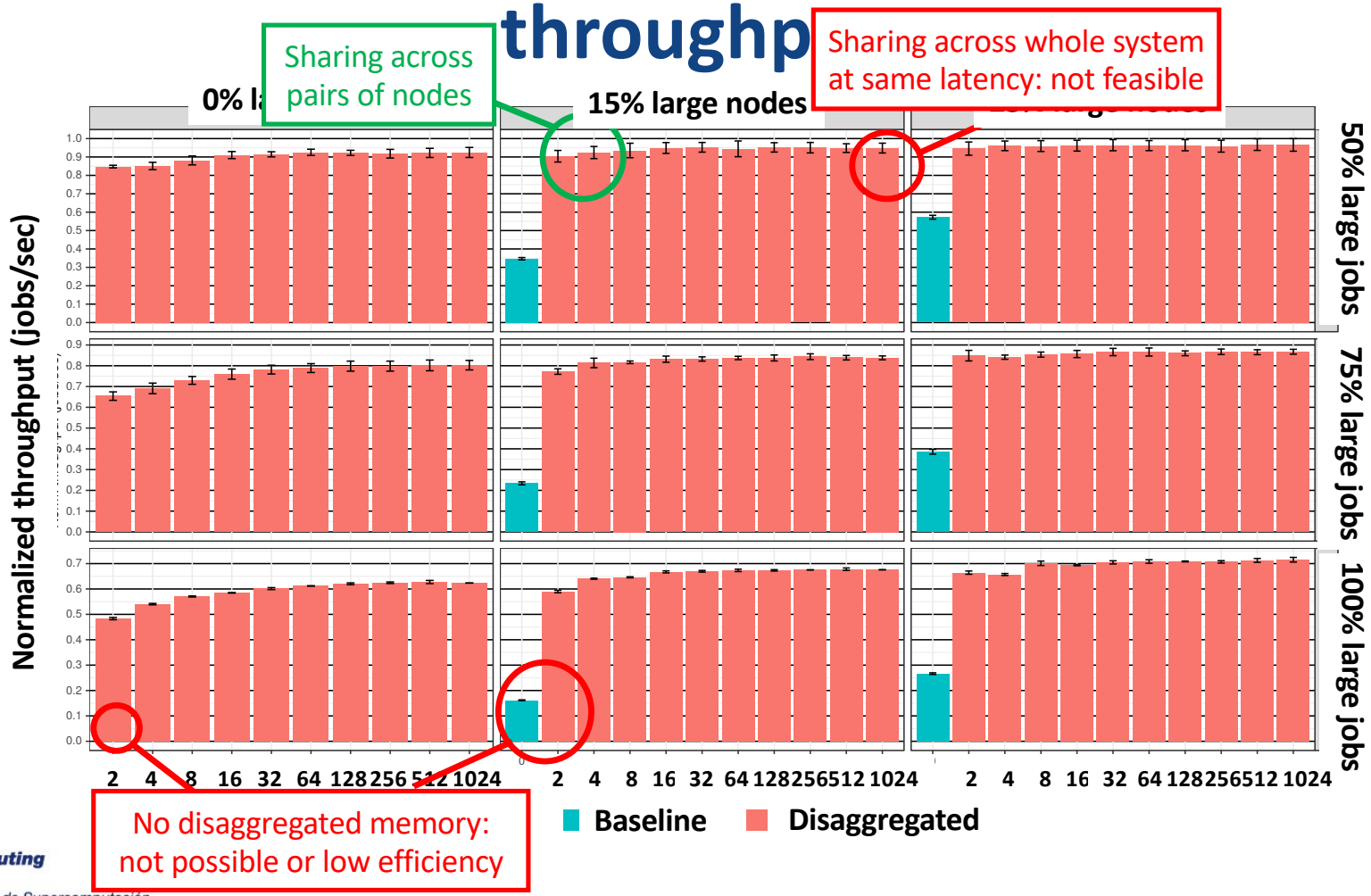
- Normal jobs fit in 32 GB standard node
- Large jobs do not fit
- Composite traces, e.g. 25% large have jobs taken from large and small with appropriate probabilities

	Normal Jobs		Large Jobs	
Metric	**Memory (GB)**	**Node–hours**	**Memory (GB)**	**Node–hours**
Min	0.12	0.0	33.0	0.0
1st Qu.	1.7	0.85	48.2	0.0
Avg	6.2	52.6	48.5	24.9
3rd Qu.	3.8	15.0	49.8	2.1
Max	27.6	6412	49.8	3659.0

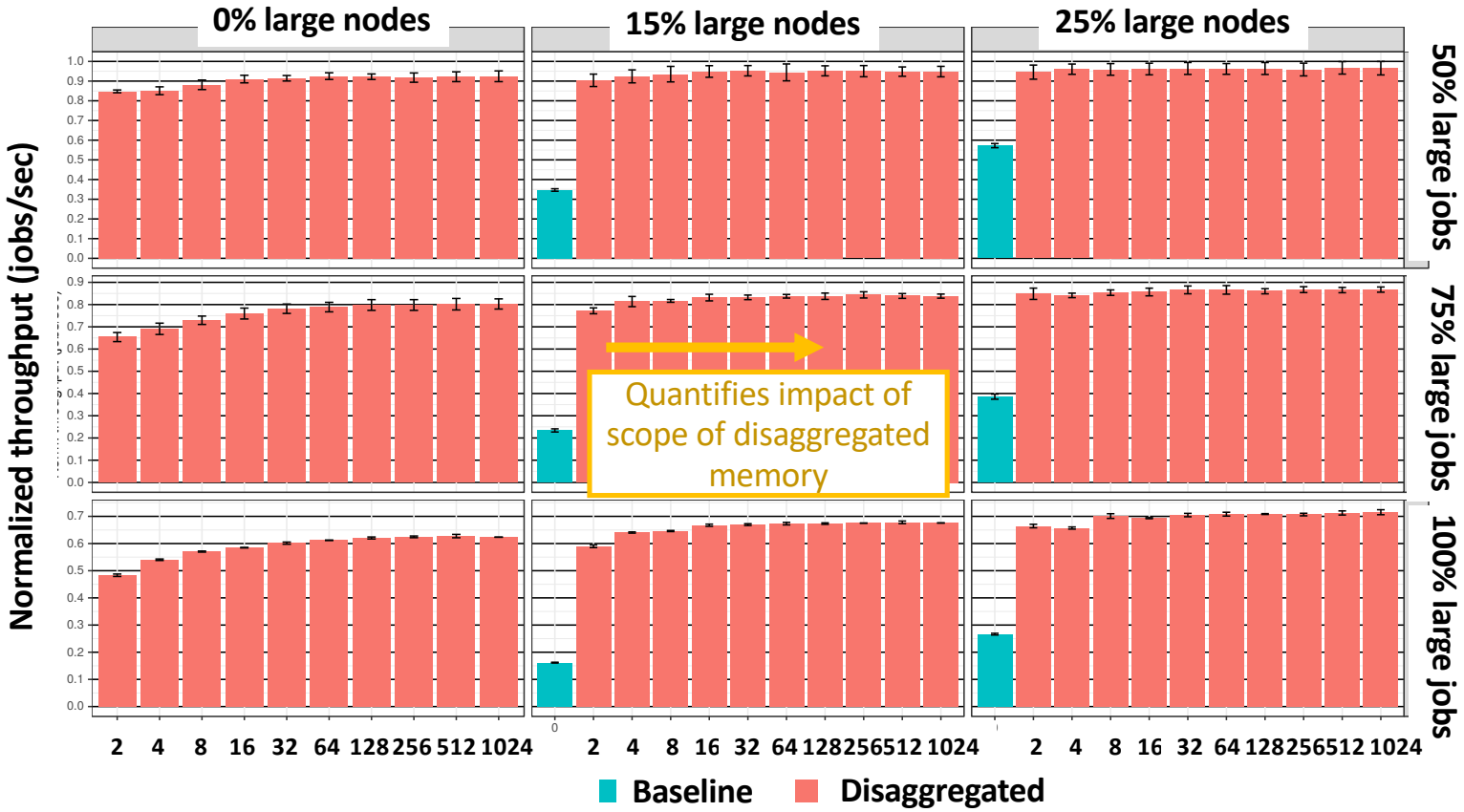
Effect of disaggregated memory group size on throughput



