News from the IT Department and Cluster Computing

PI IT Team

David Berghaus, *Oliver Freyermuth*, Frank Frommberger, *Michael Hübner*[1], Katrin Kohl, Ernst-Michail Limbach-Gorny[2], Andreas Wißkirchen & more helping hands in projects and from the HISKP IT Team

it-support@physik.uni-bonn.de

19th October, 2023

[1] started April 2023
[2] started June 2023

Physikalisches Institut UNIVERSITÄT BONN

Outline

1. News
2. Funded IT R&D projects on Research Data Infrastructures
3. Behind the scenes: Queuing jobs on the BAF cluster

Personnel Changes

- since 2020: FTD IT position not filled

- successor for Helmut Kortmann: Ernst-Michail Limbach-Gorny, started in June 2023, welcome 🙂

Project-specific helpers

- Development Team for web and database projects: Philipp König (*leaving*), Jan Heinrichs, Luka Vomberg, Oliver But
- Research data infrastructure projects: Luka Vomberg, Simon Thiele
- IT specialist trainees: 3 months every year in cooperation with HRZ

(several personnel changes in the past years also in these projects)

Personnel Changes

- since 2020: FTD IT position not filled

- successor for Helmut Kortmann: Ernst-Michail Limbach-Gorny, started in June 2023, welcome 🙂

PI Web team

Ian Brock, Daniel Jacobi, Florian Kirfel, Barbara Valeriani-Kaminski
(coordinating also with FTD, HISKP and department web teams)

Flexible project developers

David Berghaus (*leaving*), Katrin Kohl

Projects (highlights)

- still ongoing
  - Web development team: ongoing **development of HR system**
  - Joint project of PI & HISKP IT teams, secretaries, HRZ: **Indico**
  - Development of **common firewall** (HISKP, PI, FTD)
- new in 2023
  - bandwidth upgrade of Ceph fileservers (for Grid storage, new workernodes) $8 \times 10\ \mathrm{Gbit/s}$
  - Hybrid Conference equipment in Conference Room II, Seminar Room I (BCTP upcoming)
  - Mostly automated, GDPR-compliant managed Windows 11 Pro deployment
  - Work behind the scenes (construction planning, upgrade preparations etc.)

Project highlights: Indico event management and room booking

- Instance operated at HISKP, used for local events (from colloquia up to conferences and workshops)
- Successfully used for room booking at FTD, PI, HISKP
  Common room booking system across institutes!

Upcoming

- Login via eduGAIN (no local accounts for non-Uni-Bonn users needed anymore)
- Discussion on faculty level: Plan to open up Indico inside MNF, attract more users (goal: centralization)

Funded Projects: FIDIUM

FIDIUM project

- Federated Digital Infrastructures for Research on Universe and Matter
- Financed by the BMBF, part of ErUM-Data
- Groups: elementary particle physics, hadron physics, nuclear physics and astroparticlephysics
- 10 institutions and 4 associated partners (CERN, DESY; GridKa, GSI)
- One project in a series of projects:

  Q3/2017 → Q3/2021 ErUM-Data Pilot I: IDT-UM
  Q3/2021 → Q3/2024 ErUM-Data Pilot II: **FIDIUM**
  Q3/2024 → Q3/2025 **FIDIUM** extension
  Q3/2025 → Q3/2028 ErUM-Data 'föderierte Digitalinfrastrukturen' I
  Q3/2028 → Q3/2031 ErUM-Data 'föderierte Digitalinfrastrukturen' II

Federating compute resources

Tasks in Bonn

- Leverage BAF (HTC) and Uni Bonn HPC for Grid jobs via an overlay batch system operated by KIT ('Single point of Entry' for ATLAS, Belle II)
- 'Compute Site in a Box': Ease deployment for other sites

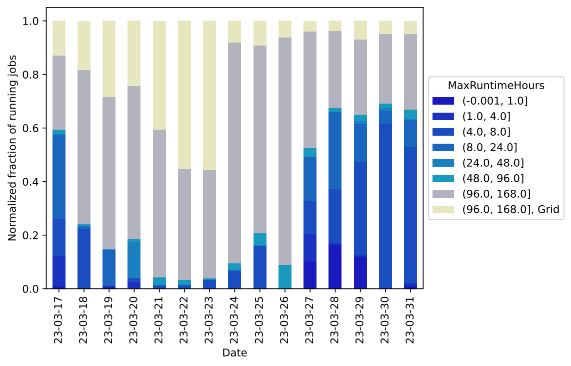
FIDIUM

UNIVERSITÄT BONN

Federating compute resources

Results so far

- Successfully operating backfilling since 2019
- Cluster is able to adapt quickly once users submit new jobs



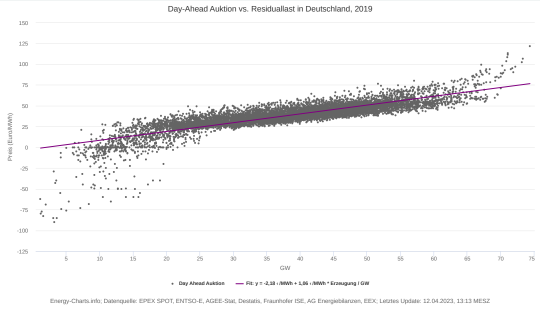
FIDIUM

UNIVERSITÄT BONN

Federating compute resources

FIDIUM successor — Planned proposal

- Premise: Heading towards an age of (renewable) energy abundance
- Consequence: Due to volatility, energy will be for free when the sun is shining & wind is blowing
- Idea: Project energy price into the future and start/stop Grid jobs accordingly



https://www.energy-charts.info/charts/price_scatter/chart.htm?l=de&c=DE&year=2019&residual_load=1&solar=0&wind_onshore=0&wind_offshore=0

FIDIUM

UNIVERSITÄT BONN

PUNCH: On the way to a Science Data Platform

Particles, **U**niverse, **N**u**C**lei and **H**adrons for the NFDI

Goal

Federated / FAIR 'Science Data Platform'
Provide infrastructure and interfaces for access to and working with data and compute resources of the communities, breaking community borders

Activities in Bonn with regard to technical infrastructure

- JupyterHub frontend for federated Compute infrastructure ('Single Point of Entry')
- Including resources in Bonn in the Compute infrastructure
- Federated storage for 'small' experiments

Other activities in Bonn

Metadata structure and definitions, Outreach activities

IT R&D for Research Data Management

- Both projects aim to federate compute and storage resources to deal with increasing resource requirements
- Connection of federated compute and storage via federated authentication
- Leverage existing compute resources via an overlay batch system
- PUNCH Goal: break community borders & offer a FAIR Science Data Platform
    - One platform / portal to allow cross-experiment analyses
    - Usage not only for Open Data
    - Allow access to data from all communities via industry standard protocols
    - Common distributed data management infrastructure including caches

PUNCH 4NFDI

FIDIUM

UNIVERSITÄT BONN

Intermediate Summary

- Continuous modernization of existing services
- Successful R&D projects in IT (FIDIUM & PUNCH)
  - Both aim to federate compute and storage with federated authentication
  - PUNCH: break community borders & develop FAIR Science Data Platform
- Debian 12 for Desktops on the horizon
- Check out our documentation pages on Confluence!

Ongoing future project

Common institute firewall for Physics institutes:
More bandwidth, IPv6, less poking of firewall holes

In case of any questions, don't hesitate to contact us!

it-support@physik.uni-bonn.de

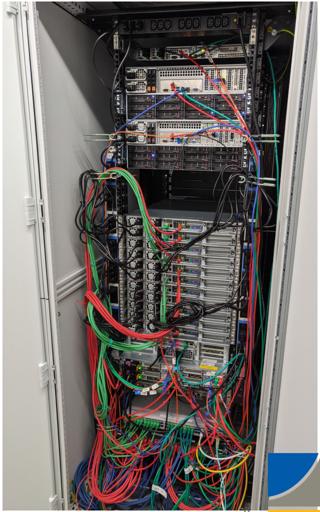
FAIR: **F**indable, **A**ccessible, **I**nteroperable and **R**e-usable

BAF Cluster

- 2017: Started with 40 worker nodes, **2240 logical cores**
- 2019 and 2020: 3 waves of memory upgrades
- February 2020: 4 x NVIDIA GeForce GTX 1080 Ti, 11 GB VRAM
- July 2020: Integration of 56 worker nodes in HRZ institute machine room (‘`CephFS_IO`’), new total: **3776 logical cores**
- November 2020: Extension with 4 worker nodes, new total: **4288 logical cores**
- April 2023: Extension with 11 worker nodes, 1 high-memory node: 4 TB RAM, new total: **7104 logical cores**
  - produce significant heat (1 kW per node)
  - fileservers upgraded to 8 × 10 Gbit/s in June 2023

