Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Federal Ministry
of Education
and Research

Towards benchmarking analysis data flows for caching applications

status report for Topic Area II

FIDIUM Collaboration Meeting | DESY, Hamburg | 20–21 October 2022

Johannes Haller, Johannes Lange, Daniel Savoiu, Hartmut Stadie

Commitments in FIDIUM

- Topic II – **Data lakes, distributed data, caching**
  - investigate and deploy data caching technologies
  - integrate dynamic data caches near newly integrated CPU resources

- Topic III – **Adaptation, testing, optimization**
  - deploy tools developed within FIDIUM to selected computing centers
  - integrate into production environment of HEP experiments
  - optimize to requirements for typical analysis workflows

First goals

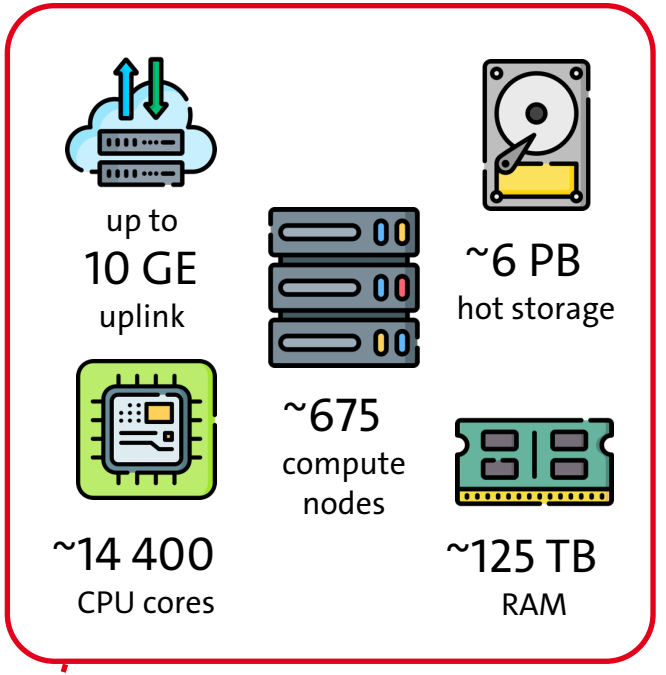
- integration of ***PHYSnet*** computing cluster at Uni Hamburg
  - (see report for Topic Area III)

- benchmark performance of resources to be integrated
  - time data transfers between ***PHYSnet*** faculty cluster and grid storage sites
  - compare with transfer rates to/from ***NAF*** as a reference compute site

PHYSnet cluster

compute resources shared by all institutes of physics faculty

- heterogeneous, multiple pools/queues for diverse applications:
  - ***idefix.q*** – mixed single-threaded applications
  - ***infinix.q*** – for multi-node applications using MPI + InfiniBand
  - ***obelix.q***, ***epyx.q*** – for large-memory applications
  - ***graphix.q*** – for GPU applications
- parts reserved for exclusive use by various project groups
  - high flexibility for tailoring to individual/group use-cases
- adaptable to HEP workflows using ***containerization*** technologies



[Icons: flaticon.com]

	PHYSnet	Typical WLCG sites / *NAF*
OS	***Ubuntu***	***RedHat***-based (SLC/CentOS)
Batch system	***SGE***	***HTCondor***

*(transition to **SLURM** planned for early 2023)*

HEP workflows at *PHYSnet*

provision necessary software using containers and CVMFS

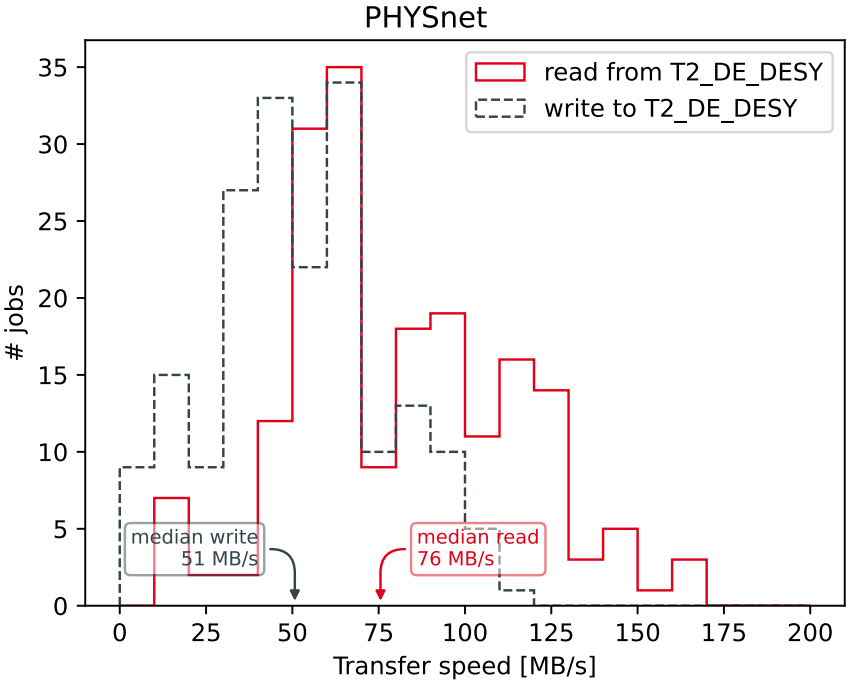
- ***Singularity*** container derived from official CERN ***Docker*** image (`cern/cc7-base`)
  - ***gfal2*** libraries installed for grid access
  - grid authentication handled via ***X.509*** user proxy
- provision ***CernVM-File System*** (CVMFS) using ***cvmfsexec***
  - scalable distributed file system designed for software distribution for HEP experiments
  - normally requires superuser privileges, with ***cvmfsexec*** it can be mounted in userspace
  - made accessible inside job containers using **bind-mounts**
- functional setup able to interact with grid storage elements and transfer files
  - sufficient for first evaluation of raw transfer rates
  - *next step*: run experiment-specific workflows

