HTCondor Tutorial

Matthias Schnepf | 17. July 2025

Computing Resources at ETP

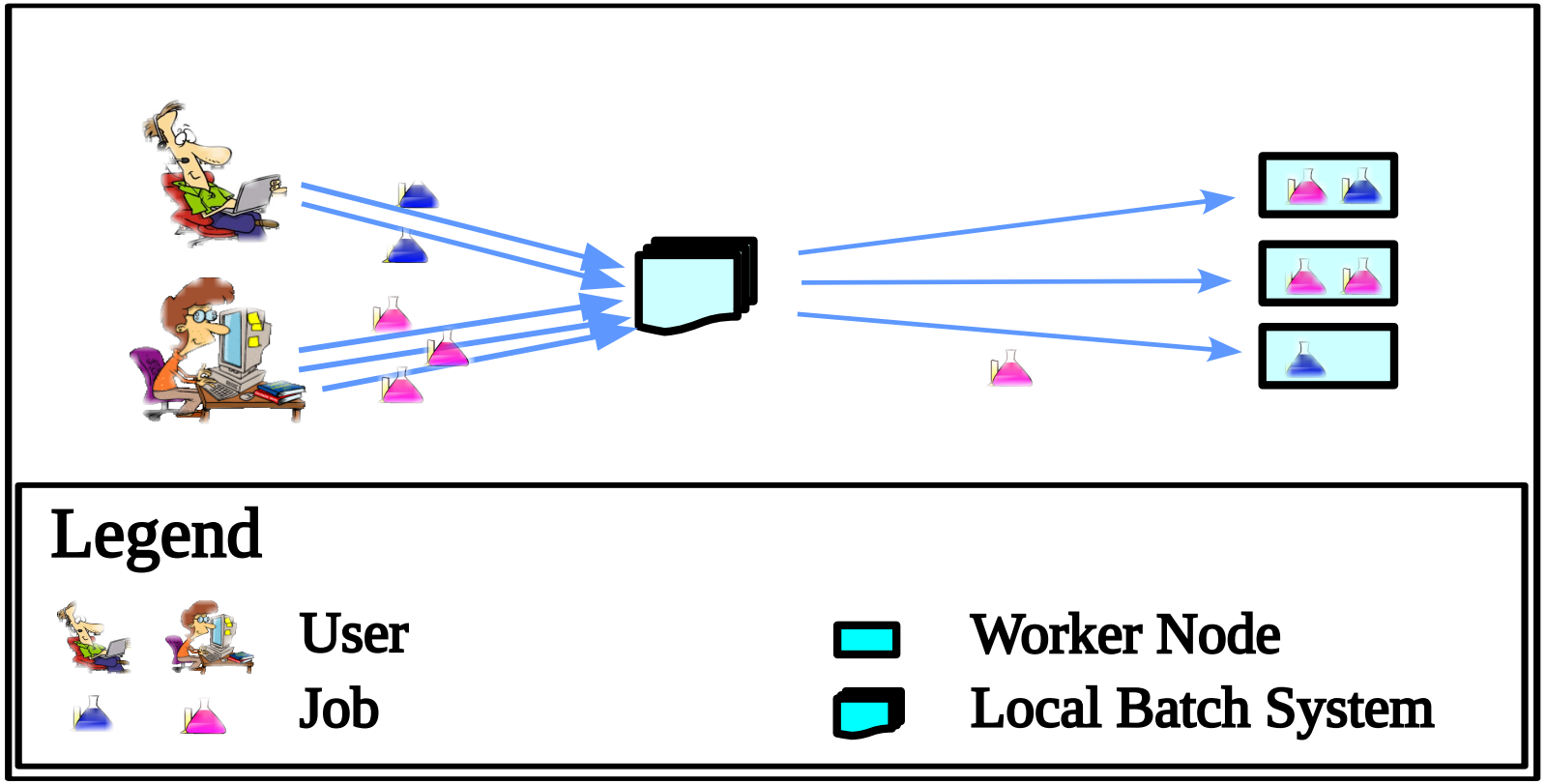
- end-user device: laptop/desktop PC/tablet,...
  - web browsing
  - video conference
  - login to other computing resources
- portal machines: bms[1,3] portal1, deepthought(GPUs)
  - powerful machines to develop and test software
  - can be used for **special** workflows
  - access to batch system/Grid
- batch system/Grid
  - distributed bunch of machines (worker nodes)
  - majority of computing resources

