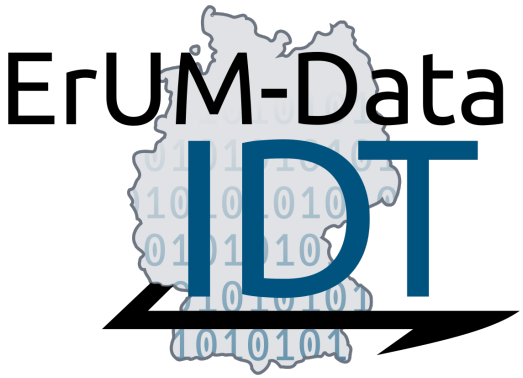
How to bring HTC data to HPC resources - A caching solution for the ATLAS computing environment in Freiburg

Michael Böhler, Anton J. Gamel, Dirk Sammel, Markus Schumacher

24.11.2021
14th Annual Meeting of the Helmholtz Alliance
"Physics at the Terascale", DESY Hamburg (virtual)

Albert-Ludwigs-Universität Freiburg

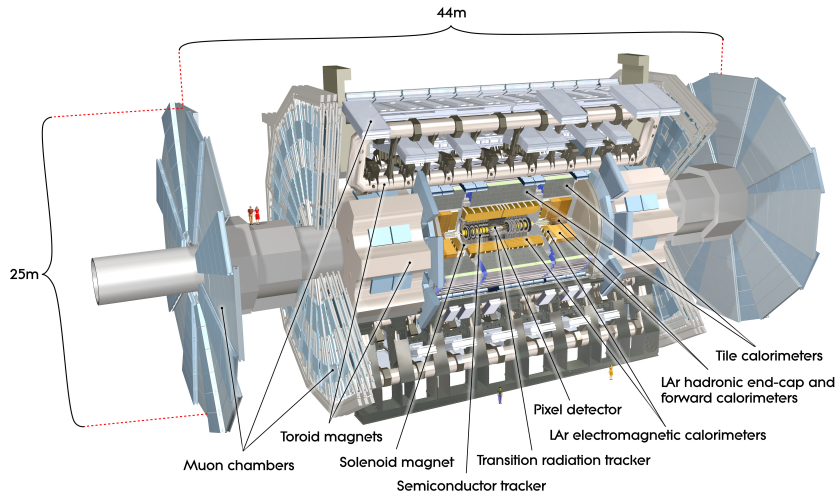
UNI FREIBURG

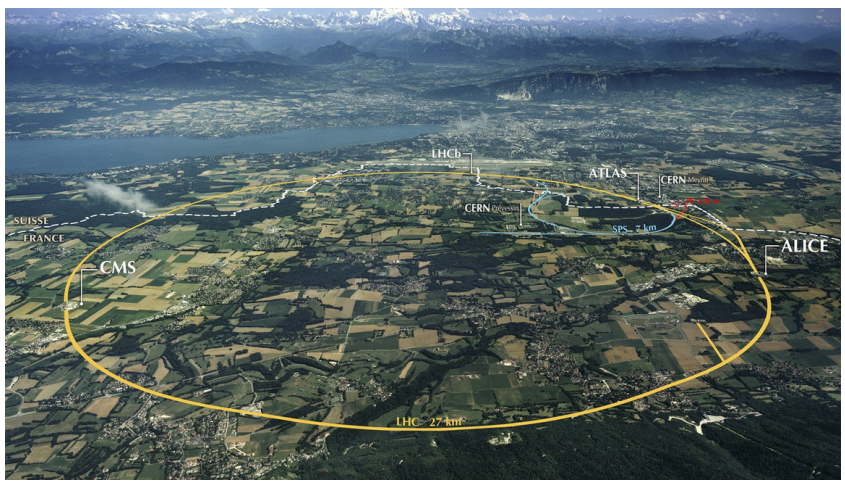


Bundesministerium
für Bildung
und Forschung

Introduction

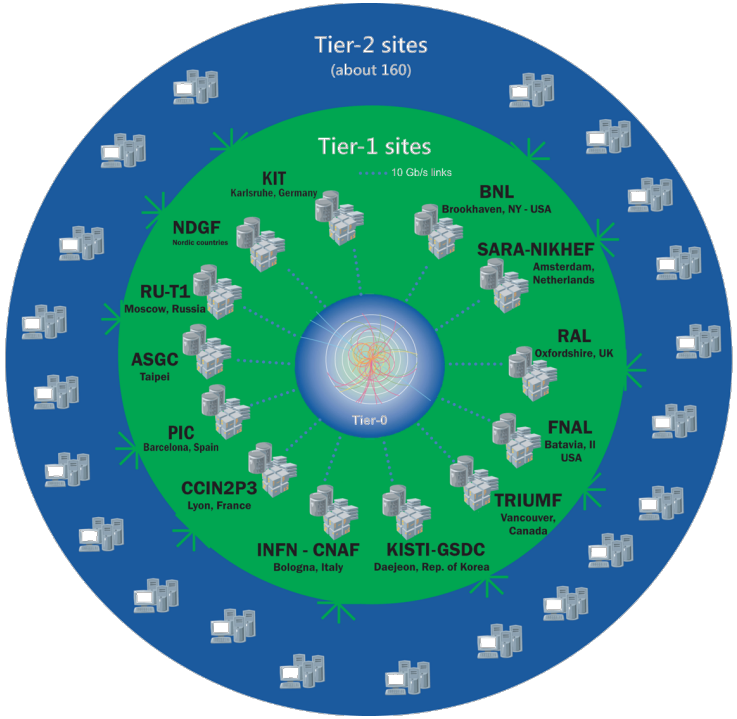
- ATLAS experiment at the LHC
- Initial data rate: $\sim 60\,\mathrm{TB/s}$!
  $\rightarrow$ trigger system
- Data written to disk: $\sim 300\,\mathrm{MB/s}$
- Data distributed & analyzed via Worldwide LHC Computing Grid (WLCG)

