Modelling Distributed Computing Infrastructures for High Energy Physics Applications

26th International Conference on Computing in High Energy & Nuclear Physics

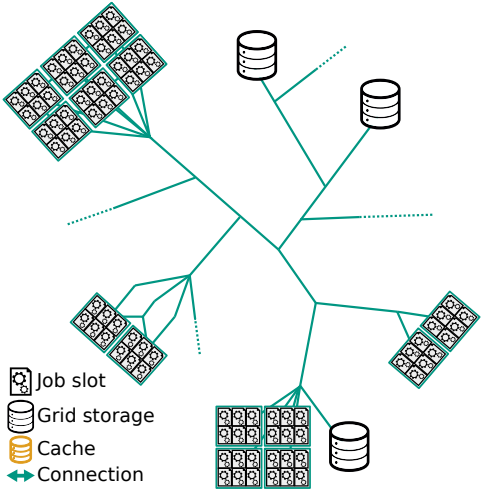
Maximilian Horzela, Henri Casanova, Manuel Giffels, Artur Gottmann, Robin Hofsaess, Günter Quast, Simone Rossi Tisbeni, Achim Streit, Frederic Suter | 08. May 2023

Planning a Computing Infrastructure for HEP

- We only have one grid to play with
  - Many connections, (managed) storages and compute resources
  - Dynamically changing workloads and data access patterns
  - Resource availability
  - Complex scheduling systems with many parameters
- **Many** competing ideas for infrastructure designs, enough resources only for **one**

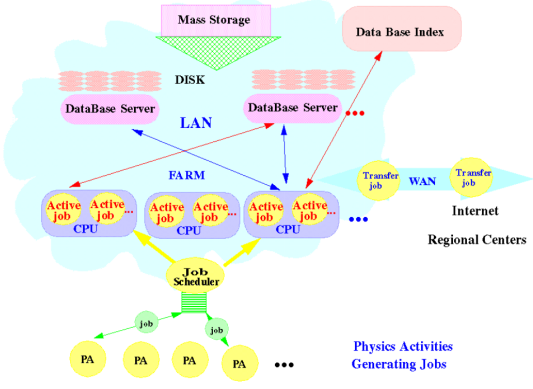
Most efficient realization of grid infrastructure is not obvious

How to study such systems?



Simulating Infrastructure was Appealing 20 Years Ago

- Proposed & Used 20 years ago: MONARC [1, 2] → original WLCG



[1]

Simulating Infrastructure is Appealing Today

[5]

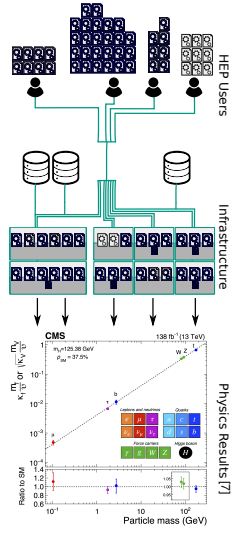
[6]

- Proposed & Used 20 years ago: MONARC [1, 2] → original WLCG
- There have been many attempts on modelling complex computing systems → many failed, some succeeded
- SIMGRID [3]: Library of **low-level** simulation abstractions for distributed systems in general (Actors using Resources through Activities)
  - **Fluid-model** for usage of network, storage, and CPU resources
  - Activities defined by a total and remaining amount of work to accomplish
  - Manager of a queue of activities, which start new and stop other activities triggered by events
- WRENCH [4]: **High-level** building blocks
  - Services specifying Activities
- Addition of **(HEP-)specific adaptions** (e.g. job-, dataset- & workflow-model, data streaming, . . .) and **logic starting and using services**

DCSim: Implementation of (HEP) Extensions and Simulator

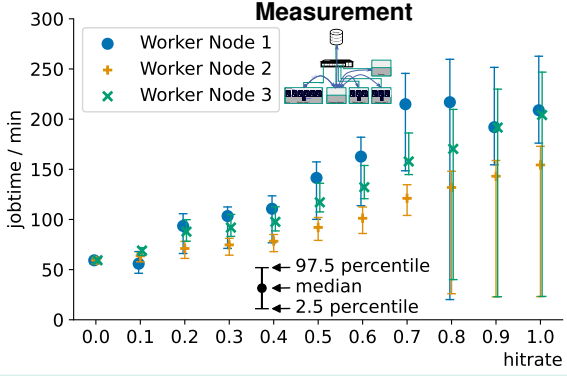
1. Define workloads of jobs with:
   - Number of operations to execute
   - Memory
   - Size of in- and output files
   - Submission time
2. Define platform (network & hosts) with
   - properties: N_{core}, CPU-speed, RAM, disk, bandwidth
   - roles: worker, storage, data cache, scheduler, ...
3. Instantiate initial deployment of files on storage systems
4. Start the simulation!
   - Jobs are scheduled and run
   - Input-files are streamed and cached
   - Caches evict files if necessary

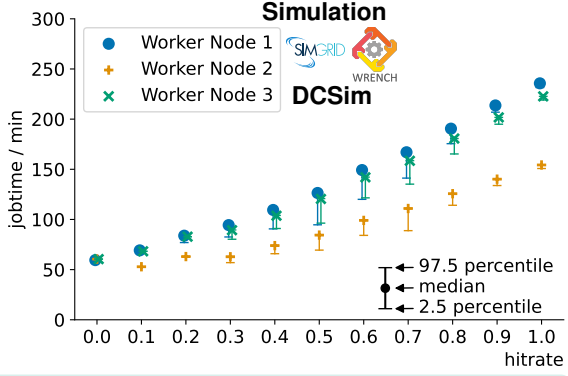
   → Job dynamics are monitored



Calibration and Validation

- Model not based on first principles → free parameters in simulation need to be tuned
- Validate tuned simulator by comparing with measured data
  e.g. job-execution times over fraction of cached input files



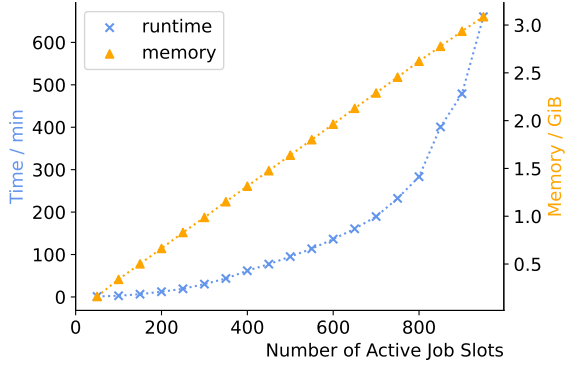


After calibration **simulation accurately reproduces general behaviour of real systems**, differences in details

Mitigating Computational Complexity of Simulation

- Superlinear increase of runtime with number of simultaneous activities
  $\rightarrow$ Detailed simulation of big platforms is problematic

$\rightarrow$ Scale size of platform to reduce number of simultaneously active entities (links, cores, hosts, . . .)

