Karlsruhe Institute of Technology

Institute of Experimental Particle Physics

Steinbuch Centre for Computing

CMS Germany

BMBF FSP-CMS

Modelling (parts of) the WLCG

CERN School of Computing 2022, Kraków

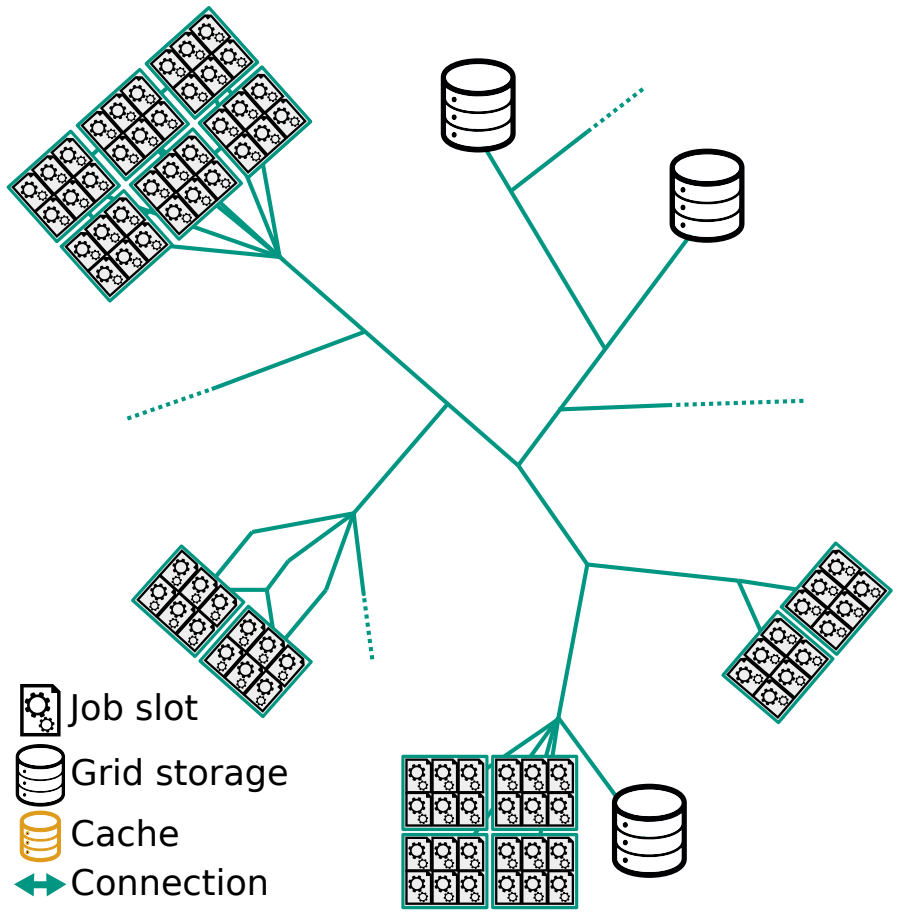
Maximilian M. Horzela | 07. September 2022

Distributed Storage and Compute Infrastructure in HEP

Karlsruhe Institute of Technology

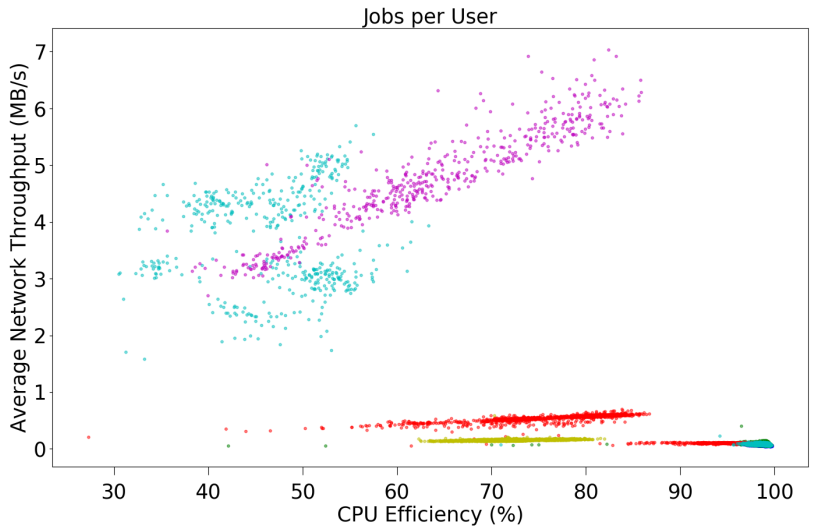
- **Heterogeneous** infrastructure (large/small grid sites, opportunistic resources, . . .)
- **Distributed** data and compute resources
- Many data transfers across WAN

→ Need to establish an **efficient** CDN / information-centric network / data lake / . . .



Use Available Resources Efficiently

- Network often limiting factor
- Correlation of network throughput and CPU efficiency



M. Schnepf

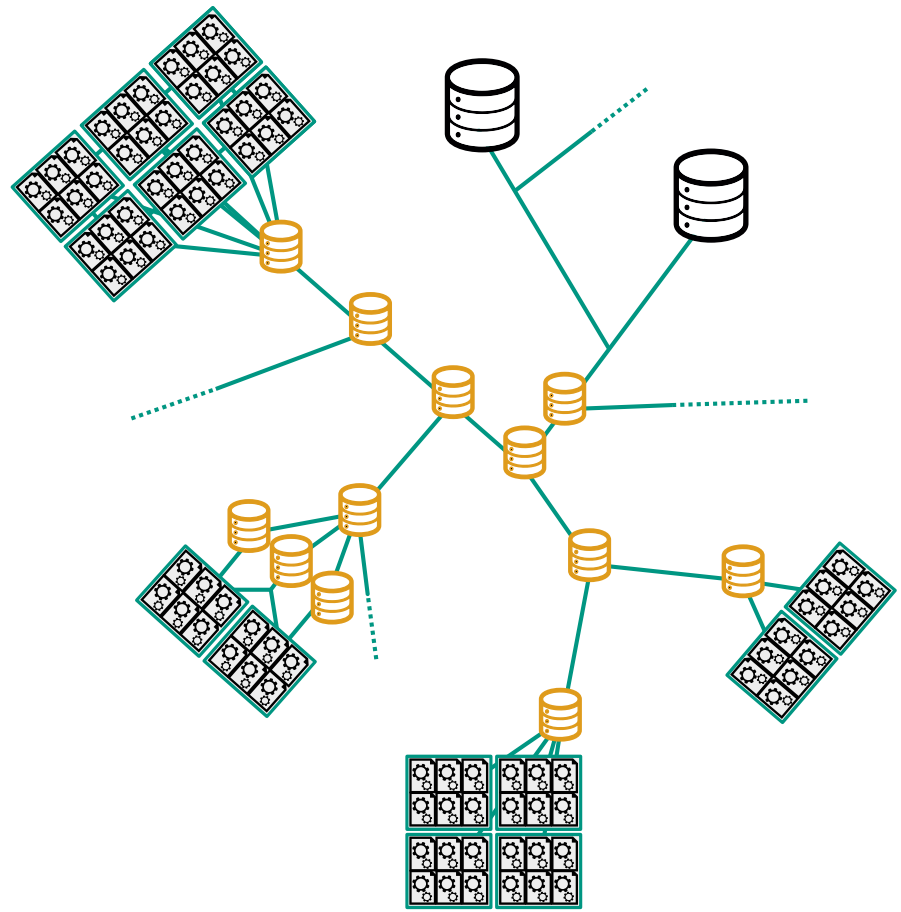
- Increase **data locality**: relieve WAN and shift transfer load to local network/connections via **data caches**

⇒ Increase overall data throughput for a more efficient usage of compute resources

Asking Questions Involving Complex Structures

Karlsruhe Institute of Technology

- Many caches, managed storages and compute resources
- Dynamically changing workloads and data access patterns (and resources)
- Complex scheduling systems with many parameters



- Real world systems are complex

→ Unclear how to use caches

→ What are efficient realizations?

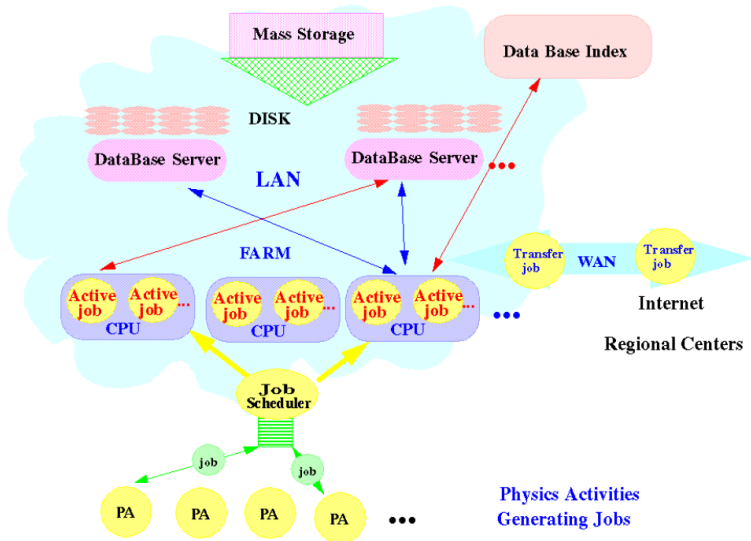
How to study these systems?

Planning the GRID Like 20 Years Ago

- **Option 1**: Performance measurements of test-beds not feasible (time and monetary costs) for big complex structures

- **Option 2:** Simulate applications on appealing infrastructure designs
- Already successful in the past: MONARC → WLCG
- MONARC(2) discontinued

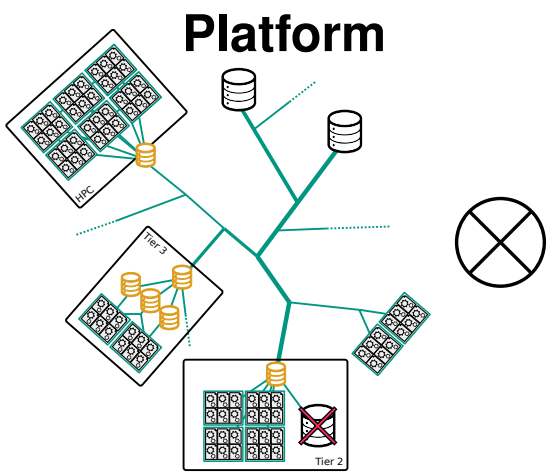
→ Modern, accurate & scaleable simulator

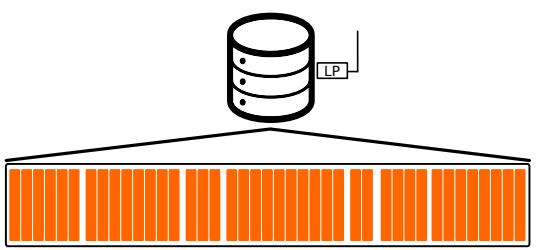


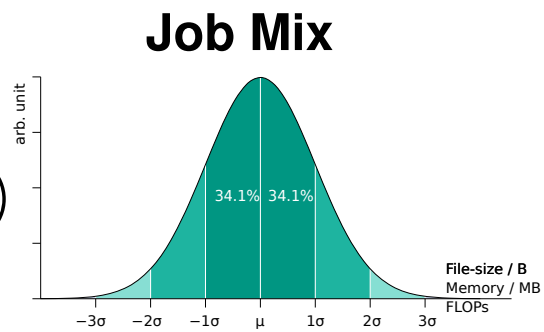
I. Legrand, H. Newman

Simulator Software

- SimGrid: Library of low-level simulation abstractions for distributed systems (Actors using Platform through Activities)
- Wrench: High-level building blocks (Services specifying Activities)
- Addition of (HEP-)specific adaptions (e.g. job-, dataset- & workflow-model, XRootD, . . .)

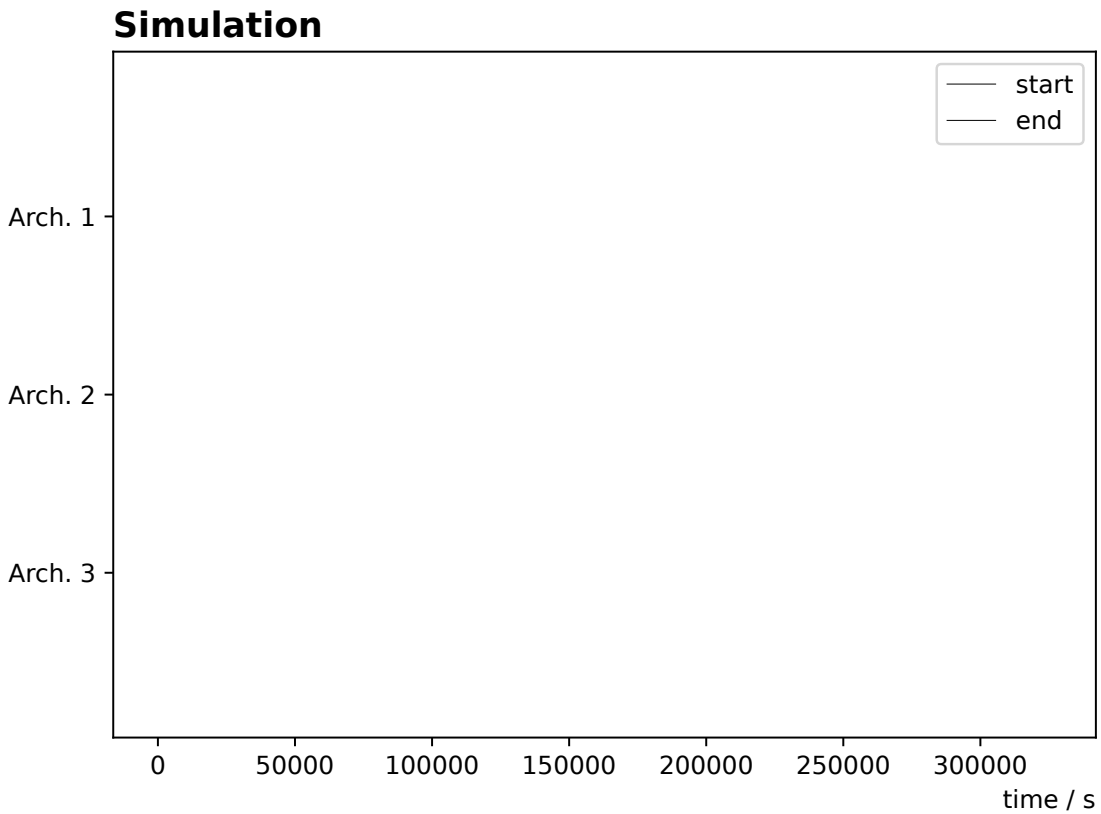






Application

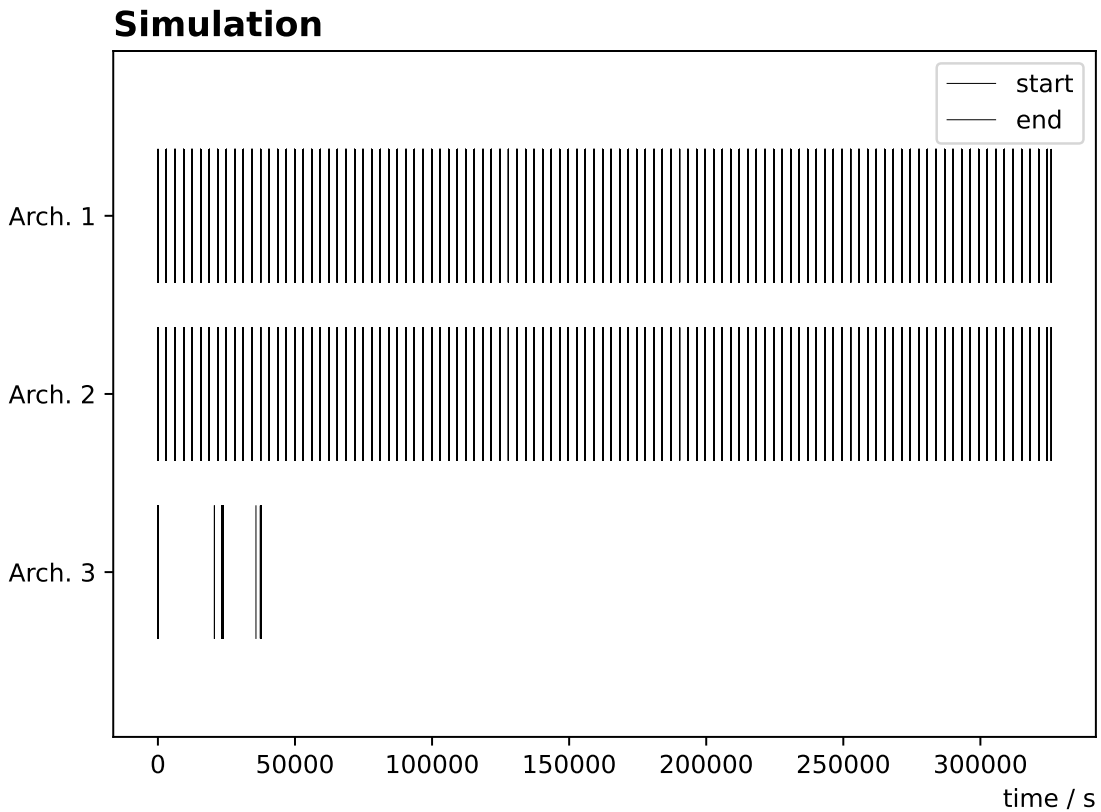
- e.g.: What is a good analysis cluster?



