Towards a Caching Infrastructure for HEP-Germany

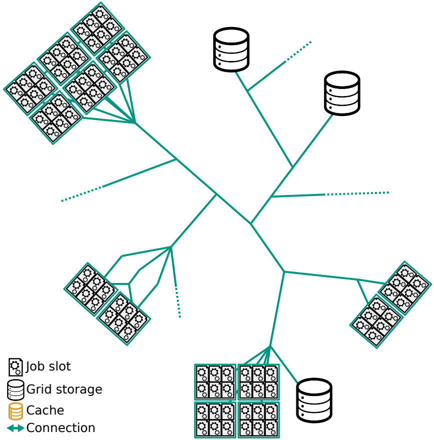
ErUM Data IDT Workshop 2022

Maximilian M. Horzela on behalf of the KIT-HEP-Computing team | 14. Februar 2022

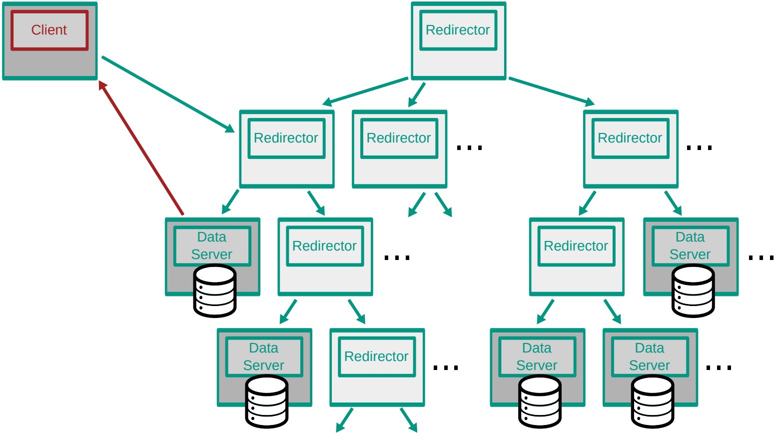
Distributed Storage and Compute Infrastructure in HEP

- 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 / . . .



Any Data, Anytime, Anywhere: XRootD

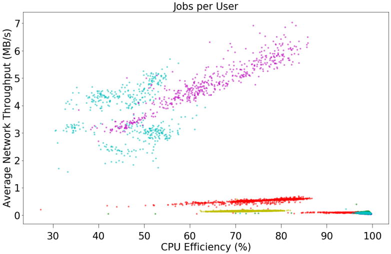


- XRootD is an established technology in HEP
- On client side, data can be accessed independent of location