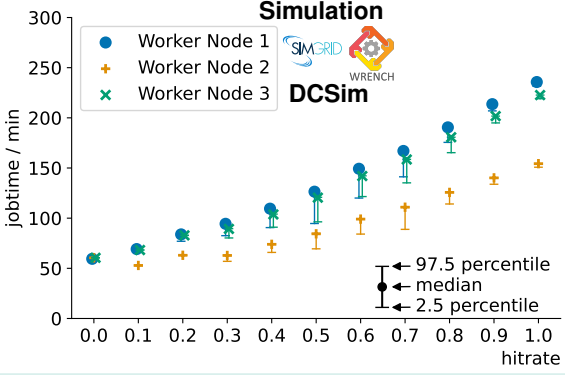


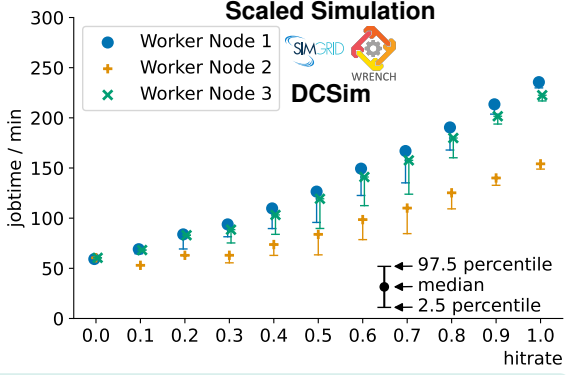
- Reduce number of active slots by decreasing number of cores on worker node hosts
- Reduce RAM on hosts to conserve the number of job slots
- Adjust link and storage bandwidths to conserve bandwidth per core
- Decrease the number of jobs per workload accordingly

Mitigating Computational Complexity of Simulation

- Superlinear increase of runtime with number of simultaneous activities
  → Detailed simulation of big platforms is problematic

→ Scale size of platform to reduce number of simultaneously active entities (links, cores, hosts, . . .)

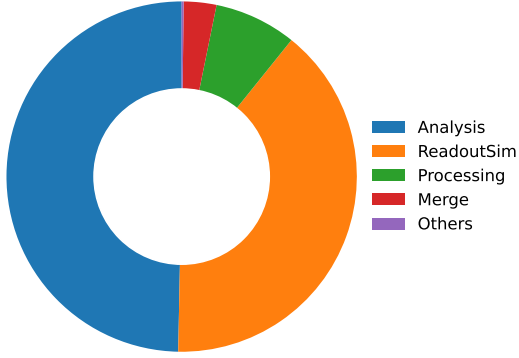




Scaling decreases runtime significantly (up to several orders of magnitude), while **conserving realism**

HEP Job Mix

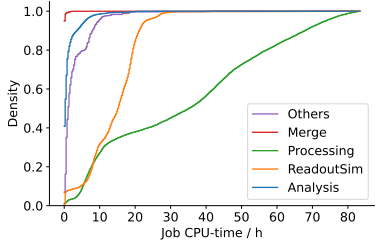
- Job characteristics gathered from job monitoring data over a representative time frame
- Group jobs into representative classes
- Main job types
  - **Analysis**: All types of jobs submitted by users, mainly MC production and data reduction
  - **ReadoutSim**: Simulation of detector readout electronics
  - **Processing**: Collective chunks of multiple data and MC processing steps from reconstruction to analysis objects data files
  - **Merge**: Reduction of number of files by merging multiple into a single

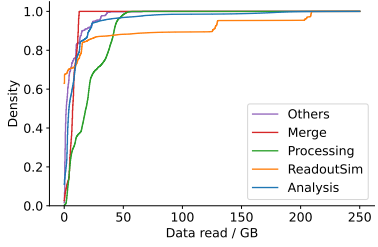


HEP Job Mix

- Job characteristics gathered from job monitoring data over a representative time frame
- Group jobs into representative classes
- Main job types
  - **Analysis**: All types of jobs submitted by users, mainly MC production and data reduction
  - **ReadoutSim**: Simulation of detector readout electronics
  - **Processing**: Collective chunks of multiple data and MC processing steps from reconstruction to analysis objects data files
  - **Merge**: Reduction of number of files by merging multiple into a single

Job types differ in their characteristics

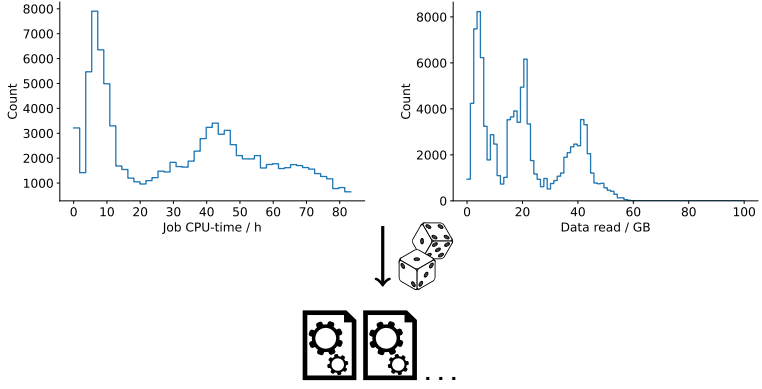




HEP Workload Sampling

- **Random sampling of job characteristics**
  - CPU-time of jobs
  - Memory required or used
  - Number of bytes read/written by the job → input/output size
- Keeping **proportions relative to the monitoring data**

Example for **Processing** jobs

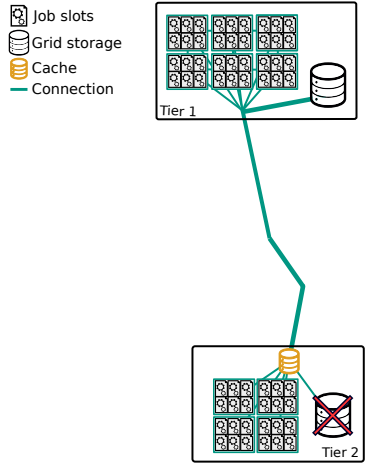


Use-case Study:
Interplay Between Tier 1 and Tier 2 with Unmanaged Storage

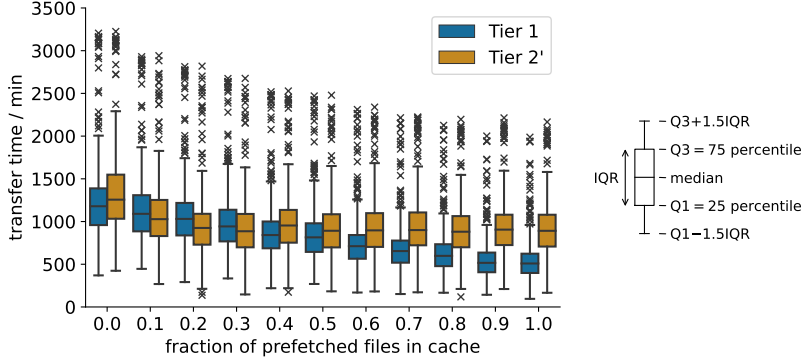
- Study **hypothetical** interplay of a Tier 1 with one **Tier 2 with grid storage replaced by cache** (Tier 2')
- Model both sites as local networks of worker nodes and storages with network in between

Tier 1	Tier 2'
$\mathcal{O}(40\text{k})$ cores	$\mathcal{O}(20\text{k})$ cores
80 Gbps bandwidth storage	80 Gbps bandwidth cache
$2 \otimes 100$ Gbps local network	40 Gbps local network
100 Gbps network between sites	

