Karlsruher Institut für Technologie

First Deployment of XCache for Workflow and Efficiency Optimizations on Opportunistic HPC Resources in Germany

CHEP 2024 – 19.10.2024 - 25.10.2024

Robin Hofsaess, M. Giffels, A. Gottman, A. Petzold, G. Quast, M. Schnepf, A. Streit | 22. October 2024

Karlsruher Institut für Technologie

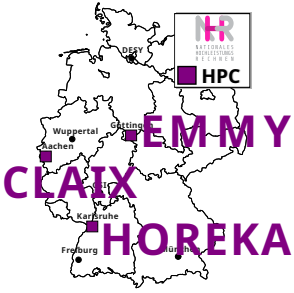
Can we use an HPC share as a WLCG Grid Site?

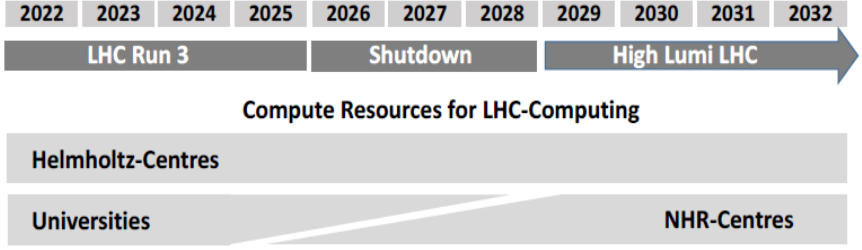
Can we use an HPC share as a WLCG Grid Site?

This means:

- A fixed pledge that needs to be guaranteed
- Reliability that is comparable to classic grid sites
- An efficient operation and utilization of the resource
- What about storage?

Why? The Future of HEP Computing in Germany!





Source: German HEP Computing Strategy paper

- Transition from Tier-2 centers **at universities**[1] to shares on national HPC centers within the **NHR computing compound**
- Storage will be moved to the Helmholtz centers KIT and DESY

[1]NOTE: This does not mean that the T2 groups are closed!

HPC Centers vs. Classic Grid Sites

A Multi-Purpose HPC Cluster

- Typically bigger centers with new, highly **efficient** hardware
- More **sustainable** on large scale
- Interdisciplinary used, HEP community with a rather small share
- Typically strong **performance** focus
- A USER among many, high security standards and few permissions

VS.

A WLCG Tier-2 Center

- Small centers with longer hardware half-life
- Cooling overhead, older hardware, etc
- HEP only, **tailored** to the WLCG requirements (e.g. cvmfs)
- Data **Throughput**-optimized
- **ADMIN** with all permissions, trusted (production) users

Conclusion

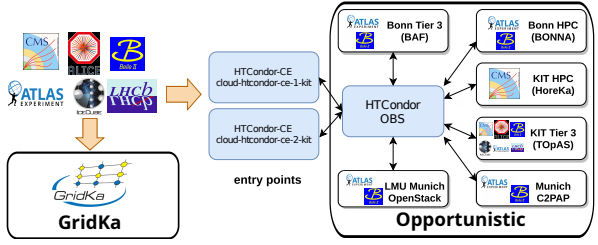
- The compute models differ, multi-purpose HPC centers are **not necessarily suited** for HEP workflows
- The utilization of HPC centers bears many advantages: **sustainability**, **performance**, **collaboration** ...

The (Opportunistic) Utilization of HPC

- In Germany, we dynamically integrate several distributed resources of **different kinds** opportunistically into the German WLCG infrastructure
- For that, we provide a **toolkit**: COBaID/TARDIS
- In production since **several years**!