Use Available Resources More Efficient

Karlsruhe Institute of Technology

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

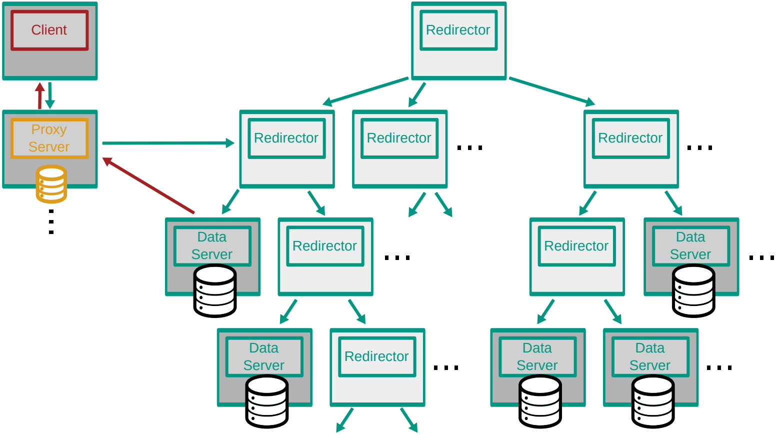


M. Schnepf

- 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

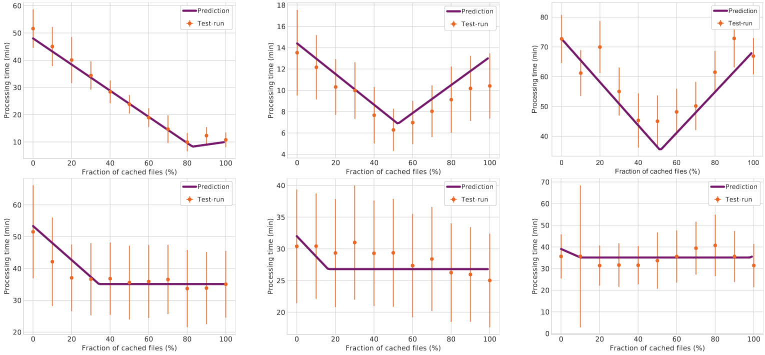
Any Data, Anytime, Anywhere: XRootD



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

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

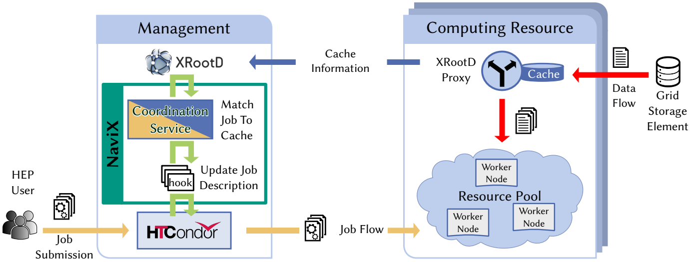
Ideas for an Efficient CDN for HEP

Karlsruhe Institute of Technology

- 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?

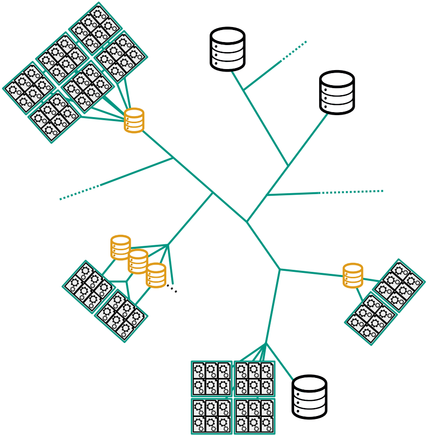
- Latest technology for **cache-aware scheduling**: NaviX



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

A Content Delivery Network for HEP

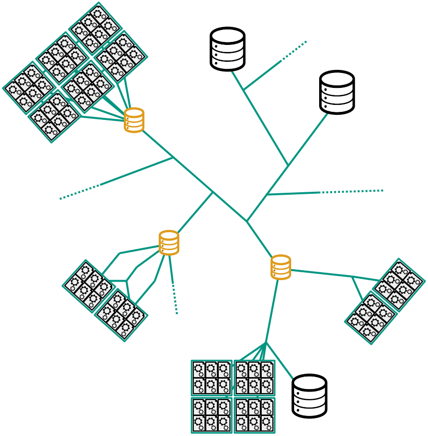
Where, how (large) and when do we place caches?



- In closest vicinity to processing sites → minimize WAN load

A Content Delivery Network for HEP

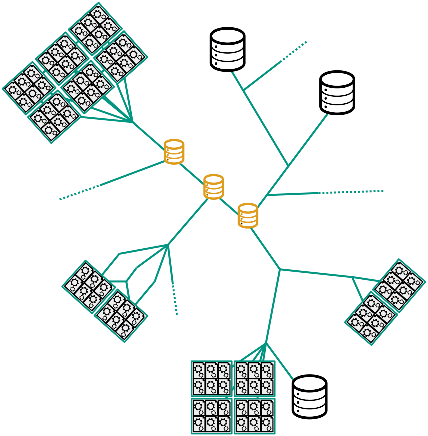
Where, how (large) and when do we place caches?



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

A Content Delivery Network for HEP

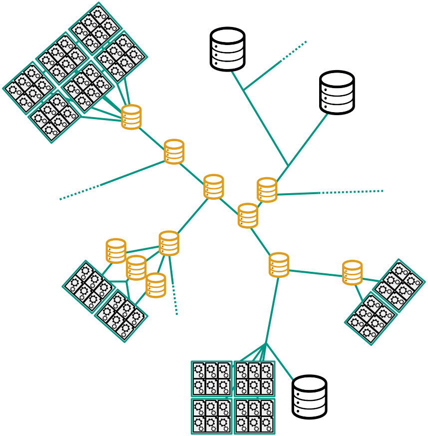
Where, how (large) and when do we place caches?