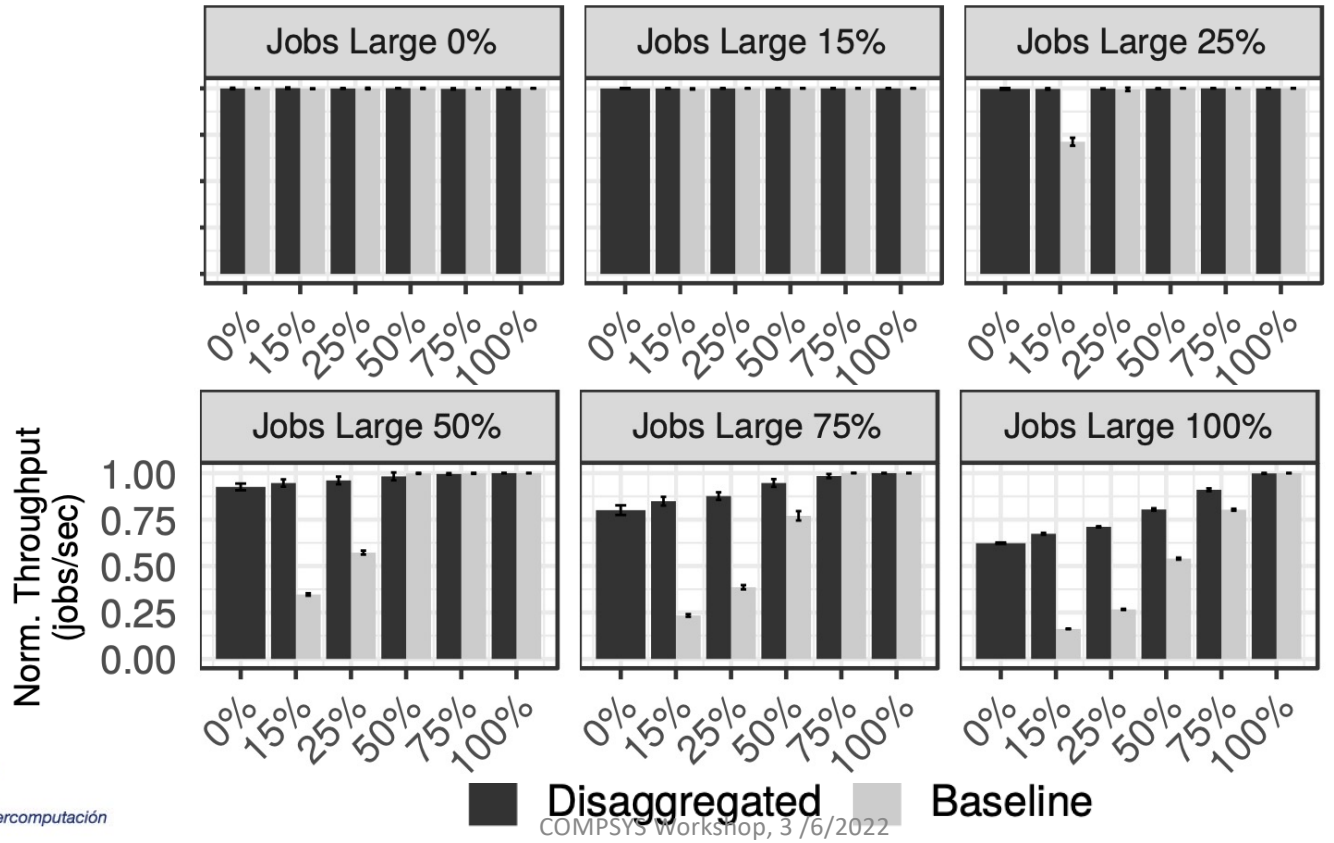
Effect of disaggregated memory group size on throughp