BAF Cluster

- 2017: Started with 40 worker nodes, **2240 logical cores**
- 2019 and 2020: 3 waves of memory upgrades
- February 2020: 4 x NVIDIA GeForce GTX 1080 Ti, 11 GB VRAM
- July 2020: Integration of 56 worker nodes in HRZ institute machine room (‘CephFS_IO’), new total: **3776 logical cores**
- November 2020: Extension with 4 worker nodes, new total: **4288 logical cores**
- April 2023: Extension with 11 worker nodes, 1 high-memory node: 4 TB RAM, new total: **7104 logical cores**
  - produce significant heat (1 kW per node)
  - fileservers upgraded to 8 × 10 Gbit/s in June 2023



BAF Cluster: Nußallee 12

BAF Cluster: Nußallee 12

BAF Cluster: Nußallee 12

BAF Cluster: Nußallee 12

BAF Cluster: Wegelerstraße 6

- 31 racks
- 1 rack filled with 56 BAF worker nodes (on the right)

BAF Cluster: News

Operating System Containers on BAF

- Ubuntu 18.04 ⇒ End of Life, not offered anymore
- Ubuntu 20.04 ⇒ End of Life in April 2025
- Debian 10 ⇒ End of Life in June 2024
- Debian 11 and 12
- CentOS 7 ⇒ End of Life in June 2024
- RockyLinux 8 and 9

BAF Cluster: News

Organizational Developments

- Ongoing convergence to one HTC cluster for Physics Institutes
- Central HPC team: https://www.hpc.uni-bonn.de
  offering courses on Linux, Python, building your own cluster,. . .
- Coming soon: Large central HPC cluster 'Marvin'
  - Inauguration October 20th (tomorrow)
  - Tests with 'power users' starting up
  - likely publicly available end of 2023
- Ongoing discussions & plans to cover HTC and HPC use cases together

HTCondor

- Workload Management system for dedicated resources, idle desktops, cloud resources, ...
- Project exists since 1988 (named Condor until 2012)
- New naming in 2022: **HTCSS** (HTCondor Software Suite)
- Open Source, developed at UW-Madison, Center for High Throughput Computing
- Key concepts:
  - '**Submit Locally. Run globally.**' (Miron Livny)
    One interface to any available resource.
  - Integrated mechanisms for **file transfer** to / from the job
  - '**ClassAds**', for submitters, jobs, resources, daemons, ...
    Extensible lists of attributes (expressions) — more later!
  - Supports Linux, Windows and macOS and has a very diverse user base
    CERN community, Dreamworks and Disney, NASA,...
  - Focus on decentralized operation models (Peer-to-Peer), heterogeneous resource ownership
  - Dynamic integration of resources

HPC vs. HTC

High Performance Computing

tightly coupled massively parallel jobs which may span many nodes and often need low-latency interconnects, e. g.

- Climate simulations (grid cells connected to each other)
- Lattice calculations

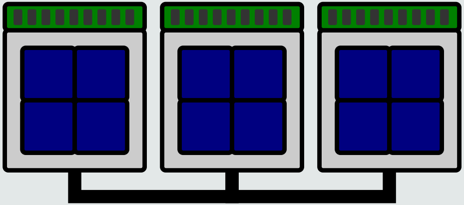
High Throughput Computing

many jobs, often submitted in large batches, usually loosely coupled or independent, goal is large throughput of jobs and / or data, e. g.

- Event-based analysis (e. g. particle physics, video rendering)
- Simulation of single events
- Parameter scans

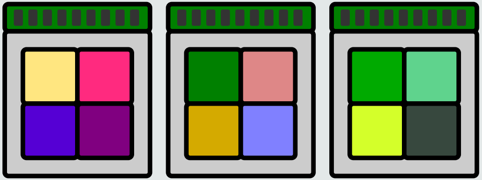
HPC vs. HTC

High Performance Computing



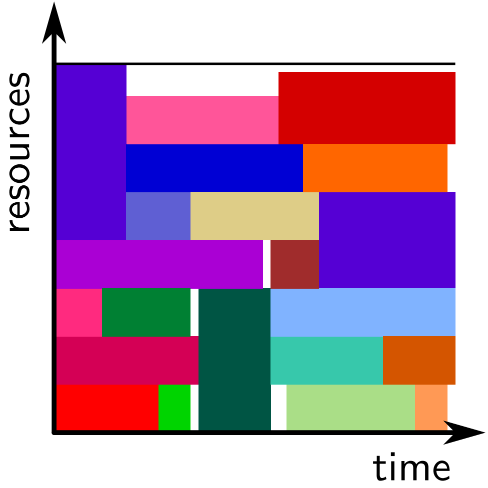
low-latency, high bandwidth interconnect
converged memory access

High Throughput Computing



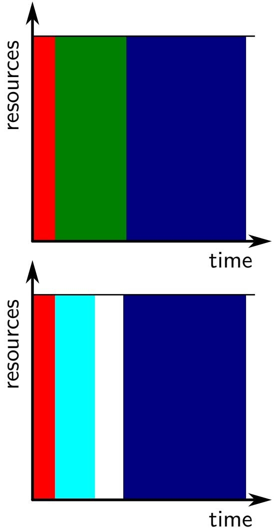
individual jobs on each CPU core,
no memory sharing

HTC: The tetris game



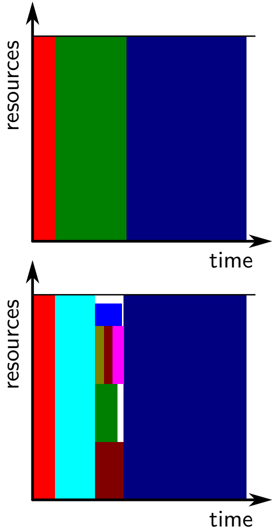
- 'Tetris' of resources: Individual, independent jobs with diverse resource requirements
- 'Fragmentation' of resources by design
- Note: The resource axis is multi-dimensional (tetris in many dimensions!)

HPC: Priority rules



- Large interconnected chunks of resources used (up to the full cluster system)
- Priority dominates scheduling, resources left empty to prepare for large jobs

HPC with backfilling



- Gaps in resource usage can be filled with shorter HTC jobs
- HPC schedulers are not well-suited for tetris with many jobs
- Overlay batch systems can work around this (large placeholder job submitted, 'tetris' within)

What HTCondor needs from the user...

A job description / Job ClassAd

Resource request, environment, executable, number of jobs,...

```
Executable = some-script.sh
Arguments  = some Arguments for our program $(ClusterId) $(Process)
Universe   = vanilla
Transfer_executable     = True

Error                   = logs/err.$(ClusterId).$(Process)
#Input                  = input/in.$(ClusterId).$(Process)
Output                  = logs/out.$(ClusterId).$(Process)
Log                     = logs/log.$(ClusterId).$(Process)

+ContainerOS="Rocky8"
Request_cpus = 2
Request_memory = 2 GB
Request_disk = 100 MB

Queue
```

What HTCondor needs from the user. . .

some-script.sh

- Often, you want to use a wrapper around complex software
- This wrapper could be a shell script, python script etc.
- It should take care of:
  - Argument handling
  - Environment setup (if needed)
  - Exit status check (bash: consider -e)
  - Data handling (e.g. move output to shared file system)

```
#!/bin/bash
source /etc/profile
set -e
SCENE=$1

cd ${SCENE}
povray +V render.ini
mv ${SCENE}.png ..
```

Submitting a job

```
$ condor_submit myjob.jdl
Submitting job(s)..
1 job(s) submitted to cluster 42.
```

There are many ways to check on the status of jobs:

- `condor_tail -f` can follow along `stdout` / `stderr` (or any other file in the job sandbox)
- `condor_q` can access job status information (memory usage, CPU time,...)
- `log` file contains updates about resource usage, exit status etc.
- `condor_history` provides information after the job is done
- `condor_ssh_to_job` may allow to connect to the running job (if cluster setup allows it)

Advanced JDL syntax

```
Executable = /home/olifre/advanced/analysis.sh
Arguments = "-i '$(file)'"
Universe = vanilla
if $(Debugging)
  slice = [:1]
  Arguments = "$(Arguments) -v"
endif
Error  = log/$Fn(file).stderr
Input  = $(file)
Output = log/$Fn(file).stdout
Log    = log/analysis.log
Queue FILE matching files $(slice) input/*.root
```

HTCondor offers macros and can queue variable lists, file names...
Can you guess what happens if you submit as follows?

```
condor_submit 'Debugging=true' analysis.jdl
```