- Inside network to maximize amount of profiting sites

A Content Delivery Network for HEP

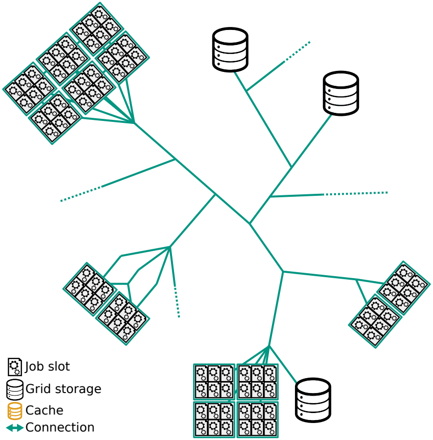
Where, how (large) and when do we place caches?



- In closest vicinity to processing sites → minimize WAN load
- Deeper inside network → 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

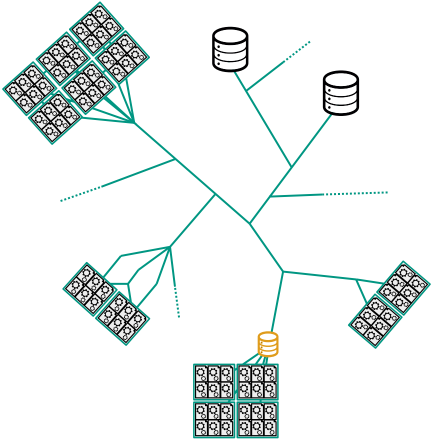
Can we even **replace managed storage** with caches . . . ?



- Replace an expensive managed storage . . .

A Content Delivery Network for HEP

. . . or just one cache?



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

Asking Questions Involving Complex Structures

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

→ Strongly coupled, dynamical multiscale system with many parameters

- Real world system is complex at big scales

→ Unclear how to use caches and decide on execution of many present ideas

→ What are efficient realizations?

How can we study these systems?

Modelling of Distributed Systems with Caches

- **Option 1**: Performance measurements of test-beds not feasible (time and monetary costs) for complex structures
- **Option 2:** Simulate interesting infrastructures
  - Caveat: Inherent simplifications in the simulation model

  $\rightarrow$ Drop effects of subleading influence

  $\rightarrow$ But keep effects of governing entities
- Similar simulation ansatz was already successful in the past: MONARC I. Legrand, H. Newman

We need simulation tools able to handle distributed infrastructures and $\mathcal{O}(100\text{k})$ jobs!

WRENCH-based Simulation

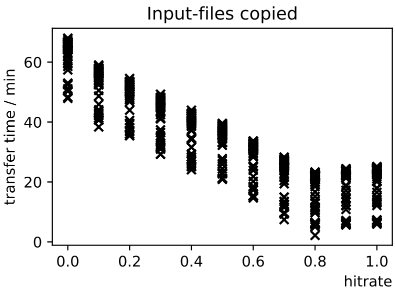
- MONARC(2) is discontinued
- WRENCH:
  - Wide user base
  - Validated accuracy
  - **Very supportive and highly motivated developers**, special thanks to Henri Casanova
  - But: missing HEP specific adaptions

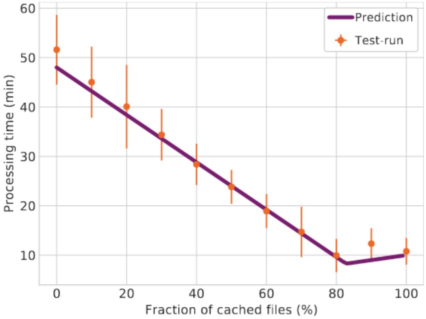
To-do list:

- Implement representations of caching and HEP specific services into the simulator
- Test and validate with simple measurements
- Start to get answers for the most urgent questions

Validate Simple Measurements

- Comparison of WRENCH simulation (left) with measurement (right)

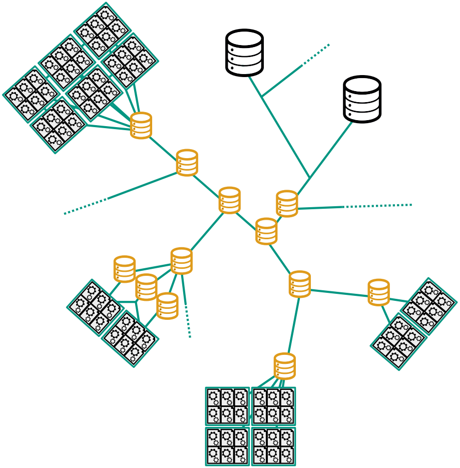




- Minimum due to the “best” combination of network throughput and cache performance

$$h_0 \approx \left(\frac{b_{\text{net}}}{n_{\text{cache}} b_{\text{cache}}} + 1 \right)^{-1} \approx 0.82$$