Effect of disaggregated memory group size on throughput

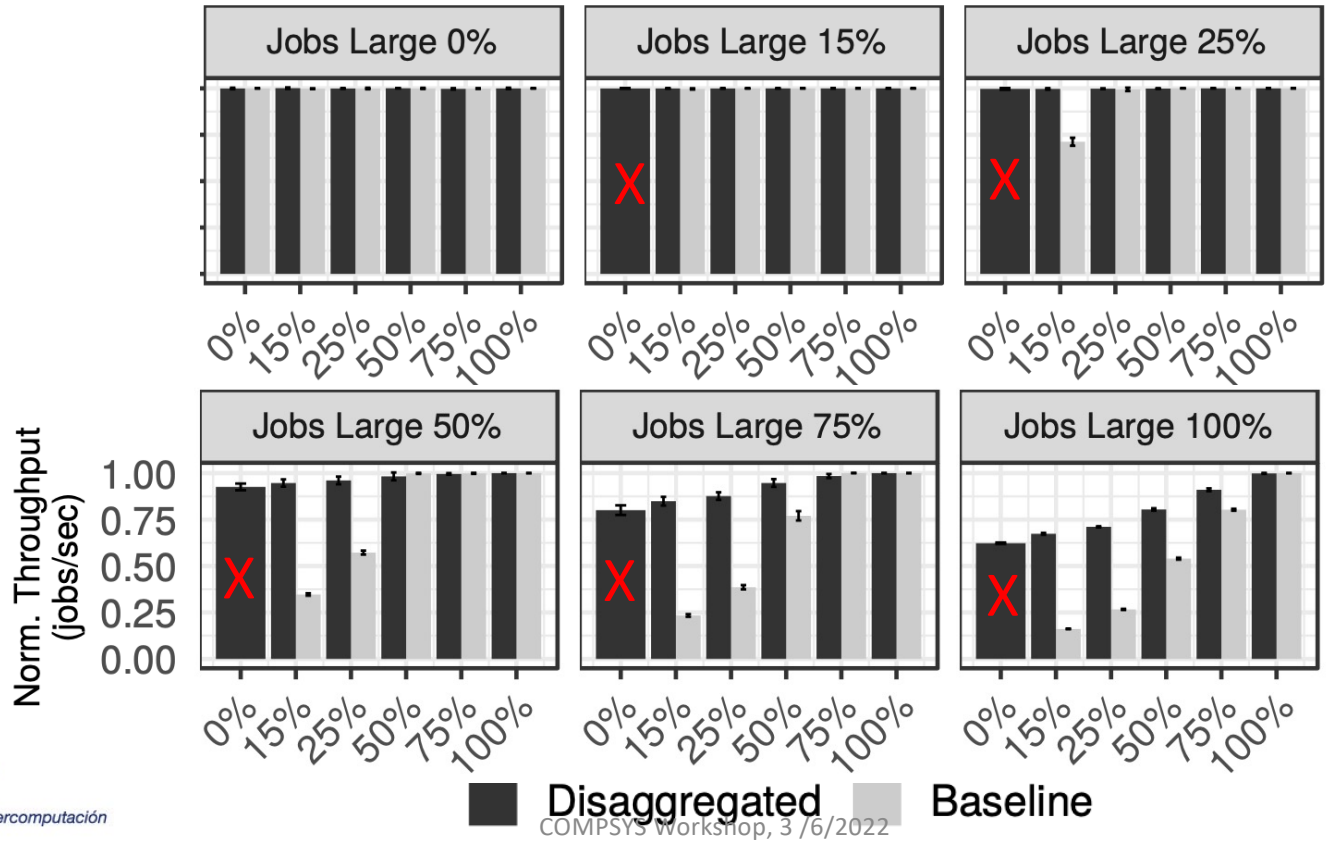


Normalized throughput for simulated systems running various job mixes



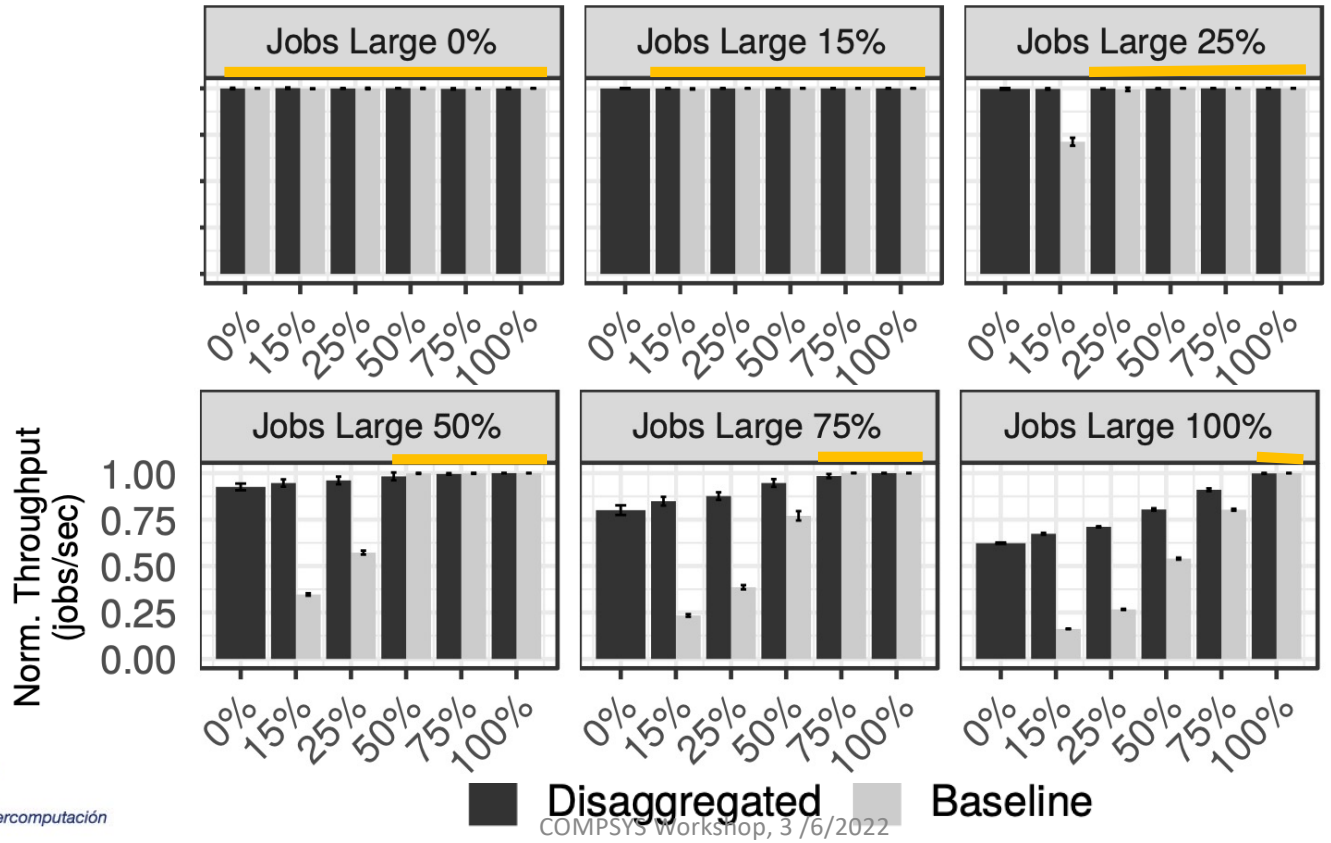
Normalized throughput for simulated systems running various job mixes