- Scan through **fraction of prefetched files at cache (prefetch-rate)**
  - $p = 0 \Leftrightarrow$ single grid storage serving both sites
  - $p = 1 \Leftrightarrow$ separate grid storage for Tier 2



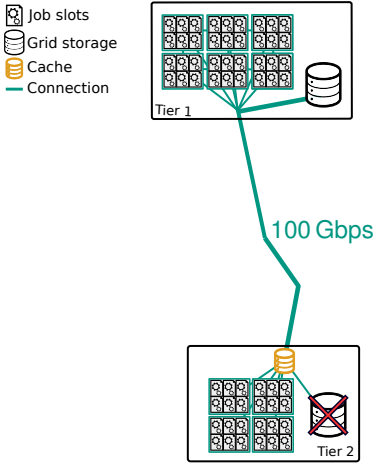
Interplay Tier 1 $\leftrightarrow$ Tier 2': Job Runtime & Transfer Duration

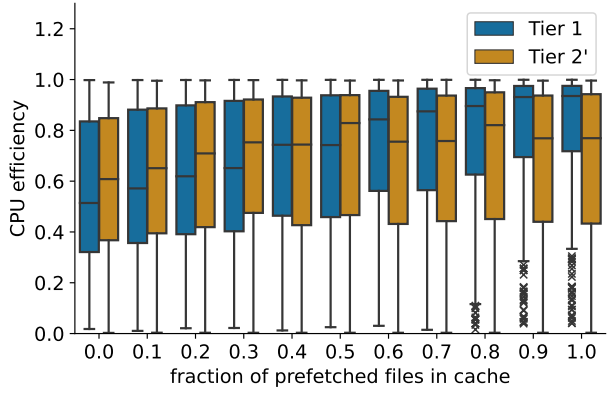


- Increasing presence of input-files at cache **reduces overall wall-times** of jobs because of faster I/O
- Benefit flattens out at Tier 2'(jobs throttled by cache and compute speed), further reduces at Tier 1

$\rightarrow$ **Both sites benefit** from reduced I/O on grid storage

Interplay Tier 1 $\leftrightarrow$ Tier 2': CPU Efficiency

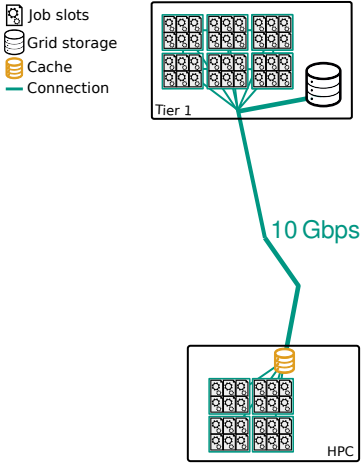


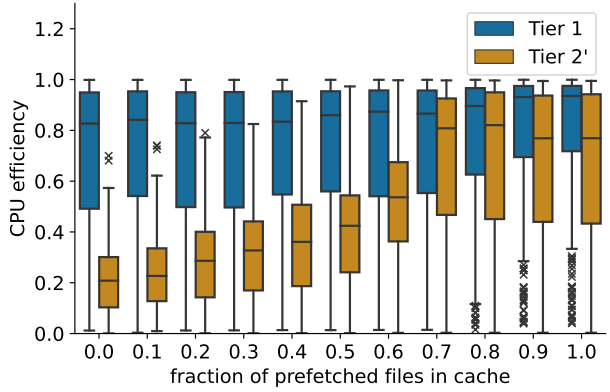


- **Job efficiency drops significantly** with lower fraction of input-files in cache due to higher load on Tier 1 grid storage

$\Rightarrow$ **Cache as replacement only reasonable with performance improvements of Tier 1 grid storage**

What if. . . ? – Tier 2 with Cache Connected with Lower Bandwidth (e.g. Cloud/HPC)





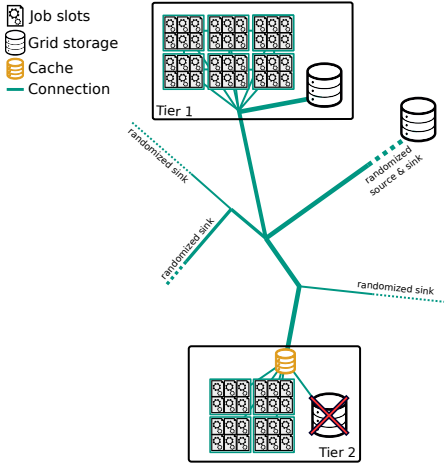
- **When slow network between sites**, effect on Tier 1 significantly reduced

⇒ **Job efficiency significantly boosted with cache** running at ”HPC“ **without efficiency loss at Tier 1**

Conclusions and Future Work

- Developed **flexible, performant and accurate tool**
  - **Calibrated and validated** using measured data
  - **Enabling practicable research on interesting and realistic computing infrastructures** without deploying them
- **More studies** on scalability, realism and more complex scenarios **in the pipeline**
  - Improve speed of the simulation
  - Influence of network load introduced by other sites
  - Datasets instead of job-specific input-files
  - Automated calibration and validation

On behalf of the collaboration, contact maximilian.horzela@kit.edu, if you are interested in using or in helping us develop the tool!



References I

[1] Iosif C. Legrand and Harvey B. Newman. “The MONARC Toolset for Simulating Large Network-Distributed Processing Systems”. Proceedings of the 32nd Conference on Winter Simulation. WSC ’00. Orlando, Florida: Society for Computer Simulation International, 2000, pp. 1794–1801. ISBN: 0780365828.

[2] Harvey B. Newman and Iosif C. Legrand. “Simulating distributed systems”. AIP Conference Proceedings 583.1 (Aug. 2001), pp. 164–166. ISSN: 0094-243X. DOI: 10.1063/1.1405293. eprint: https://pubs.aip.org/aip/acp/article-pdf/583/1/164/11368715/164_1_online.pdf.

[3] Henri Casanova et al. “Versatile, Scalable, and Accurate Simulation of Distributed Applications and Platforms”. Journal of Parallel and Distributed Computing 74.10 (Oct. 2014), pp. 2899–2917. DOI: 10.1016/j.jpdc.2014.06.008.

[4] Henri Casanova et al. “Developing Accurate and Scalable Simulators of Production Workflow Management Systems with WRENCH”. Future Generation Computer Systems 112 (2020), pp. 162–175. DOI: 10.1016/j.future.2020.05.030.

[5] SimGrid Team. “simgrid.org”. 2022. URL: https://simgrid.org/.

References II

[6] WRENCH Team. “wrench-project.org”. 2022. URL: https://wrench-project.org/.

[7] CMS Collaboration. “The Higgs Boson turns 10: Results from the CMS experiment”. 2022. URL: https://cms.cern/news/higgs-boson-turns-10-results-cms-experiment.