Storage Resources

- storage mounted on all desktops, portal machines and some of the worker nodes:
  - */home*
    - for configs, mails, ...
    - 50 GB limit per user
    - backup every second day
  - */work*
    - for software and files with fast access
    - 200 GB limit per user
    - weekly backup
  - */ceph*
    - for large datasets
    - about 1.3 PB total
    - please clean up from time to time
    - no backup
- Grid storage
  - for datasets and software sandboxes
  - accessible from everywhere via Grid protocols (see later)
  - collaboration storage
    - official dataset
  - NRG (**N**ational Grid **R**esource **G**ermany)
    - for local group
    - located at GridKa with 80 Gbit s^{-1} network connection to ETP
    - CMS: 2.75 PB
    - Belle II 250 TB

Batch System

- batch system: coordinated run of batch jobs (non-interactive programs) on distributed resources
  - submit nodes (portal machines)
  - worker nodes
  - central node
- batch job
  - executable
  - meta data
    - name of executable
    - memory requirements
    - submit host
    - worker node
    - ...



Based on "The Pilot Way to Grid Resources Using glideinWMS" from I. Sfiligoi at. all

Batch Job (1)

- executable (program/script that run on a worker node)
  1. (worker node information)
  2. setup environment
  3. copy data to worker node if necessary
  4. run workload
  5. copy output to storage e.g. */ceph* or NRG

```
#!/bin/bash

##### WN informaton
echo -e "Hostname : $(hostname)"
echo -e "user : $(whoami)"
echo -e "spawndir : $(pwd)"

##### setup environment
source /cvmfs/belle.cern.ch/tools/b2setup release-09-00-01

##### start workload at spawndir and create output.root with results
basf2 /work/$(whoami)/HTCondor_Tutorial/code/01_simple_submit/mc_example.py -o output.root

##### copy output from WN
cp output.root /work/$(whoami)/HTCondor_Tutorial/code/01_simple_submit/simple_submit_output.root
```

Batch Job (1)

- executable (program/script that run on a worker node)
  1. (worker node information)
  2. setup environment
  3. copy data to worker node if necessary
  4. run workload
  5. copy output to storage e.g. */ceph* or NRG
- should be tested on portal machine before submission
  - check for errors
  - check for resource demand via htop: CPU, number of processes, Memory,...

```
#!/bin/bash

##### WN informaton
echo -e "Hostname : $(hostname)"
echo -e "user: $(whoami)"
echo -e "spawndir: $(pwd)"

##### setup environment
source /cvmfs/belle.cern.ch/tools/b2setup release-09-00-01

##### start workload at spawndir and create output.root with results
basf2 /work/$(whoami)/HTCondor_Tutorial/code/01_simple_submit/mc_example.py -o output.root

##### copy output from WN
cp output.root /work/$(whoami)/HTCondor_Tutorial/code/01_simple_submit/simple_submit_output.root
```

Batch Job (1)

- executable (program/script that run on a worker node)
  1. (worker node information)
  2. setup environment
  3. copy data to worker node if necessary
  4. run workload
  5. copy output to storage e.g. */ceph* or NRG
- should be tested on portal machine before submission
  - check for errors
  - check for resource demand via htop: CPU, number of processes, Memory,...
- develop portable code
- job runtime should be longer than 20 min
- output file size should be bigger than 1 GB