- Baseline cannot run large jobs on 0% large system



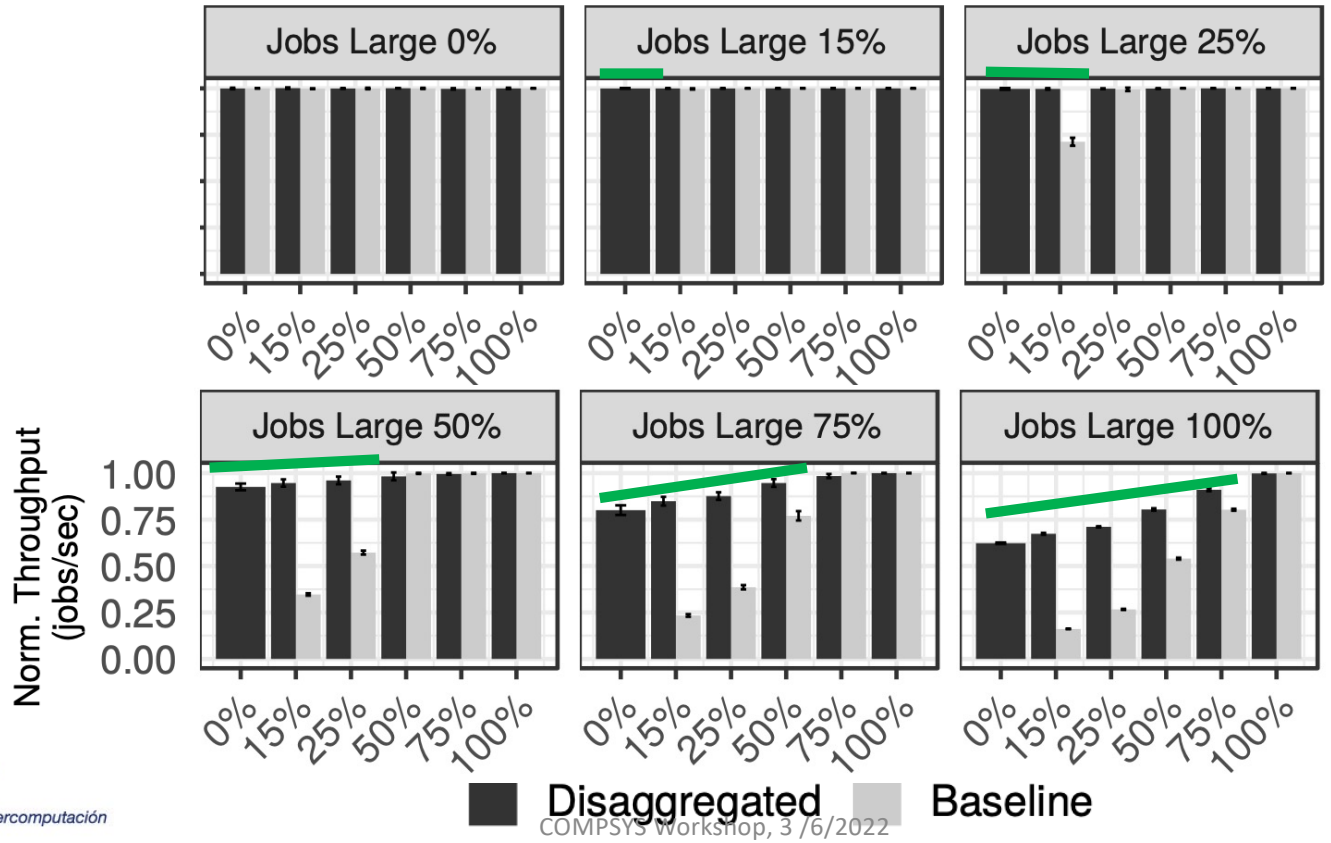
Normalized throughput for simulated systems running various job mixes

- Disaggregated approach matches baseline when sufficient memory



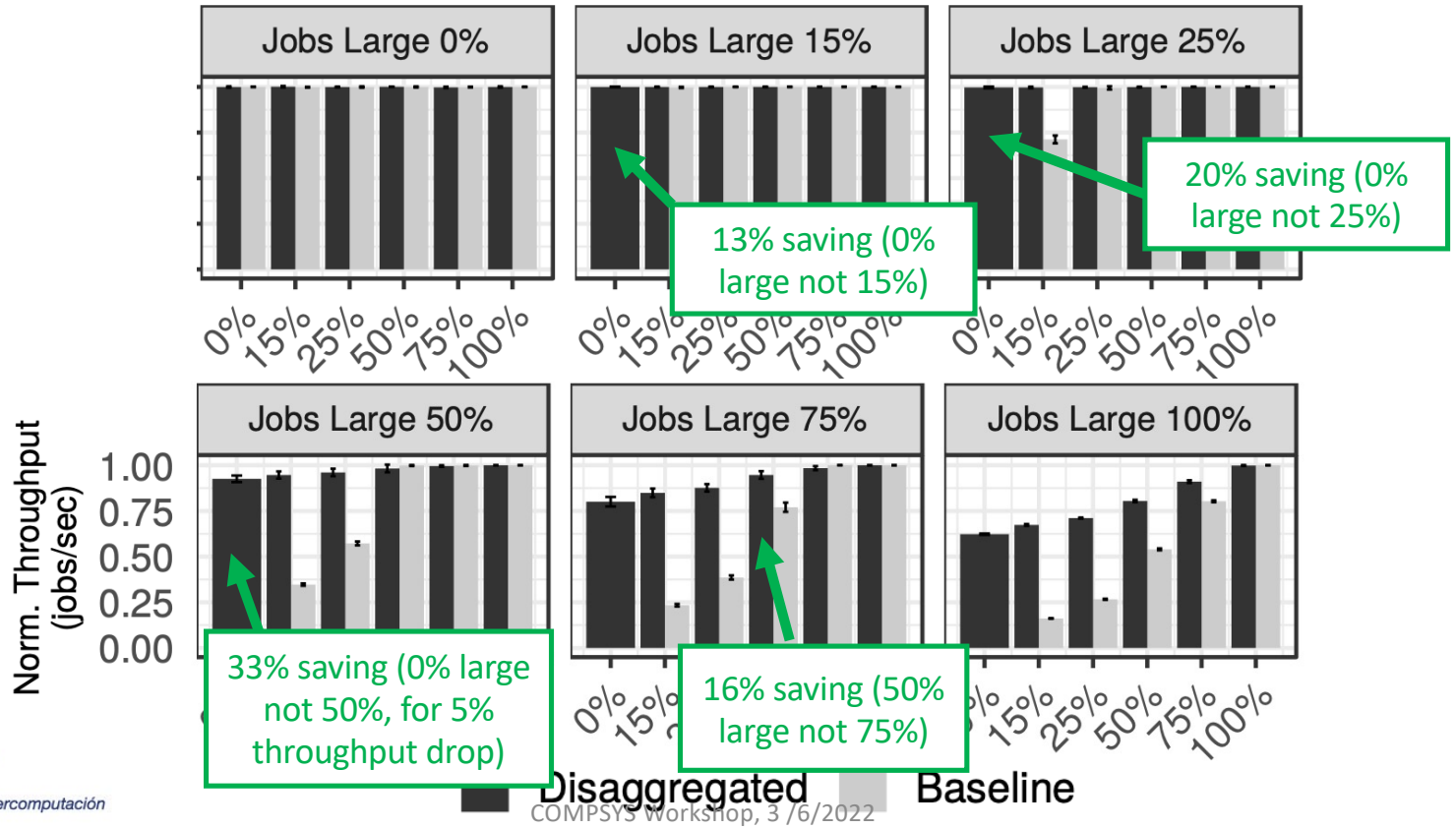
Normalized throughput for simulated systems running various job mixes