Grid transfer benchmarks

first measurements of transfer speeds between ***PHYSnet*** and grid storage

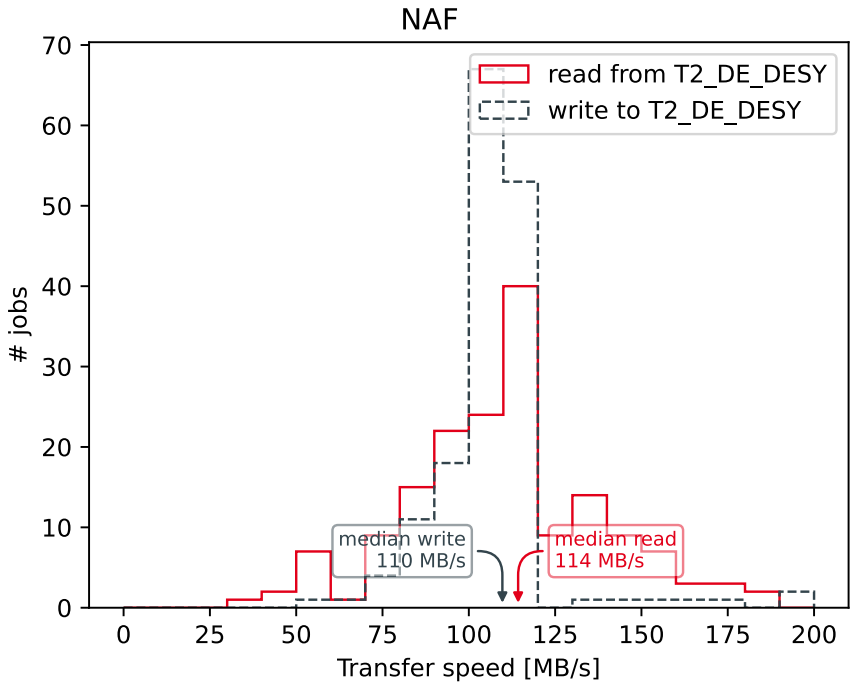
- schedule 200 jobs with simple payload:
  copy a test file (~800 MB) to local storage
  and then write the same file back to grid
- transfer via SRM protocol
- jobs are run sequentially (no bandwidth sharing)

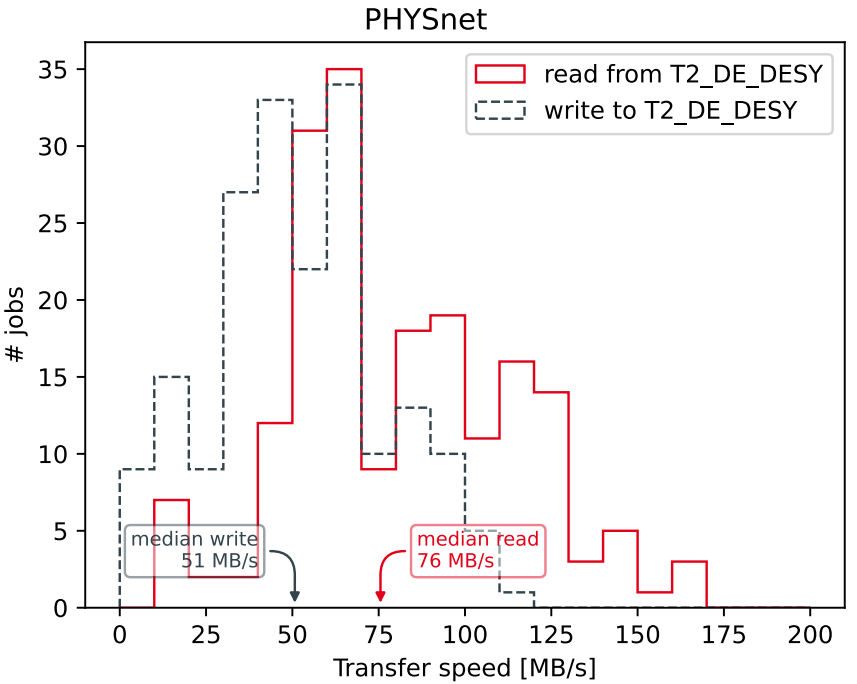
- results for dCache storage @ DESY (`T2_DE_DESY`):
  - median **read** (**write**) speed of **76** MB/s (**51** MB/s)