- Compare three architectures:
  - Define the platforms for each scenario
  - Place the input-data
  - Create a job mix of 5000 analysis jobs

Application

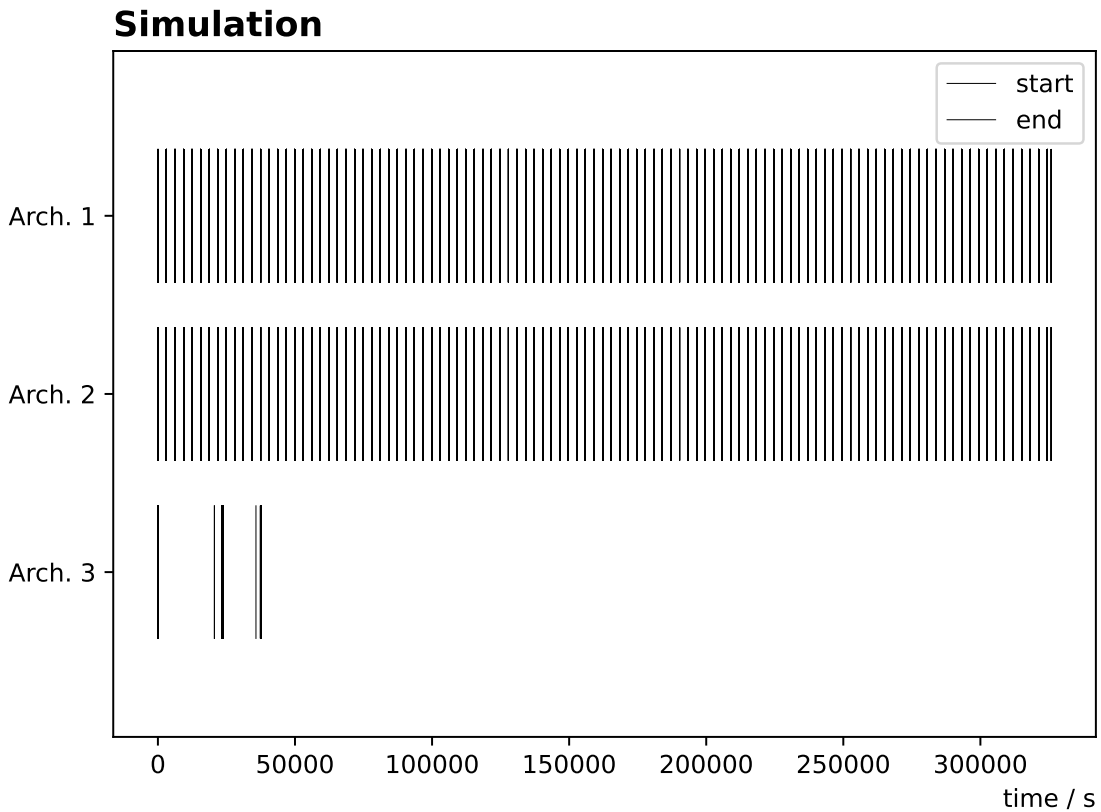
- e.g.: What is a good analysis cluster?



- Compare three architectures:
  - Define the platforms for each scenario
  - Place the input-data
  - Create a job mix of 5000 analysis jobs
  - Compare the performance

Application

- e.g.: What is a good analysis cluster?



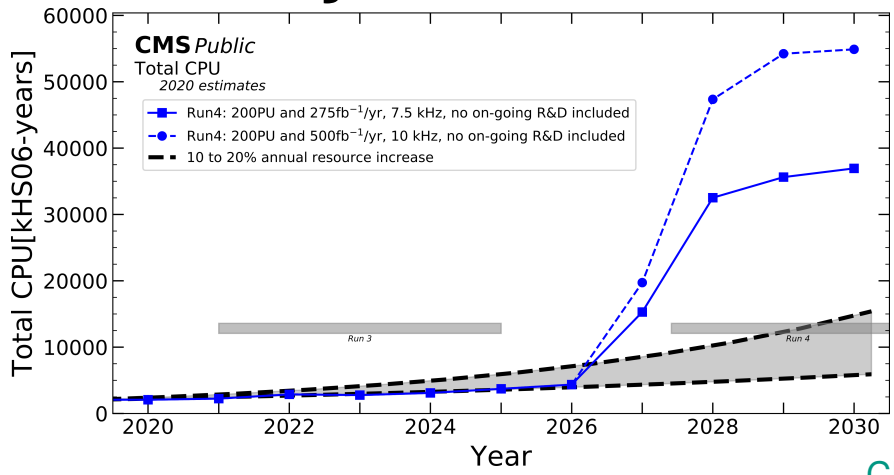
- Compare three architectures:
  - Define the platforms for each scenario
  - Place the input-data
  - Create a job mix of 5000 analysis jobs
  - Compare the performance

⇒ Extend to: **How do we optimize the grid?**

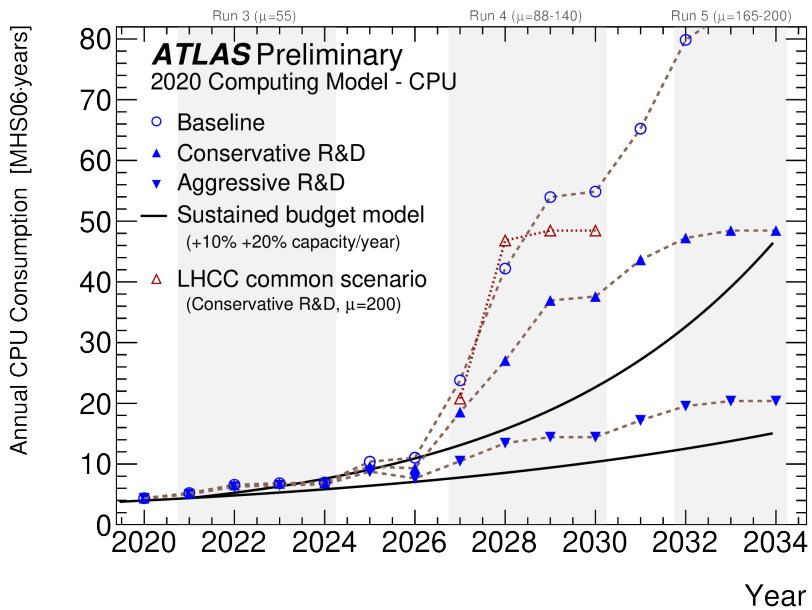
Backup

Computing for the High-Luminosity-LHC

- Clear challenge
  - Expect exploding demand for computing infrastructure
- Proposed solutions
  - Software improvements
  - **Integration of additional non-HEP resources**
  - **Optimization of existing computing model**



CMS

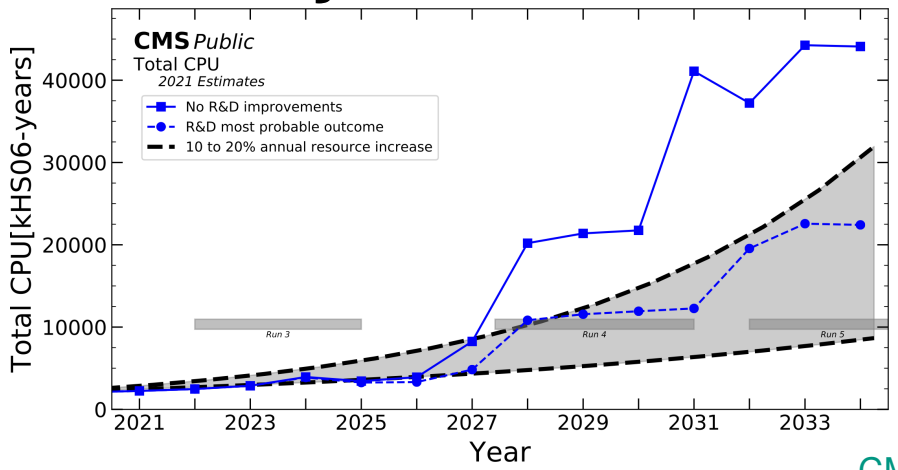


ATLAS

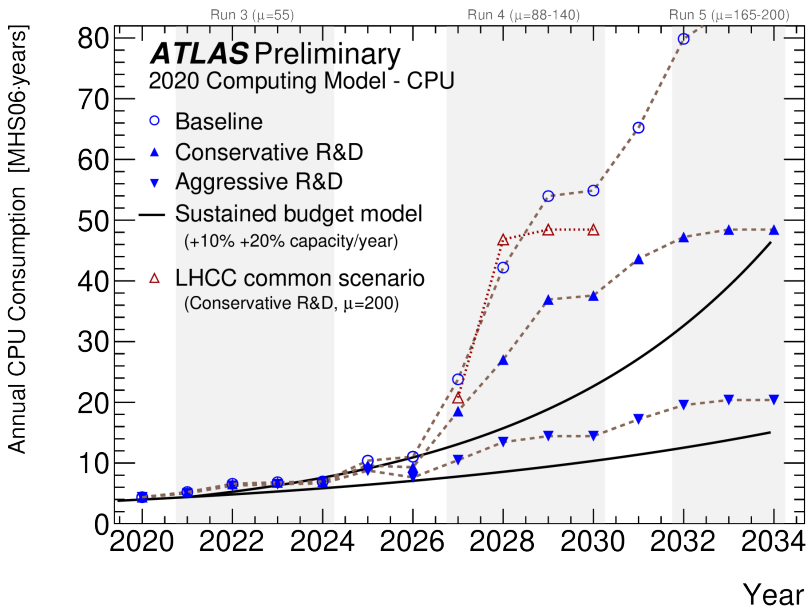
Computing for the High-Luminosity-LHC

- Clear challenge
  - Expect exploding demand for computing infrastructure
- Proposed solutions
  - Software improvements
  - **Integration of additional non-HEP resources**
  - **Optimization of existing computing model**
- Pursuing further efficiency improvements and making additional resources available

SUPERSEDED



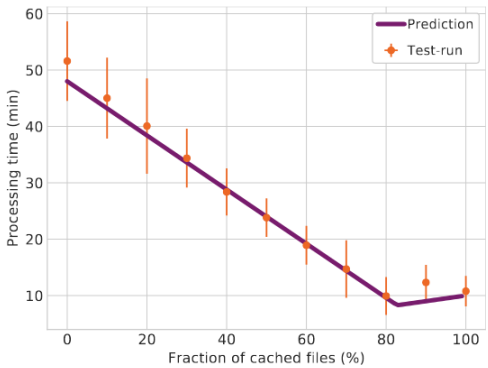
CMS

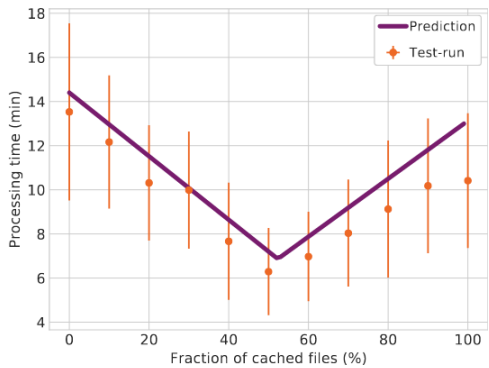


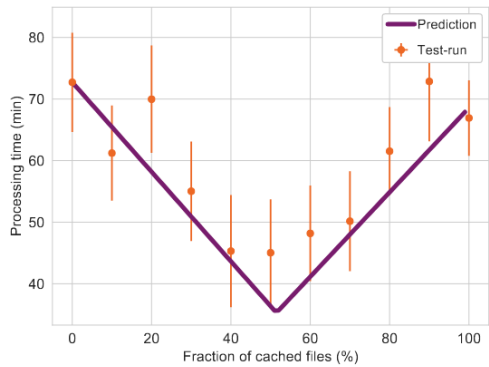
ATLAS

Operation of Individual Local Caches

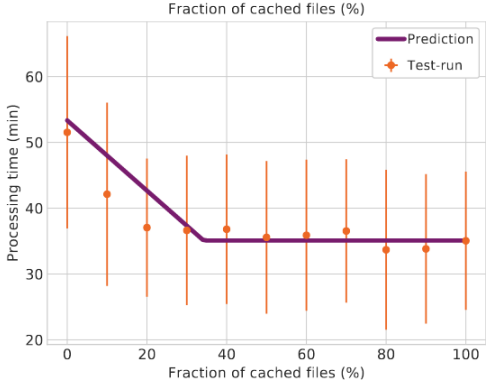
- Tested on several clusters with different workflows
- Used to measure caching performance by M. Sauter

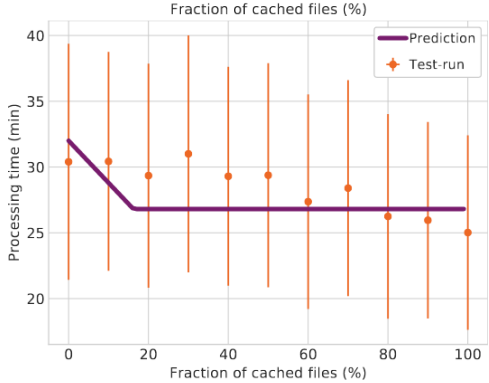


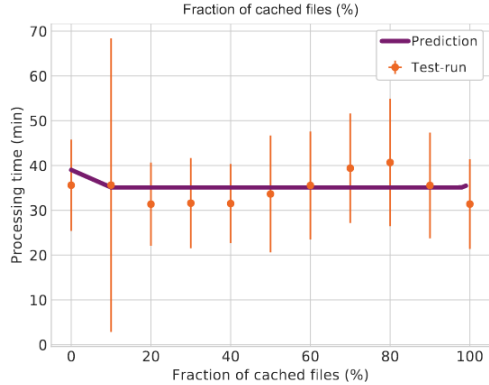




copy jobs







HEP workflow

ETP High Throughput Nodes | TOPAS (KIT Tier 3) | NEMO Freiburg

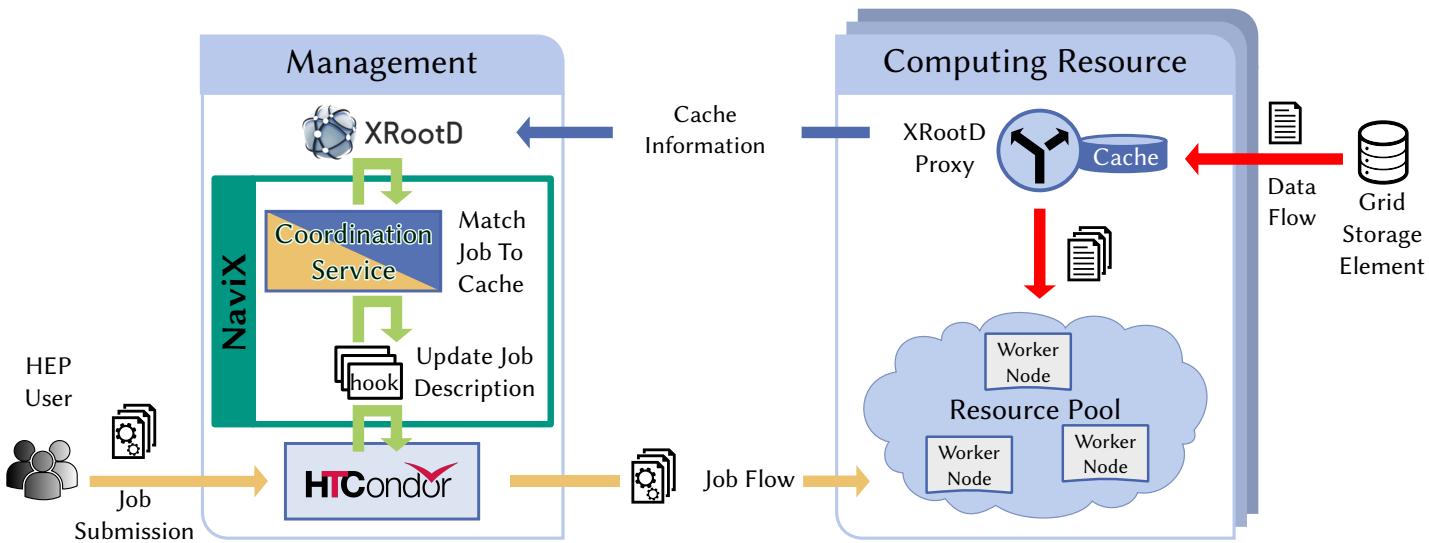
⇒ Simple caching scenarios work

Ideas for an Efficient CDN for HEP

- Wildly distributing jobs on distributed infrastructure might lead to
  - → Significant amount of jobs not benefiting from cached data
  - → Redundant replicas of data wasting cache space

⇒ Actively coordinate jobs?