- Disaggregated approach has benefit when memory constrained



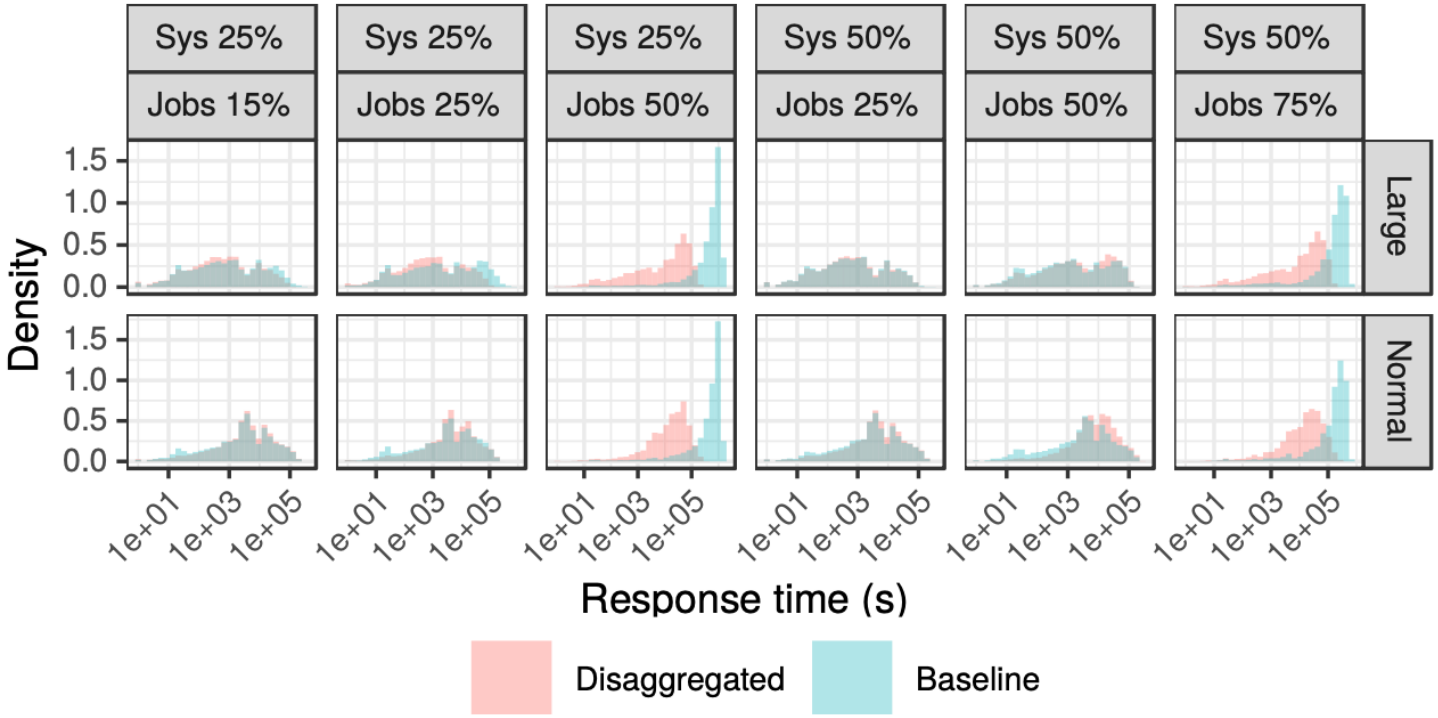
Normalized throughput for simulated systems running various job mixes

- Disaggregated approach gets similar performance with less memory



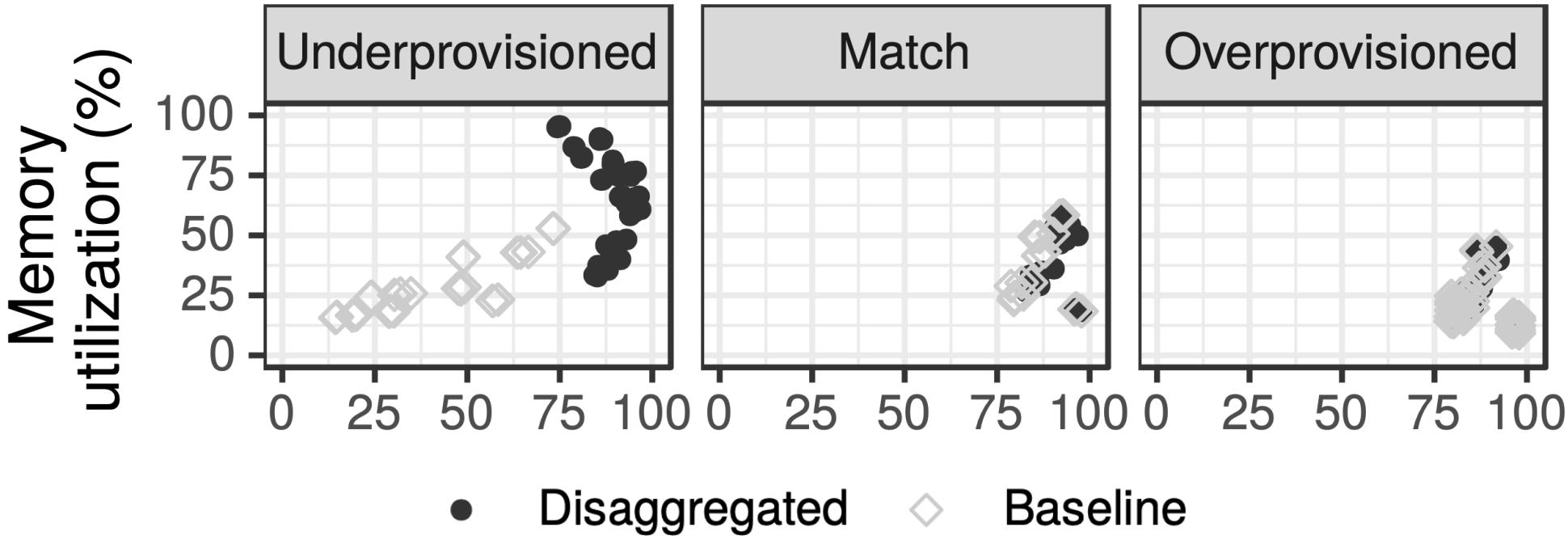
Response time* distribution for large and normal jobs