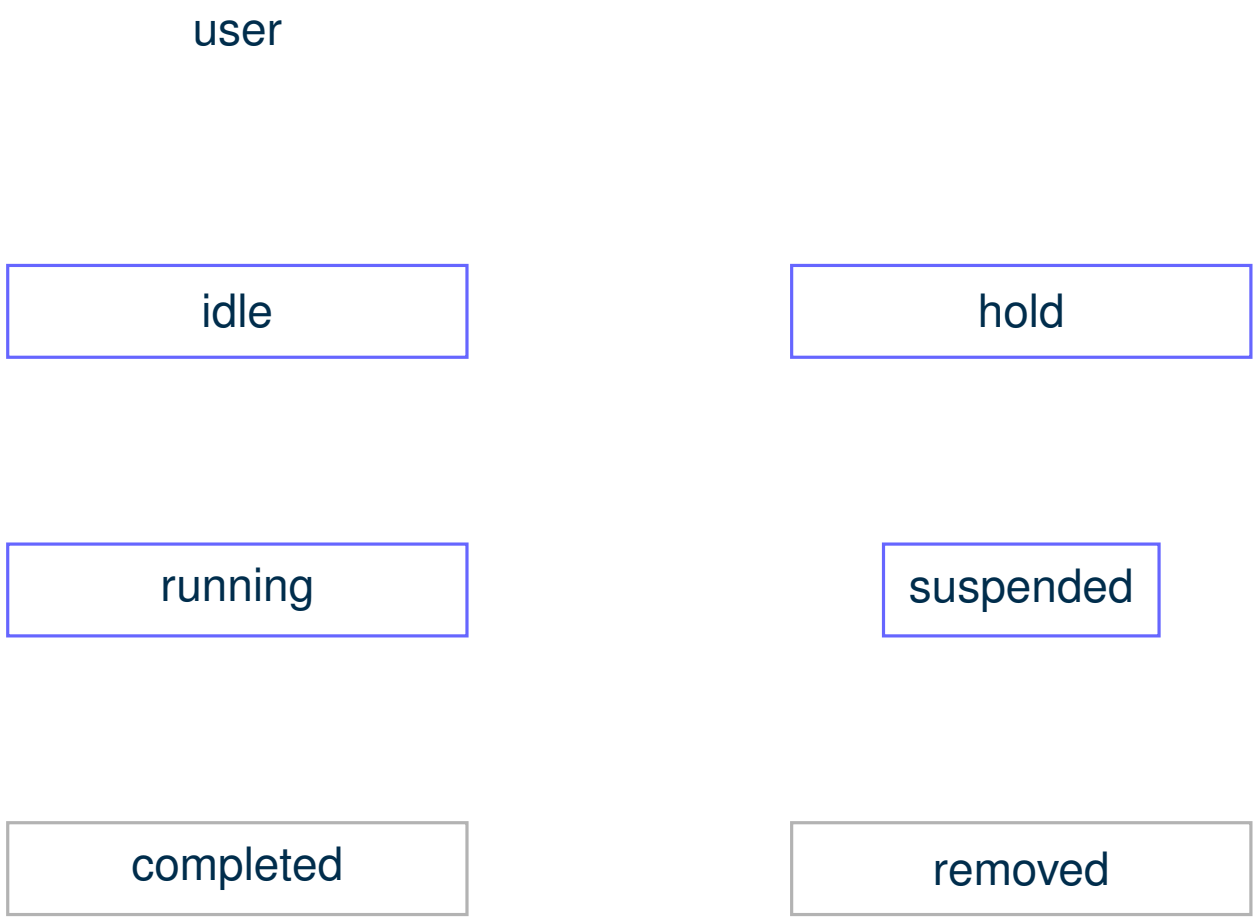
```
#!/bin/bash

##### WN informaton
echo -e "Hostname : $(hostname)"
echo -e "user: $(whoami)"
echo -e "spawndir: $(pwd)"

##### setup environment
source /cvmfs/belle.cern.ch/tools/b2setup release-09-00-01

##### start workload at spawndir and create output.root with results
basf2 /work/$(whoami)/HTCondor_Tutorial/code/01_simple_submit/mc_example.py -o output.root

##### copy output from WN
cp output.root /work/$(whoami)/HTCondor_Tutorial/code/01_simple_submit/simple_submit_output.root
```

Batch Job (2)