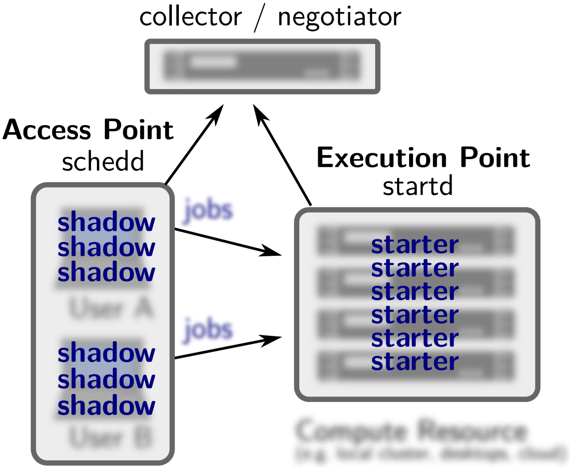
HTCondor's commandline tools (in PATH)

condor_adstash condor_annex condor_check_config condor_check_password
condor_check_userlogs condor_config_val condor_continue condor_dagman
condor_docker_enter condor_drain condor_evicted_files condor_findhost condor_gather_info
condor_history condor_hold condor_job_router_info condor_now condor_nsenter condor_ping
condor_pool_job_report condor_power condor_prio condor_q condor_qedit condor_qsub
condor_release condor_remote_cluster condor_reschedule condor_rm condor_router_history
condor_router_q condor_router_rm condor_run condor_scitoken_exchange
condor_ssh_to_job condor_stats condor_status condor_submit condor_submit_dag
condor_suspend condor_tail condor_test_match condor_token_create condor_token_fetch
condor_token_list condor_token_request condor_token_request_approve
condor_token_request_auto_approve condor_token_request_list condor_top
condor_transfer_data condor_transform_ads condor_update_machine_ad condor_userlog
condor_userlog_job_counter condor_userprio condor_vacate condor_vacate_job
condor_vault_storer condor_version condor_wait condor_watch_q condor_who

HTCondor's commandline tools (in PATH)

condor_adstash condor_annex condor_check_config condor_check_password condor_check_userlogs condor_config_val condor_continue condor_dagman condor_docker_enter condor_drain condor_evicted_files condor_findhost condor_gather_info condor_history condor_hold condor_job_router_info condor_now condor_nsenter condor_ping condor_pool_job_report condor_power condor_prio condor_q condor_qedit condor_qsub condor_release condor_remote_cluster condor_reschedule condor_rm condor_router_history condor_router_q condor_router_rm condor_run condor_scitoken_exchange condor_ssh_to_job condor_stats condor_status condor_submit condor_submit_dag condor_suspend condor_tail condor_test_match condor_token_create condor_token_fetch condor_token_list condor_token_request condor_token_request_approve condor_token_request_auto_approve condor_token_request_list condor_top condor_transfer_data condor_transform_ads condor_update_machine_ad condor_userlog condor_userlog_job_counter condor_userprio condor_vacate condor_vacate_job condor_vault_storer condor_version condor_wait condor_watch_q condor_who

Structure of HTCondor



see also Architecture talk:
https://htcondor.org/event_summary/htcondor_week_2020

HTCondor's processes

on access points (where you submit jobs)

condor_schedd Scheduler, keeps track of queue, spawns `condor_shadow`

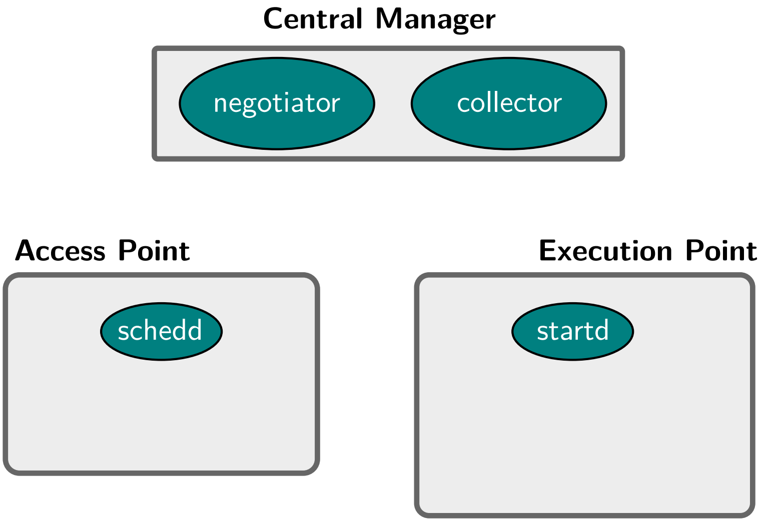
condor_shadow Monitors a single job (plus logs etc.)

on execute points (worker nodes)

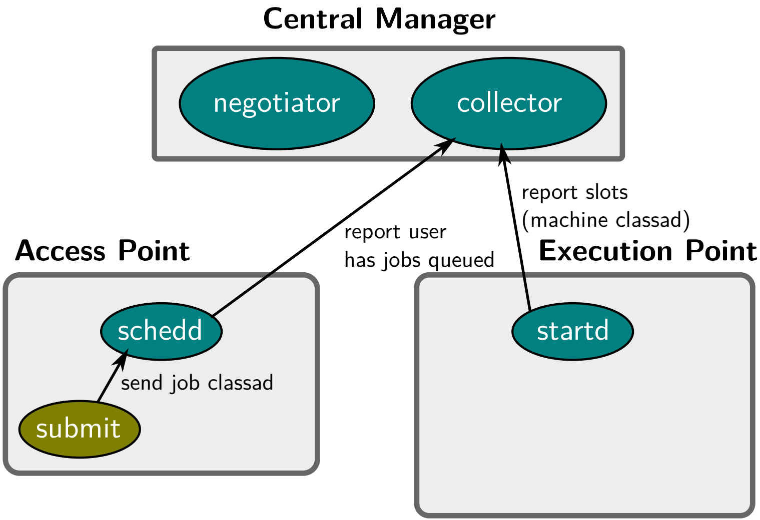
condor_startd Spawns `condor_starter`

condor_starter For each slot, takes care of jobs

Structure of HTCondor

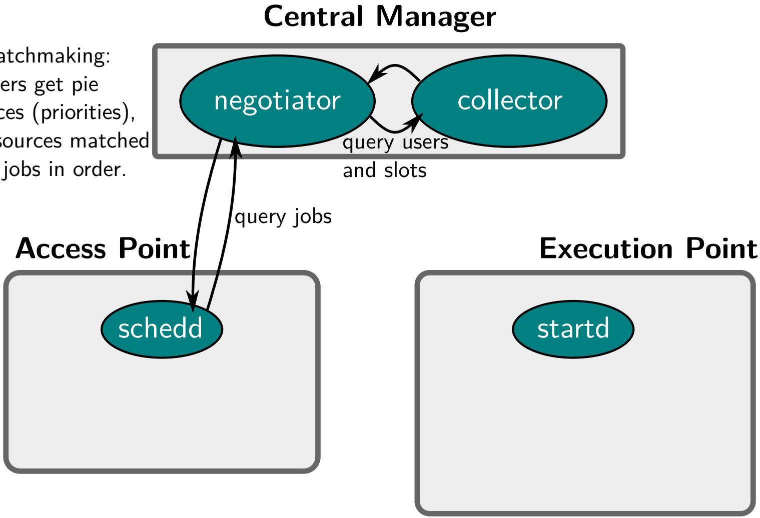


Structure of HTCondor

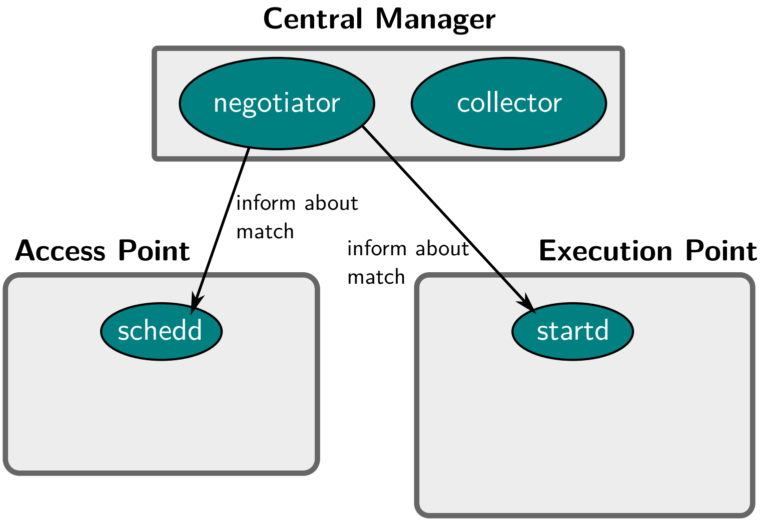


Structure of HTCondor

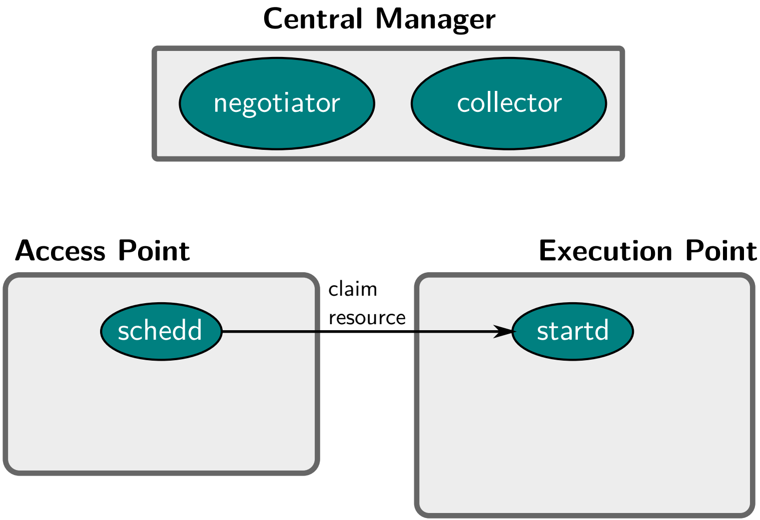
Matchmaking:
Users get pie slices (priorities), resources matched to jobs in order.