- Large and normal jobs benefit roughly equally

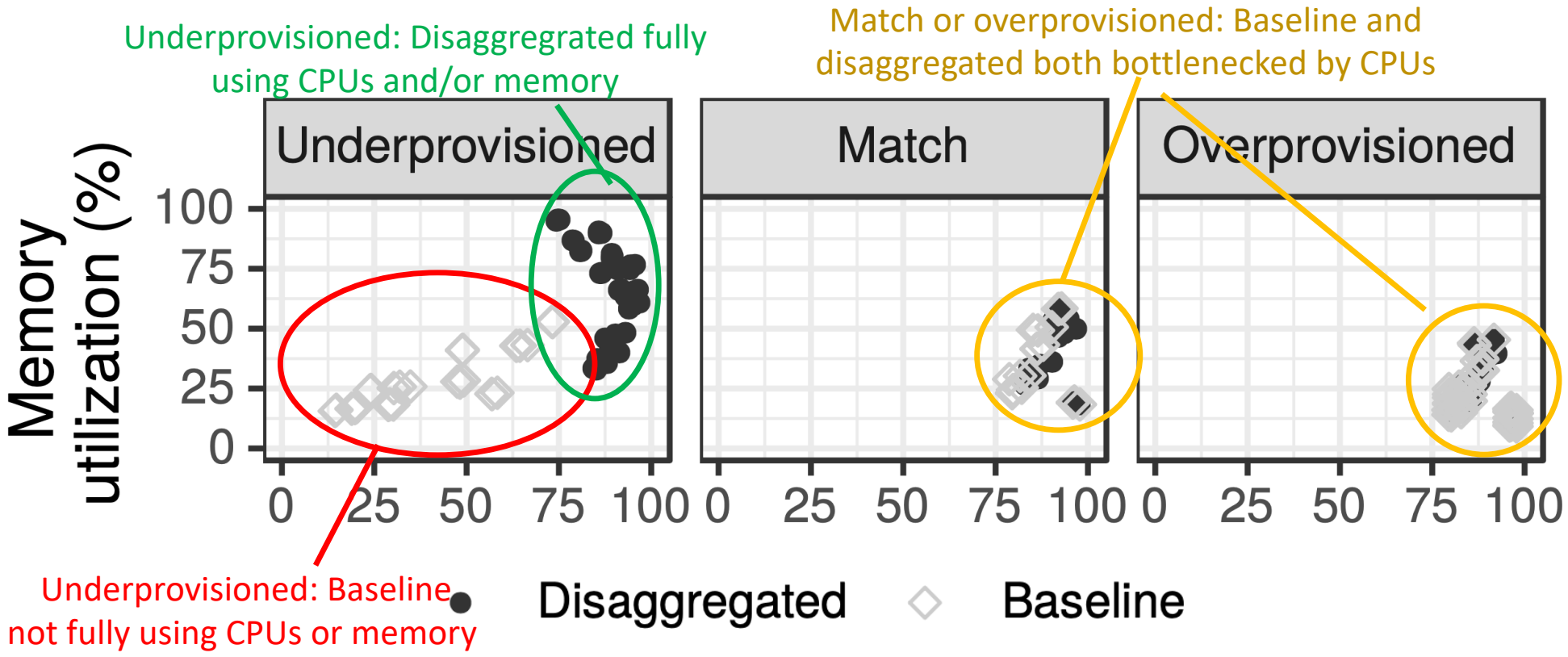


* Response time is queuing time plus runtime

Scatter graph of average memory and CPU utilization

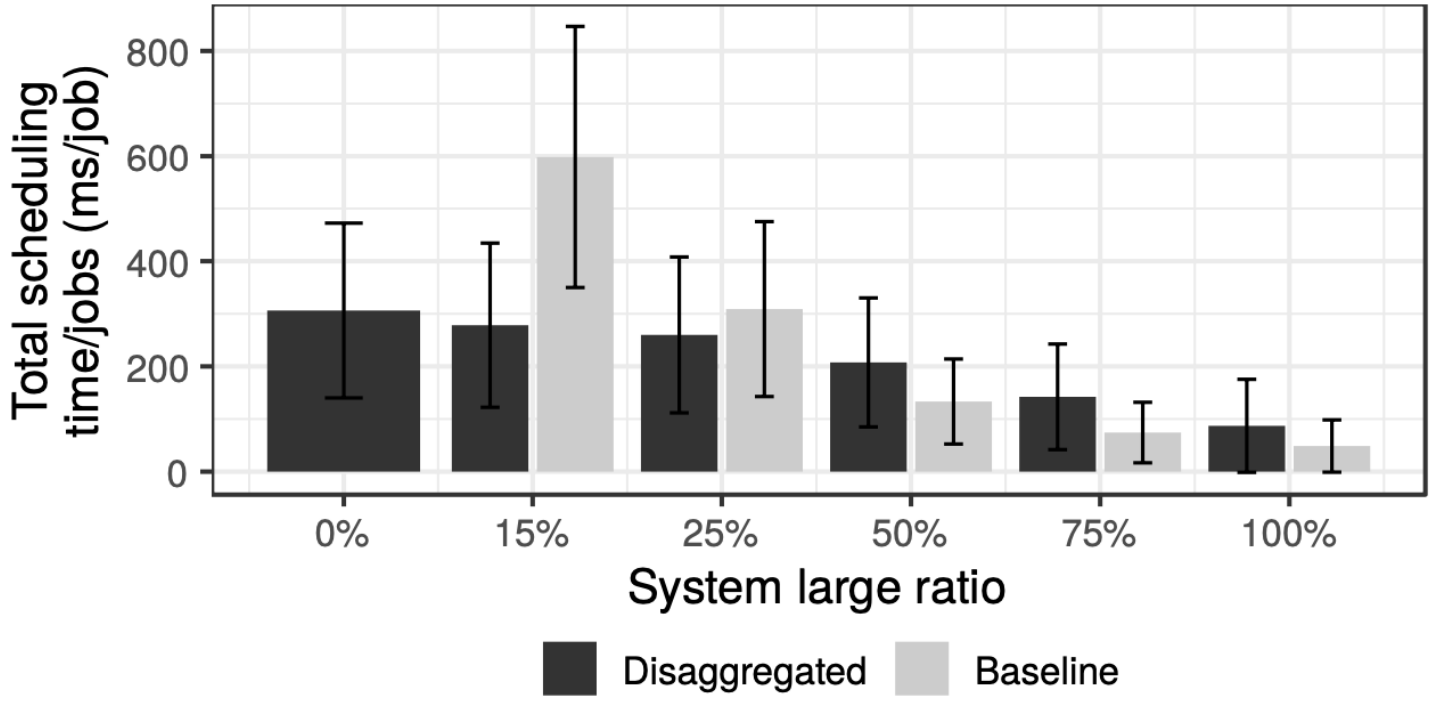


Scatter graph of average memory and CPU utilization



Scheduling time per job

- Approx. 2x slower per job when no benefit (right-hand side)
- Approx. 2x faster per job when there is a benefit (left-hand side)
  - Due to fewer attempts to backfill each job