COBaID/TARDIS

- The software is designed for the dynamic integration of resources according to the demand into an overlay batch system
- More details in backup
- **Interested? Feel free to contact us, we are glad to help!**

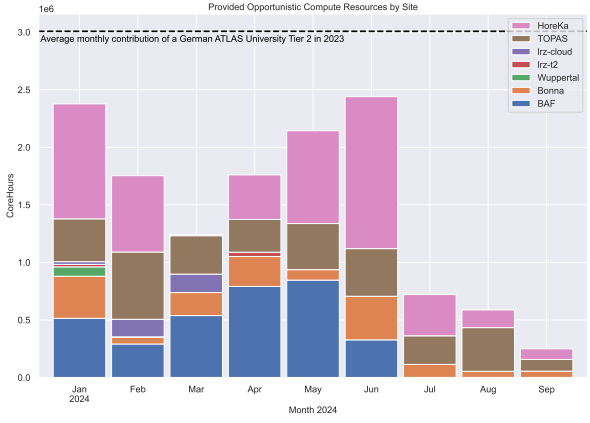


The (Opportunistic) Utilization of HPC

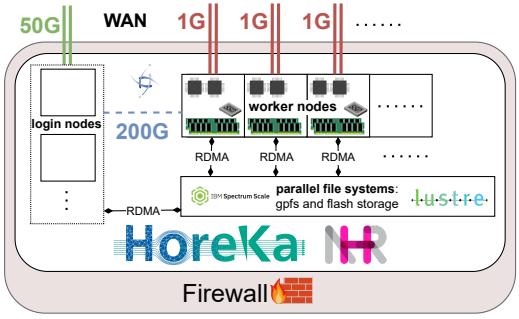
- In Germany, we dynamically integrate several distributed resources of **different kinds** opportunistically into the German WLCG infrastructure
- For that, we provide a **toolkit**: COBaID/TARDIS
- In production since **several years**!

COBaID/TARDIS

- The software is designed for the dynamic integration of resources according to the demand into an overlay batch system
- More details in backup
- **Interested? Feel free to contact us, we are glad to help!**



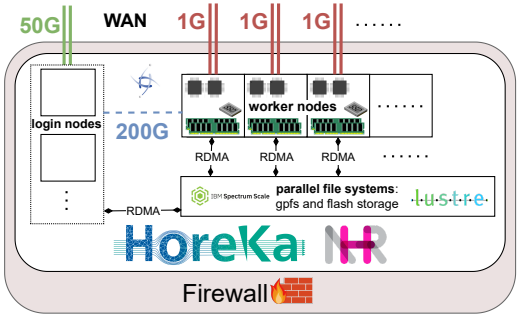
The (Opportunistic) Utilization of HPC: HoreKa



HoreKa

- HoreKa is a **multi-purpose HPC** cluster within the NHR computing compound
- Scientists of German universities can apply
- Integrated opportunistically for **over 3 years**
- We are allowed to **backfill** + a **test project** for CMS has recently been approved

The (Opportunistic) Utilization of HPC: HoreKa



HoreKa

- HoreKa is a **multi-purpose HPC** cluster within the NHR computing compound
- Scientists of German universities can apply
- Integrated opportunistically for **over 3 years**
- We are allowed to **backfill** + a **test project** for CMS has recently been approved

Observations

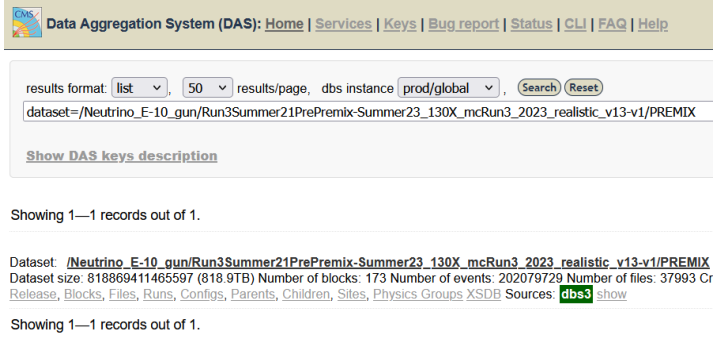
- HPC integration **works in general**, but ...
  - Higher failure rate
  - Overall lower CPU efficiency

  in **comparison** to our **Tier-1** and **Tier-3** centers

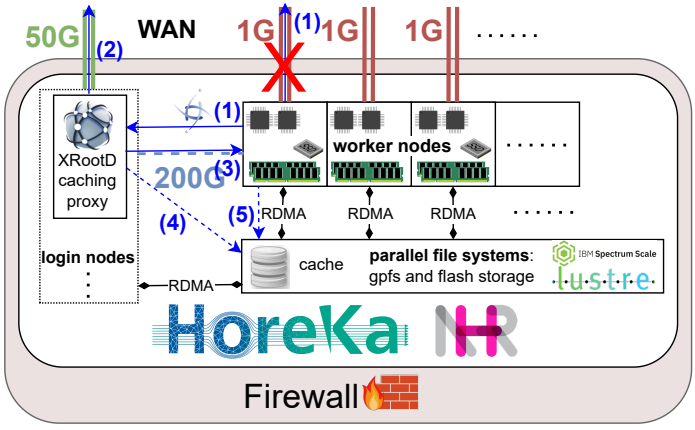
Possible Solution: Pre-fetching the Data?