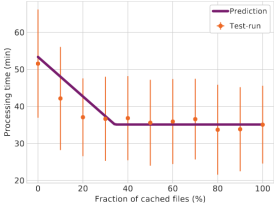
Ongoing and Planned

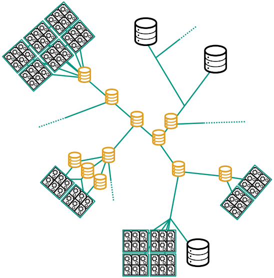


- Support sufficiently complex architectures
- Enable simulation of various steerable workflows
- Implement (HEP-)specific services into simulation, i.e. XRootD

Summary

- Simple caching scenarios work and technology is established
- Caching promising approach to optimize data analysis tasks, many ideas exist
- Finding optimized caching configuration at bigger scales is hard due to complex nature of distributed systems
- Simulation ansatz to identify beneficial setups seems natural
- Design of a simulation suite for HEP computing based on WRENCH has started
- First sanity checks look promising



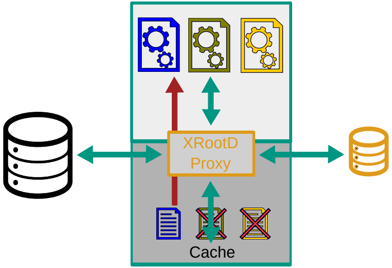


Backup

Caching Tools

XCache

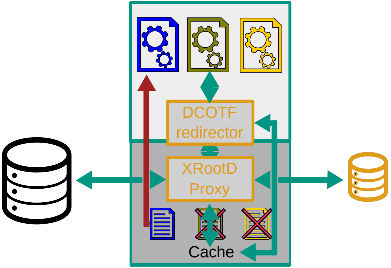
- Plugin for the XRootD proxy file caching



- 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

A Content Delivery Network for HEP

Karlsruhe Institute of Technology

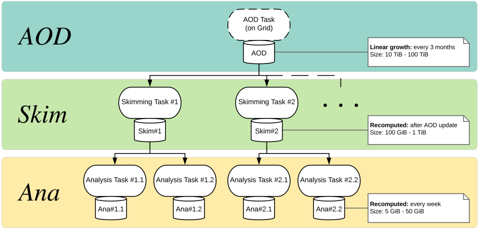
What is a **desirable cache size and caching logic**?

- Least Recently Used: standard in XRootD proxy file caching
- MIN: Minimum based on full knowledge
- RAND: random
- EVABIT: probabilistic algorithm

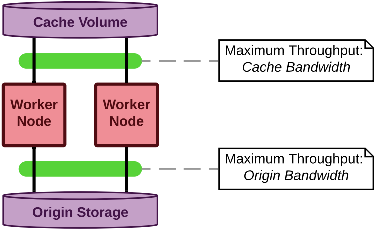
Cache-Logic Simulation Model

- Specific simulation model for a specific question:
  What is the impact of different cache algorithms on an idealized HEP workflow for reactive caching?

Workflow generation



Access generation



P. Skopnik, *Reactive Caching to Accelerate High-Throughput Computing Workloads*

Finding an Optimal Cache

What is a **desirable cache size and caching logic**?

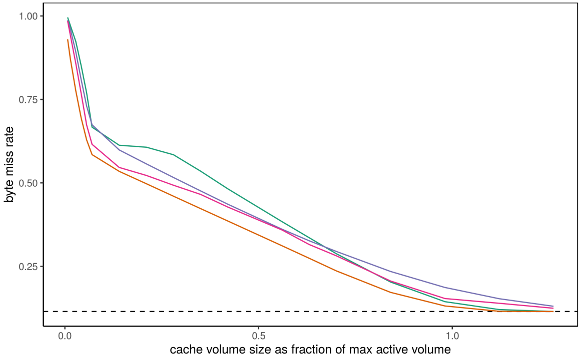
Hitrate/missrate

- Fraction of files read from cache instead of remote storage
- $1 - \text{missrate}$
- High hitrate (low missrate) $\rightarrow$ in many cases higher cache utilization and lower WAN load

- Least Recently Used: standard in XRootD proxy file caching
- MIN: Minimum based on full knowledge
- RAND: random
- EVABIT: probabilistic algorithm

Finding an Optimal Cache

What is a **desirable cache size and caching logic**?