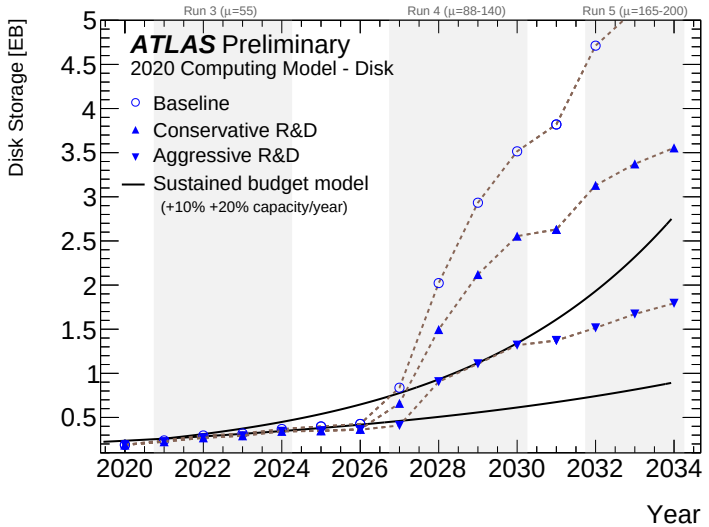




Introduction

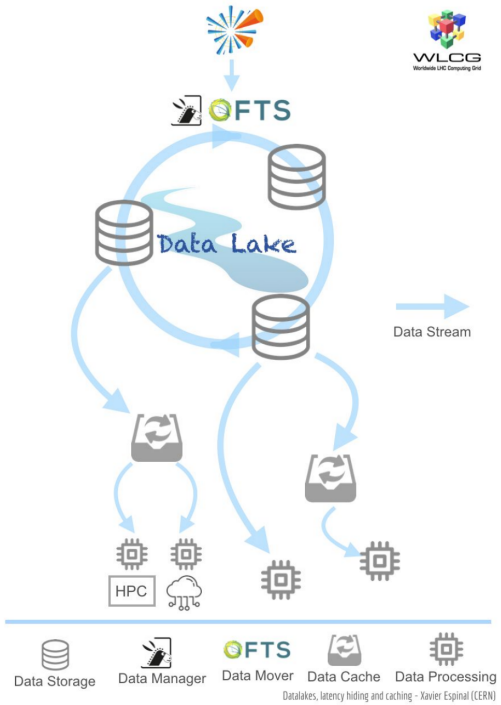
- WLCG: three different levels
  - Tier-0: CERN data centre
  - Tier-1: 13 computer centres with large storage
  - Tier-2: $\sim$ 160 smaller storage sites
- Tier-2 site in Freiburg: ATLAS-BFG
  - 2.5 PB of disk storage
  - Local users of 4 research groups
- Total ATLAS disk storage
  - $\sim$ 235 PB
  - After start of High-Luminosity LHC: $\geq$ 2.5 EB in 2030
  - Huge amounts of disk space required





Introduction

- ATLAS plans
  - Centralize data storage