Could we pre-fetch the data **before** a job is starting?

- Complicated to realize (for CMS), would require changes in the submission infrastructure
- Overall, not very efficient or even impossible with current setup, e.g. in case of MC PREMIX:



XBuffer: Deployment of XCache for Data-Access Bottleneck Mitigation on HPC



Setup

- XRootD proxy file cache running within an apptainer instance on a login node
- The worker nodes are configured to redirect their traffic

Why X*Buffer*?

The focus lies more on the pre-fetching of small amounts of data (*buffering*) than on the caching

LINK: Data access steps described in BU

To cache or not to cache? That is the question!

- Can be very useful with a decent **cache hit rate**:
  - Strongly depends on job mix, data sets, cache size, ...
  - May be useful for **analysis**, but rather not for production
  - From our **experience on HPC**: **bleeding edge** and still **error-prone**

Plans

- We plan to conduct further studies
- Especially **selective caching** might be an option, e.g. only cache **NANOAOD** datasets, certain **sites**, or may only use it for **Analysis** jobs
- Big **THANKS** to the XRootD developers for the support and very fruitful collaboration!

Benefits

Faster Bandwidth

- Accelerated data access
- Pre-fetching (*buffering*) helps utilizing the available bandwidth more balanced
- XCache's caching may be beneficial

Monitoring

- HPC typically does not provide/allow much
- **Adds capabilities**:
  - XRootD monitoring + logs
  - apptainer instance stats
  - Own tools

New Possibilities!

- Feature Request: Support of native RDMA for XRootD
- Our setup may could enable centers without external connectivity on the worker nodes in the future

Benefits

Faster Bandwidth

- Accelerated data access
- Pre-fetching (*buffering*) helps utilizing the available bandwidth more balanced
- XCache's caching may be beneficial

Monitoring

- HPC typically does not provide/allow much
- **Adds capabilities**:
  - XRootD monitoring + logs
  - apptainer instance stats
  - Own tools

New Possibilities!

- Feature Request: Support of native RDMA for XRootD
- Our setup may could enable centers without external connectivity on the worker nodes in the future

Summary

- The setup not only **works well**, but also offers other advantages and benefits in addition to speeding up data access, e.g. for site operation
- Still in testing phase; we are constantly improving the setup in close contact with the XRootD team

Conclusion and Outlook

Conclusion and Outlook

Yes!

Conclusion and Outlook

Yes!

Conclusion

- We run a **fully functioning**, containerized setup on the HoreKa HPC center, despite very **limited permissions**
- Our prototype shows that a **reliable and efficient** integration of HPC is possible
- With the setup:
  - The failure rate dropped to an average level
  - CPU efficiency is now comparable with our Tier-1 and Tier-3!

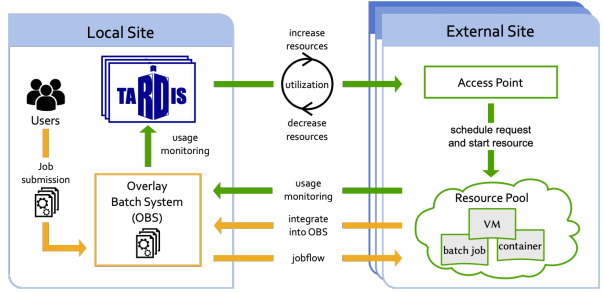
Further Plans

- Further **studies and optimizations** will be carried out in the future, especially focusing on **caching** and **data locality**
- Long term plans:
  - Dedicated **transfer and caching nodes** placed at the attached WLCG site (e.g. the T1 for HoreKa) may enhance the setup
  - However, this needs firewall bypasses et cetera and is therefore very site specific