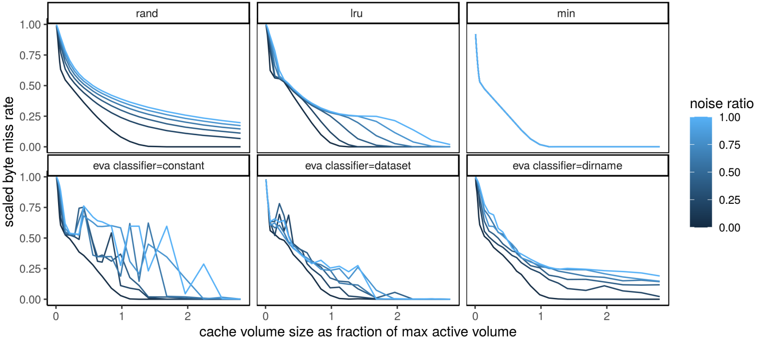


P. Skopnik

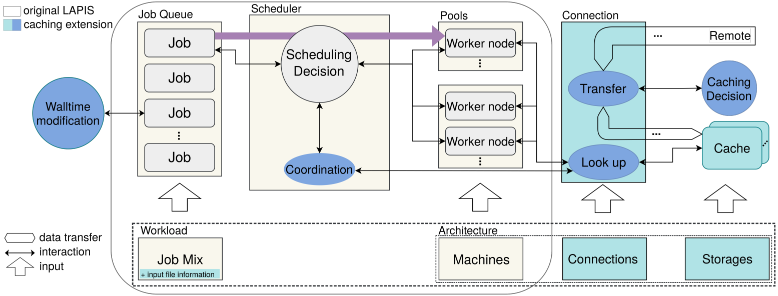
- Least Recently Used: standard in XRootD proxy file caching
- MIN: Minimum based on full knowledge
- RAND: random
- EVABIT: probabilistic algorithm

Cache logic - Noise stability

- Adding single accesses to files belonging to an external dataset at a given rate
- Since single accesses, no benefit from caching



Simulator LAPIS.CACHING



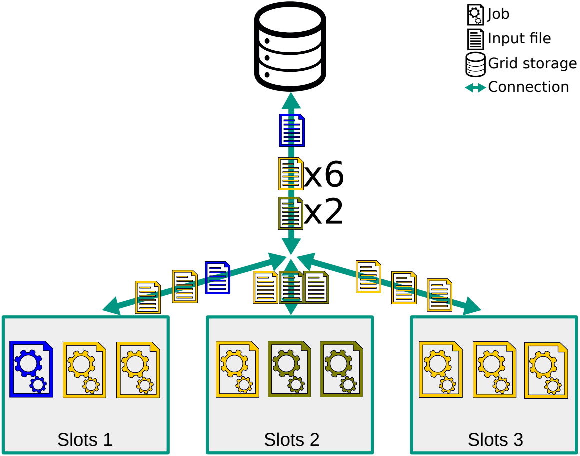
- Walltime modification:

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

A simple example

- Group of nodes connected to a remote storage running several jobs with input data

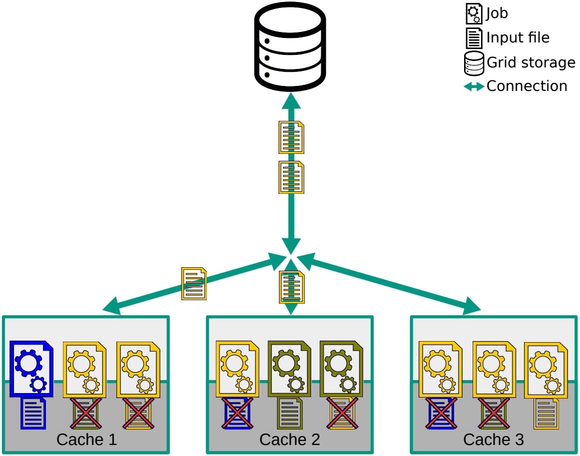
→ Large load on network connection to remote grid storage