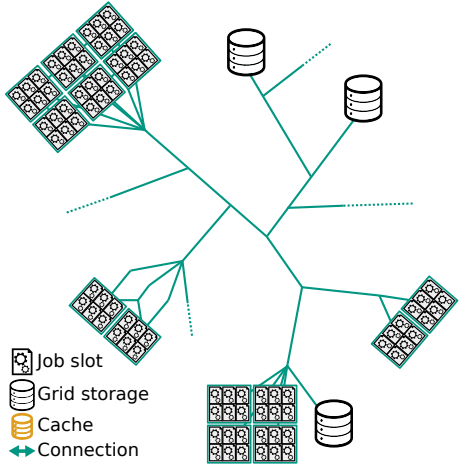
[8] Pedro Velho et al. “On the Validity of Flow-Level Tcp Network Models for Grid and Cloud Simulations”. ACM Trans. Model. Comput. Simul. 23.4 (Dec. 2013). ISSN: 1049-3301. DOI: 10.1145/2517448. URL: https://doi.org/10.1145/2517448.

Backup

Static Diskless Tier 2 with Cache

- Study interplay of a Tier 1 with one (or more) Tier 2 with **grid storage replaced by cache**
- Investigate “static” collection of production and analysis jobs

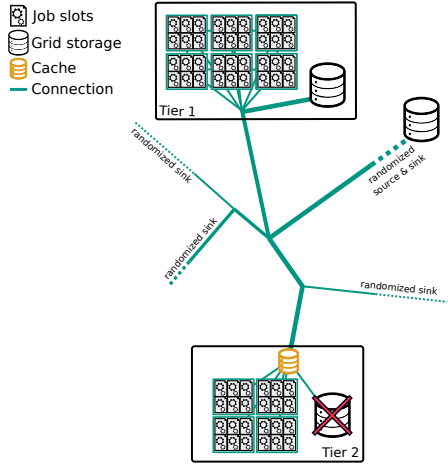
→ **Scan prefetch-fraction of relevant input-files** at cache
(0: no cache, 1: grid storage)



Static Diskless Tier 2 with Cache

- Study interplay of a Tier 1 with one (or more) Tier 2 with **grid storage replaced by cache**
- Investigate “static” collection of production and analysis jobs
- Approximate dispensable sites’ effects with randomized links/bws

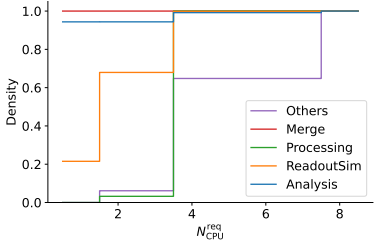
→ **Scan prefetch-fraction of relevant input-files** at cache
(0: no cache, 1: grid storage)

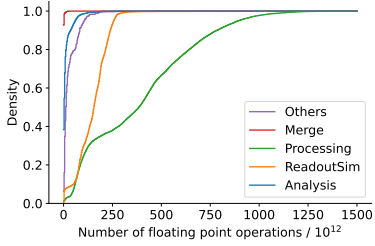


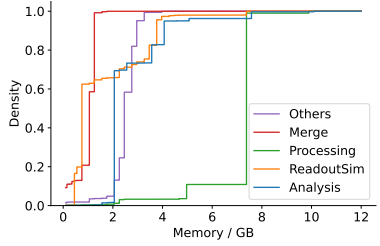
HEP Job Characteristics

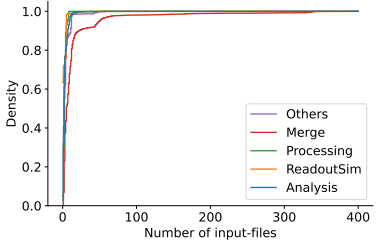
- Job characteristics gathered from job monitoring data of time frame 24.02.-07.03.23
- CMS jobs executed at sites `T1_DE_KIT` and `T2_DE_DESY` analysed
- Job types
  - **Analysis**: All types of jobs submitted by users, mainly MC production and data reduction
  - **ReadoutSim**: Simulation of detector readout electronics
  - **Processing**: Collective chunks of multiple data and MC processing steps from reconstruction to analysis objects data files
  - **Merge**: Reduction of number of files by merging multiple into a single
  - **Others** including
    - Generation of MC events and simulation of Detector effects
    - Reconstruction of digital detector signals
    - Analysis object data derivation
    - Collection of job logs
    - Cleanup of logs and data
    - Test jobs

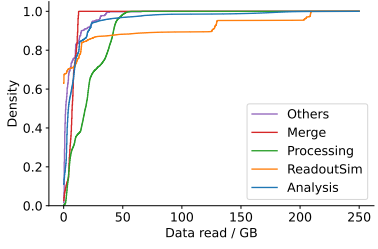
CMS Job Characteristics – T1_DE_KIT and T2_DE_DESY