BACKUP

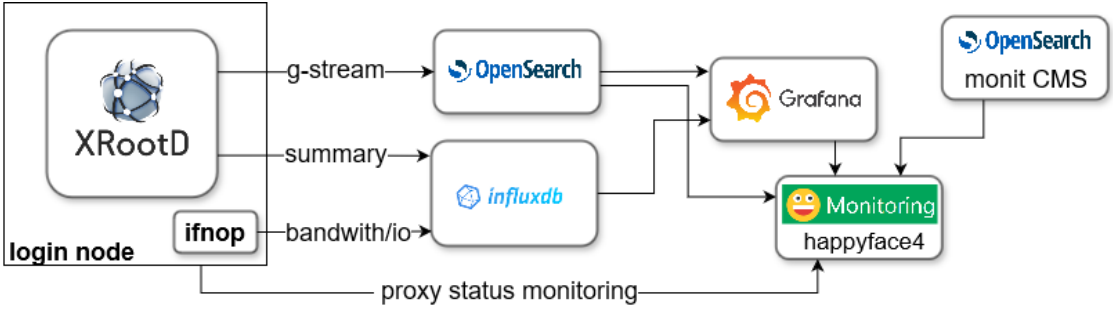
Dynamic Integration of (Opportunistic) Resources

- COBaID/TARDIS is a tool for the dynamic on-demand integration of resources into an HTCondor overlay batch system
- The demand is determined by the allocation and utilization of a resources
- Allowing for feedback-based meta-scheduling
- The tool greatly simplifies the integration of additional resources and is widely used in the German HEP computing



More info: **https://matterminers.github.io/** and **https://github.com/MatterMiners**

Monitoring



- Very comprehensive monitoring stack, helping a lot with site operation and overcoming the limitations of HPC
- All containerized and easily to deploy. On HPC, everything is running in apptainer containers without additional permissions
- Unifying experiment monitoring with our own and site monitoring data in our meta-monitoring **HappyFace4**

Using XCache as a Proxy for Accelerated Data Access

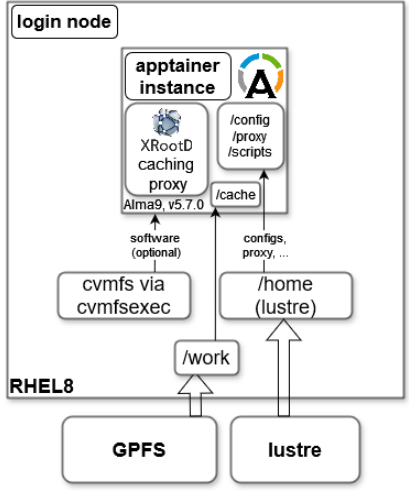
Karlsruher Institut für Technologie

Description of: link

1. Instead of initiating a data transfer over the 1G WAN connection of the worker nodes, the data request is **redirected** to the XRootD (caching) proxy on a login node.
2. The transfer is executed on the login node with the **faster 50G connection** by the XRootD caching proxy. **Pre-fetching** may be enabled to create a *buffering* effect for the data access.
3. Internally, the data is transported via the **200G IPoIB** internal network connecting the login node and the workers.
4. Transferred data can be **cached on-the-fly** on the parallel filesystem, mounted on the login node and the workers as well, if enabled.
5. In case a file is completely cached, a **direct file access (dca)** over the filesystem is possible, accelerating the transfer by a **factor of 10-100** (over 3GB/s)

Implementation Details (I)

- Host: RHEL8
- Config: usernamespaces, CGroups v2, systemd user services
- Currently running: XRootD v5.7.1 as Alma9 apptainer instance (image bootstrapped from docker)
- In principle, up to 76c and 500GB RAM, 50G WAN
  - But shared with other users (limitation via apptainer instance with CGv2 possible)
  - Current usage: 32t, 64GB memory
- 250T quota on gpfs (via IB)



Implementation Details (II)

- To enable the proxy for transfers, we currently do not use the *XrdClProxyPlugin*
- Instead, the proxy is added to the siteconf directly
- Advantages:
  - We can use the intended fallback mechanism if something fails
  - It is only enabled for file reads
- Disadvantages: mainly flexibility
  - Changes always require a change in the repo and a full restart of the service