A simple example

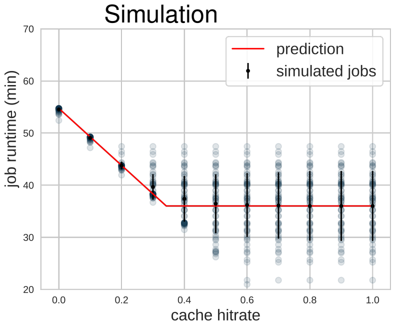
- Placement of one cache on each node

→ Load on network connection to remote grid-storage decreases

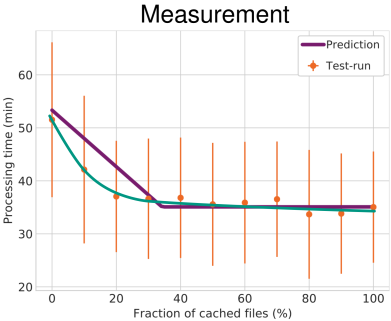


Institute of Experimental Particle Physics, Steinbuch Centre for Computing

Simulating a simple example



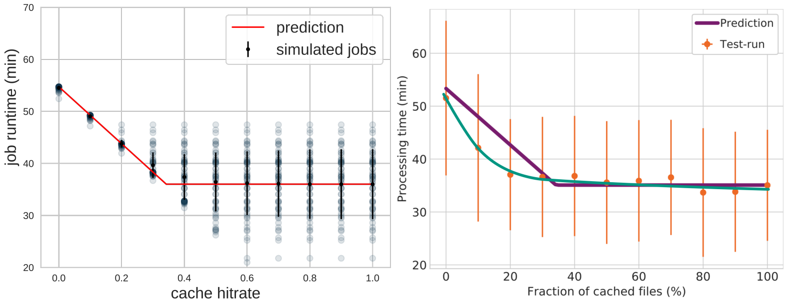
T. Feßenbecker



M. Sauter

- Simulation able to catch general features of real systems
- Encouraging to continue with the simulation ansatz!

Simulation-Data-Comparison



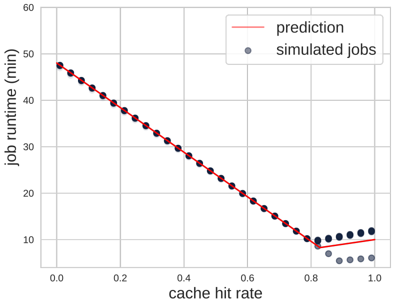
T. Feßenbecker

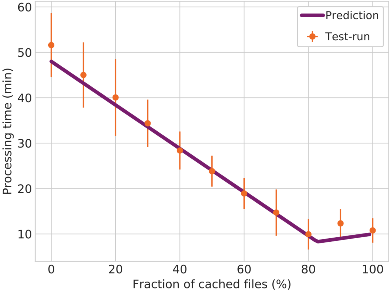
M. Sauter

$$\text{hitrate} = \frac{\text{input-files}_{\text{cache}}}{\text{input-files}_{\text{remote}} + \text{input-files}_{\text{cache}}}$$

- Crucial component *Network* not supported

Simulation-Data-Comparison





T. Feßenbecker

M. Sauter

$$\text{hitrate} = \frac{\text{input-files}_{\text{cache}}}{\text{input-files}_{\text{remote}} + \text{input-files}_{\text{cache}}}$$

- Crucial component *Network* not supported

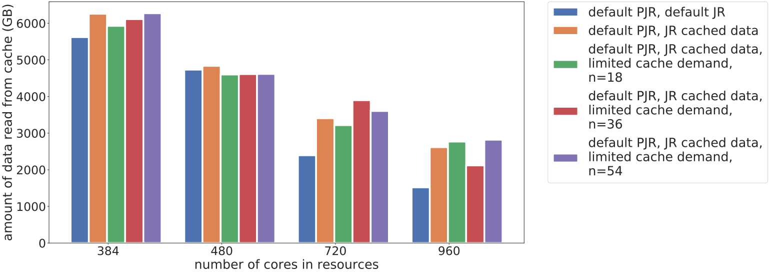
Simulating Coord. Caching

Karlsruhe Institute of Technology

- Tested cache rank scenarios T. Feßenbecker:
  - No coordination: CACHE_RANK$= 0$
  - Hit-rate based: CACHE_RANK$= \sum_{\text{files}} h_{\text{file}} V_{\text{file}}$
  - Hit-rate based modified with throttling-factor: CACHE_RANK$= \sum_{\text{files}} h_{\text{file}} \, V_{\text{file}} \, \left(\frac{1}{\tau} < n\right)$