  - Huge storage centers with long-term storage
  - Compute sites without local long-term storage
- How get compute sites the data?
  → implement a caching setup



- What is caching?
  → local, temporary storage of data
- Why caching?
  - Local: fast access for subsequent requests
  - Temporary: limited resources for long-term storage, automatic deletion if not accessed again

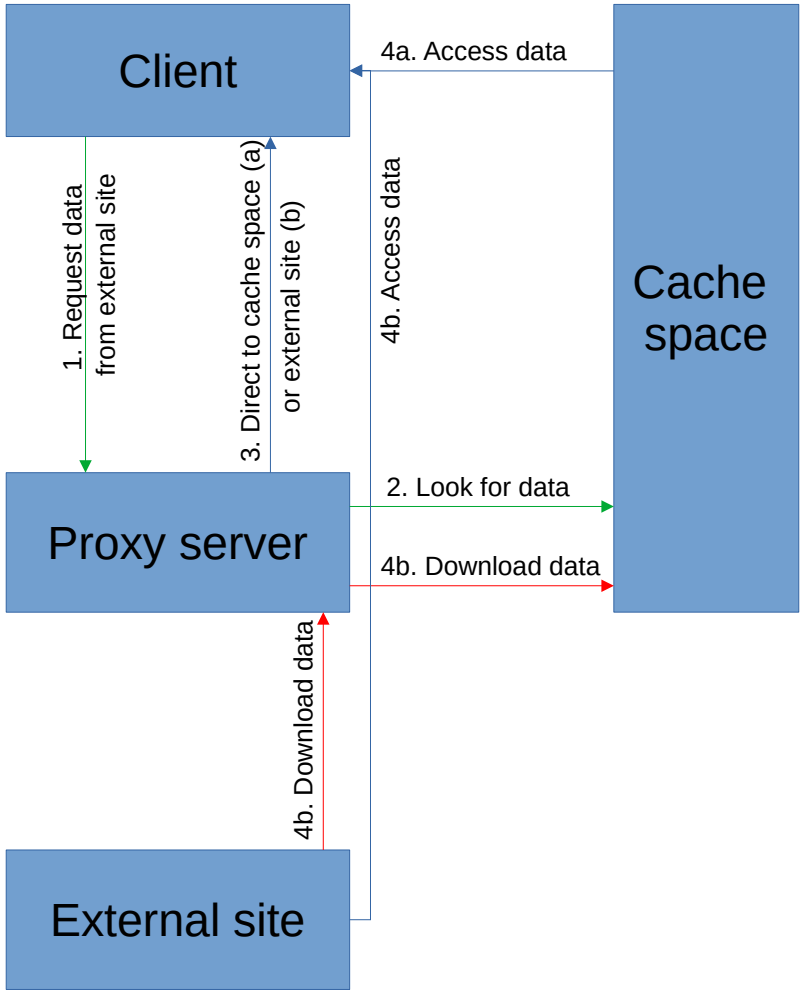
Caching in Freiburg

- Caching setup in Freiburg with XRootD
- Starting point: sandbox implementation by group of Dr. Kilian Schwarz at Uni. Frankfurt/GSI