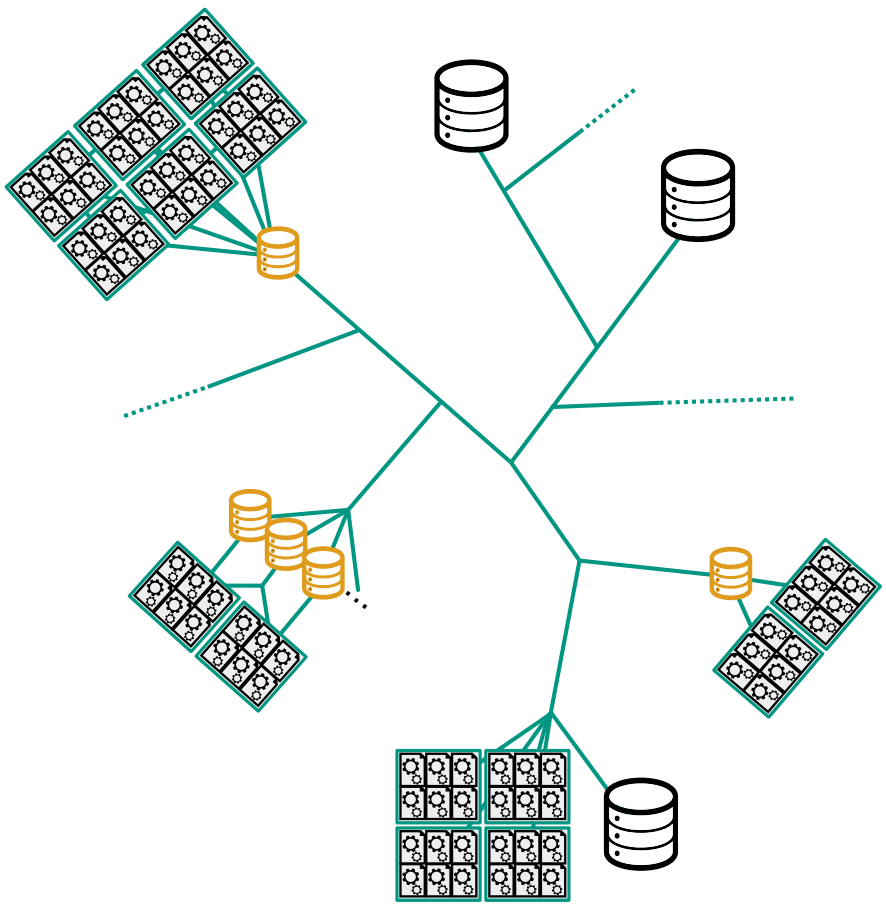
- Latest technology for **cache-aware scheduling**: NaviX



- Proof-of-concept: Used in production on a local Tier-3 cluster at KIT

A Content Delivery Network for HEP

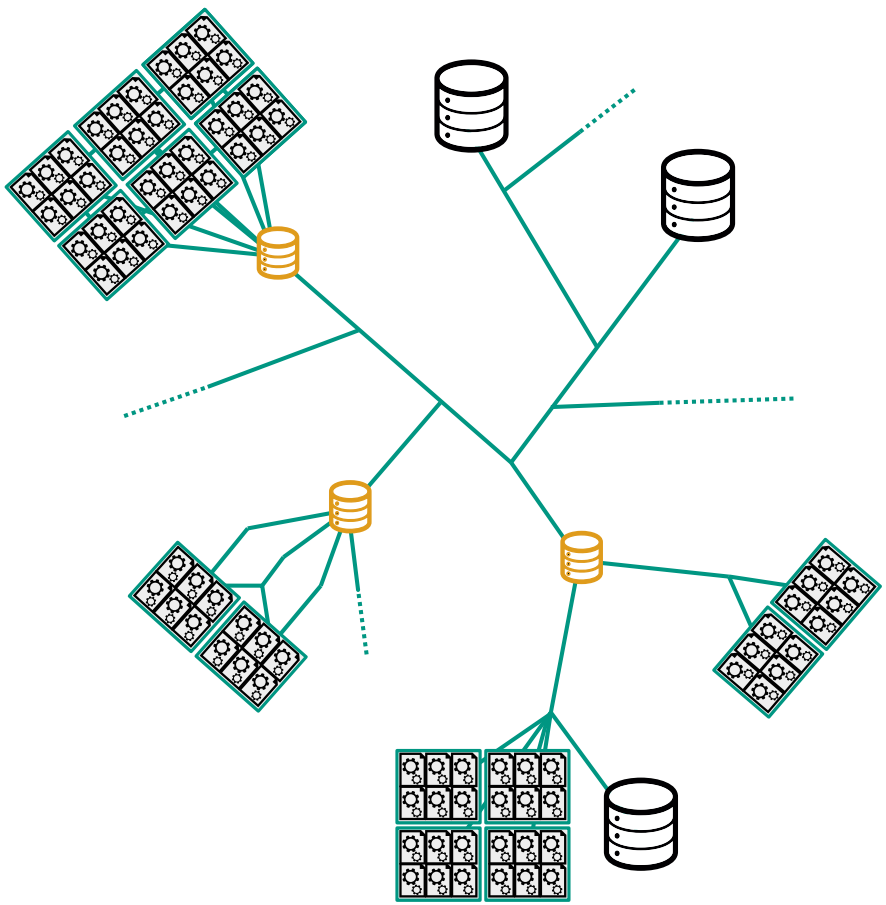
Where, how (large) and when do we place caches?



- In closest vicinity to processing sites $\rightarrow$ minimize WAN load

A Content Delivery Network for HEP

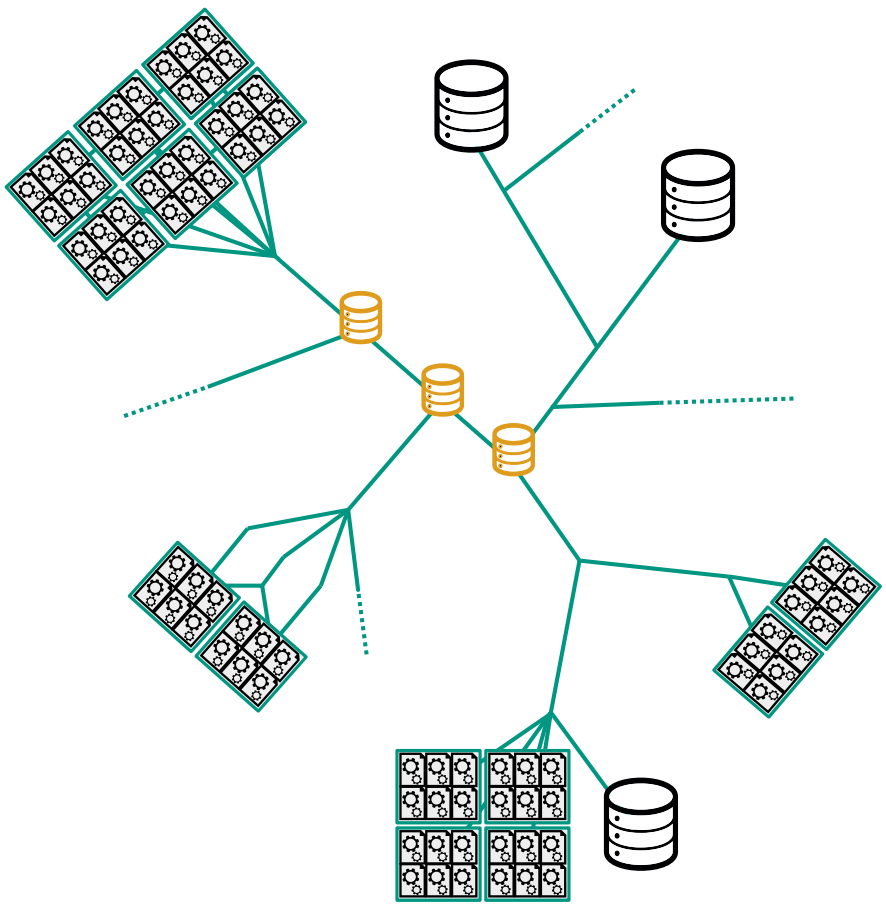
Where, how (large) and when do we place caches?



- Deeper inside network → benefit more sites, distribute network loads to WA(ish)N and LA(ish)N

A Content Delivery Network for HEP

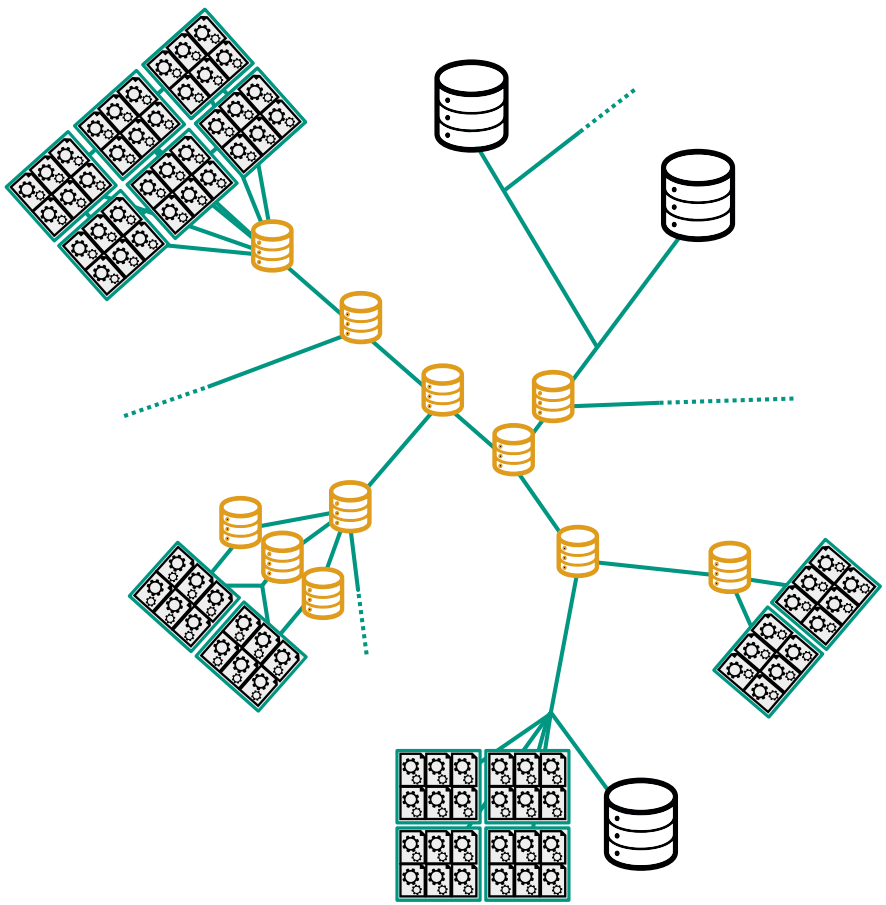
Where, how (large) and when do we place caches?



- Inside network to maximize amount of profiting sites

A Content Delivery Network for HEP

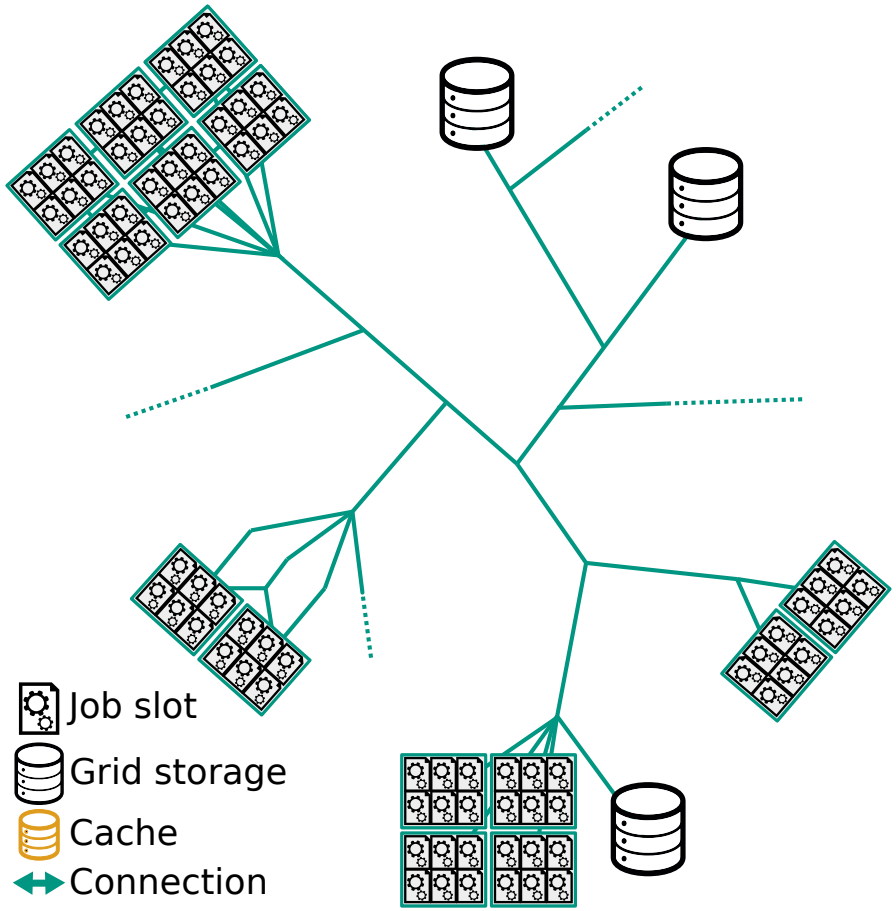
Where, how (large) and when do we place caches?



- In closest vicinity to processing sites $\rightarrow$ minimize WAN load
- Deeper inside network $\rightarrow$ benefit more sites, distribute network loads to WA(ish)N and LA(ish)N
- Inside network to maximize amount of profiting sites
- Combination of all

A Content Delivery Network for HEP

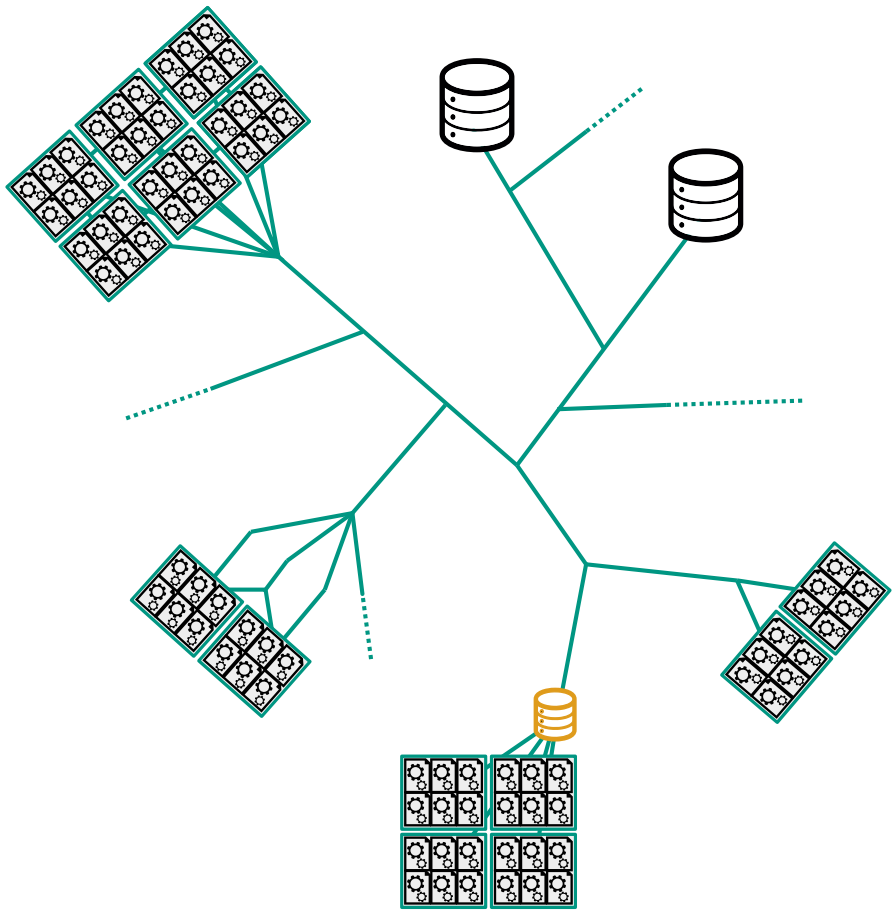
Can we even **replace managed storage** with caches . . . ?



- Replace an expensive managed storage . . .

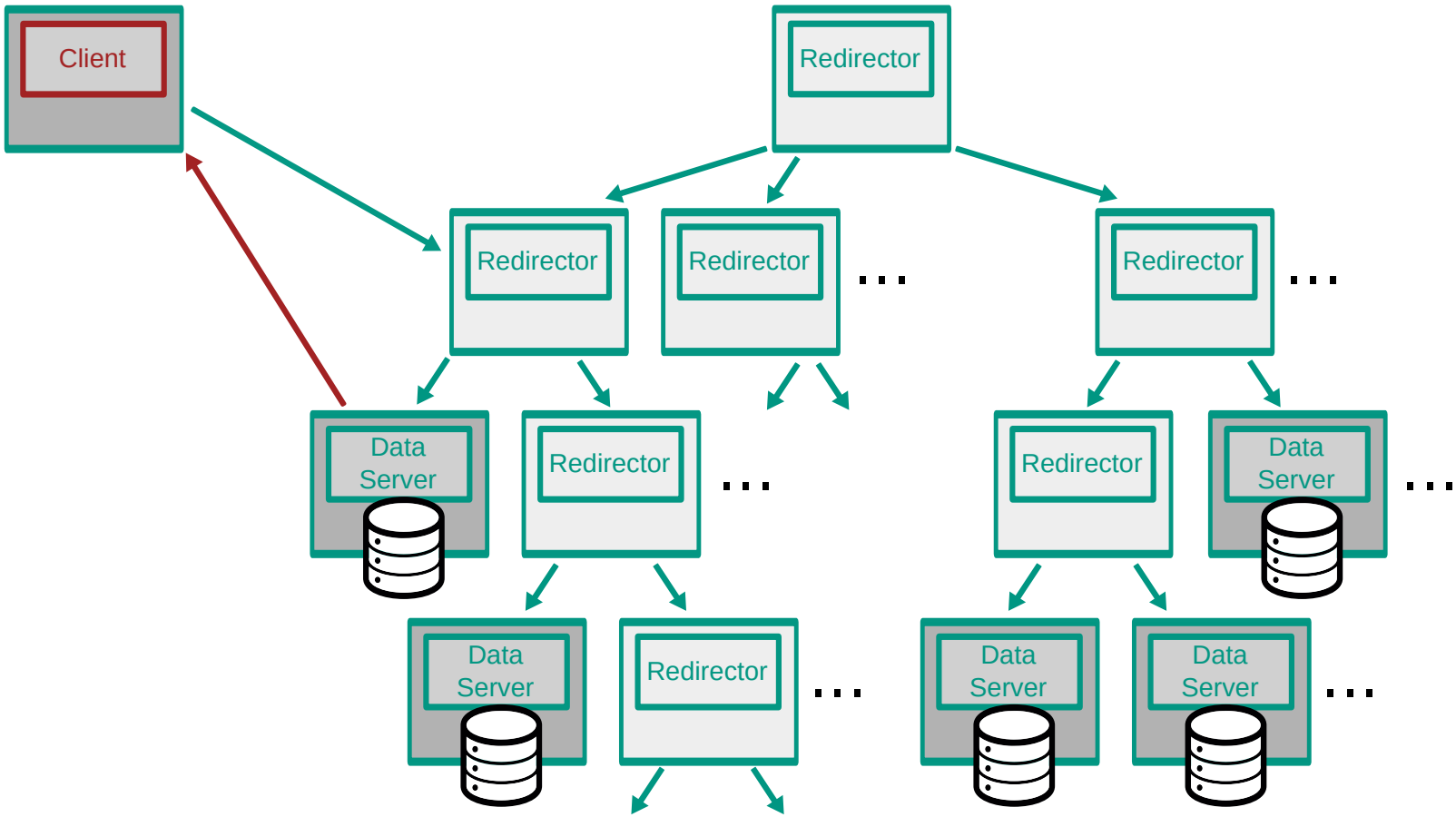
A Content Delivery Network for HEP

. . . or just one cache?



