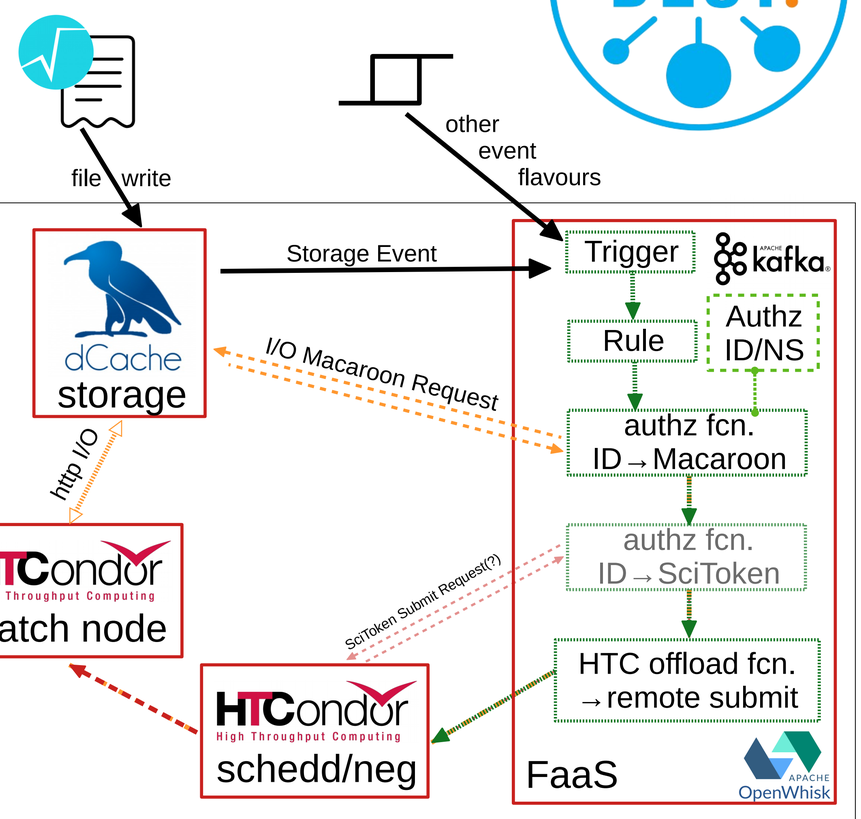
Consolidating the interactive analysis and Grid infrastructure at DESY.

Chaining Storage and High-Throughput Computing to each other

C. Beyer, T. Finnern, M. Flemming, A. Gellrich, J. Hannappel, T. Hartmann, Y. Kemp, B. Lewendel, M. Sahakyan, M. Schuh, C. Voss



Event-driven Automatised Workflows and FaaS-Offloading

Storage | f(x) | LRMS | Storage

Classic workflow engines rely primarily on “polling” the different states in their connected systems such as storage instances and compute clusters.
In contrast event based workflows allow for a “push” approach, where different actions can be combined, chained and forked to powerful workflows. To integrate various systems into such a workflow engine, a common message bus for exchanging events as well as a function registry for acting on events is needed.
With dCache Storage Events as prime example [4], we are working on combining our storage instances and offline computing resources into one workflow environment, e.g., to allow new files to automatically initiate their own processing as for calibration tasks.
Here, a dCache instance creates for each file change a Storage Event and relays it over the Kafka bus. To react on matching events and initiated further actions, functions are registered in a dynamically scalable OpenWhisk Function-As-A-Service platform. However, FaaS lambdas are intended as primarily low-latency, fast processing programs with a standardised API, we are working on offloading the actual heavy processing tasks via functioninalised remote submission to our HTCondor batch system. Combined with anonymous jobs, this allows for dynamically scaling out to computing resources.

Storage Event augmented by Authorization Tokens

```
{
  "read": {
    "path": "/pnfs/.../myinfile.root",
    "macaroon": "MDAx...97MK",
    "endpoint": "https://dcache-door.in:2880"
  },
  "write": {
    "path": "/pnfs/.../output.d",
    "macaroon": "MDAx...5JCg",
    "endpoint": "https://dcache-door.out:2880"
  }
  "exec": {
    "function": "my_fcn_ns.my_reprocessing_fcn"
  },
...
}
```

Anonymous Job

- predef. Fcn/Job
- LRMS Token
- File I/O Macaroon

Anonymous Jobs with Authorization Tokens

While *classical* jobs run under a user ID and read files belonging to such an user ID, this requires a common global *identify namespace* over all involved resources. Moving from authentication to an authorisation based resource control will allow to be more secure and especially flexible with function workflows.
In a token-based scheme, only a central token generator and authentication instance is be needed. Here, a lambda token generator is the only compute instance matching an identity to resources. On a matching event, it request on behalf of the ID authorisation tokens from the storage and compute elements. The tokens can be strictly limited to the needs of the actual processing event, e.g., requesting from a dCache instance read-only Macaroon tokens [4] for input files and limit write abilities to a confined output path. Similarly a submit token to the HTCondor batch system might be requested and added to the compute event.
The aim is for a self-sufficient event, that includes as tokens all necessary file inputs and outputs as well as compute resources.
Followingly, a compute job based on such a self-sufficient event could run locally under an anonymous user and stage files in/out by its Macaroons. Such an anonymous job could also be scaled out to other compute resources, as it is decoupled from the inital ID namespace.