- Ingredients:
  - Client
    - Virtual machine on node of NEMO HPC cluster
    - Default client configuration
    - Non-evasive setup of caching via environment variables
    - Enable and configure XrdClProxyPlugin
  - Proxy server
    - Virtual machine independent from NEMO cluster
    - XRootD server daemon configured for disk caching and as "forward proxy"
  - Cache space
    - beeGFS file system in Freiburg

Workflow of caching setup

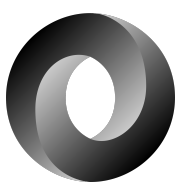
- Client:
  - Requests data from external site
  - XrdClProxyPlugin: redirects request to proxy server
- Proxy server:
  - Checks if data is already in cache
  - Yes:
    Forwards client to location in cache
  - No:
    Forwards client to external site and downloads data to cache



Benchmarking

- Python script with ROOT module
- 'Pseudo analysis', implements features of typical user analysis:
  - Open file
  - Retrieve data
  - Loop over events
  - Fill information in histograms
  - Write histograms to disk
- Output: information in JSON format
  $\rightarrow$ visualize with pandas & Matplotlib



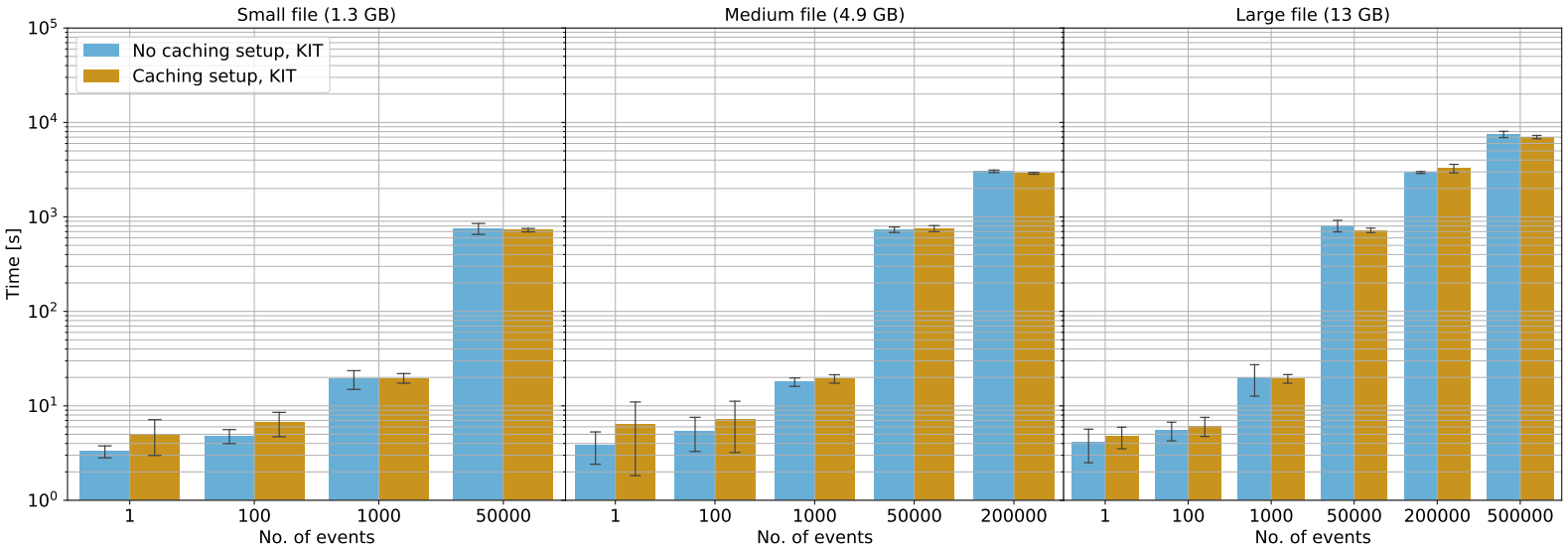



Benchmarking

- Access configurations:
  - Caching without file in cache space
    → caching setup, file read and downloaded from external site
  - Caching with file in cache space
    → caching setup, file read from local beeGFS
  - Local access
    → caching setup not active, file read from local beeGFS
  - External access
    → caching setup not active, file read from external site
- ATLAS sites
  - KIT (Karlsruhe)
  - LRZ (Munich)
  - TRIUMF (Vancouver, Canada)
  - BNL (Brookhaven, USA)
  - DESY (Hamburg)
  - UNI-FREIBURG (Freiburg)
- For each setup: vary file sizes and number of events
  - 1.3 GB, 4.9 GB, 13 GB (s, m, l)
  - 1, 100, 1000, 50k,
    200k (only m & l), 500k (only l)
- Repeat each test multiple times
  → mean value of elapsed time + standard deviation

Results - Overhead

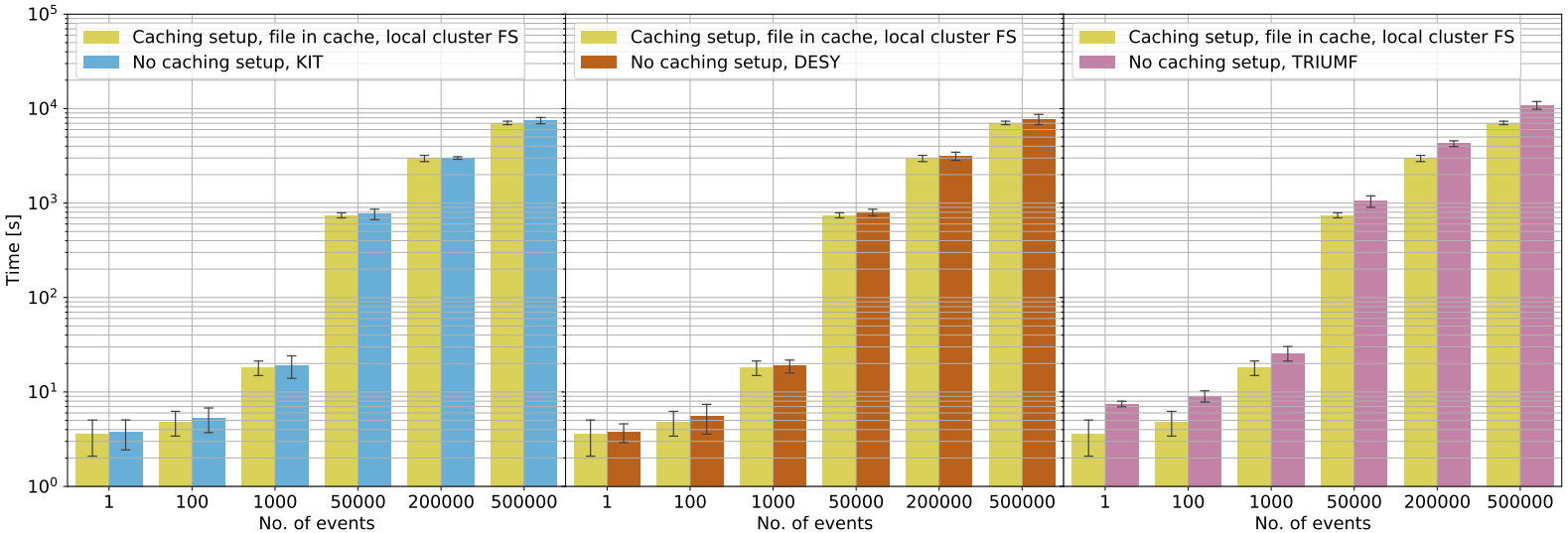
- Direct access to KIT vs. access with caching setup



- Both setups are comparable
  $\rightarrow$ no overhead by caching setup
- Results differ for number of events, but not for file sizes
  $\rightarrow$ merge file sizes in remaining figures