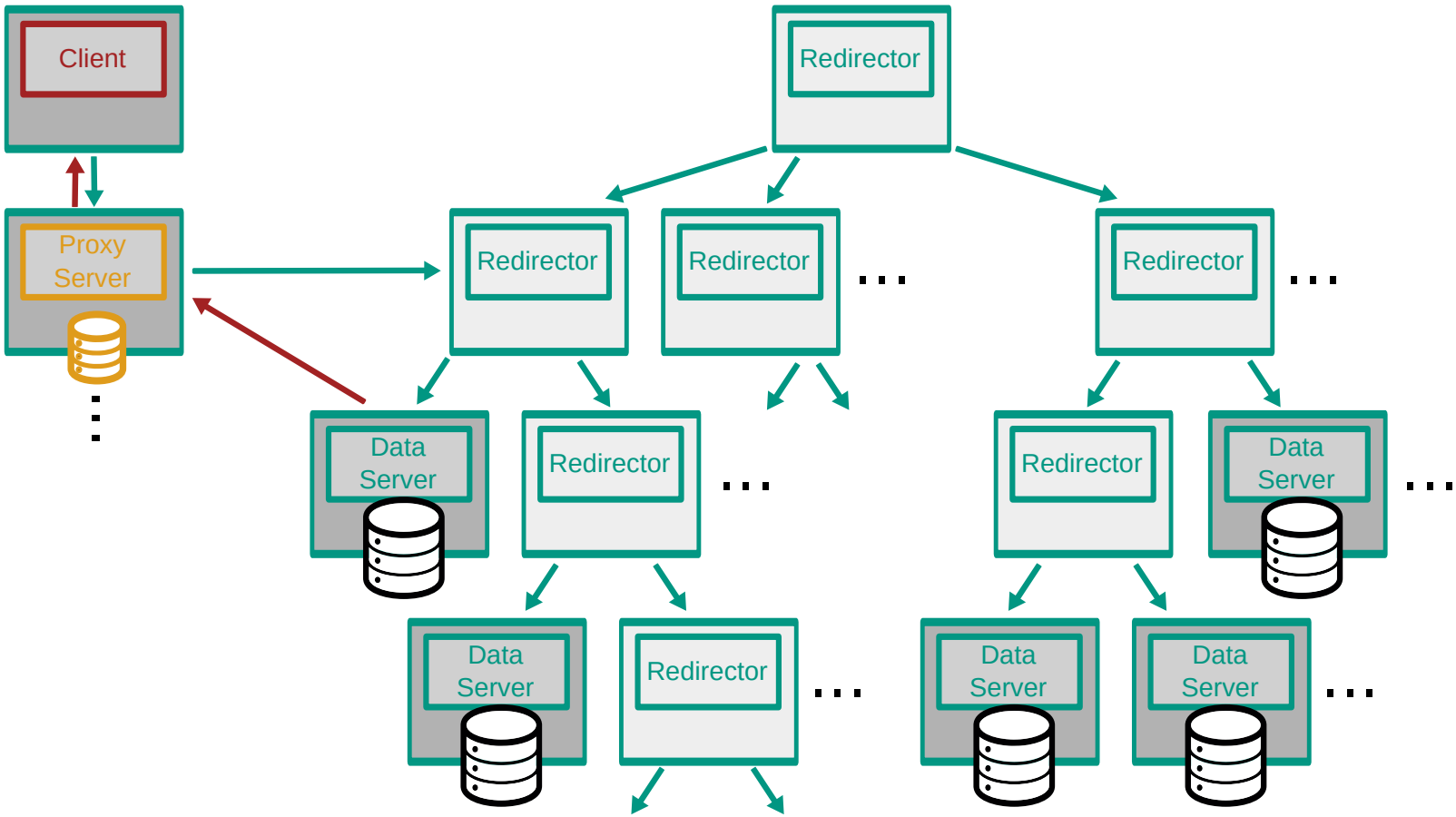
- . . . with a non-crtitical and cheap cache system

Any Data, Anytime, Anywhere: XRootD



- XRootD is an established technology in HEP
- On client side, data can be accessed independent of location

Any Data, Anytime, Anywhere: XRootD

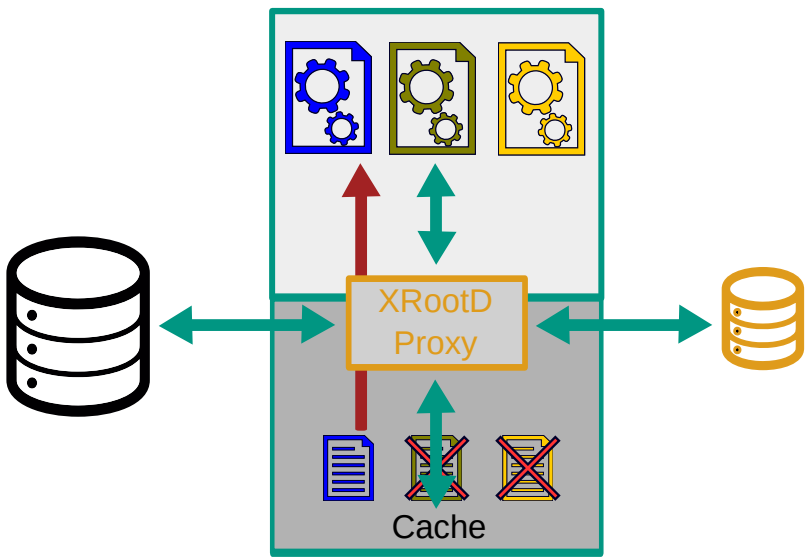


- Transparent usage of proxy servers to cache data already included in XRootD

Caching Tools

XCache

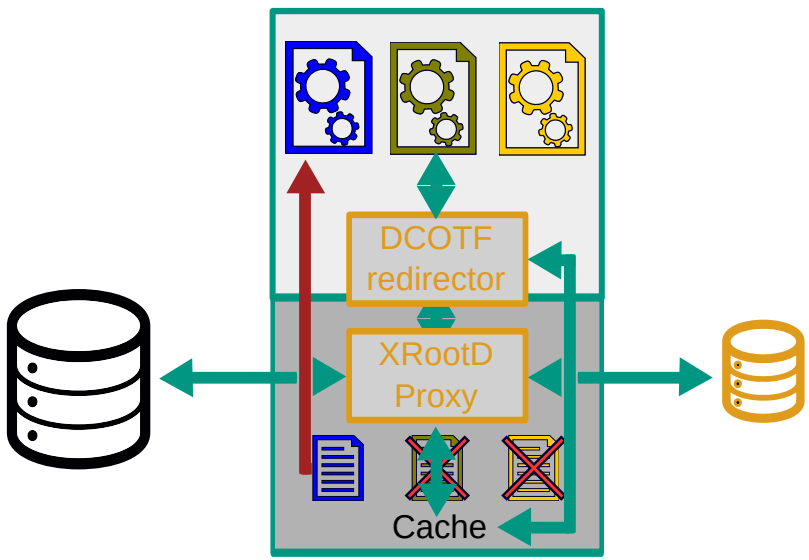
- Plugin for the XRootD proxy file caching



- File access via XRootD proxy

DCOTF

- **D**isk-**C**aching-**O**n-**T**he-**F**ly
- Extension of XCache



- Direct access to local file system

Simulator Software

- MONARC(2) is discontinued → Need modern solution
- SIMGRID: Library of low-level simulation abstractions for distributed systems in gen. (Actors using Platform through Activities)
- WRENCH: High-level building blocks (Services specifying Activities)
  - Wide user base
  - Validated accuracy
  - **Very supportive and highly motivated developers**, special thanks to Henri Casanova, University of Hawai´i
- Addition of (HEP-)specific adaptions (e.g. job-, dataset- & workflow-model, XRootD)

⇒ Reimplementation of MONARC-inspired simulator tool based on WRENCH

SimGrid and Wrench

Karlsruhe Institute of Technology

SimGrid

- 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

- High-level simulation abstractions based on SimGrid

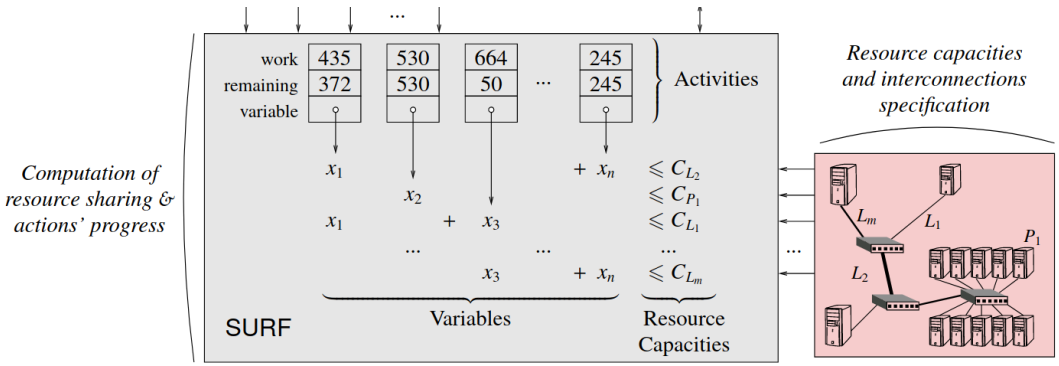
SimGrid Engine

- “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



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

SimGrid 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 [DOI:10.1145/2517448]
  - and improved execution time

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

with TCP version specific parameters α and β tuned by packet-level simulation valid for transfers of data of size $\geq$ 100KiB

SimGrid Platform Description

- Simulated hardware platform consisting of clusters of hosts, storage resources, links, routes, etc.

```
<platform version="4.1">
    <zone id="AS0" routing="Full">

        <!-- The host on which the WMS will run -->
        <host id="WMSHost" speed="10Gf" core="1">
            <disk id="hard_drive" read_bw="100MBps" write_bw="100MBps">
                <prop id="size" value="5000GiB"/>
                <prop id="mount" value="/"/>
            </disk>
        </host>

        <!-- The host on which the BareMetalComputeService will run -->
        <host id="ComputeHost" speed="1Gf" core="10">
            <prop id="ram" value="16GB" />
        </host>

        <!-- A network link that connects both hosts -->
        <link id="network_link" bandwidth="50MBps" latency="20us"/>
        <!-- WMSHost's local "loopback" link -->
        <link id="loopback_WMSHost" bandwidth="1000EBps" latency="0us"/>
        <!--ComputeHost's local "loopback" link -->
        <link id="loopback_ComputeHost" bandwidth="1000EBps" latency="0us"/>

        <!-- Network routes -->
        <route src="WMSHost" dst="ComputeHost">
            <link_ctn id="network_link"/>
        </route>

        <!-- Each loopback link connects each host to itself -->
        <route src="WMSHost" dst="WMSHost">
            <link_ctn id="loopback_WMSHost"/>
        </route>
        <route src="ComputeHost" dst="ComputeHost">
            <link_ctn id="loopback_ComputeHost"/>
        </route>

    </zone>
</platform>
```

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

WRENCH Workflow-Management-System

- Workflow: collection of tasks with file- and task-dependencies
- “Workflow Management Systems” provide mechanisms for executing workflow applications via jobs (cluster of tasks combined with file location information)

Algorithm 1 Blueprint for a WMS execution

```
1:  procedure MAIN(workflow)
2:      Obtain list of available services
3:      Gather static information about the services
4:      while workflow execution has not completed/failed do
5:          Gather dynamic service/resource information
6:          Make data/computation scheduling decisions
7:          Interact with services to enact decisions
8:          Wait for and react to the next event
9:      end while
10:     return
11: end procedure
```

- A WMS-API provides the interface for the user to simulate a workflow
- In future workflow part factorized out of WRENCH

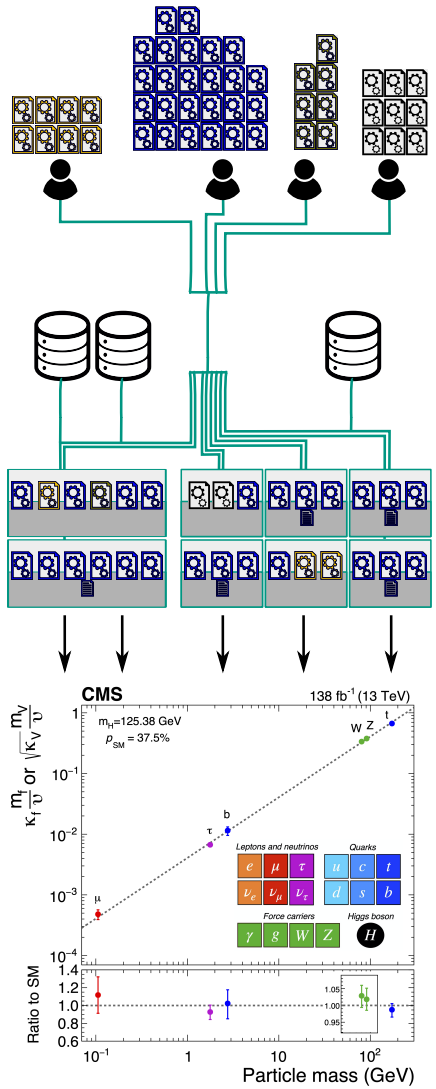
HTCondor Compute-Service

- 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 include ClassAds

Distributed-Cache-Simulator

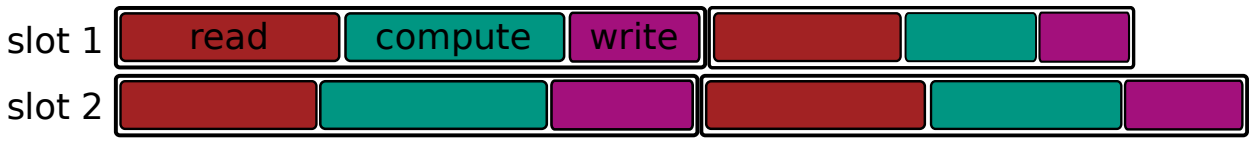
Designed to simulate HEP workflows

1. Define workflow of jobs with certain characteristics (FLOP, Memory, In-&Output-files)
2. Define platform (network & hosts) with certain characteristics (N_{core}, CPU-speed, RAM, disk, bandwidth) and roles (worker, storage, cache, scheduler, . . .)
3. Instantiate input-files
4. **Start the simulation!**
   - Jobs are scheduled and run
   - Input-files are streamed and cached
   - Caches evict files if necessary
   - Job dynamics are monitored



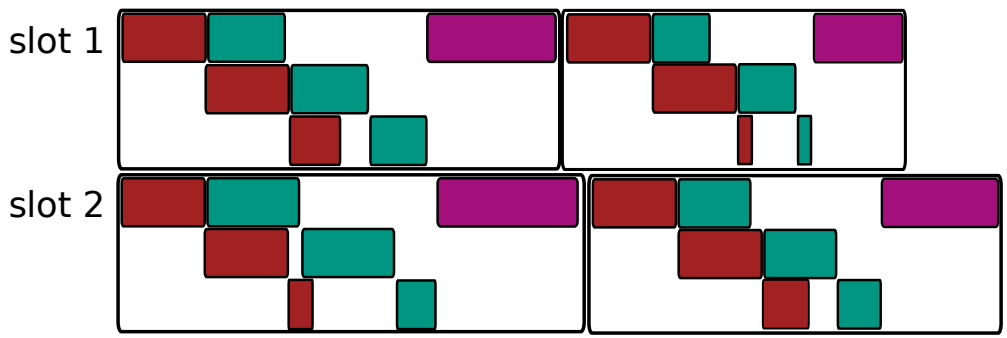
Copy and Streaming Jobs

Copy jobs



- Sequentially read, compute and write

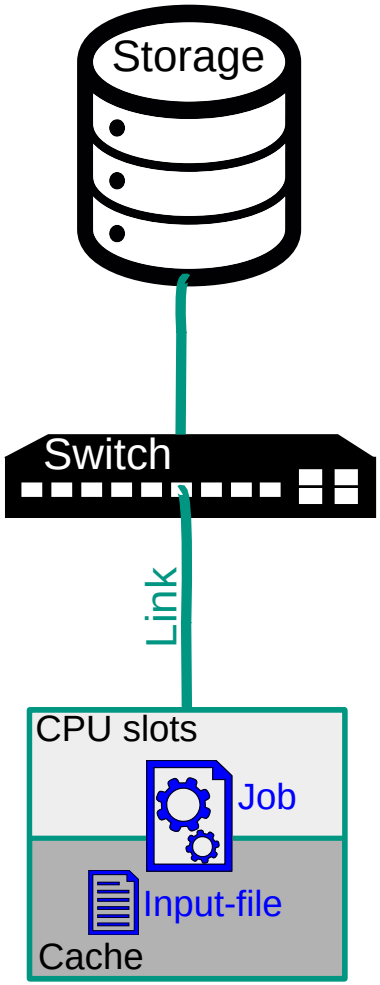
Streaming jobs

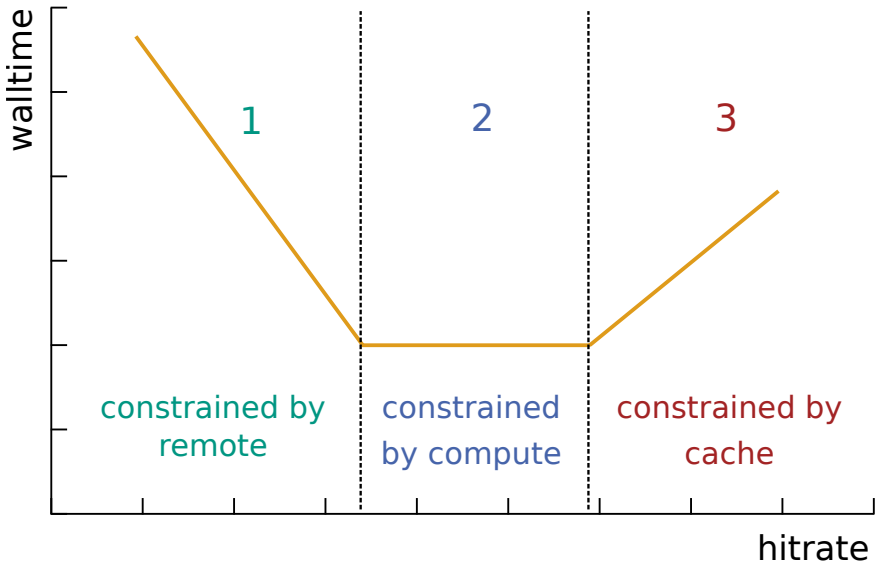


- Can concurrently read and compute (enabled by XRootD in HEP)