Offline Computing Applications and Challenges at DESY

Next to online computing for photon light sources as PETRA III or the European XFEL with significant demands on the latency and memory, about 30 000 cores are dedicated to offline computing tasks and are managed with HTCondor as batch system. The two major offline use cases are for one WLCG jobs and for another end user jobs in the National Analysis Facility (NAF). While Grid production & MC jobs are mostly predictable in their behavior, user jobs can vary in a broad range of runtime, CPU and I/O patterns making their optimal handling a challenge. While both use cases still submit directly to the batch system in a polling way, in the future we envisage to use the offline batch system also to offload function tasks for streaming processing as described above. Here again the actual emerging usage patterns might pose a further interesting challenge.

Varying Workload Patterns

As Grid workloads are not latency sensitive, resources dedicated to such workloads are aimed for an utilization as high as possible. In contrast, NAF users expect a fast and high turnaround of jobs for an interactive experience and workloads vary more in their computing and I/O patterns than Grid jobs.

Utilization: Grid Jobs

Utilization: User Jobs

Authentification

As users require long-running jobs over several days and rely on their AFS-HOME directories as well as scratch and long-term storage mounted over NFSv4, token handling is crucial. We implemented a token renewal service to allow for such long-running jobs to access the mounted network home directories over their whole lifetime.

Jupyter Notebooks

We have deployed a Jupyterhub for our NAF groups, that starts dynamically user notebooks on HTCondor as dedicated jobs on GPU or standard resources.
Since especially for interactively notebooks users are sensitive on job upstart times, we worked on a “*fast-lane*” for notebook jobs. One hurdle were the high number of short user jobs, that slowed down the match making, which we tackled with a dedicated prioritised scheduler. Thus users have their familiar environment already available in notebooks and we can schedule dynamically resources on established workflows without allocating them constantly.

Container Abstraction

Docker

To ease maintenance and expand the operating system support for users, we are working on moving jobs into containers. While users can already run their jobs in Singularity containers, we envisage containers to wrap all jobs, which would also allow to provide a reliable and exportable API to local resources.
We are setting up a CI/CD pipeline to build and distribute to batch nodes Docker containers consisting of tools and environments familiar to the users. Jobs are intended to be run transparently in the target containers independent on the batch nodes’ OS or kernels.

Singularity for legacy Grid SL6 Jobs

As all Grid resources have been migrated to CentOS7, we provide Scientific Linux 6 containers in Singularity for legacy users. Grid SL6 submitted to an ARC CE are transparently started in a Singularity container, that is distributed over CVMFS.

On Demand Spark Cluster

To be able to scale an Apache Spark cluster on demand beyond its set of static base resources, we have investigated how to expand dynamically a cluster on the HTC farm. Deploying Spark-flavoured Singularity containers over CVMFS, we have been able to increase a cluster through ordinary batch jobs running containerized Spark workers. [6]

Opportunistic Resources

In addition to static HTC/offline resources, we investigate dynamic utilization of on- and off-site computing resources for opportunistic workloads.

Dynamic Cloud Cluster

Since the OpenStack cluster at DESY is for elasticity not run at full utilization, we are working on integrating it as an opportunistic resource to the Grid. To allow for standard Grid submissions and accounting, we have setup a dedicated ARC CE and HTCondor head node.
Preconfigured HTCondor VMs are spawned/stopped dynamically on the available resources by the Cloudscheduler instance at the University of Victoria [5]. After startup, the HTCondor VMs announce themselves as batch nodes to the local collector/negotiator to receive jobs.

Backfilling the Maxwell HPC

User demands on resources the Maxwell HPC cluster correlates with the data taking of the EuXFEL and photon experiments on site.
During low demand, we backfill sections of the cluster with opportunistic grid jobs through a dedicated ARC CE.
As demand can change quickly and for data taking responsiveness is crucial, evacuation time has to be in the order of seconds.

Talk @ CHEP 2019
https://indico.cern.ch/event/773049/contributions/3473380/

NAF Entry Page
https://bird.desy.de

Grid @ DESY
https://grid.desy.de

dCache Macaroon Documentation
https://www.dcache.org/manuals/UserGuide-5.2/macaroons.shtml

University of Victoria
HEP Research Computing
http://heprc.phys.uvic.ca/

Talk @ CHEP 2019
https://indico.cern.ch/event/773049/contributions/3473858/

This Poster @ CHEP 2019
https://indico.cern.ch/event/773049/contributions/3473851/

Using Images from: https://www.flaticon.com/free-icon/graphics-card_100310