Results - File in cache space

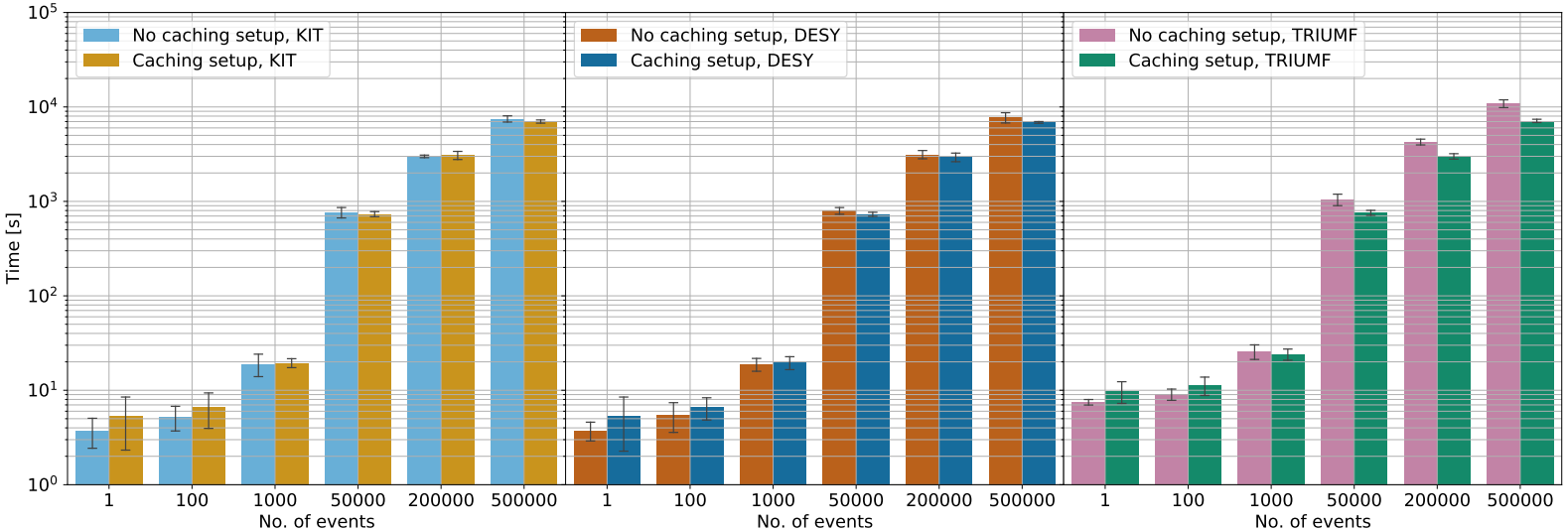
- Direct access to KIT/DESY/TRIUMF vs. file available in cache space



- Near sites (KIT/DESY): results are comparable
- Distant sites (TRIUMF): access to cached file faster
  $\rightarrow$ client profits from caching setup