Grid transfer benchmarks

- identical measurement at ***NAF*** (best link to T2_DE_DESY)

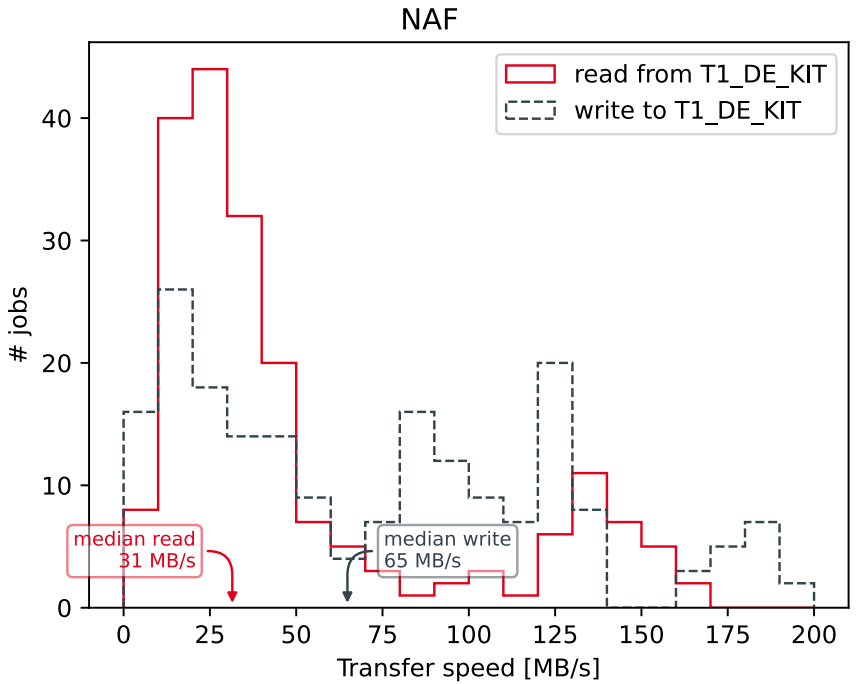


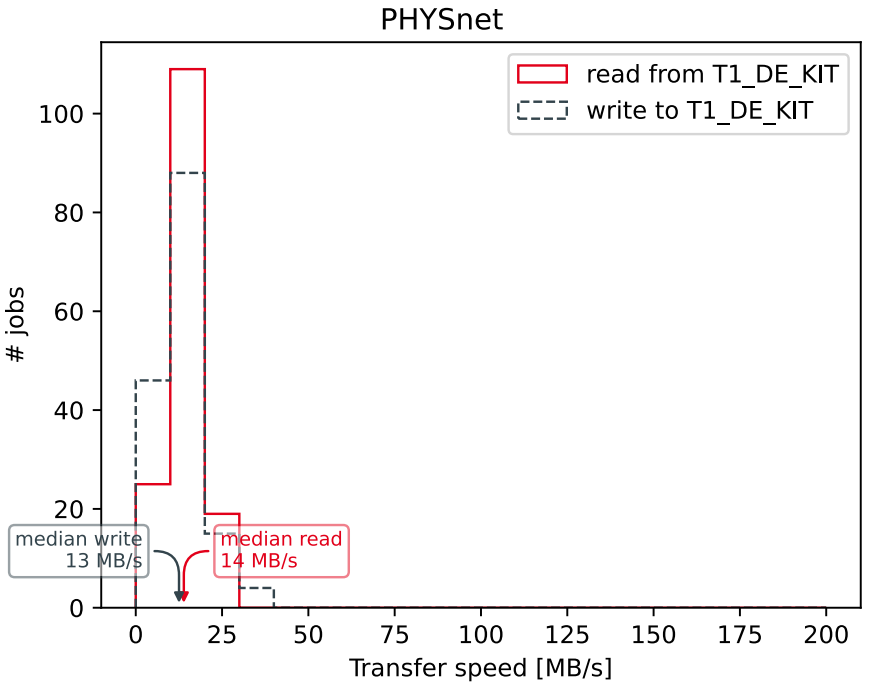


- benchmarks to serve as baseline for comparing to full analysis workflows and evaluating optimizations through caching

Grid transfer benchmarks

- long-distance transfer (T1_DE_KIT instead of T2_DE_DESY)





- caching particularly beneficial for increasing I/O performance with distant data

Summary

- developed containerized setup for running HEP-specific software on ***PHYSnet*** cluster
  - successfully provisioned grid access utilities and CVMFS using ***Singularity*** and ***cvmfsexec***

- benchmarked transfer speeds to/from grid storage at *DESY*
  - measurements establish a baseline for evaluating performance of caching technologies

grid storage
T2_DE_DESY

	median **Read**	median **Write**
NAF *@ DESY*	114 MB/s	110 MB/s
PHYSnet *@ UHH*	76 MB/s	51 MB/s

Next steps

- run experiment-specific workflows
- deploy, test (and adapt) available caching technologies