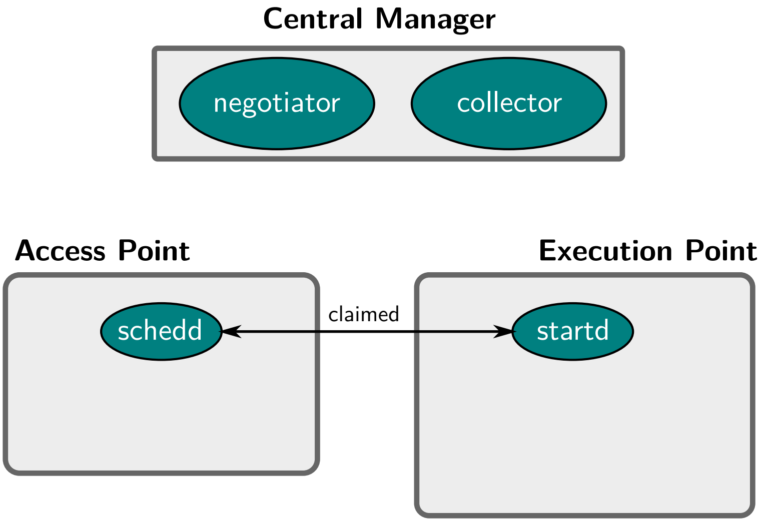
Structure of HTCondor



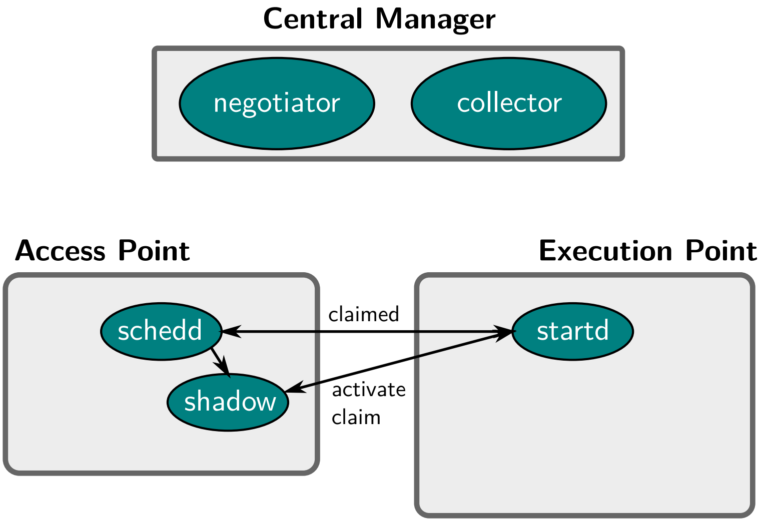
Structure of HTCondor



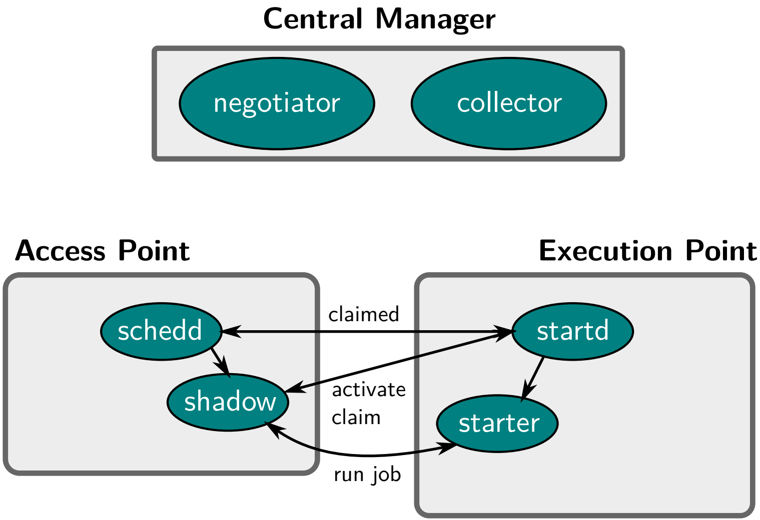
Structure of HTCondor



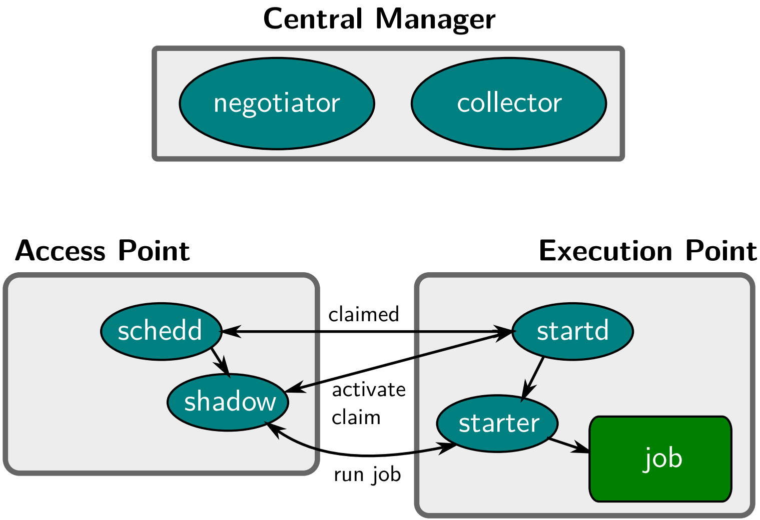
Structure of HTCondor



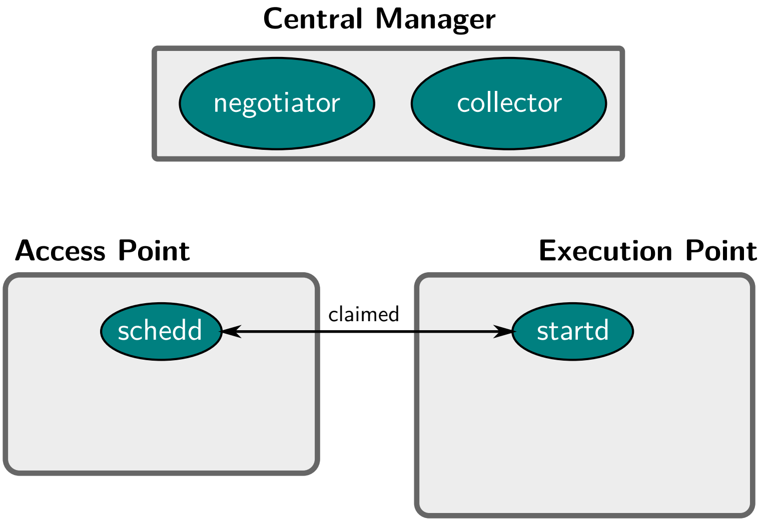
Structure of HTCondor



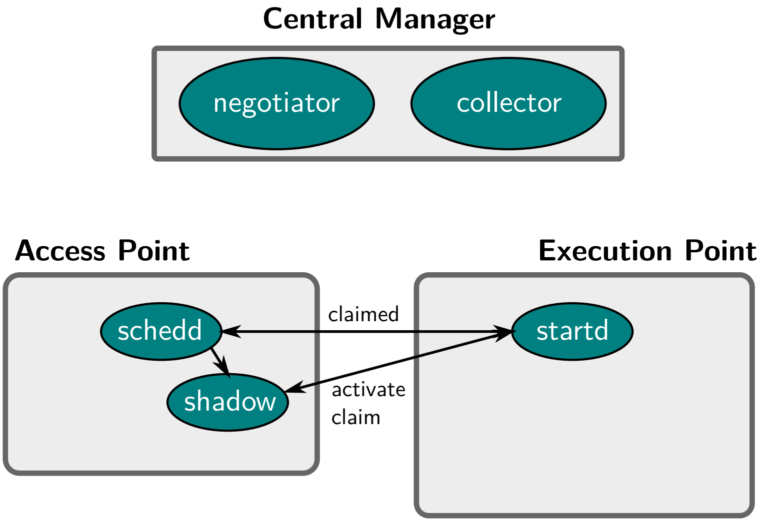
Structure of HTCondor



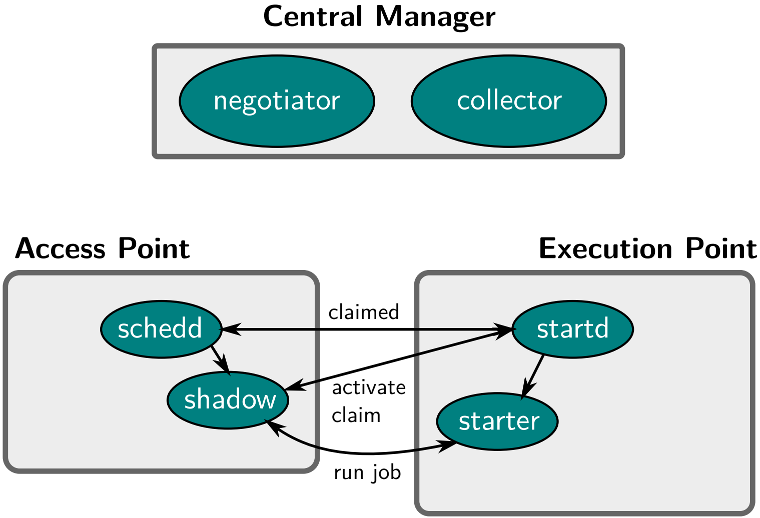
Structure of HTCondor



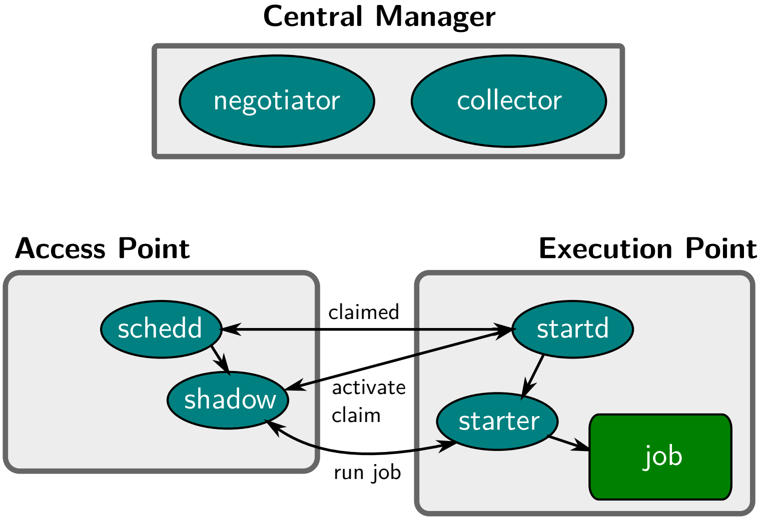
Structure of HTCondor



Structure of HTCondor



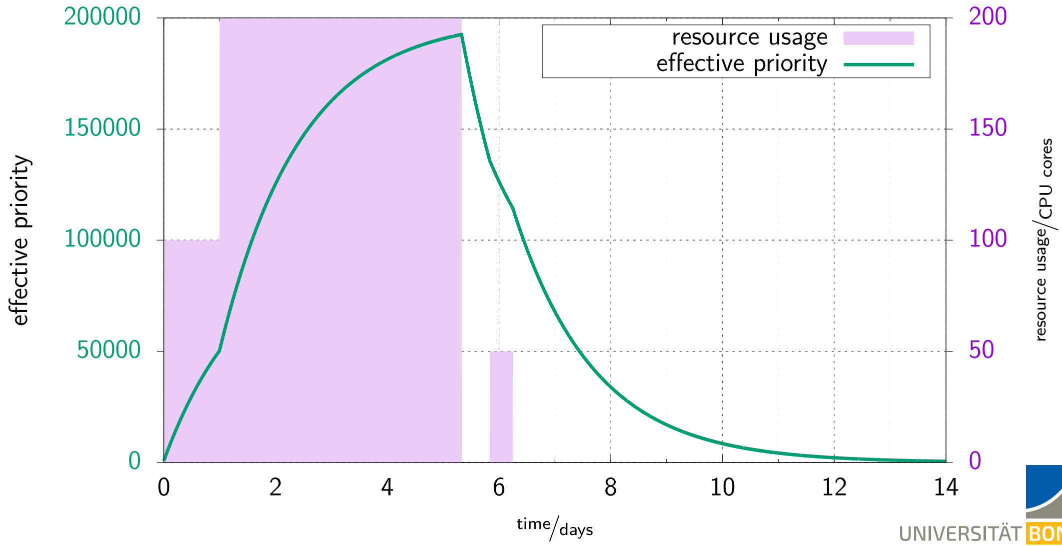
Structure of HTCondor



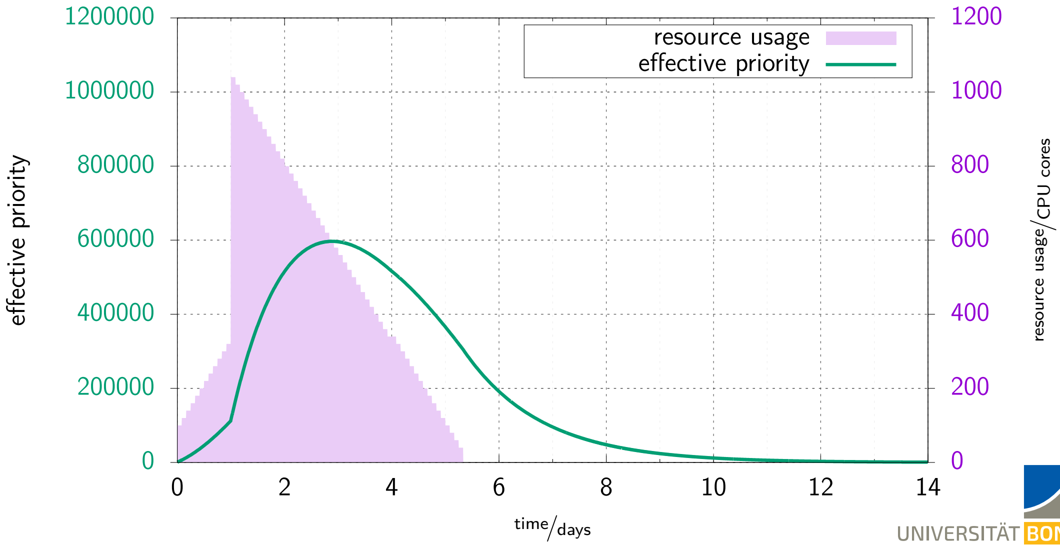
User Priorities in HTCondor

- Every user / accounting group is given an effective priority
- Effective priority approaches weighted resource usage (cores multiplied with priority factor of 1000) in an exponential manner
- Half-life constant configurable, in our case: 24 hours
- Resources are distributed amongst accounts with queued jobs proportionally, weighted by priority ('pie slices')

User Priorities in HTCondor



User Priorities in HTCondor



HTCondor's ClassAds

- Any submitter, job, resource, daemon has a ClassAd
- ClassAds are basically just expressions (`key = value`)
- Dynamic evaluation and merging possible

Job ClassAd

```
Executable = some-script.sh
+ContainerOS = "Rocky8"

Request_cpus = 2
Request_memory = 2 GB
Request_disk = 100 MB
```

Machine ClassAd

```
Activity = "Idle"
Arch = "X86_64"
Cpus = 8
DetectedMemory = 7820
Disk = 35773376
has_avx = true