Results - Initial access

- Direct access to KIT/DESY/TRIUMF vs. access with caching setup



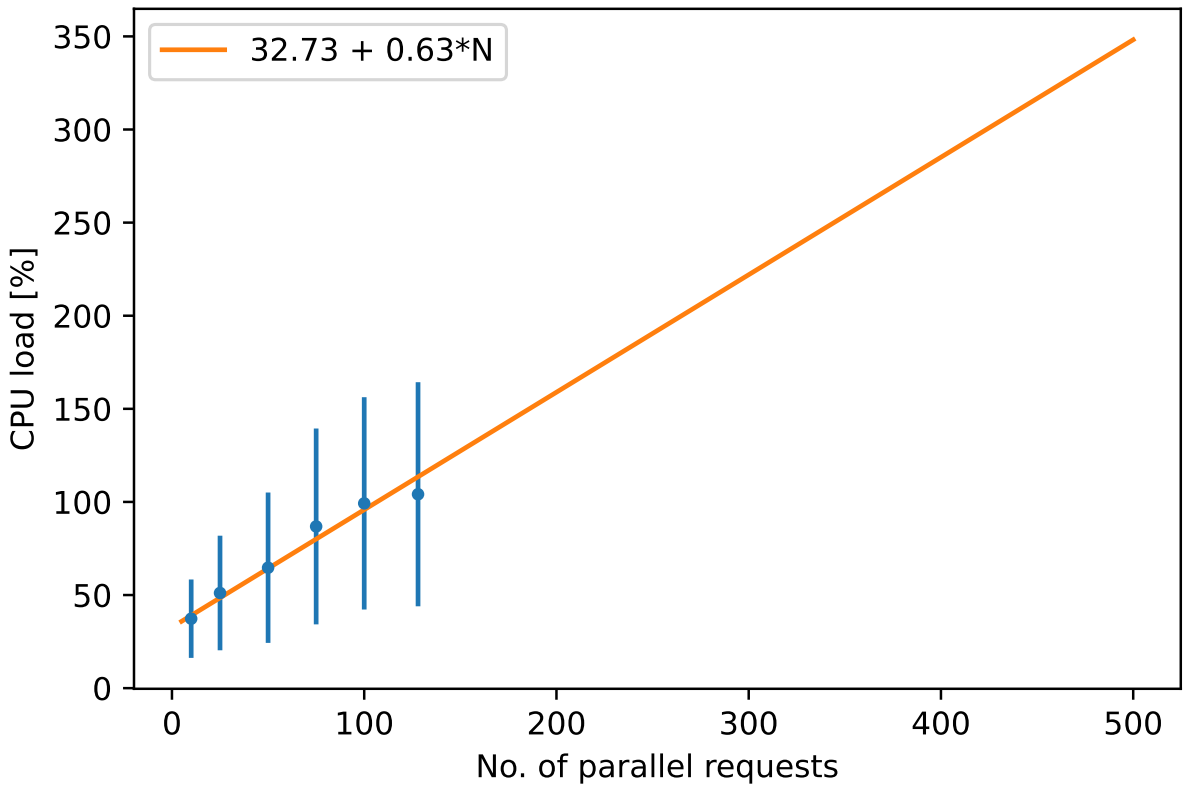
- Near sites (KIT/DESY): results are comparable
- Distant sites (TRIUMF):
  access with caching setup is faster for large numbers of events
  - Download of file via proxy server faster than benchmark runtime
  - After download: client reads file in cache $\rightarrow$ faster access
- Without caching setup: file access completely from external site

Parallel requests

- All results so far: only single file request on proxy server at once
  - Real world scenario: ∼500-1000 jobs at once
- How does the resource consumption scale for parallel file requests?
- Specs of proxy server:
  VM on machine with Intel(R) Xeon(R) CPU E5-2640 v3 @ 2.60 GHz
  - 4 vcores
  - 8 GB RAM
- Tested number of parallel requests:
  - 10, 25, 50, 75, 100, 128 ← client with 128 CPU threads
- 128 copies of large file at KIT
- Measure CPU load and memory consumption at proxy server during benchmark with 10k events