→ More compact pattern and reduced duration

Simplest Model Expectation "Bathtub Pattern"





Wall-time estimation T. Feßenbecker:

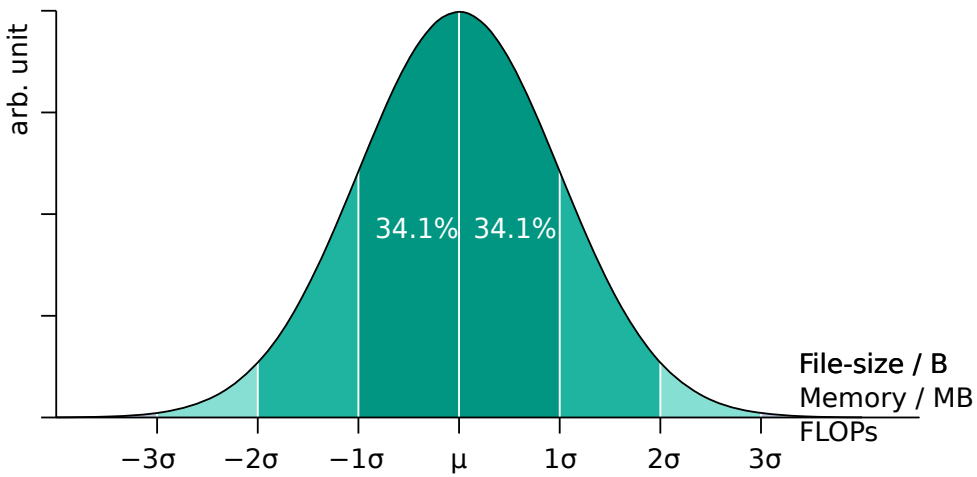
$$t_{\text{wall}}(t) = \max\left(\frac{V\cdot(1-h)}{b_{\text{remote}}(t)}, t_{\text{CPU}}, \frac{V\cdot h}{b_{\text{cache}}(t)}\right)$$

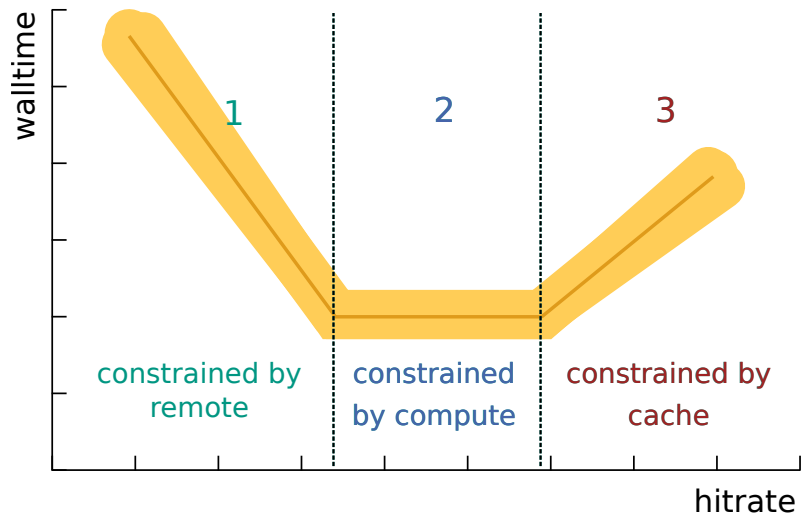
Hit-rate:

$$h = \frac{\text{input-file-sizes on cache}}{\text{all input-file-sizes}}$$

Bathtub ⊗ Job Characteristics

- Sampling job characteristics from a (truncated) Gaussian distribution



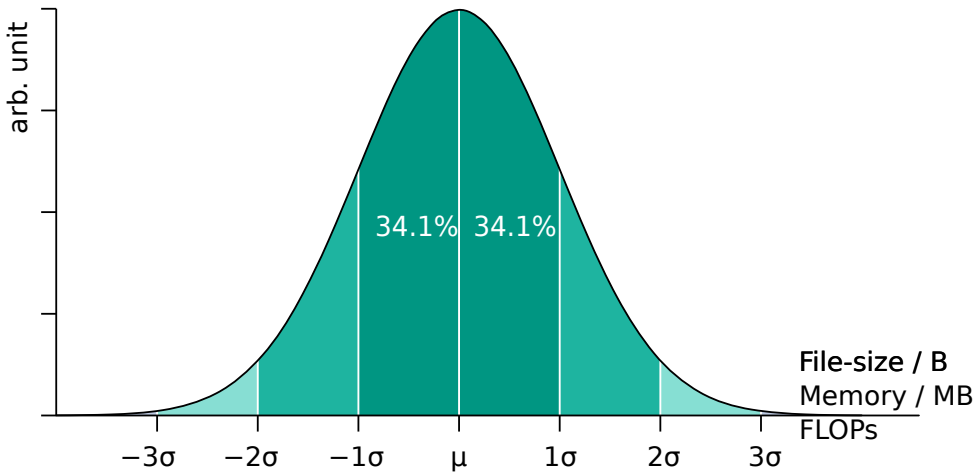


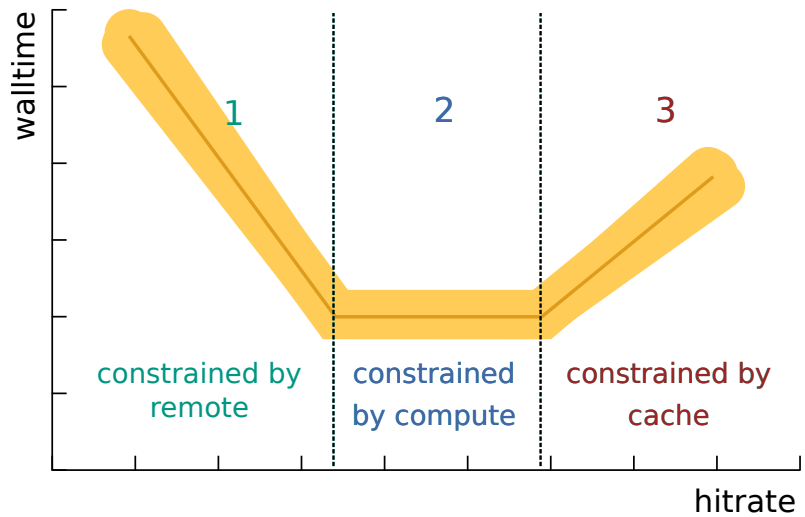
⇒ pdf_{Job} ⊗ Bathtub = Smeared Bathtub

- Is this the whole story?

Bathtub ⊗ Job Characteristics

- Sampling job characteristics from a (truncated) Gaussian distribution

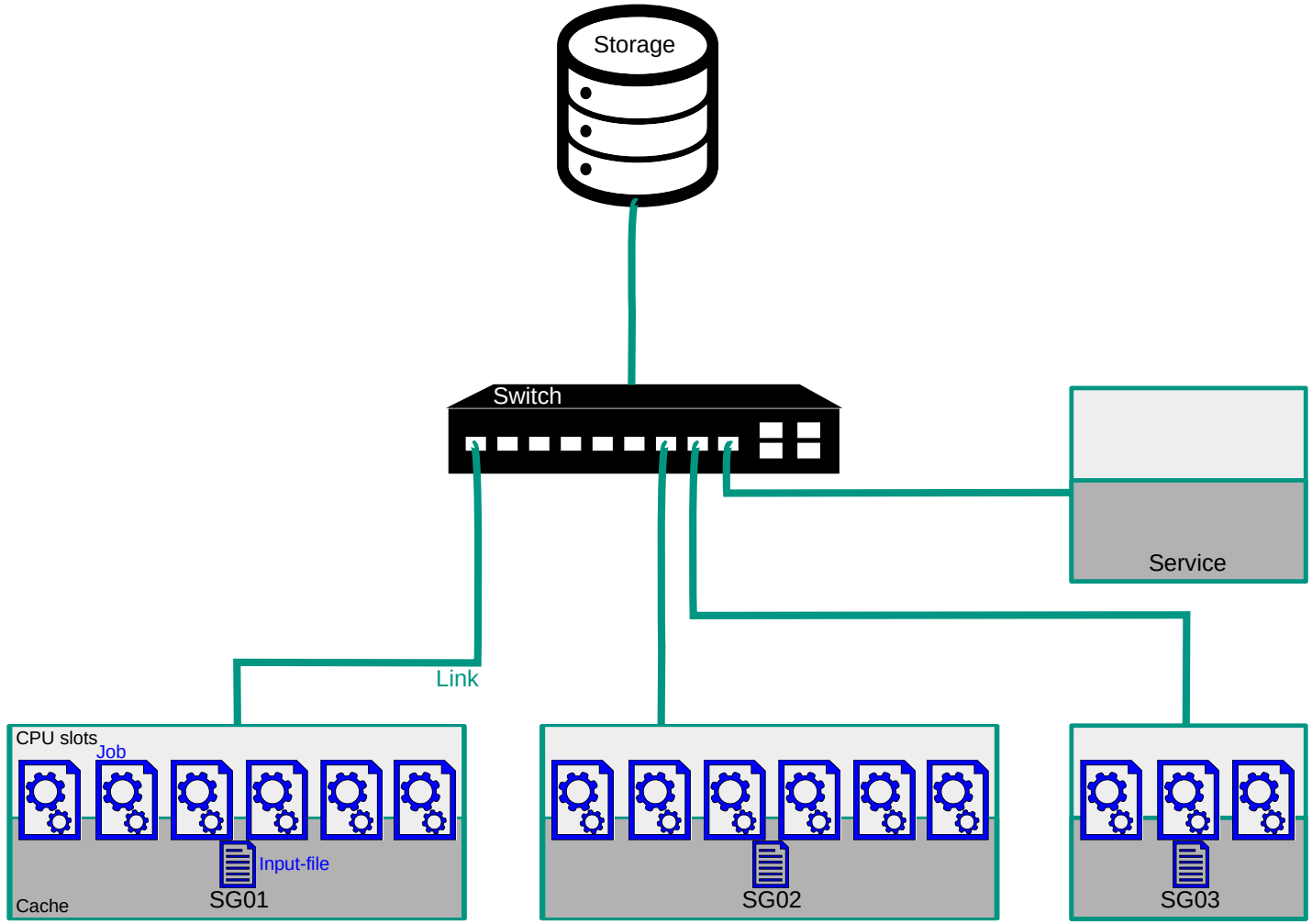




⇒ pdf_{Job} ⊗ Bathtub = Smeared Bathtub

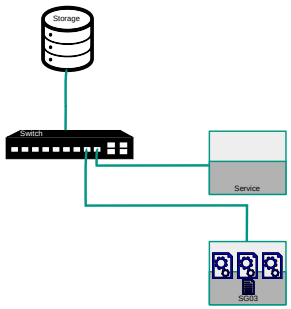
- Is this the whole story? → Didn't include the influence of other entities!

Hypothetical Test Platform



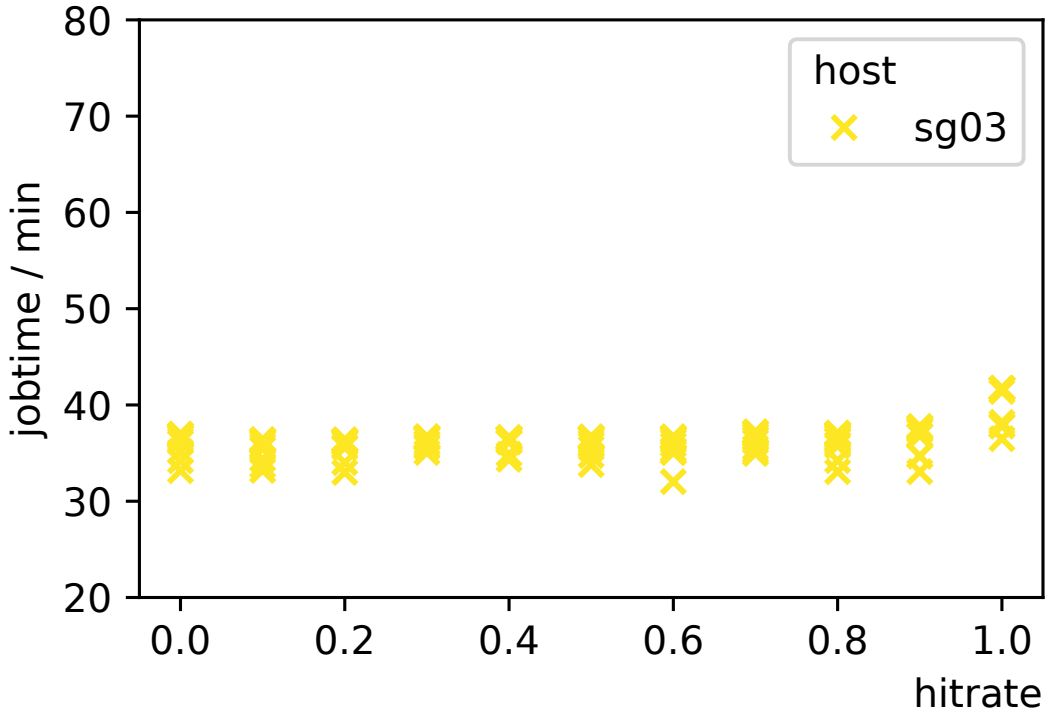
- Same caches & network speeds, different slot numbers & speeds

One Worker

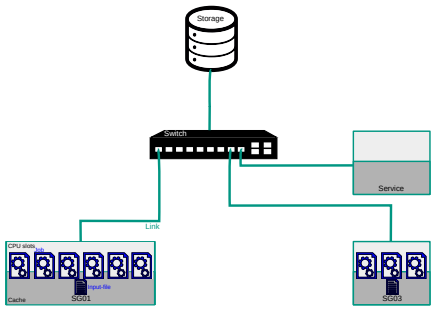


- Starting as many jobs as slots available
- Preparing fixed hitrate at simulation start

- Nice smeared “Bathtub-Type-2&3”
- Small number of jobs → network still fast enough

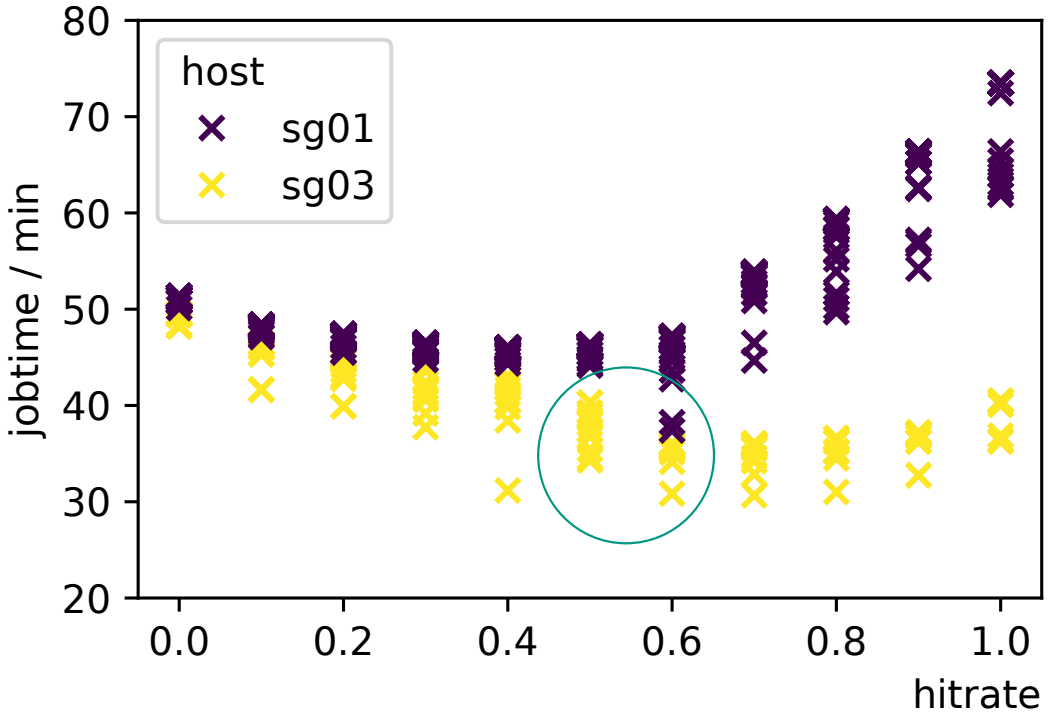


Two Workers

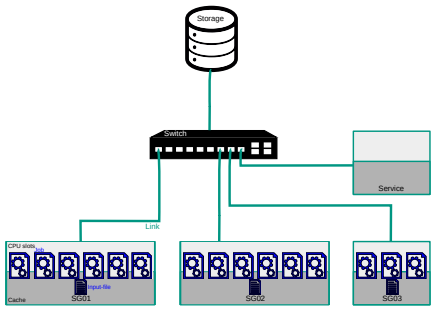


- Above N_{job} threshold → Network throttling
- Unexpected dip at pivot point for `sg03`