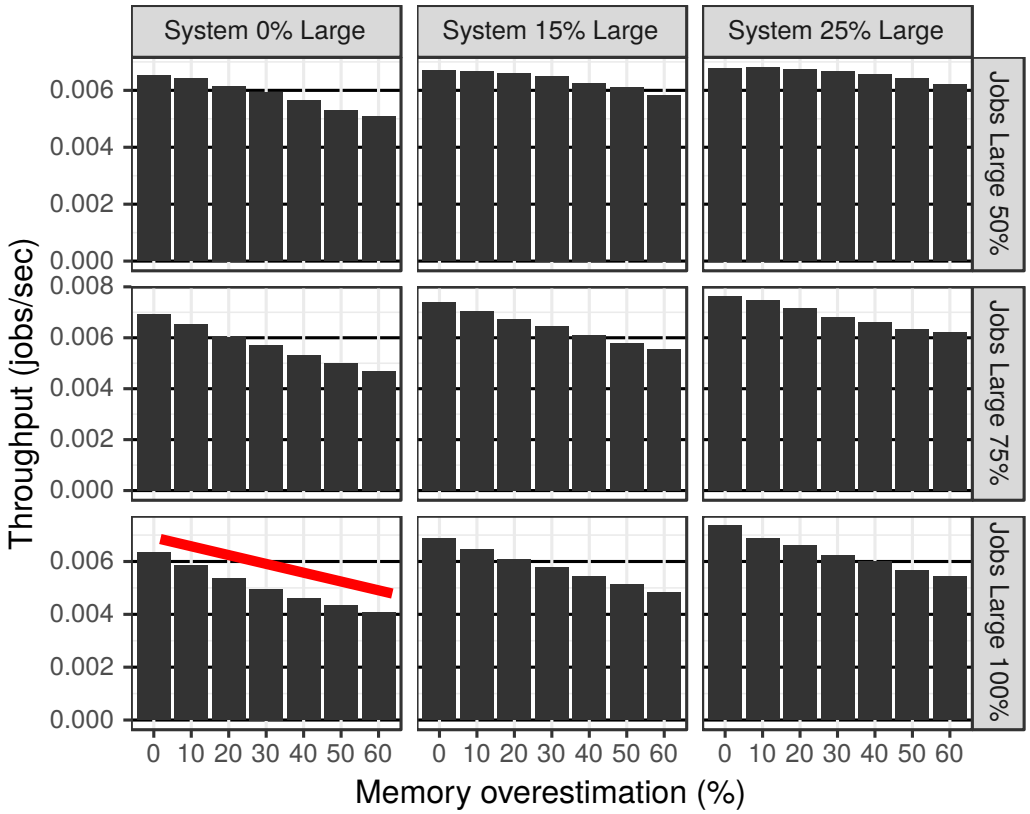


Problem: User must specify max memory demands

- HPC schedulers typically assign resources to jobs *statically*
- Users are required to *estimate job memory requirements* at submission time
  - Too low: jobs may be cancelled/killed
  - Too high: may waste HPC resources
- Key point #1: How important is accuracy in memory demands to system throughput and response time?
- Key point #2: What is user's incentive to be accurate, in terms of individual job response time?

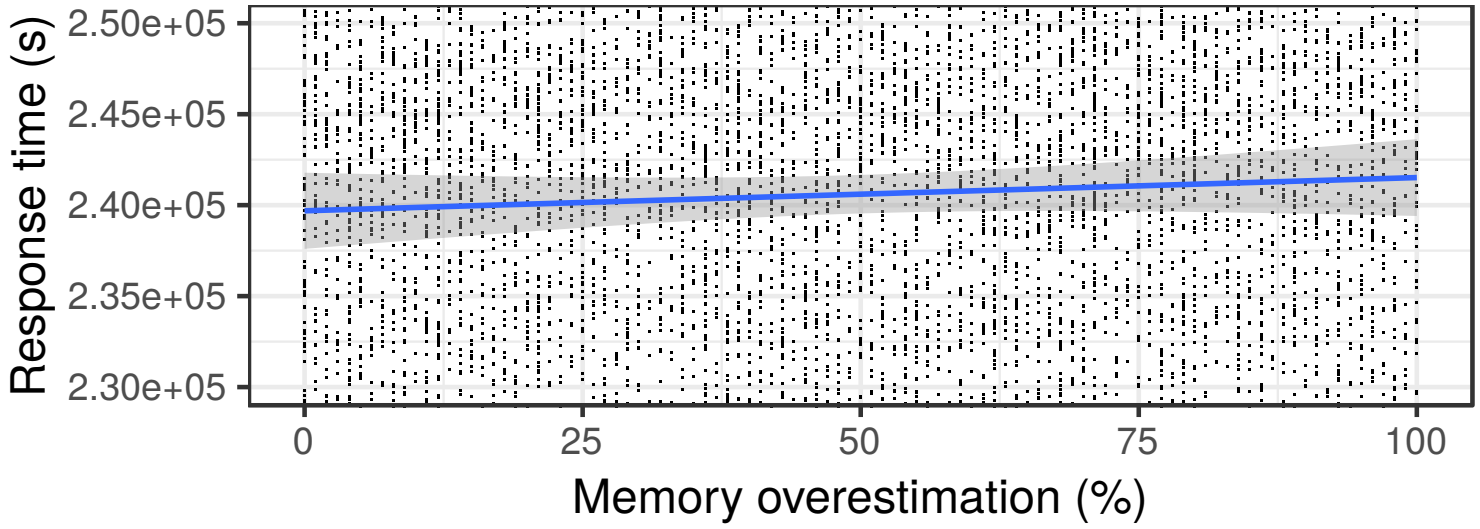
System throughput effect of overestimation

- Degradation in system throughput increases with system and job mix mismatch
- Up to 25% reduction in throughput at 60% increase in demands
- **Conclusion: System throughput is sensitive to memory estimates**



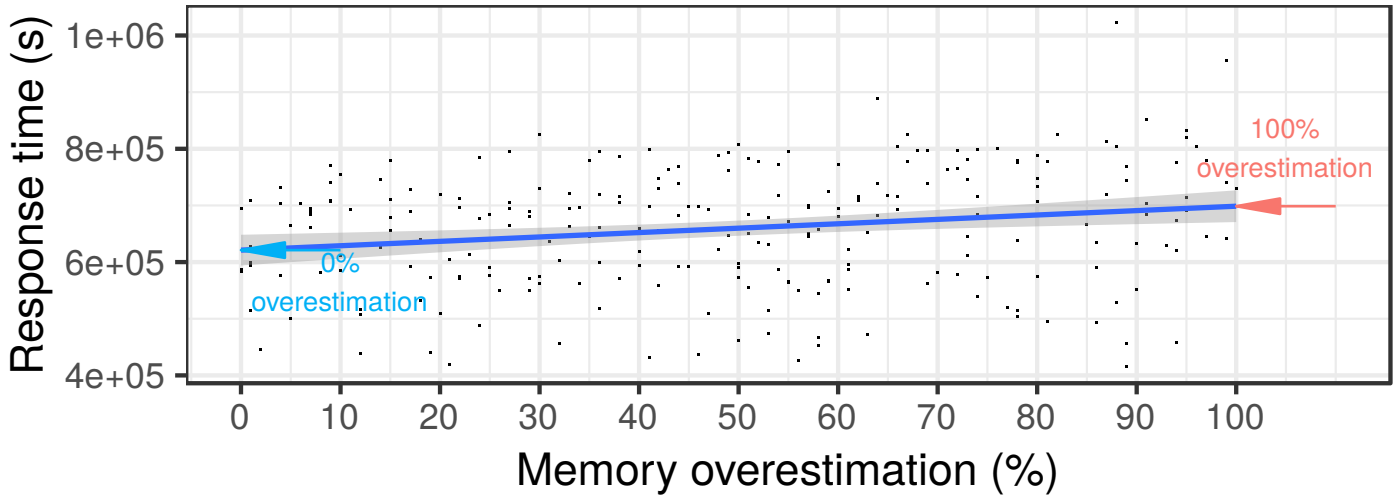
Implications of memory demands - Individual Job