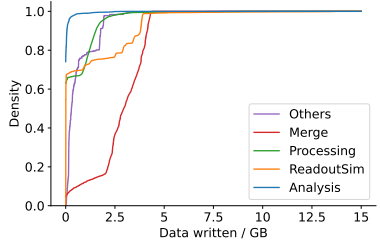




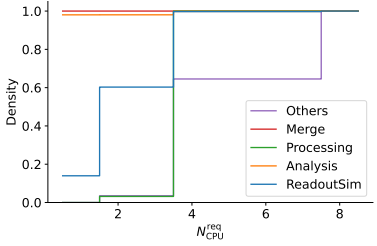


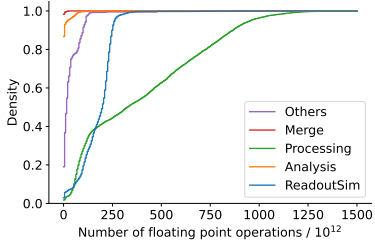


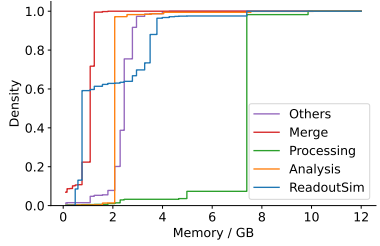


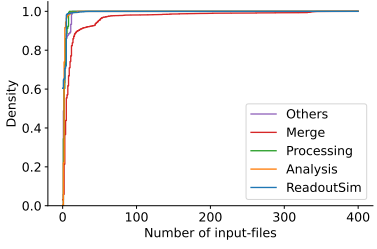


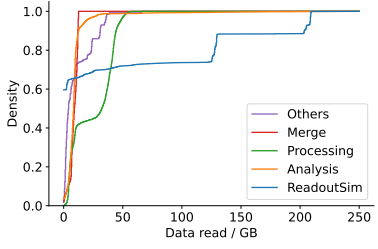
CMS Job Characteristics – T1_DE_KIT

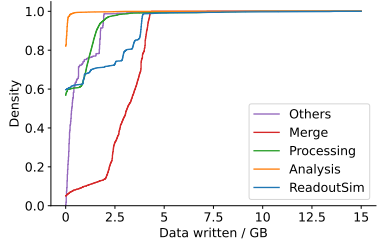




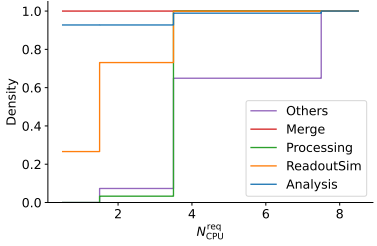


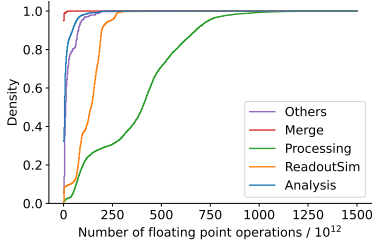


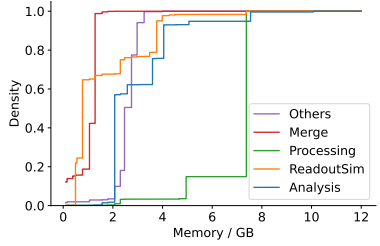


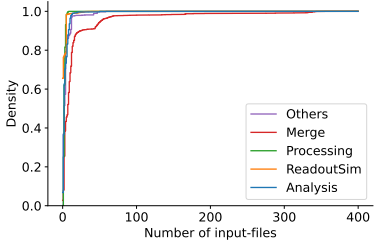


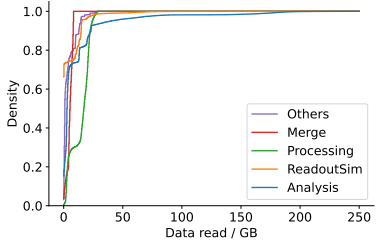
CMS Job Characteristics – T2_DE_DESY

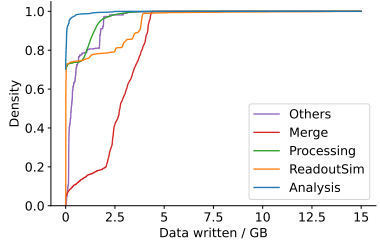












Underlying Models

SimGrid and Wrench

SimGrid [3]

- Library of exposed functions (low-level simulation abstractions) written in C++
- Framework for building own simulator of distributed computer systems in C/C++, Python or Java
- Accurate (validated), scalable (low ratio of simulated versus real time) and expressive (able to simulate arbitrary platforms, applications and execution scenarios)
- Large user-base

Wrench [4]

- High-level simulation abstractions based on SimGrid
- C++ API to define services and execution controllers for the implementation of a simulator

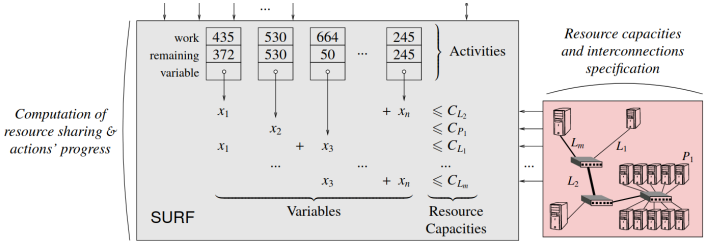
SimGrid Engine

- “Actors” using “Platform” of resources through corresponding “Activities”
- “Activities” have both qualitative (synchronization between actors / locking) and quantitative (consumption of resource capacities) components
- “Actor” management by a central actor, called “Maestro”, in scheduling round:
  1. Assign control flow to each actor not blocked on a simcall, start sub-scheduling:
     1. Actors execute activities and return intermediate simcalls that take no time to execute (e.g. spawn new actor)
     2. Repeat until all actors return a blocking simcall or terminate
  2. Advance time to that point at which first next activity terminates
  3. Start new scheduling round

  Context switching between actor and maestro is highly optimized

SIMGRID Resource-Activity Model

- Same analytical flow-model for simulation of network, storage and CPU
- Activities defined by a total and remaining amount of work to accomplish



[3]

- Resource capacity of resource C_r is assigned to a set of concurrent activities $\mathcal{A}$ at time t_0 is determined by solving $\max\left[\min_{a \in \mathcal{A}}\left(x_a\right)\right]$ under constraints $\sum_{a \in \mathcal{A}} x_a \leq C_r$
- Simulation time is advanced to time at which first activity is completed t_1

SimGrid

CPU and Storage Model

- Max-min of n resources sharing a single resource r leads to fair share $\frac{C_r}{n}$ assigned to each activity
- For CPU-shares optional scaling by normalized priorities possible
- Fair sharing of storage I/O bandwidth plus additional fixed initial delay at simulation time advance due to seek time

TCP Network Model

- TCP doesn't show Max-Min fairness, two options:
- Packet level network simulator ns-3
- Improved flow-level network model with
  - modified constraints accounting for RTT unfairness of TCP and throughput degradation due to reverse traffic [8]
  - and improved execution time

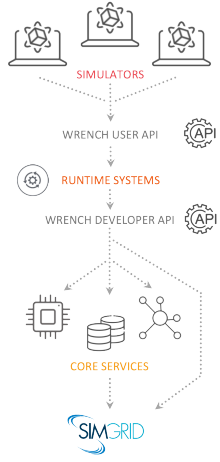
$$T = \alpha l_f + \frac{V}{\beta x_f}$$

with TCP version specific parameters α and β tuned by packet-level simulation

valid for transfers of data of size $\geq$ 100KiB

WRENCH

- Layer of high level building blocks on top of SimGrid
- Collection of Services using the simulated platform to define activities and events, i.e.:
  - Computing services implement usage of CPU resources
  - Storage services provide management of storage resources and read/write of files
  - Execution controllers schedule and start actions on other services
- Provides an API to define services and execution controllers for the implementation of a simulator



[6]

WRENCH Services

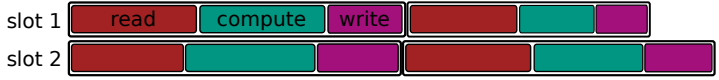
- Adds high level abstractions (“services”) on top of SIMGRID
  - Compute services knows how and where to compute tasks, e.g. bare-metal, cloud, virtualized cluster, batch-scheduled cluster platforms and HTCondor
  - Storage services know how to store and give access to files
  - File-registry services know where files reside
  - Network proximity services monitor network and maintain database of host-to-host distances
  - Energy-meter services periodically measure energy-consumption of all resources
- All services introduce their own messages (activities) and according payloads

HTCondor Compute-Service

- Starting HTCondor-compute-service spawns a central-manager and a negotiator service
  - Compute service: entry point for job submission to WMS
  - Central manager: management of available resources (pool of bare-metal or batch-compute-services), manages job submission, initializes negotiation cycles
  - Negotiator: matches jobs to resources (based on #CPU and memory requirements)
- Actual task execution and resource allocation on host managed by matched compute-service
- Main focus so far on simulation of grid-universe jobs, need to be reviewed in the future to support more complex scheduling scenarios

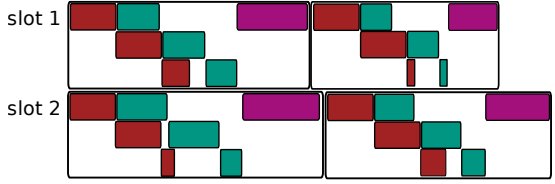
Batch and Streaming Jobs

Batch jobs



- Sequentially read a batch of data, compute and write

Streaming jobs



- Can concurrently read and compute blocks of data (enabled by XRootD in HEP)

→ More compact pattern and reduced duration

Caching Job

- XRootD inspired caching of input-files
- Information about reachable caches definite only once job started

$\rightarrow$ Active job

Once job is scheduled

1. Determine reachable (local, same local network, same network tier) caches
2. For each file, determine source storage
   - If cache, read from cache
   - If remote, read from remote, cache (instantly create) & evict (LRU), when applicable
3. Perform I/O and computation (copy & compute, stream blocks)