Study Effect of Coordination

Karlsruhe Institute of Technology

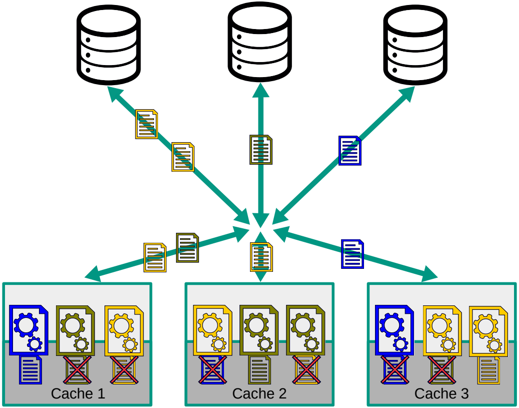


T. Feßenbecker

- Naive assumptions can be confirmed

Study Effect of Coordination

Not answered: What is the effect of coordination on the rest of the network?



SimGrid and Wrench

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 Platform Description

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

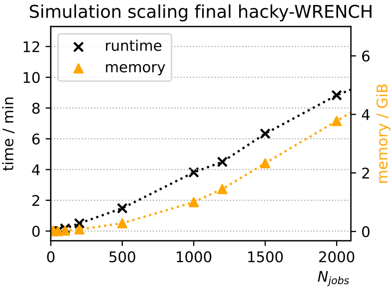
```xml
<platform version="4.1">
    <zone id="AS0" routing="Full">

        <!-- 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

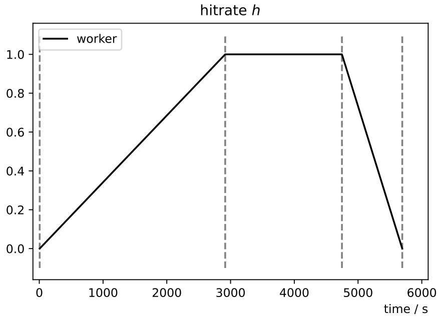
Memory and Runtime-Scaling

- First version of a hacked WRENCH to enable caching simulation

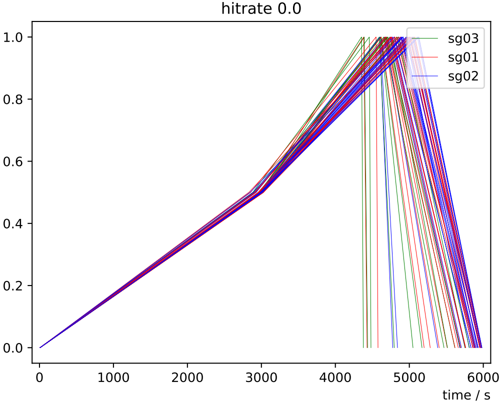


Trapezoid-Plots I

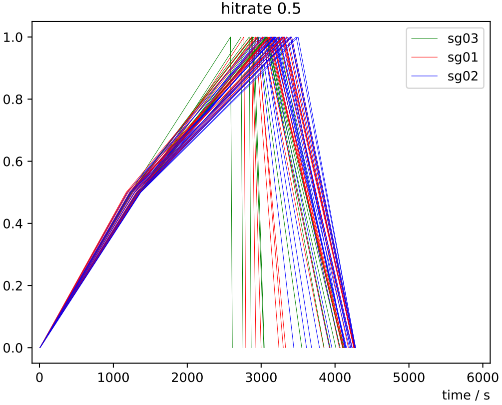
- Trapezoid-plots depict the lifetime of a job
- Segmented in three phases:
  - input-file read
  - workload run
  - output-file write



Trapezoid-Plots II



Trapezoid-Plots III



Trapezoid-Plots IV