has_sse4_1 = true
has_sse4_2 = true
has_ssse3 = true
KFlops = 1225161
Name = "slot1@htcondor-wn-7"
OpSys = "LINUX"
OpSysAndVer = "Rocky8"
OpSysLegacy = "LINUX"
Start = true
State = "Unclaimed"
```

HTCondor's ClassAds

- Job and Machine ClassAd extended / modified by HTCondor configuration
- Merging these ClassAds determines if job can run on machine
- Examples for dynamic parameters:
  - Select a different binary depending on OS / architecture
  - Machine may only want to 'Start' jobs from some users
- You can always check out the ClassAds manually to extract all information (use the argument `-long` to commands!)
- To extract specific information, you can tabulate any attributes (JSON also works!):

```
$ condor_q -all -global -af:hj Cmd ResidentSetSize_RAW RequestMemory RequestCPUs
 ID       Cmd         ResidentSetSize_RAW RequestMemory RequestCPUs
   2.0    /bin/sleep 91168               2048          1
```

DAGs: Directed Acyclic Graphs

- Often, jobs of different type of an analysis chain depend on each other
  Example: Monte Carlo, comparison to real data, Histogram merging,. . .
- These dependencies can be described with a DAG
- Condor runs a special 'DAGMAN' job which takes care of submitting jobs for each 'node' of the DAG, check status, limit idle and running jobs, report status etc. (like a *Babysitter job*)
- DAGMAN comes with separate logfiles, DAGs can be stopped and resumed
- DAGs ae often used behind workflow frontends (e.g. video rendering,. . .)

Working with different environments

How to compile and test code?