Validation & Calibration

Calibration/Validation Strategy

Real-world system:

- Assemble real test systems with a control on parameters
- Start collections of jobs with known/steerable characteristics
- Measure job dynamics

Simulation:

- Define platforms as close to the test-system as reasonable
- Start workloads of jobs with similar job characteristics
- Read-out job dynamics

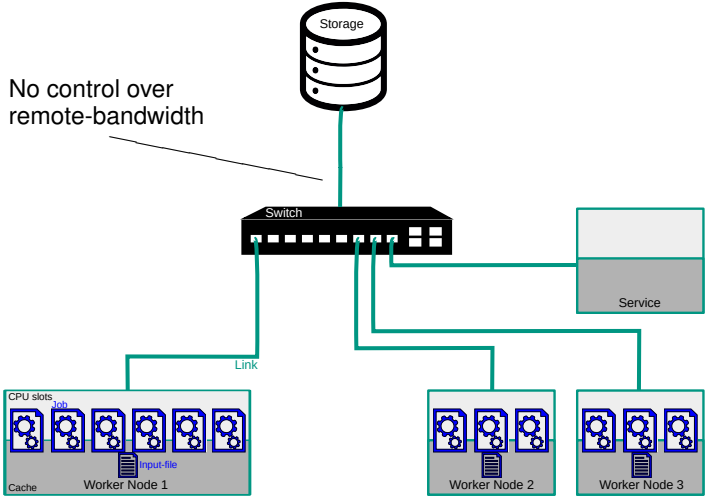
Define a set of observables, which are sensitive to simulation parameters

Tune the simulation parameters until simulation fits measurements

Combine differently configured measurements to learn something about the system

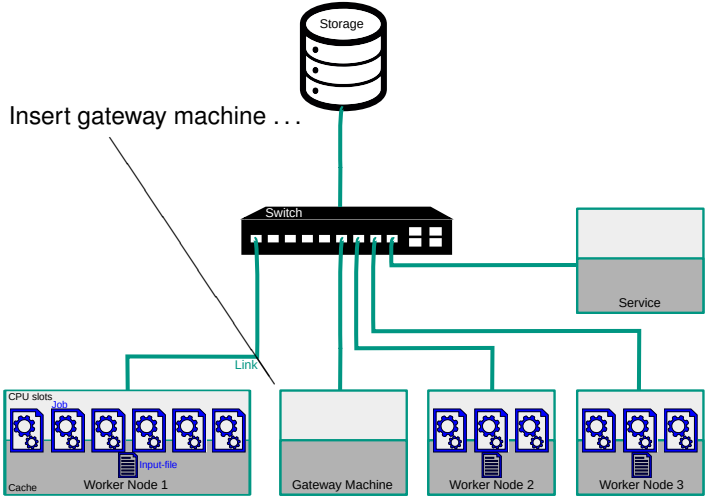
$\Rightarrow$ Use chain of slightly varied scenarios to calibrate in steps

Benchmark Setup



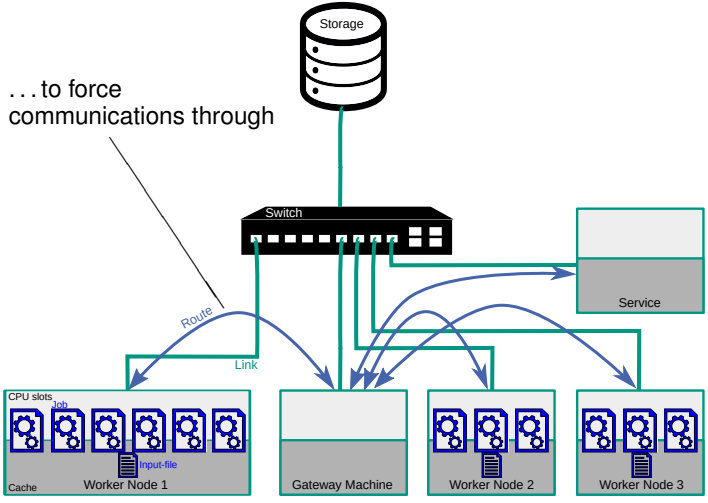
WN 1	WN 2	WN 3
24 job slots	12 job slots	12 job slots
HDD data cache each		
10 Gbps link to switch each		

Benchmark Setup



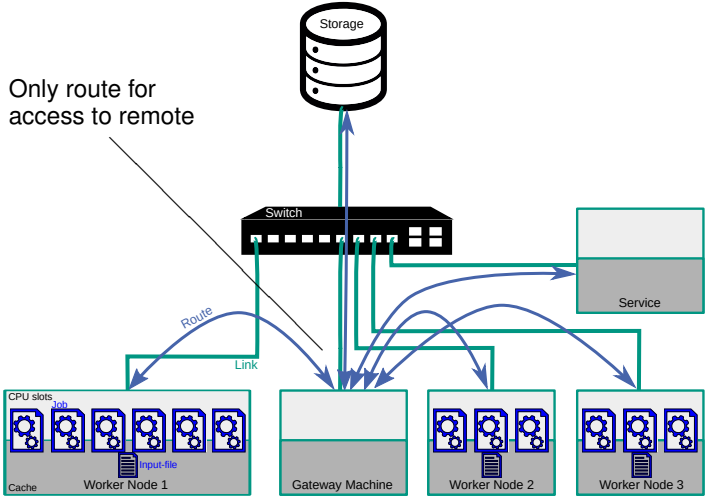
WN 1	WN 2	WN 3
24 job slots	12 job slots	12 job slots
HDD data cache each		
10 Gbps link to switch each		

Benchmark Setup



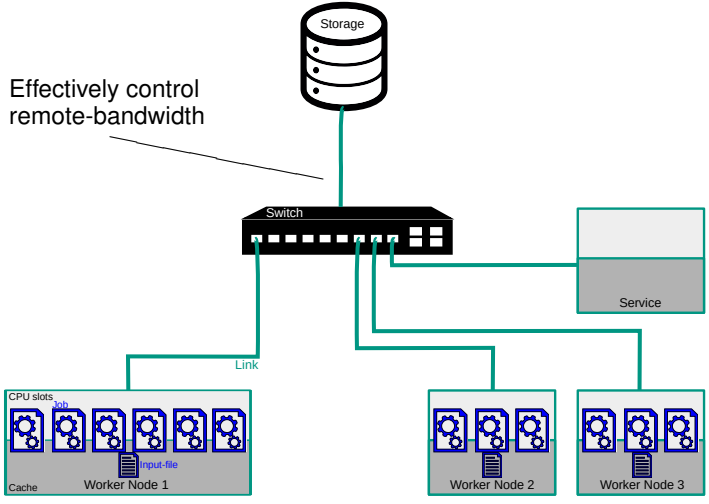
WN 1	WN 2	WN 3
24 job slots	12 job slots	12 job slots
HDD data cache each		
10 Gbps link to switch each		

Benchmark Setup



WN 1	WN 2	WN 3
24 job slots	12 job slots	12 job slots
HDD data cache each		
10 Gbps link to switch each		

Benchmark Setup



WN 1	WN 2	WN 3
24 job slots	12 job slots	12 job slots
HDD data cache each		
10 Gbps link to switch each		