- Adding host `sg01` with slower CPU and higher number of slots



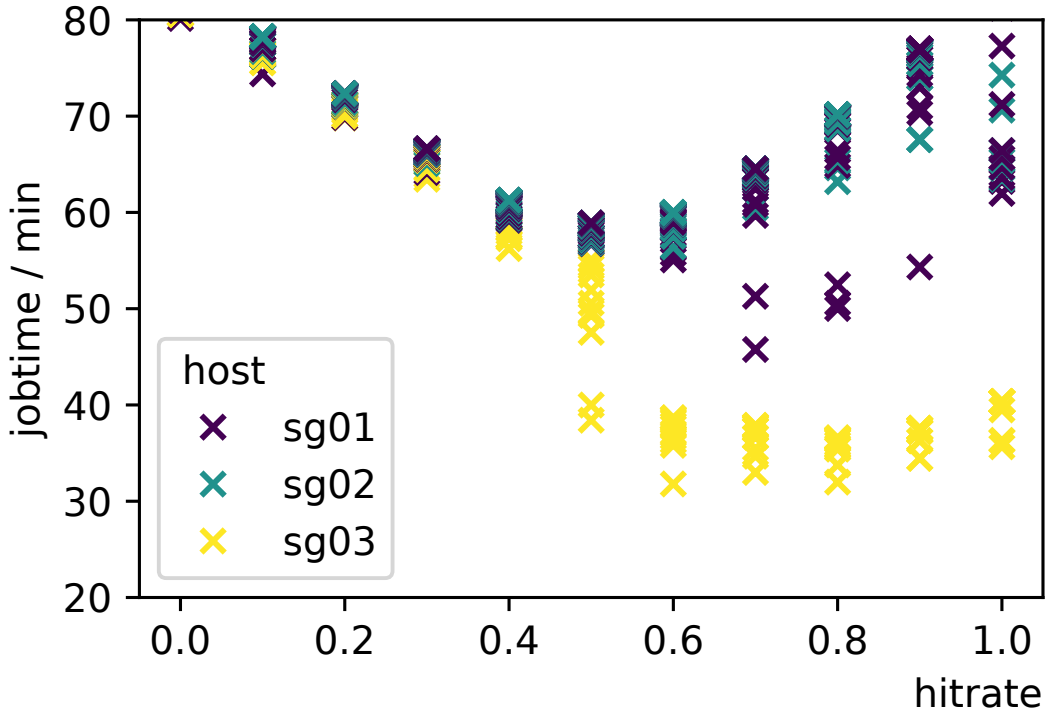
Three Workers



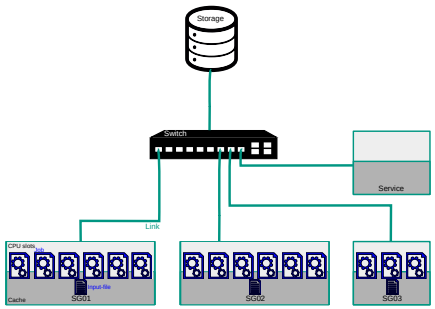
- Adding host `sg02` identical to `sg01`

- Nice “Bathtub-Type-1&3” for `sg01` and `sg02`
- More pronounced inter-machine influence (network throttling & dip)

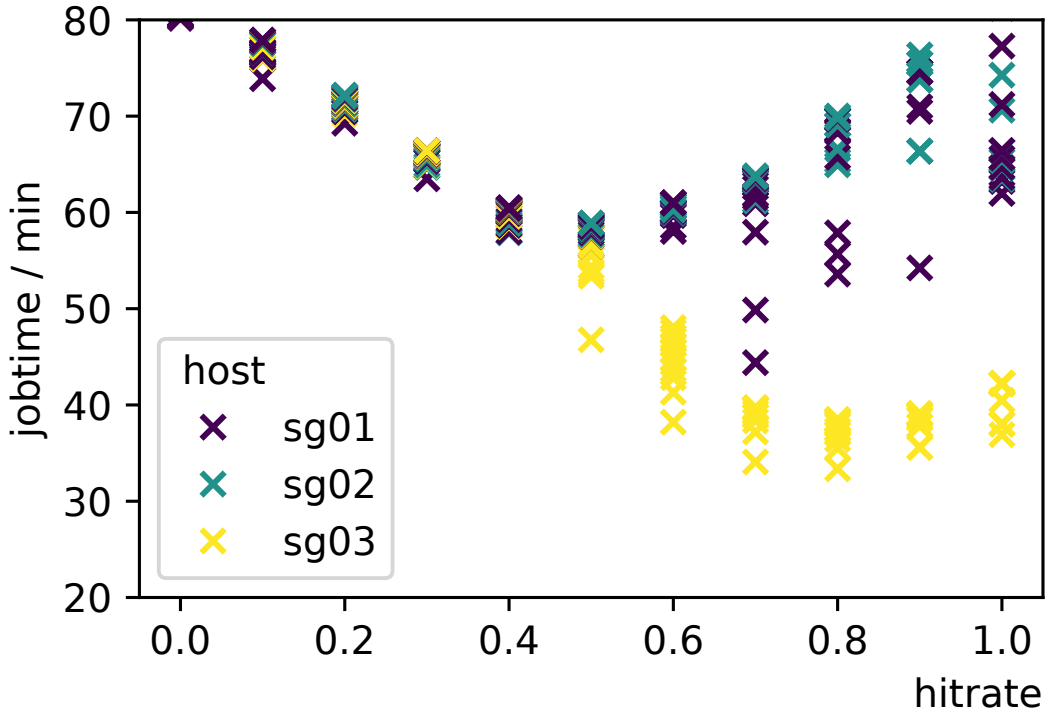
⇒ What is the dip?



Three Workers

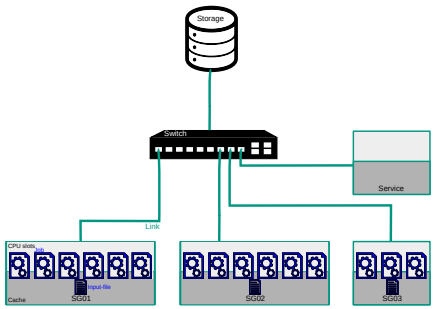


- Setting CPU-speeds on all hosts to same value



- Nice “Bathtub” for all hosts
- The dip is gone

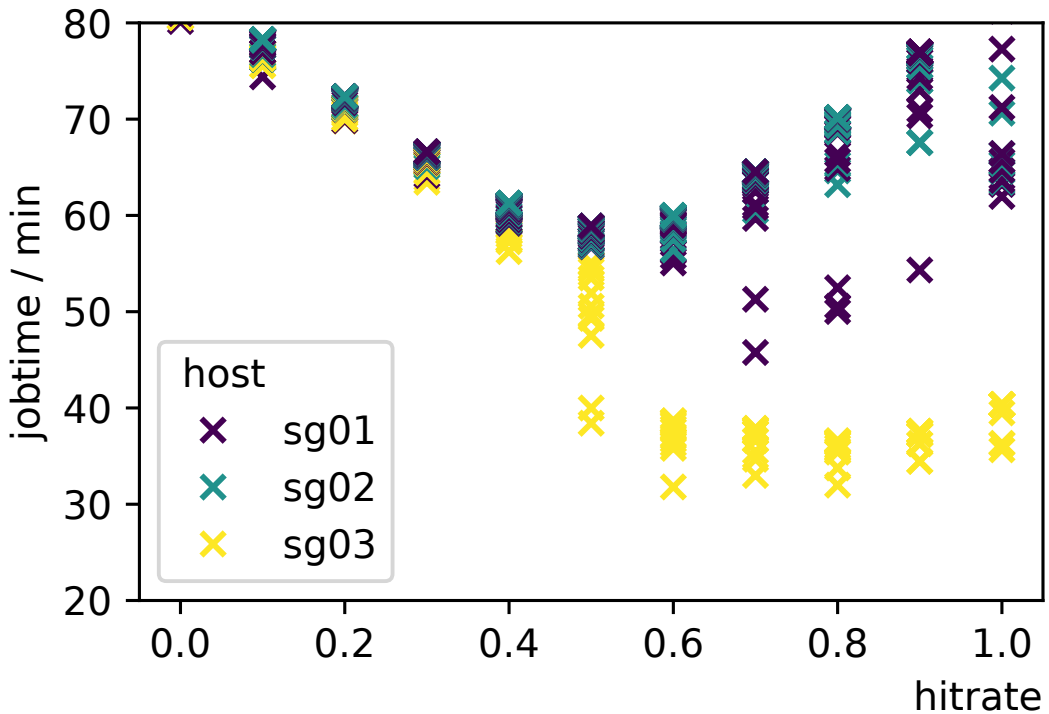
Three Workers



- Throughput limited by CPU/cache for `sg01` & `sg02` → More throughput for `sg03`

⇒ Different CPU-speed of `sg03` decouples the host

Simulator prediction: Interference between jobs due to network is not negligible!



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

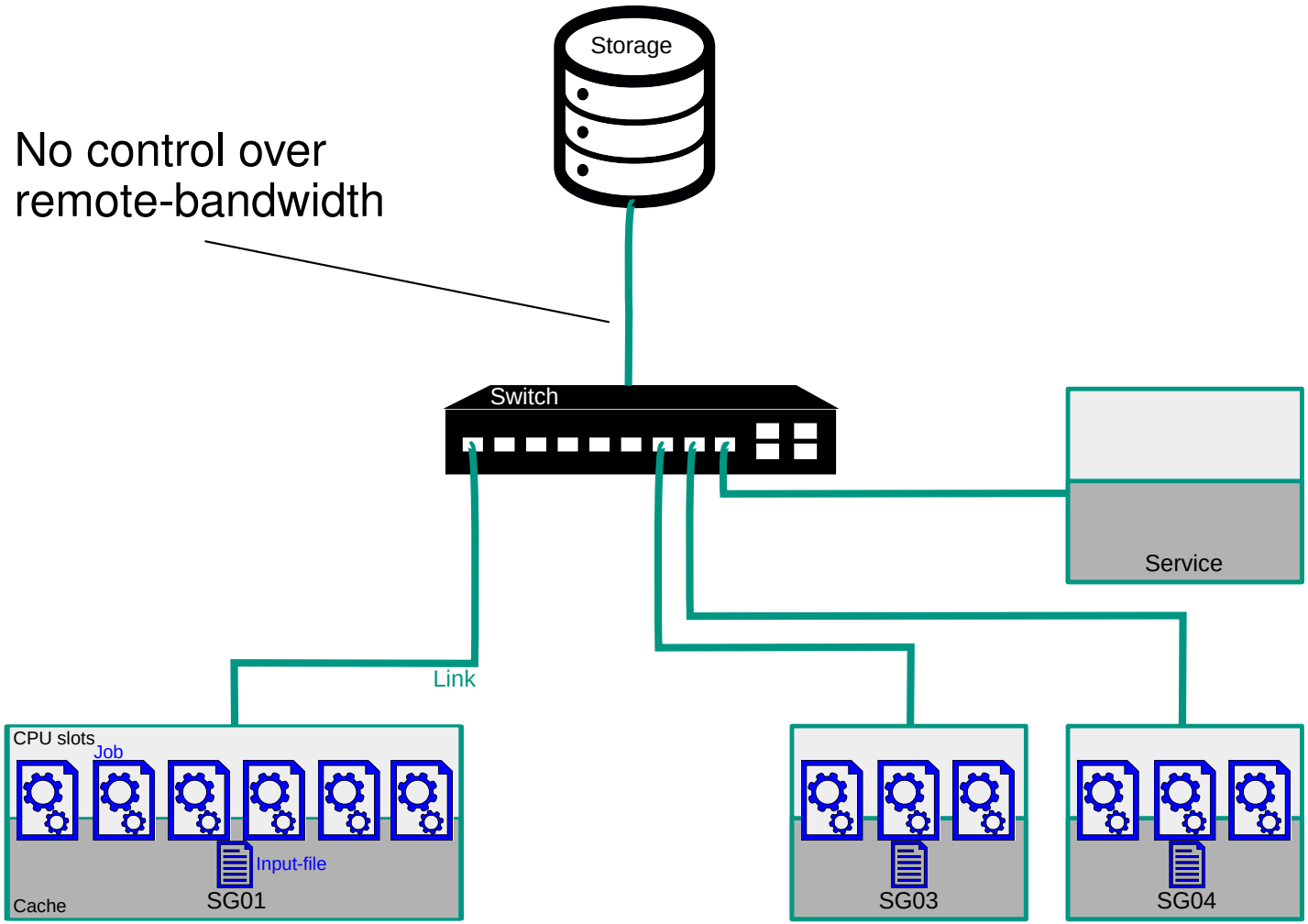
Simulation:

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

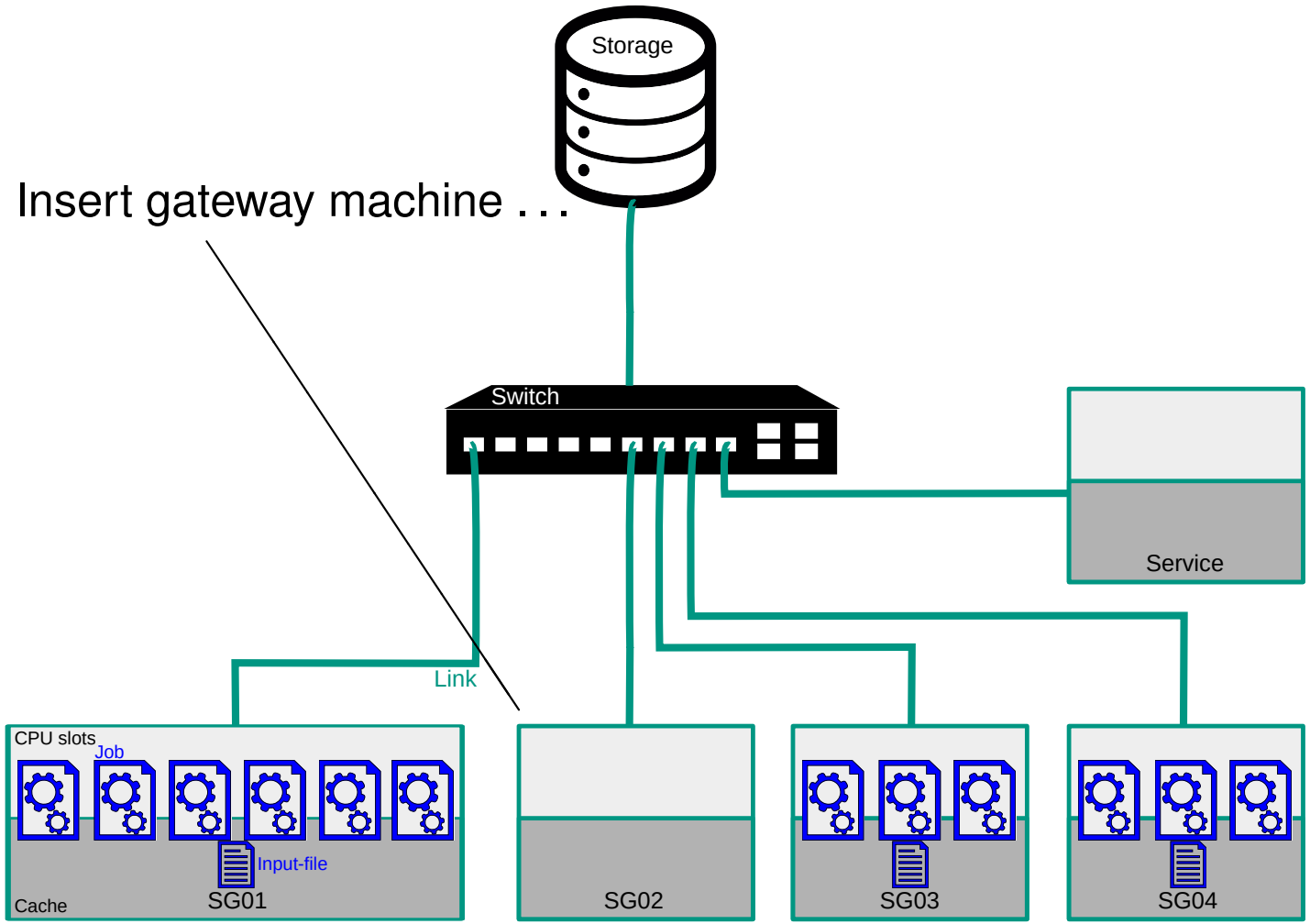
Tune the simulation parameters until simulation fits measurements