- Approach to access special environments or resources: **interactive jobs**
  - Advantage for admins: No separate bare metal machines
  - Advantage for users: Environment the same as in the job!
- Compile the code, pack it into a tarball, copy to shared FS / condor file transfer / CVMFS
- Can be automated with scripts / if offered, job start hooks (like '`.bashrc`')

Advantages of this approach

- Portable and stable job executables
- If combined with containers and 'mobile data': Mostly cluster independent jobs possible

‘Choose your OS’

- You add to the Job ClassAd:

```
+ContainerOS = "Rocky8"
```

- Jobs run in a container
- Same for interactive jobs (‘login-node experience’!)
- Small fractions of worker nodes exclusively for interactive jobs
  But: Interactive jobs can go to any slot!
- Resource-request specific tuning via `/etc/profile` possible:

```
REQUEST_CPUS=$(awk '/^RequestCpus/{print $3}' ${_CONDOR_JOB_AD})
export NUMEXPR_NUM_THREADS=${REQUEST_CPUS}
export MKL_NUM_THREADS=${REQUEST_CPUS}
export OMP_NUM_THREADS=${REQUEST_CPUS}
export CUBACORES=${REQUEST_CPUS}
export JULIA_NUM_THREADS=${REQUEST_CPUS}
```

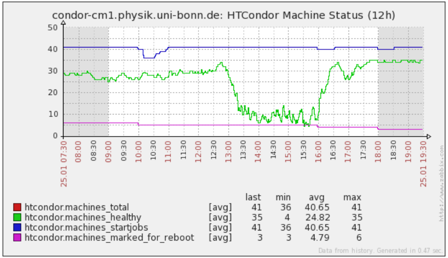
Noteworthy tools in and around HTCondor

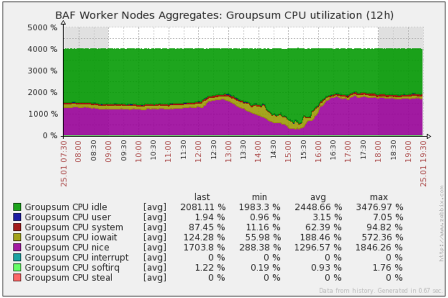
- Well-maintained Python API to directly talk to HTCondor daemons
- HTMap allows to scale map-reduce like algorithms from Python into HTC clusters
- HTCondor Adstash allows to push ClassAds from jobs / workers into ElasticSearch
- HEP-Puppet/htcondor for managed deployment and configuration of HTCondor
- MPI possible via `parallel` universe, even with containers, but manually tweaked start script and dedicated `schedd` required, and would need to teach HTCondor about interconnect topology
  ⇒ Usually not a good fit for HTC

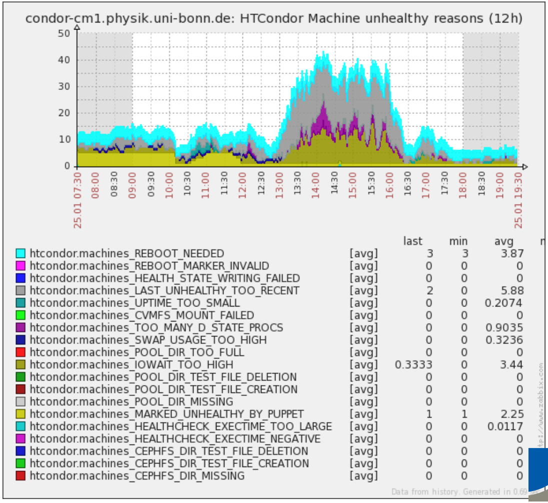
Node health checking: Reasons for 'unhealthiness'

- last 'UNHEALTHY' too recent (debouncing, $\leq 10\,\text{min}$)
- writing of status files failed or syntax bad
  (drain configuration, reboot marker, health state)
- failed reboot actions
- reboot scheduled (i.e. `shutdown` command with timeout)
- minimum uptime ($\leq 20\,\text{min}$)
- slow network interface ($\leq 100\,{}^{\text{Mbit}}/_{\text{s}}$)
- bad kernel command line (e.g. should contain '`console=`')
- unhealthy CVMFS mounts
- swap usage is too high ($> 80\,\%$, HTCondor does not monitor swap)
- iowait too high ($> 15\,\%$)
- number of processes in D state too large ($> \frac{\#\text{logical cores}}{2}$)
- read / write of execute directory or $> 80\,\%$ used (don't limit disk use yet)
- administrative 'UNHEALTHY' marker
- read / write of cluster file system, check if mount healthy
- execution time of health check ($> 10\,\text{s}$)

Node health checking







UNIVERSITÄT BONN

Node health checking

- All health information accessible via ClassAds of the machines:

```
$ condor_status -compact -af:h Machine NODE_REBOOT_REASONS
Machine                               NODE_REBOOT_REASONS
wn000.baf.physik.uni-bonn.de
wn001.baf.physik.uni-bonn.de
wn002.baf.physik.uni-bonn.de
↪   UPTIME_TOO_LARGE:39d_7h_27m_11s,NEEDS_RESTARTING_REBOOTHINT
wn003.baf.physik.uni-bonn.de
↪   UPTIME_TOO_LARGE:38d_23h_27m_19s,NEEDS_RESTARTING_REBOOTHINT
```

- Used also for monitoring, transparent for the users
- Similarly done for draining, planned reboots, node reservations, maintenances, backfilling etc.

Conclusion

- Key features of HTCondor
  - Decentralized operation model / Peer-to-Peer design
  - ClassAd system
  - Exponential evolution of user priority when fairshare is used
  - Potentially heterogeneous machine ownership supported
  - Opportunistic ressources can be integrated dynamically
  - File transfer possible

Quite some documentation on Confluence, online, passed down through PhD generations,... **How to get started?**

User Tutorial

User tutorial

https://unibonn.github.io/htcondor-bonn/

The examples teach...

- Interactive jobs and basic job submission
- Submitting job arrays
- Submitting DAGs
- Checking on your jobs status, output, and acting on errors

Game-like (playing lottery with random numbers, rendering a video),
all examples produce visible output, but still cover features used in physics analysis.

Thank you

for your attention!

FIDIUM Structure & Goals

Topic Area I **Development of tools to leverage heterogeneous resources**
Coordinators: Oliver Freyermuth, Manuel Giffels

Topic Area II **Data Lakes, Distributed Data, Caching**
Coordinators: André Brinkmann, Kilian Schwarz

Topic Area III **Testing, tuning & optimization in production & analysis environments**
Coordinators: Christian Zeitnitz, Günter Duckeck

Goals

- Enable transparent use of HPCs & Cloud infrastructures
- Flexible use of resources which are only available temporarily (e.g. backfilling)
- Concentration of most production data on large centres with local caches:
  $\Rightarrow$ Concept of data lakes

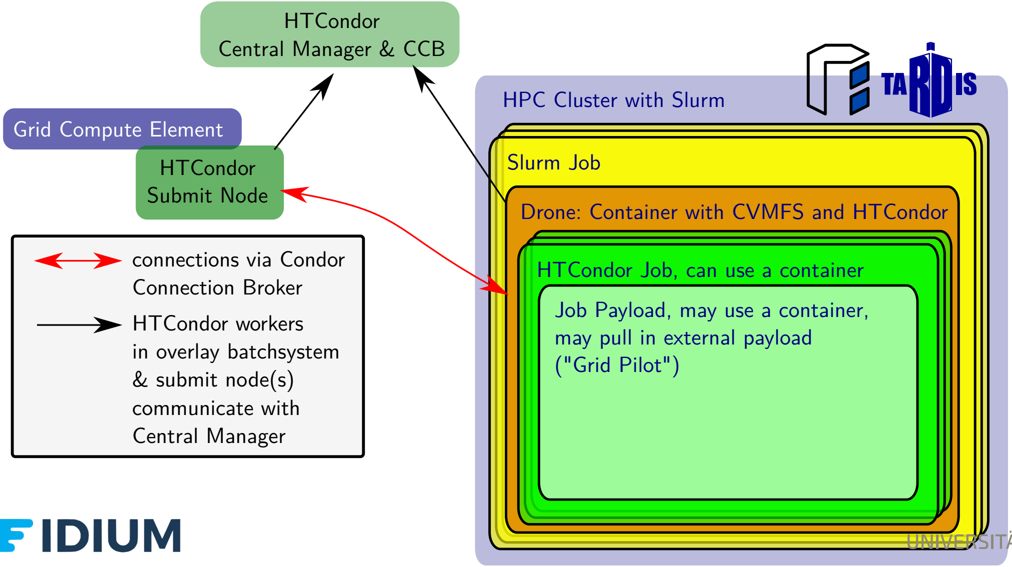
FIDIUM Activities in Bonn

- Project ‘Compute Site in a Box’
  - Usage of COBalD/TARDIS (tools developed at KIT for opportunistic computing)
  - Opening and extending existing Puppet configuration, test deployment with partners
  - Generalize unprivileged container workflows for COBalD/TARDIS resources
  - Leverage publicly accessible container infrastructure
  - Partners Mainz and GSI Darmstadt as ‘testbeds’
  - Documentation and workshops for inside and outside particle physics community
- Funding: 1 FTE for three years, equal own contribution
- Close integration with developments in other topic areas (Accounting, Caching, Monitoring)

Future

Application for the successor of FIDIUM in preparation (sustainability, checkpointing compute jobs)