Benchmark Jobs

- Exact replications of the same data analysis job
  - Same executable, executing data transformation and reduction
  - Read (copies of) same input-files via XRootD in the same order
  - Fraction of input-files read from local cache
  - Number of jobs matches number of available slots
- Input-files on remote storage and prefetched on caches
- Only metadata transfer at stage-out
- Job monitoring part of the executable

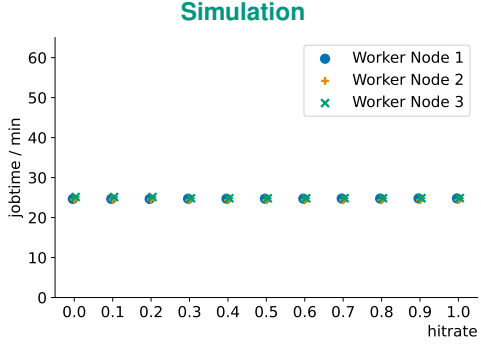
Calibration Step 1: 10G Remote with Memory-Cache

- Setting gateway machine's network interface to 10 Gbit s^{-1}
- Each job reading same input-files → Memory cache operative

- Clearly CPU limited workflow, no I/O throttling → Tune CPU speeds in simulation

Calibration Step 1: 10G Remote with Memory-Cache

- Setting gateway machine's network interface to 10 Gbit s^{-1}
- Each job reading same input-files $\rightarrow$ Memory cache operative

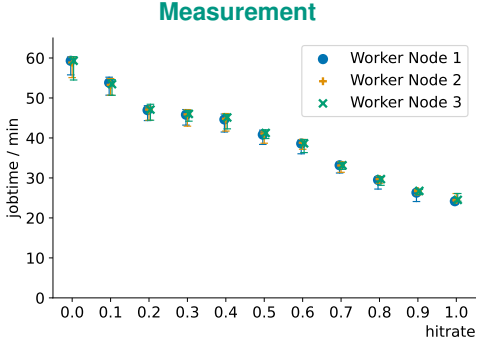


- Clearly CPU limited workflow, no I/O throttling $\rightarrow$ Tune CPU speeds in simulation

Calibration Step 2: 1G Remote with Memory-Cache

- Lower gateway machine's network interface to 1 Gbit s^{-1} $\rightarrow$ expect throttling by network

$\rightarrow$ Reuse pre-calibration information from previous measurement



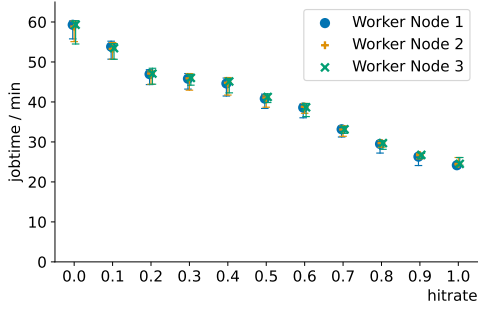
- Limited by I/O via remote network

Calibration Step 2: 1G Remote with Memory-Cache

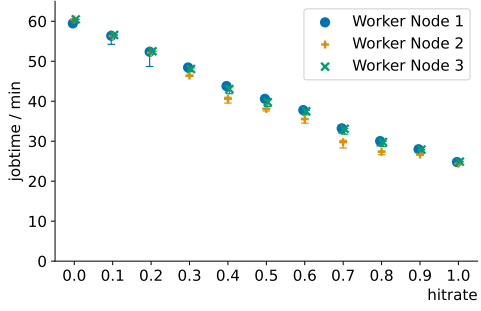
- Lower gateway machine's network interface to 1 Gbit s^{-1} $\rightarrow$ expect throttling by network

$\rightarrow$ Reuse pre-calibration information from previous measurement

Measurement



Simulation

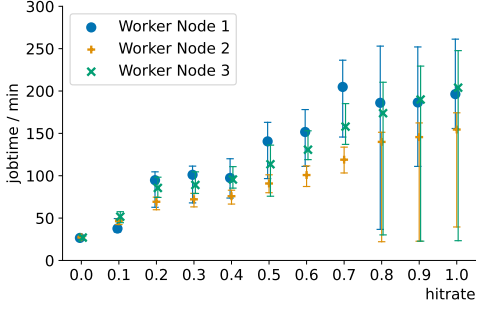


- Limited by I/O via remote network $\rightarrow$ 15% higher network bandwidth than expected

Calibration Step 3: 10G Remote with HDD-Cache

- Setting gateway machine's network interface to 10 Gbit s^{-1}
- Each job reading copies of same input-files $\rightarrow$ data cache operative

Measurement



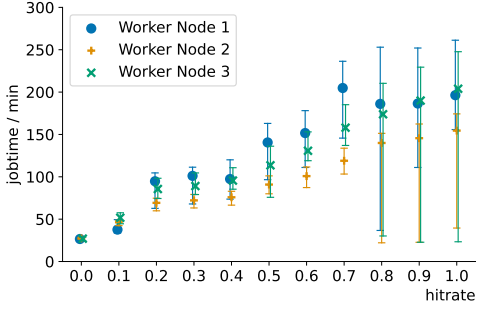
- Limited I/O by cache bandwidth

$\Rightarrow$ Now all simulation parameters are set!

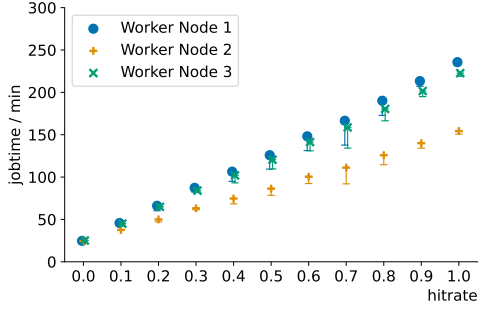
Calibration Step 3: 10G Remote with HDD-Cache

- Setting gateway machine's network interface to 10 Gbit s^{-1}
- Each job reading copies of same input-files $\rightarrow$ data cache operative

Measurement



Simulation



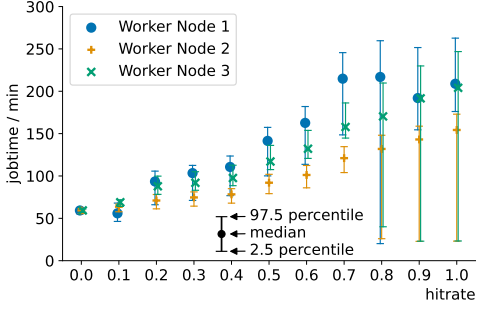
- Limited I/O by cache bandwidth $\rightarrow$ hard drive bandwidth $\mathcal{O}(10\,\mathrm{MB\,s^{-1}})$

$\Rightarrow$ Now all simulation parameters are set!