Combine differently configured measurements to learn something about the system

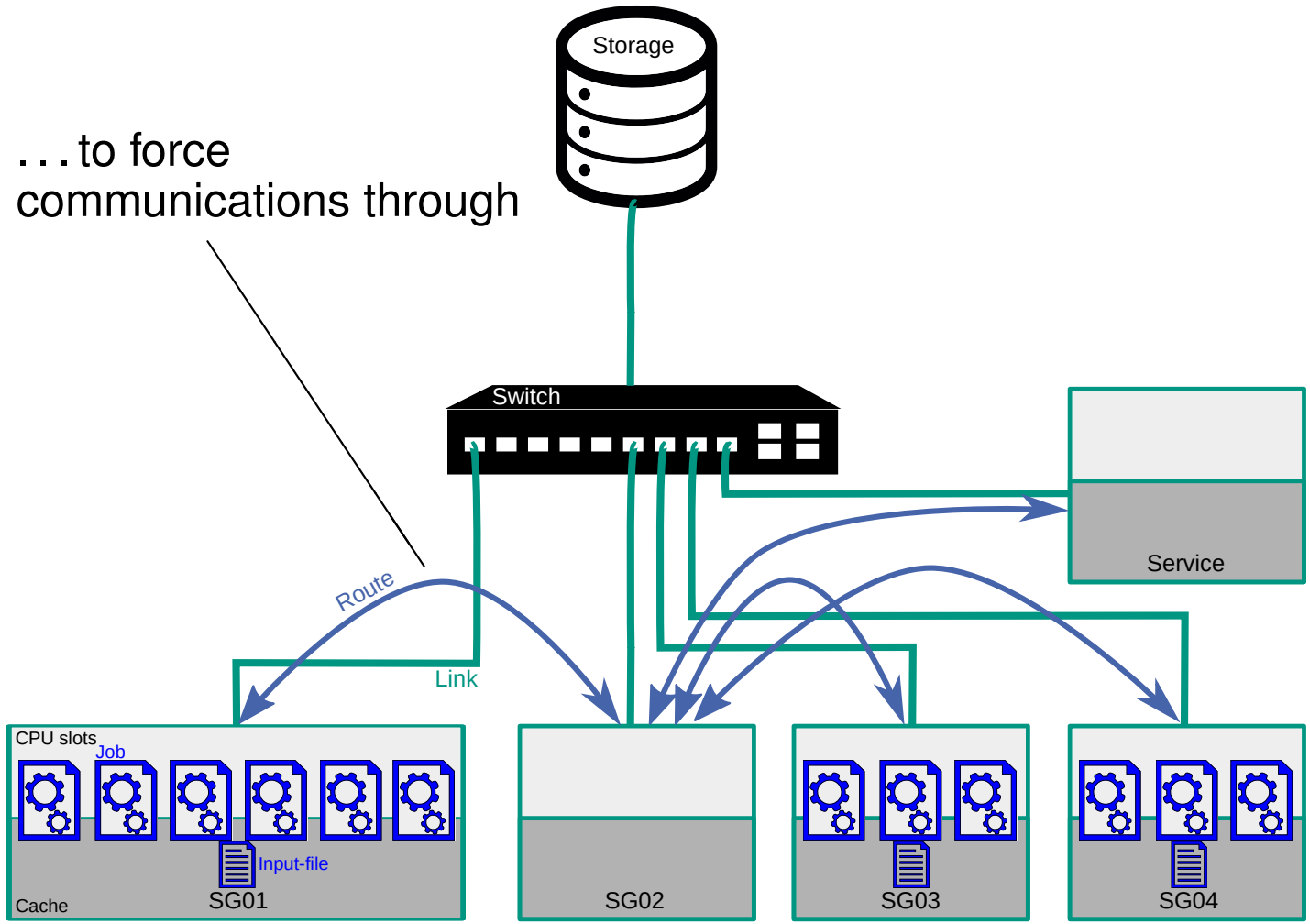
Benchmark Setup



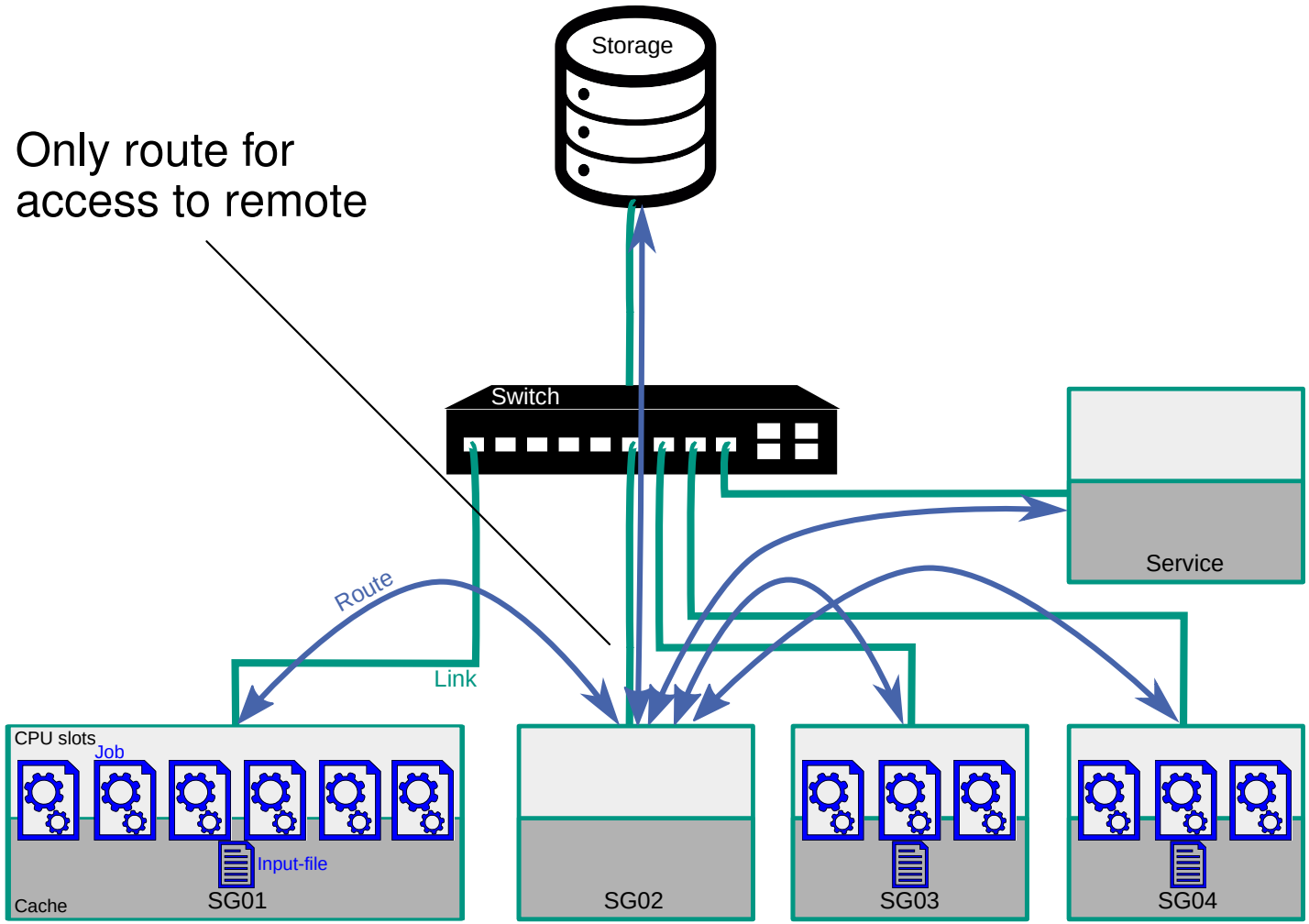
Benchmark Setup



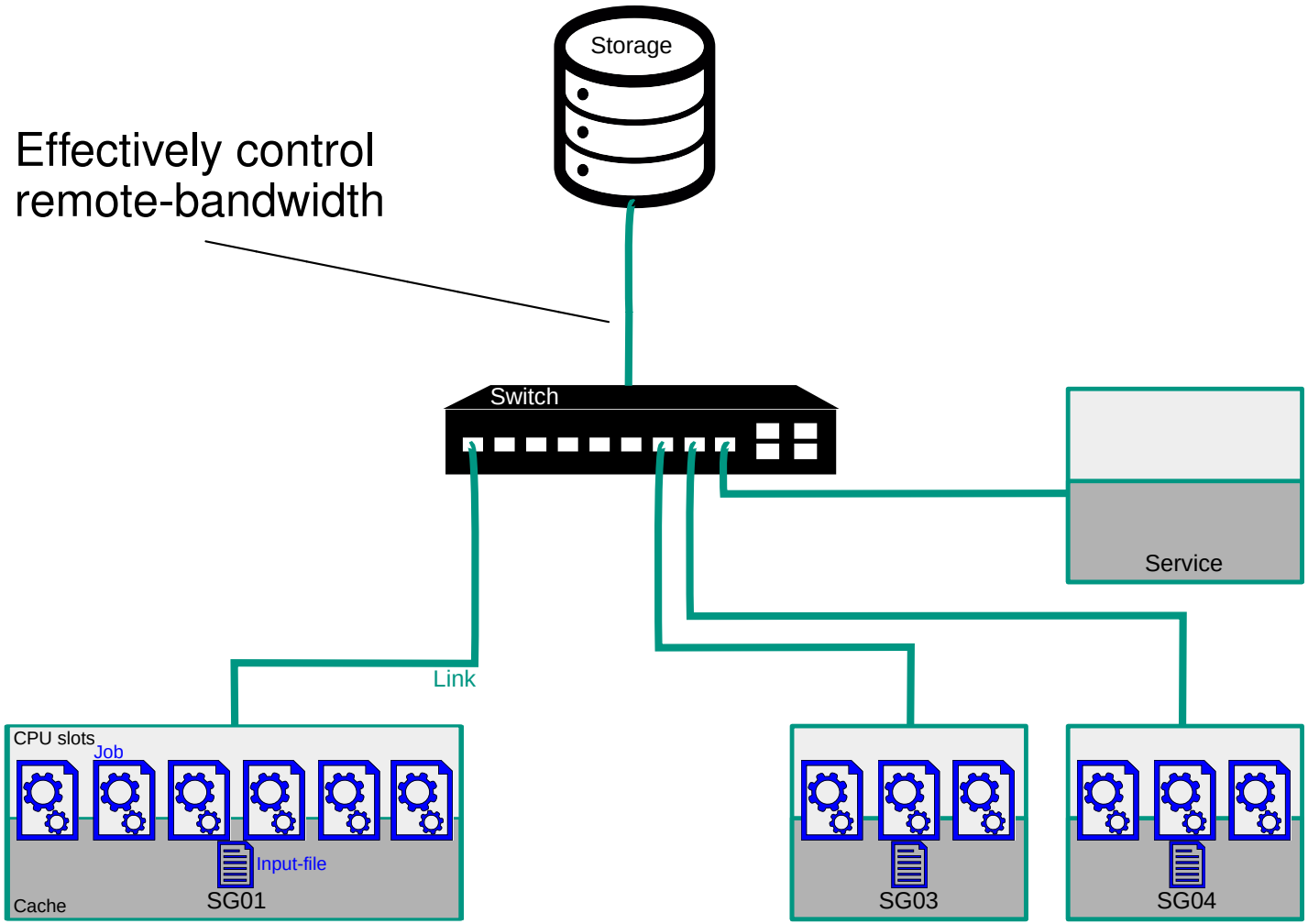
Benchmark Setup



Benchmark Setup



Benchmark Setup



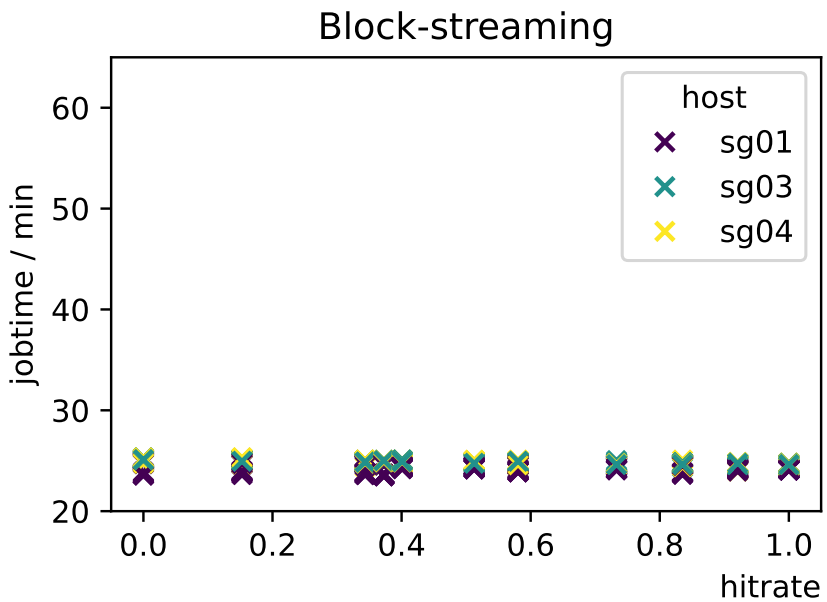
Benchmark Jobs

- Exact replications of the same data analysis job
  - Same executable, executing data transformation and reduction
  - Read 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

Scenario 1: 10G Remote

- Setting gateway machine's network interface to 10 Gbit s^{-1}

Measurement

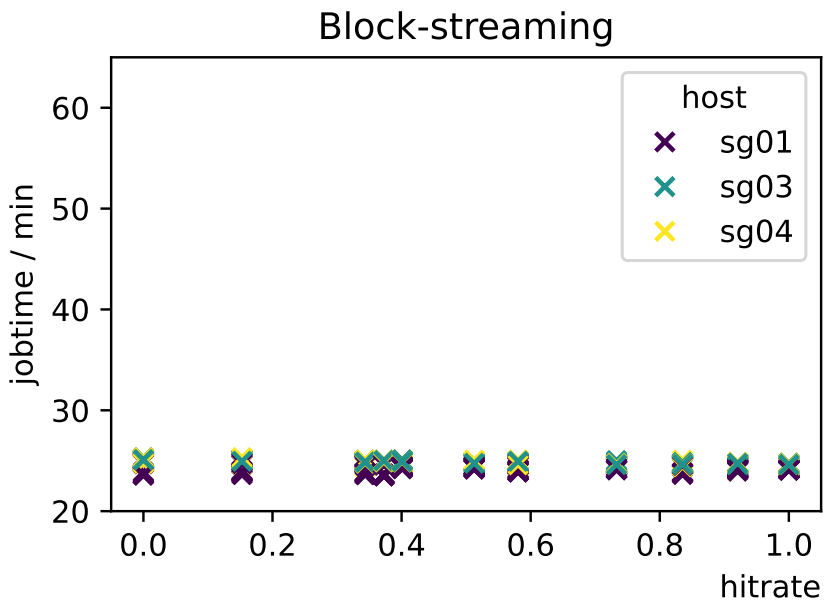


- Clearly CPU limited workflow, no I/O throttling

Scenario 1: 10G Remote

- Setting gateway machine's network interface to 10 Gbit s^{-1}

Measurement



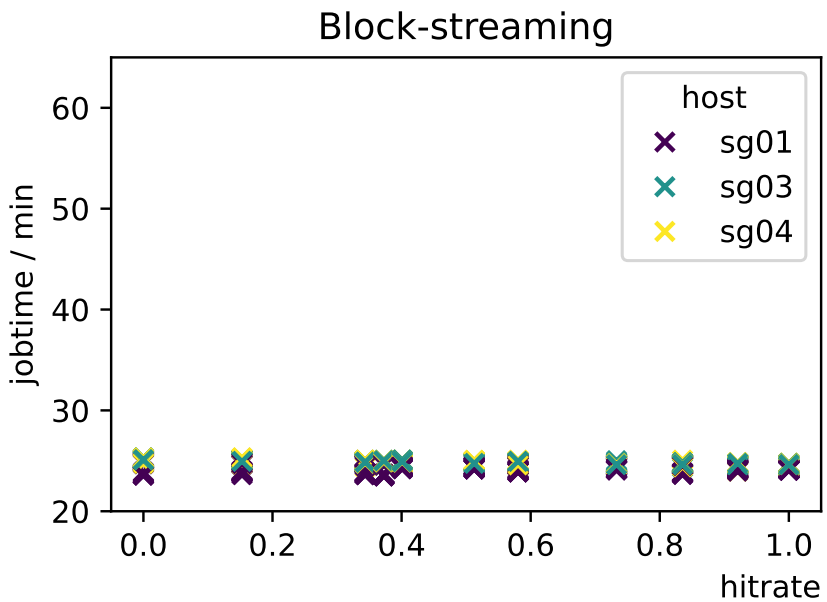
- Clearly CPU limited workflow, no I/O throttling

→ Tune CPU speeds in simulation

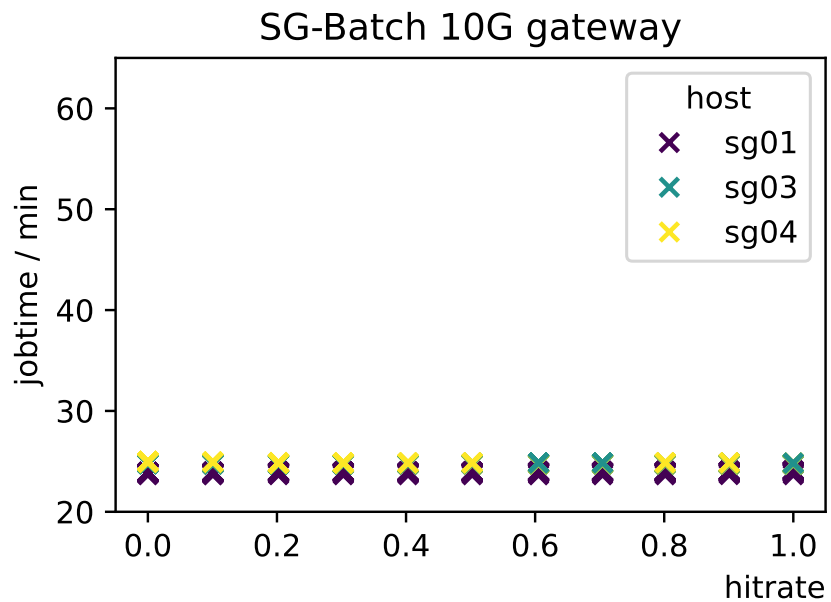
Scenario 1: 10G Remote

- Setting gateway machine's network interface to 10 Gbit s^{-1}

Measurement



Simulation



- Clearly CPU limited workflow, no I/O throttling

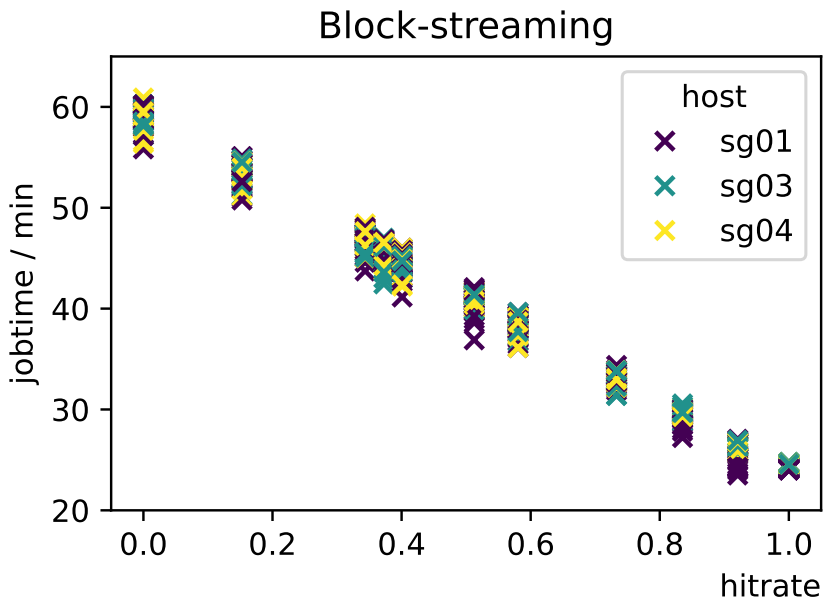
→ Tune CPU speeds in simulation

Scenario 2: 1G Remote

- Lower gateway machine's network interface to 1 Gbit s^{-1}, else the same

→ Reuse condensed information from previous measurement

Measurement



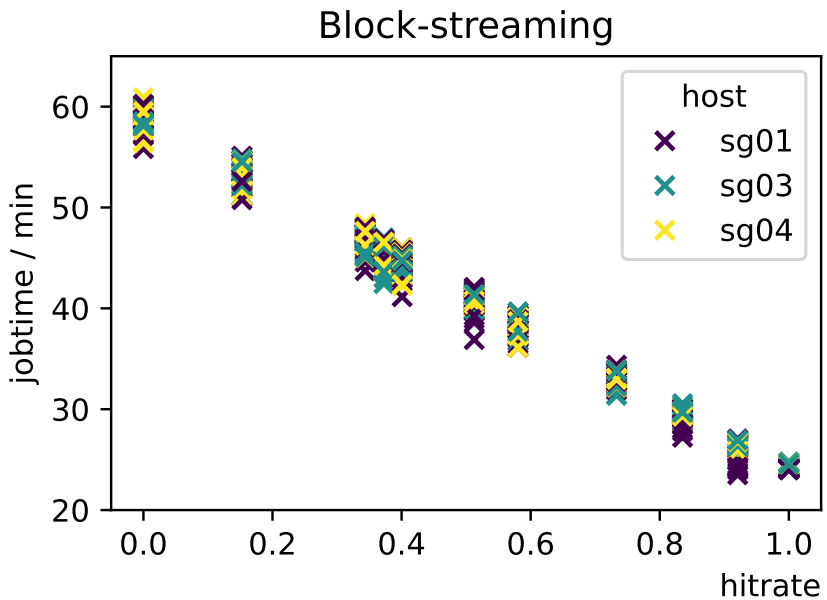
- Limited by I/O via remote network

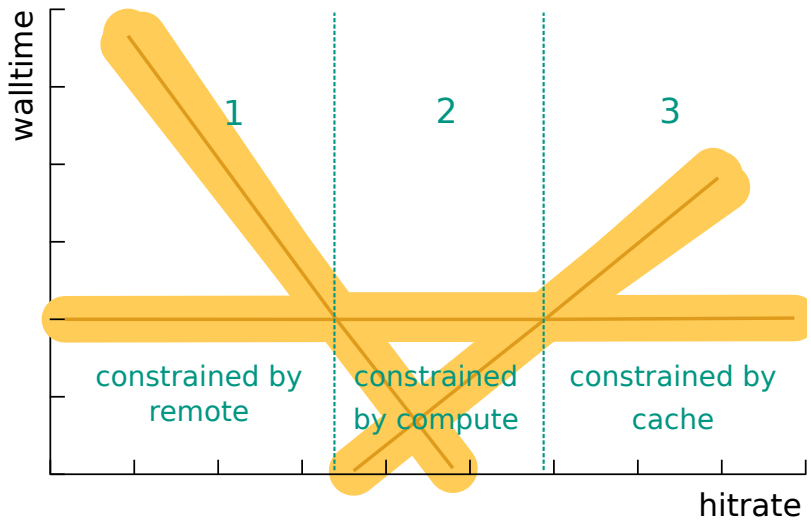
Scenario 2: 1G Remote

- Lower gateway machine's network interface to 1 Gbit s^{-1}, else the same

→ Reuse condensed information from previous measurement

Measurement





- Limited by I/O via remote network
- Not compatible with "Bathtub" model!