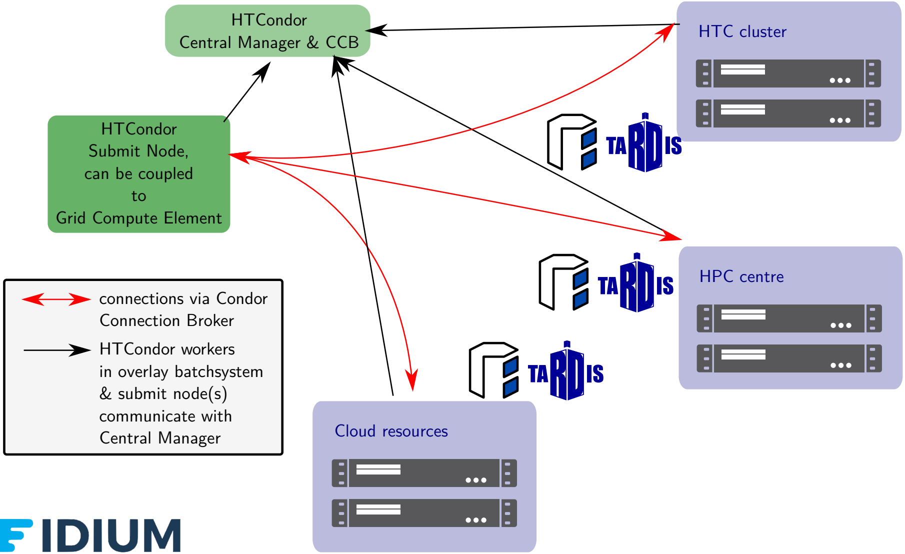
Federating compute resources



IDIUM

UNIVERSITÄT BONN

Federating compute resources



IDIUM

UNIVERSITÄT BONN

Transparent integration of Compute resources

Main steps

- Jobs submitted locally ⇒ Execution on **Overlay Batch System**
- Software stack as container (shipped via Cern-VM FS)
- Unprivileged containers started as jobs on site: Leverage User Namespaces
- Inside, an HTCondor `startd` (execute node) is started ⇒ 'Drone'
- Drone registers with Central Manager of the Overlay Batch System
- Jobs inside Overlay Batch system can use containers themselves

Scaling of number of 'Drones' based on resource usage

- COBalD — the Opportunistic Balancing Daemon
- TARDIS — The Transparent Adaptive Resource Dynamic Integration System

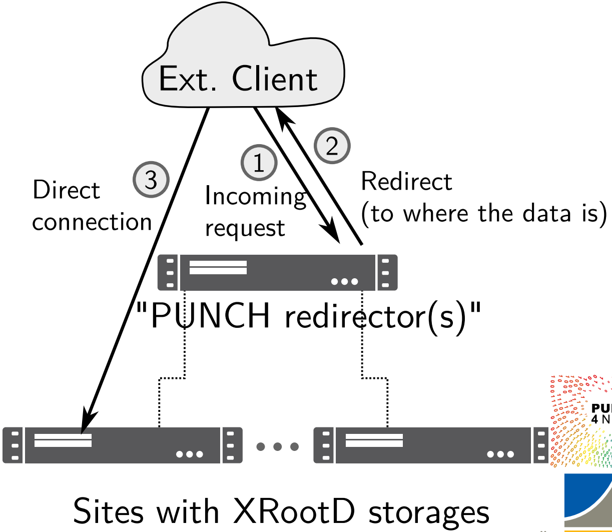
Successfully used with HPC BONNA

IDIUM

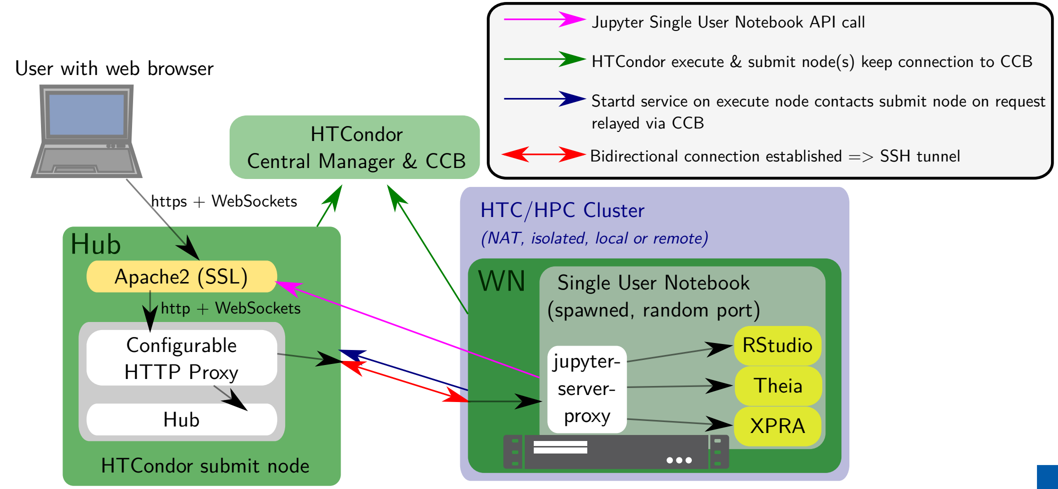
UNIVERSITÄT BONN

XRootD: Federating data access across sites

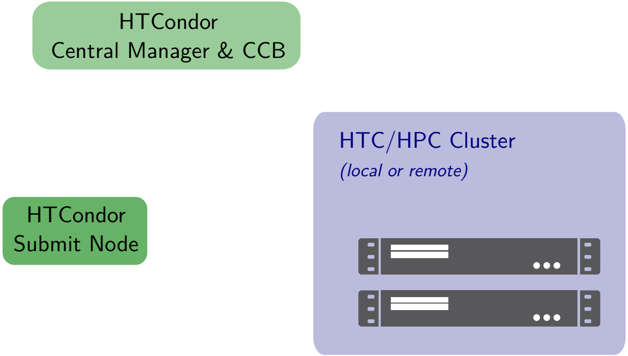
- XRootD storage provides HTTPS / WebDAV and XRootD protocols
- Global namespace: Use **g**lobal **L**ogical **F**ile **N**ames
- Redirection from central redirector to site with physical files ('merge' sites)
- Compute jobs can be routed to sites with data, or close to them
- Concept also works with caches



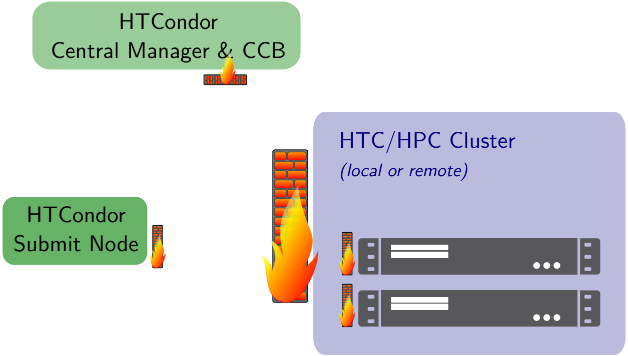
HTCondor Networking: JupyterHub



HTCondor Networking



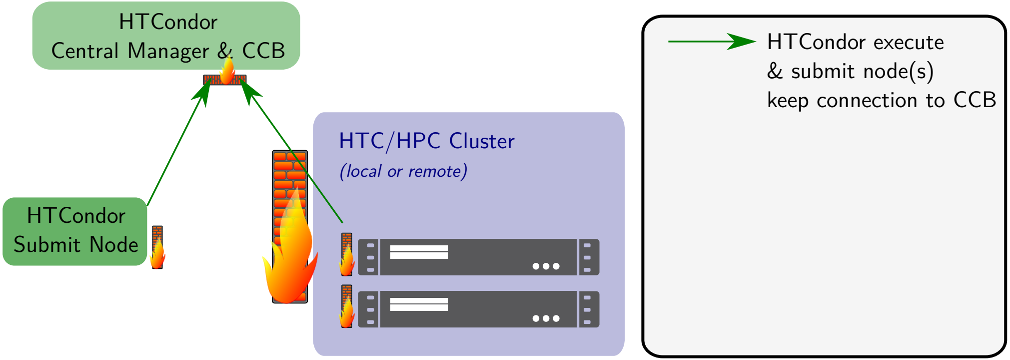
HTCondor Networking



Firewalling & NAT

- FW on each node (HTCondor port open)
- NAT(s), router(s), FWs in front of cluster networks

HTCondor Networking

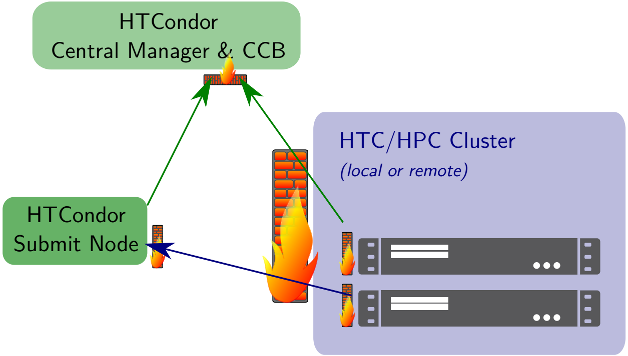


Firewalling & NAT

- FW on each node (HTCondor port open)
- NAT(s), router(s), FWs in front of cluster networks