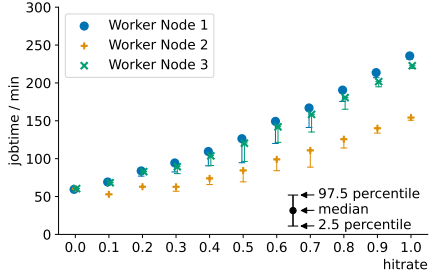
Validation: 1G Remote with HDD-Cache

- Setting gateway machine's network interface to 1 Gbit s^{-1}
- Each job reading copies of same input-files → data cache operative

Measurement



Simulation



- Limited I/O by cache bandwidth & network to remote → Simulation parameters obtained previously

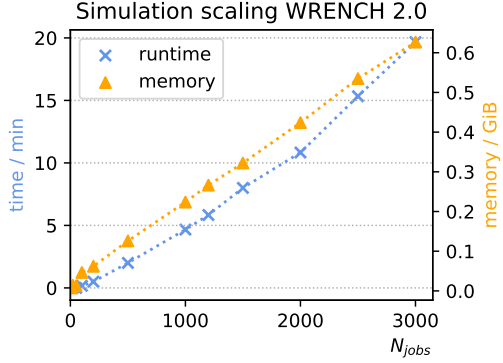
→ Simulation reproduces measurement without further adjustments

The simulator is able to reproduce reality!

Simulation Scaling

Scaling of $\mathcal{O}(10)$-core system

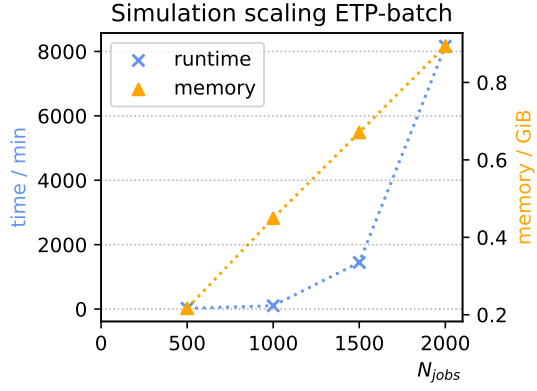
- Performant enough to simulate (parts-of) the grid with $\mathcal{O}\left(10^5\right)$ jobs?



- Linear scaling of memory and simulator run-time $\rightarrow$ 15 h runtime, 20 GB memory

Scaling of $\mathcal{O}(1000)$-core system

- Performant enough to simulate (parts-of) the grid with $\mathcal{O}\left(10^5\right)$ jobs?



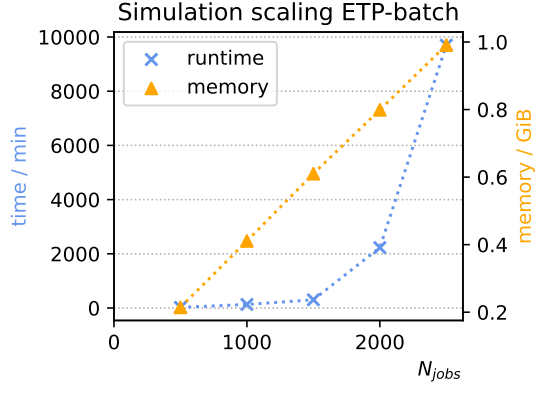
- **Exponential scaling** of simulator run-time

Scaling of $\mathcal{O}(1000)$-core system

- Performant enough to simulate (parts-of) the grid with $\mathcal{O}\left(10^5\right)$ jobs?

- Simplifying platform (condense workers, simplify network) $\rightarrow$ Simpler assignment of resource shares

$\rightarrow$ Factor ≈ 4 saving in runtime



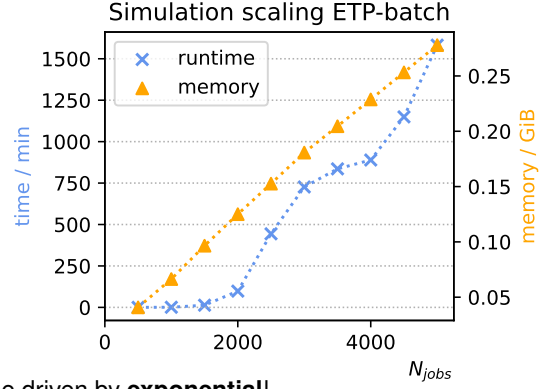
- **Still exponential scaling** of simulator run-time

Scaling of $\mathcal{O}(1000)$-core system

- Performant enough to simulate (parts-of) the grid with $\mathcal{O}(10^5)$ jobs?

- Scaling I/O $\rightarrow$ less total number of activities $\propto$ SimGrid solver calls

$\rightarrow$ Factor ≈ 20 saving in runtime



- ~~**Exponential**~~ **Linear scaling** of simulator run-time driven by **exponential**!

$\rightarrow$ Exponential increase with platform size

Scaling to $\mathcal{O}(100\text{k})$-core systems

- Hard to win against exponential (this project's no-free-lunch)
- Modelling tricks: platform simplification, I/O scaling, buffer- & block-size tweaking, ...
  - Proof-of-concept (show that tricks retain realism) ongoing

$\rightarrow$ Scale down the exponential, but still very much exponential

- How to proceed?

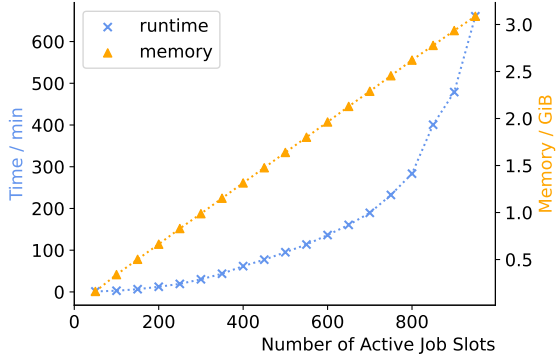
Simulation-Scaling Challenge

- Exponential increase of simulation time with number of concurrently active job slots

→ No point in running test simulations with $\approx$ 75k concurrent jobs without changes

- Scaling I/O bandwidths and file sizes or XRootD-block and buffer-sizes reduces simulation time by constant factors $\mathcal{O}(10)$

→ Maybe run it once everything is set

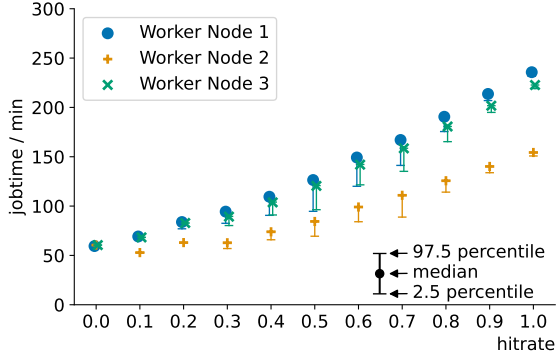


Improve Scaling by Exponential

- Exponential scaling with number of slots is a problem
- Why not scale along this exponentially manifesting dimension?

→ Scale the size of the platform, number of slots and bandwidths

  - Since jobs anyway drawn from distribution, decreasing the statistics is fine
    → Statistics can be recovered by repeating simulation with different random seed
  - Must assure that situation on platform is still the same



Improve Scaling by Exponential

- Exponential scaling with number of slots is a problem
- Why not scale along this exponentially manifesting dimension?

→ Scale the size of the platform, number of slots and bandwidths

- Since jobs anyway drawn from distribution, decreasing the statistics is fine
  → Statistics can be recovered by repeating simulation with different random seed
- Must assure that situation on platform is still the same

⇒ Works for validation platform!