- User's experience: how will memory demands affect my job response time?
- Plot single job response time as a function of memory overestimation, everything else constant
  - Not practical
- We apply a Correlation analysis
  - Running the original trace several times
  - Demands are uniformly-random overestimated between 0% and 100% for each job



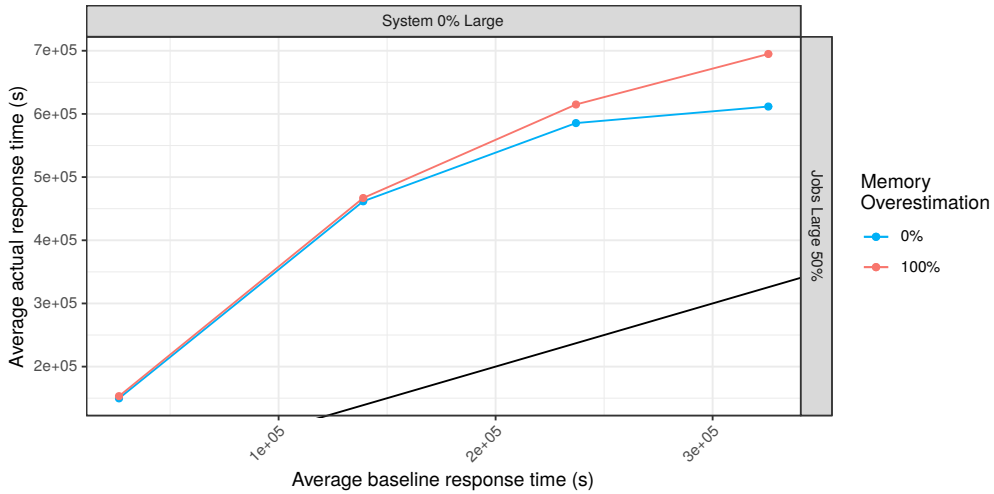
Implications of memory demands - Individual Job II

- Jobs are filtered considering a fixed interval of the baseline response time
  - The result shows the trend line correlating the response time and the memory overestimation
  - There is still significant noise to notice the difference



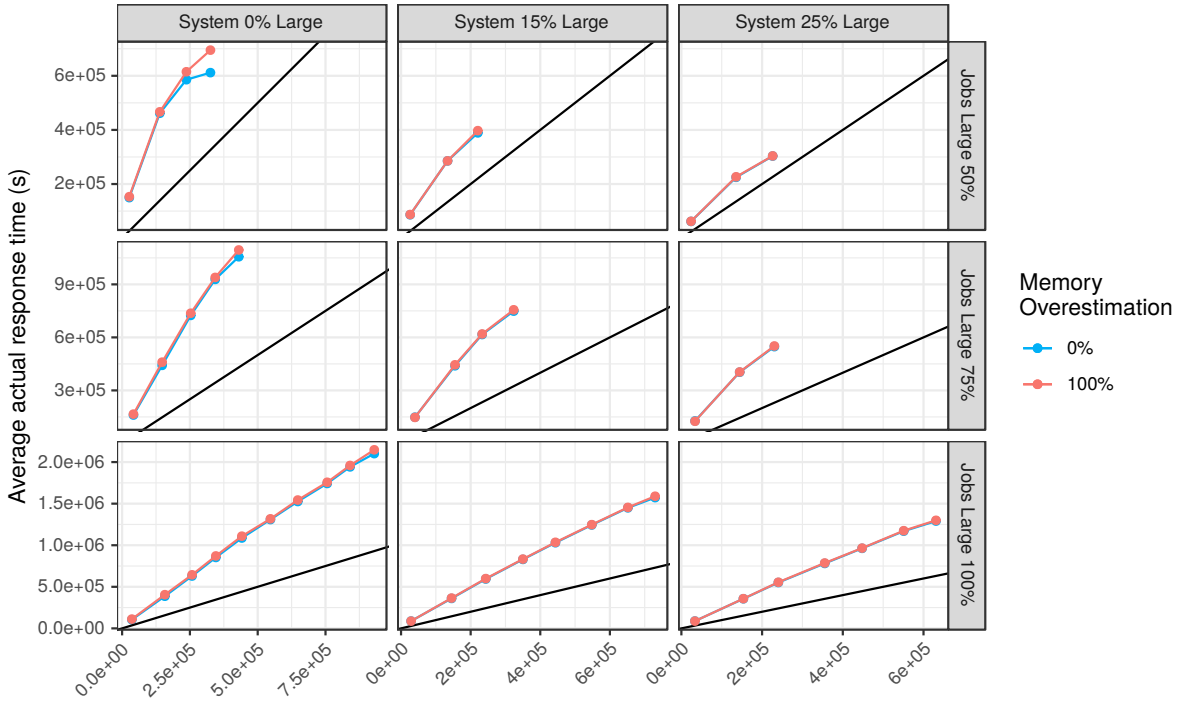
Results - Individual job response time I

- We show the average response time (*f*-axis) as a function of the baseline response time (*x*-axis)}
- The 0% line is for a **diligent** user whereas the 100% line is for a **careless** user
- *Large increase in response time* even for a diligent user
- Difference between the users is less than 10%
- **Conclusion: There is *little* incentive to the user be accurate**



Results - Individual job response time

- Little or no difference in the response time when we increase the number of large nodes in the system
- **Conclusion: No incentive for users to be accurate when others users are not**



Conclusion

- Composability and disaggregated memory show great promise in HPC
- Will need work across the whole system software (and hardware) stack
- We motivate some results using simulation with Slurm
- Static resource assignment may be an issue (as users have incentive overestimate memory demands)

- For more information
  - Felippe Vieira Zacarias, Vinicius Petrucci, and Paul Carpenter. Improving HPC System Throughput and Response Time using Memory Disaggregation. *ICPADS 2021*
  - Felippe Vieira Zacarias, Vinicius Petrucci, and Paul Carpenter. Memory Demands in Disaggregated HPC: How Accurate Do We Need to Be? *PMBS Workshop 2021*