CPU load

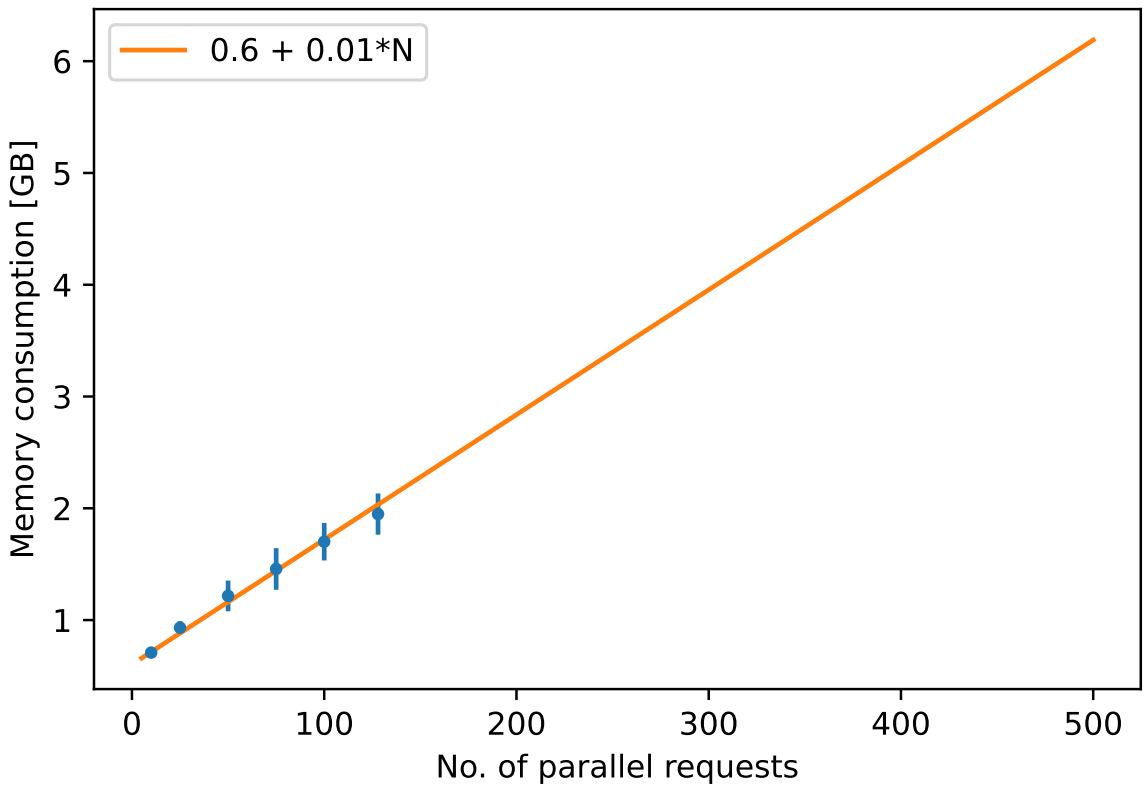
- Subtract CPU load from other processes
- Derive mean and standard deviation
- Simple approximation via linear fit



- 1 core $\equiv$ 100 % $\rightarrow$ in this setup max. 400 %
- Projection for 1k parallel requests: at least $\sim$7 cores required

Memory consumption

- Subtract memory consumption from other processes
- Derive mean and standard deviation
- Simple approximation via linear fit



- Projection for 1k parallel requests: at least ∼11 GB RAM required

Conclusion & Outlook

- Conclusion:
  - Successfull implementation of caching setup for NEMO HPC cluster
    - Clients: no additional installations, setup via environment variables
    - Proxy server: setup on VM, independent from NEMO cluster
  - Benchmark results
    - Caching setup causes no overhead
    - Access to already cached files is faster than access to distant external sites
    - Initial access to files on distant external sites is faster with caching setup
  - Caching is a good alternative to locally managed storage!
  - Estimated requirements for 1k parallel file requests at least
    - 7 cores
    - 11 GB RAM
    - Better: arrange several proxy servers as proxy cluster with one redirector

- Outlook:
  - Perform full user analysis with caching setup

Conclusion & Outlook

- Conclusion:
  - Successfull implementation of caching setup for NEMO HPC cluster
    - Clients: no additional installations, setup via environment variables
    - Proxy server: setup on VM, independent from NEMO cluster
  - Benchmark results
    - Caching setup causes no overhead
    - Access to already cached files is faster than access to distant external sites
    - Initial access to files on distant external sites is faster with caching setup
  - Caching is a good alternative to locally managed storage!
  - Estimated requirements for 1k parallel file requests at least
    - 7 cores
    - 11 GB RAM
    - Better: arrange several proxy servers as proxy cluster with one redirector

- Outlook:
  - Perform full user analysis with caching setup

Thanks!