Note:
Via the shared port daemon,
only a single port needs to be open
on the submit node and CCB node

HTCondor Networking



HTCondor execute & submit node(s) keep connection to CCB

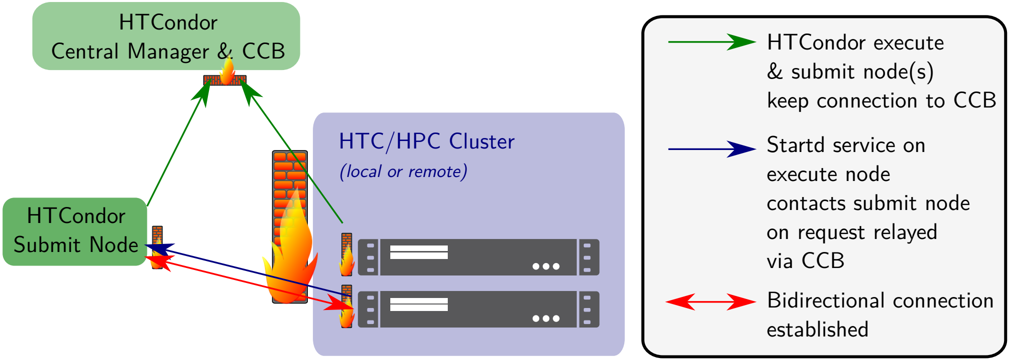
Startd service on execute node contacts submit node on request relayed via CCB

Firewalling & NAT

- FW on each node (HTCondor port open)
- NAT(s), router(s), FWs in front of cluster networks

Note:
Via the shared port daemon, only a single port needs to be open on the submit node and CCB node

HTCondor Networking

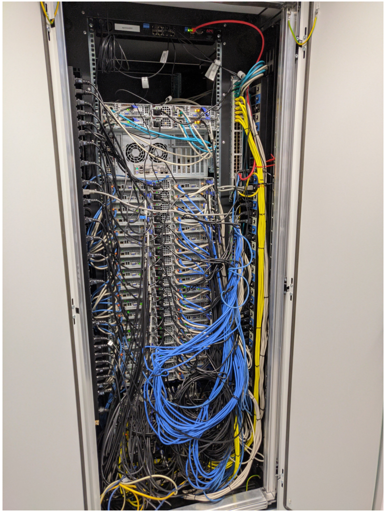


Firewalling & NAT

- FW on each node (HTCondor port open)
- NAT(s), router(s), FWs in front of cluster networks

Note:
Via the shared port daemon,
only a single port needs to be open
on the submit node and CCB node

Server Rooms: HRZ Institute Machine Room



- 56 worker nodes ('rear view')
- 1 Gbit/s ethernet, switches with 10 Gbit/s uplink $\Rightarrow$ `CephFS_IO` 'medium'
- Nodes have to be drained (starting 7 days before!) if outside temperature exceeds $\approx 35\,^\circ\mathrm{C}$
- Relying on DWD MOSMIX (Model Output Statistics-MIX) calculations, quite reliable (with error bands!)

Server Rooms: FTD

- 6 racks:
  - 2 network distribution and file servers
  - 2 service machines
  - 2 phone infrastructure
- central 60 kW UPS

Server Rooms

HRZ machine room	HISKP	PI	FTD

BAF Cluster: Compute Nodes

(location HISKP coming soon)

BAF Cluster: Storage

Virtualization infrastructure

almost 120 VMs, hypervisors and storage
redundant, Ceph RADOS Block devices, 3 copies

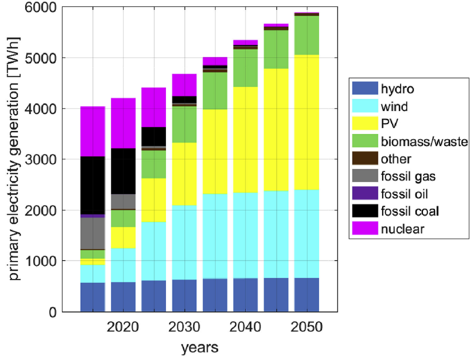
Overview

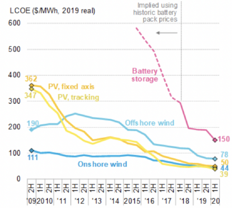
- Continue toolbox around COBalD/TARDIS to enable opportunistic computing everywhere
- Finalize connected older ideas on Compute Site in a Box
- Catchy new project (in collaboration with others?): **Computing in the age of strong sector coupling**
- Premise: We are headed towards a 100 % Renewable Energy (RE) based power infrastructure
- Transition has to take place in $\mathcal{O}(10\text{–}20)$ years, and there are no alternative technologies at even remotely reasonable price available on that timescale
- Volatility of RE sources has far reaching implications

The 2 Key Paradigms of the New Age Of Energy Abundance

Paradigm 1

There will never be a cheaper source of energy again than solar (and to a certain extent, for a limited time, wind).




BloombergNEF: 'Already cheaper to install new-build battery storage than peaking plants'

By Andy Colthorpe

April 30, 2020

Africa & Middle East, Americas, Asia & Oceania, Europe | Grid Scale | Market Watch

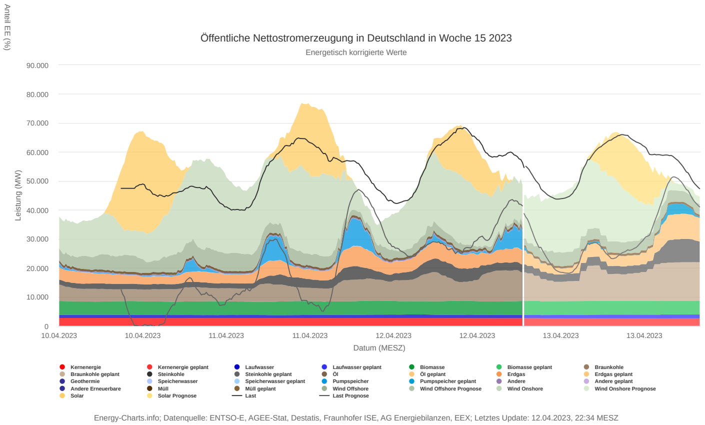
LinkedIn | Twitter | Reddit | Facebook | Email

C. Breyer et al., https://doi.org/10.1016/j.renene.2019.02.077, https://www.energy-storage.news/

The 2 Key Paradigms of the New Age Of Energy Abundance

Paradigm 2

Volatility of source availability will (probably — with caveats depending on the cost & price of Power-to-X) mean that *energy is for free when the sun is shining and the wind is blowing.*



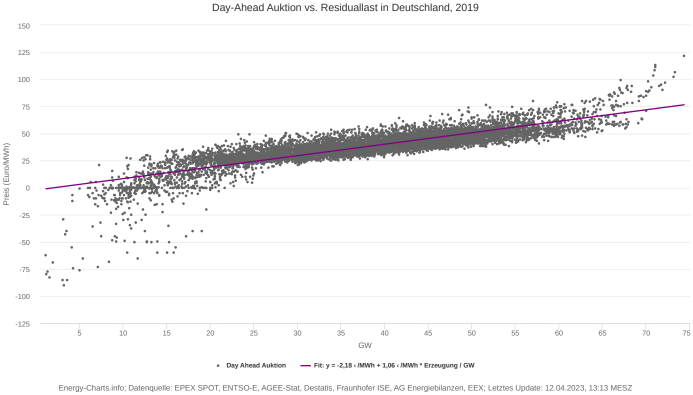
https://www.energy-charts.info/charts/power/chart.htm?l=de&c=DE

UNIVERSITÄT BONN

The 2 Key Paradigms of the New Age Of Energy Abundance

Paradigm 2

Volatility of source availability will (probably — with caveats depending on the cost & price of Power-to-X) mean that *energy is for free when the sun is shining and the wind is blowing.*



https://www.energy-charts.info/charts/price_scatter/chart.htm?l=de&c=DE&year=2019&residual_load=1&solar=0&wind_onshore=0&wind_offshore=0

Project Idea

- Get long-term real fluctuations of renewables from https://energy-charts.info/
- Parametrize energy price dependents of RE power on wind/sunshine from https://www.energy-charts.info/charts/price_scatter/chart.htm?l=de&c=DE&wind_onshore=1&wind_offshore=1&hydro_run-of-river=1&biomass=1&geothermal=1&hydro_water_reservoir=1&hydro_pumped_storage=1
- Get energy price projections e.g. from https://ieeexplore.ieee.org/document/9837910
- Project computing hardware price from past experience
- Optimize ‘Computing Overcapacity’ as a function of the duty cycle
- ‘If we want to have a duty cycle of X %, we will save Y k€ on electricity and have to spend Z k€ more on hardware to make up for lost time’
- Regulation of COBalD/TARDIS drone count, feedback from AUDITOR possible
- Key ingredient: ‘Checkpoint and restore’ of compute jobs such that jobs can be started, stopped or moved without loss, ideally on the container level.

UNIVERSITÄT BONN