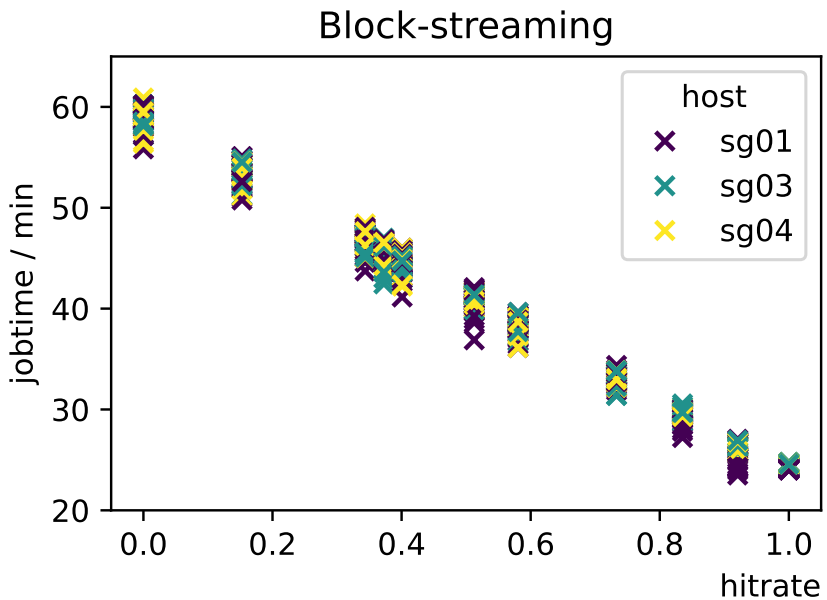
⇒ Why? Synchronized file read actions lead to overhead!

Scenario 2: 1G Remote

- Lower gateway machine's network interface to 1 Gbit s^{-1}, else the same

→ Reuse condensed information from previous measurement

Measurement



- Limited by I/O via remote network

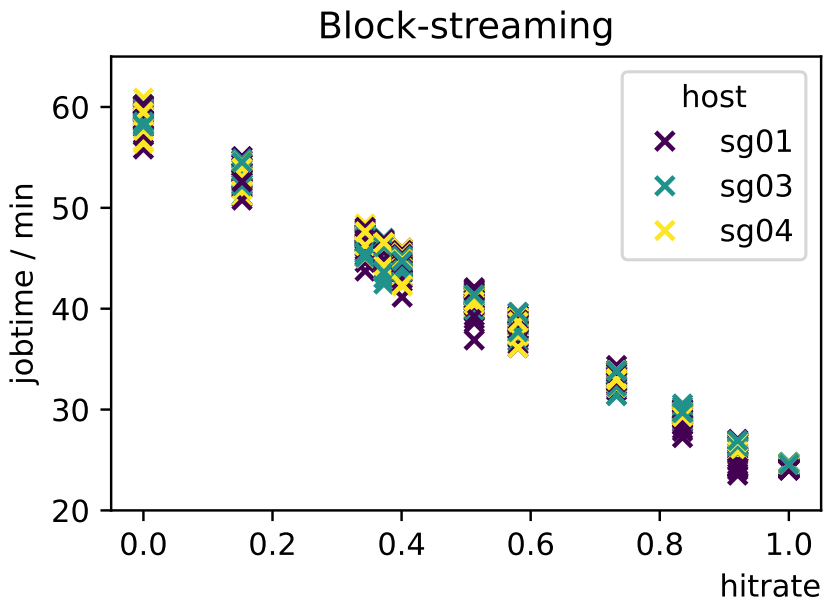
→ Simulate exactly duplicated jobs with synchronized file reads

Scenario 2: 1G Remote

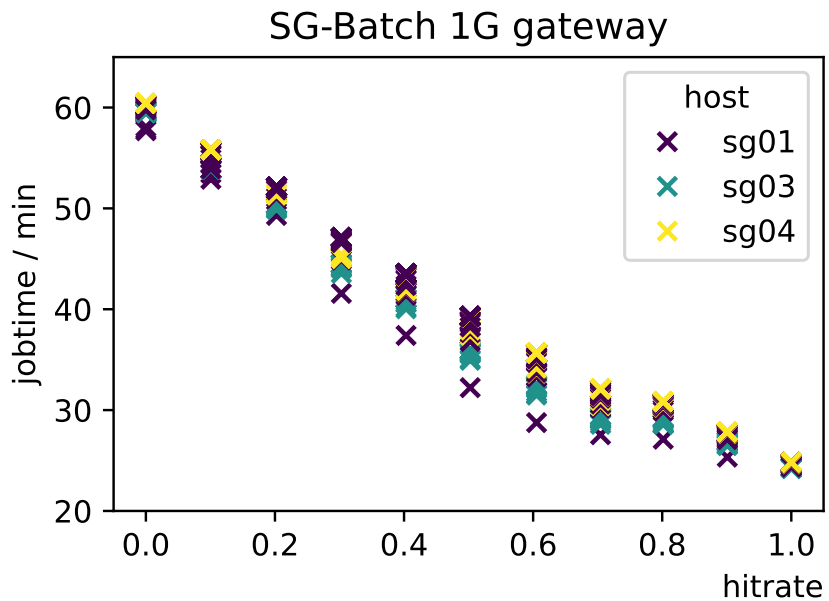
- Lower gateway machine's network interface to 1 Gbit s^{-1}, else the same

→ Reuse condensed information from previous measurement

Measurement



Simulation



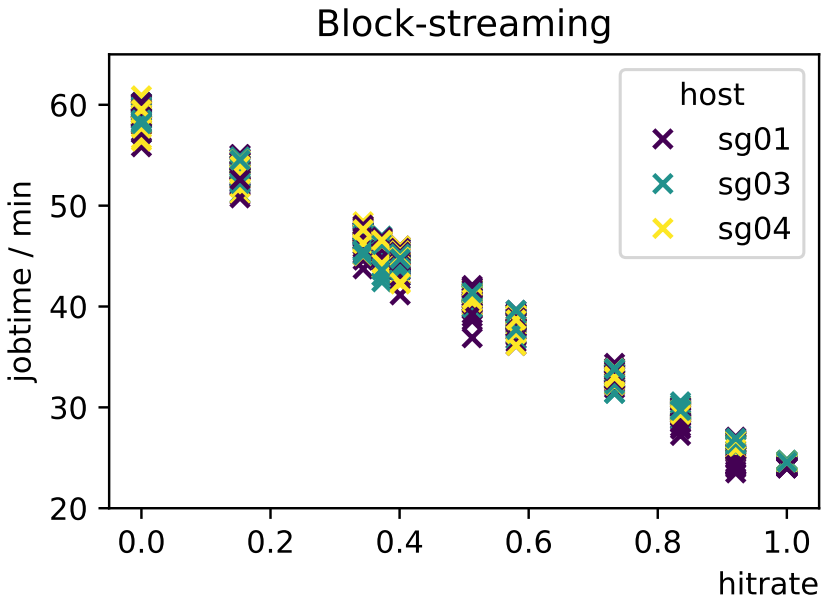
- Limited by I/O via remote network

→ Simulate exactly duplicated jobs with synchronized file reads

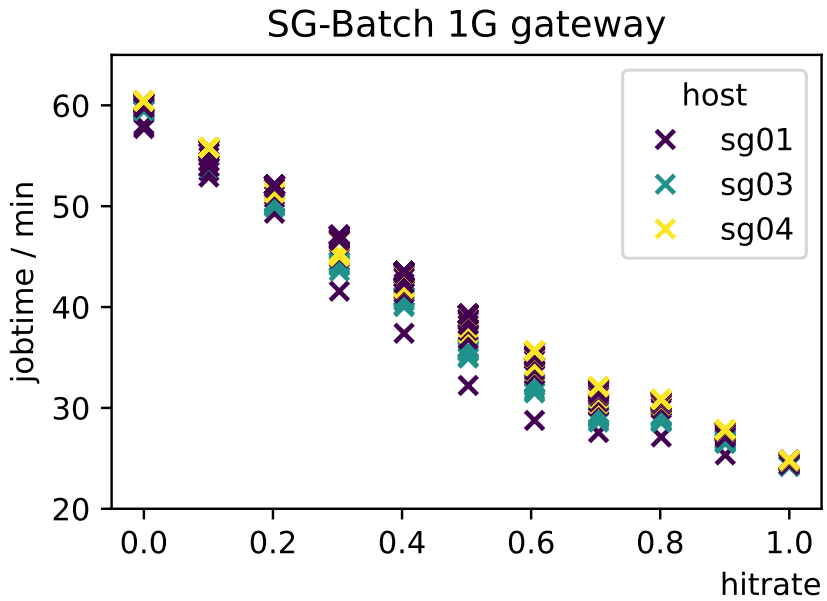
Calibrated Simulation

- Naive real world platform parameter values not necessarily “true”

Measurement



Simulation



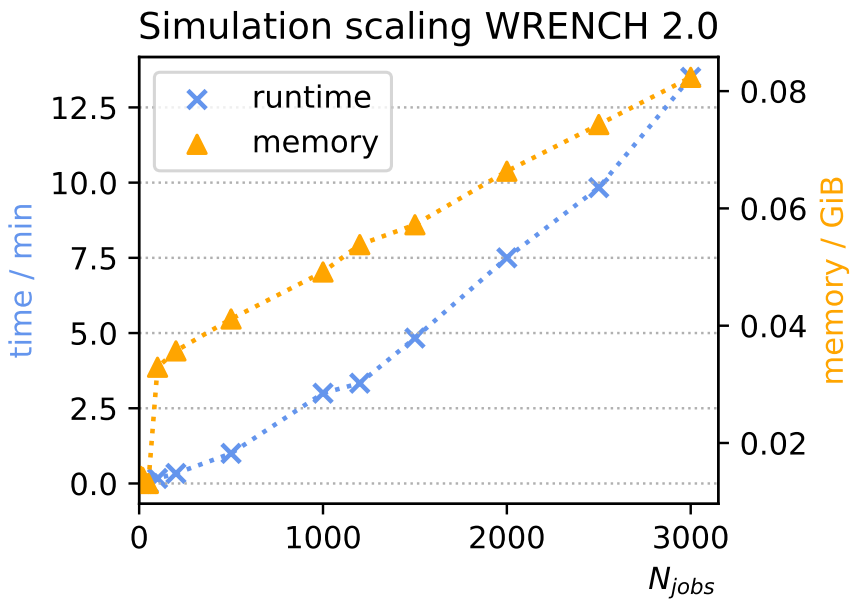
- Elaborate calibration of simulation parameters (fine-tuning involved)

→ Good analogue of a real-world computing system via simulation

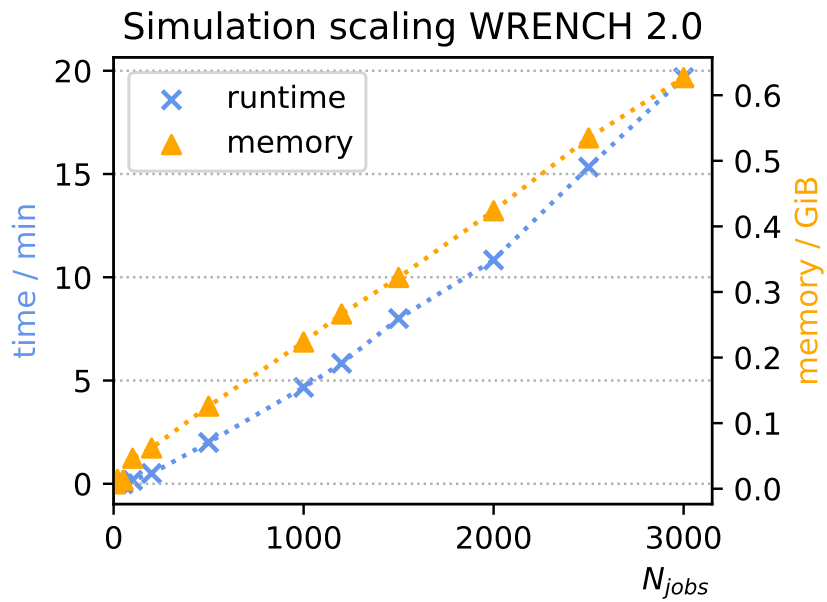
- The simulator is able to reproduce reality!

Memory and Runtime-Scaling WRENCH 2.0

- Copy-jobs



- Streaming-jobs with 0.1 GB blocks

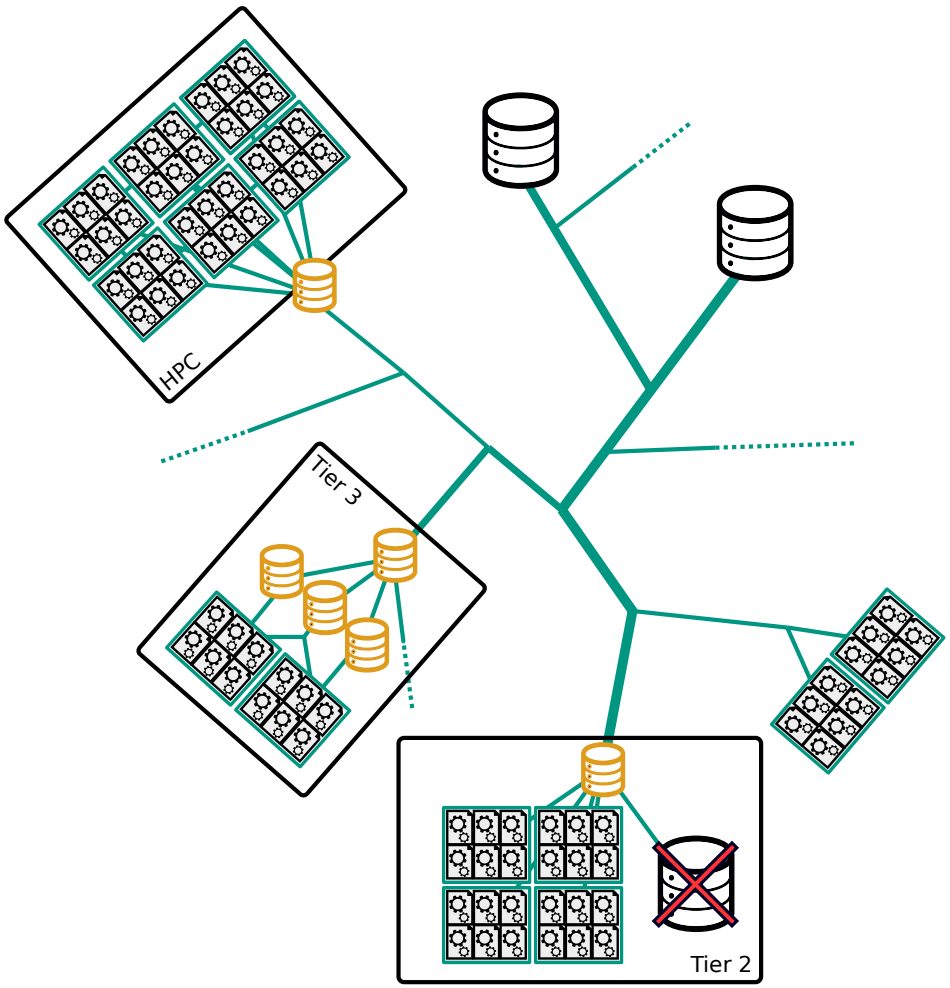


- Streaming increases simulation resource consumption and runtime

→ But $\mathcal{O}(100)$k jobs not beyond reach ($\Leftrightarrow$ 20 GB memory, 15 h runtime)

- Further optimizations ongoing in particular for memory

Ongoing and Planned



- Comparison of large scale platforms with(-out) caches as proof-of-concept
- Detailed XRootD simulation (In progress, **Hawaii**) and comparison with my naive streaming and source-identification implementations
- Automatic simulator calibration methods, e.g. NN (In progress, **Hawaii**)