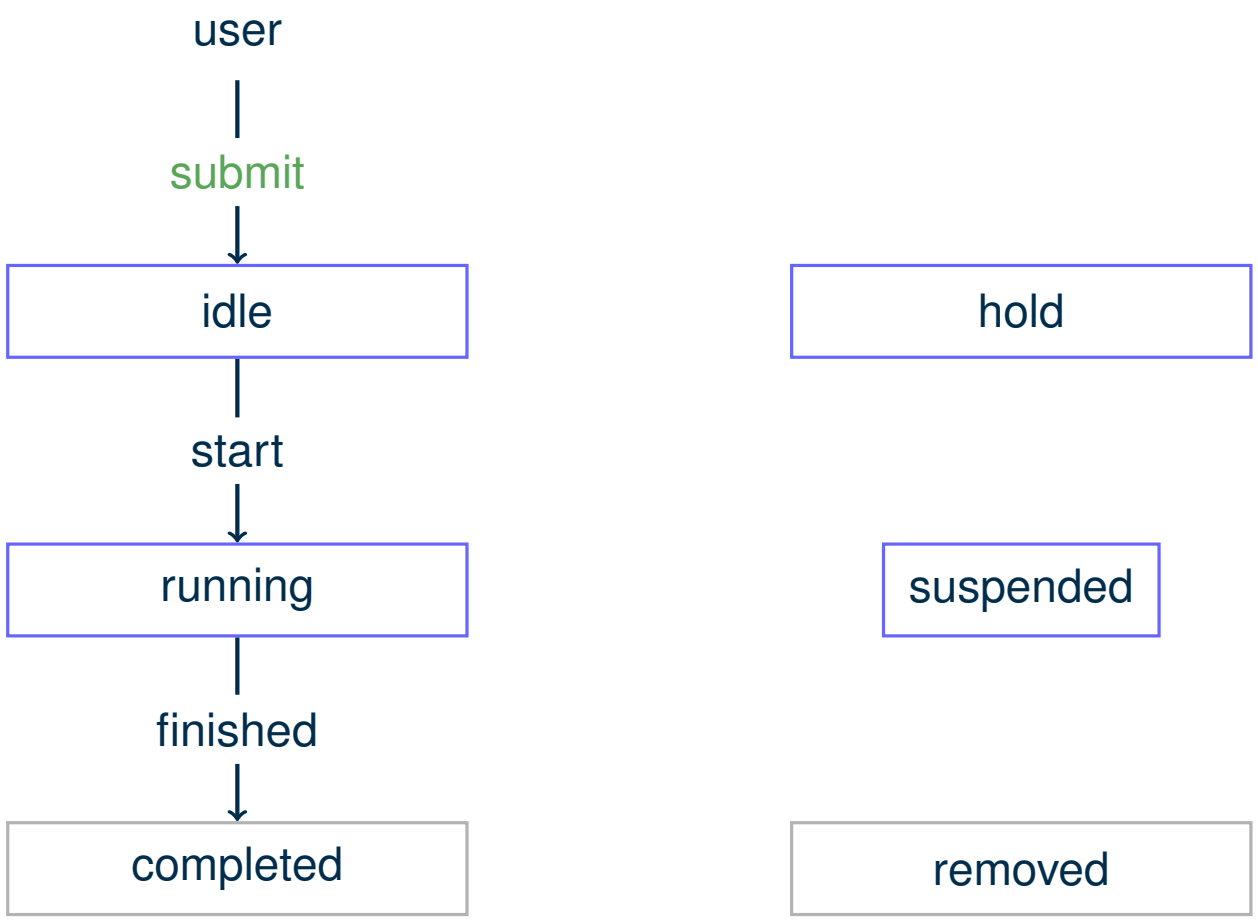
- submit file (meta data about the job)
  - container environment
  - executable
  - resource demand
    - CPU cores (default 1)
    - RAM (default 2000 MB)
    - Disk (default 500.000 kB)
    - Walltime (default 86 400 sec)
  - logging
  - accounting group
    - beamdump (5%)
    - belle (25%)
    - cms (70%)
  - submit (queue)
- submit with *condor_submit submit_file.jdl*

```
Universe = container
container_image = /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cverstege/alma9-gridjob:latest

executable = ./simple_submit.sh

request_cpus = 1
request_memory = 2000
request_disk = 500000
+RequestWalltime = 86400

log = log.log
stdout = stdout.log
stderr = stderr.log

accounting_group=belle

queue
```

Job Status

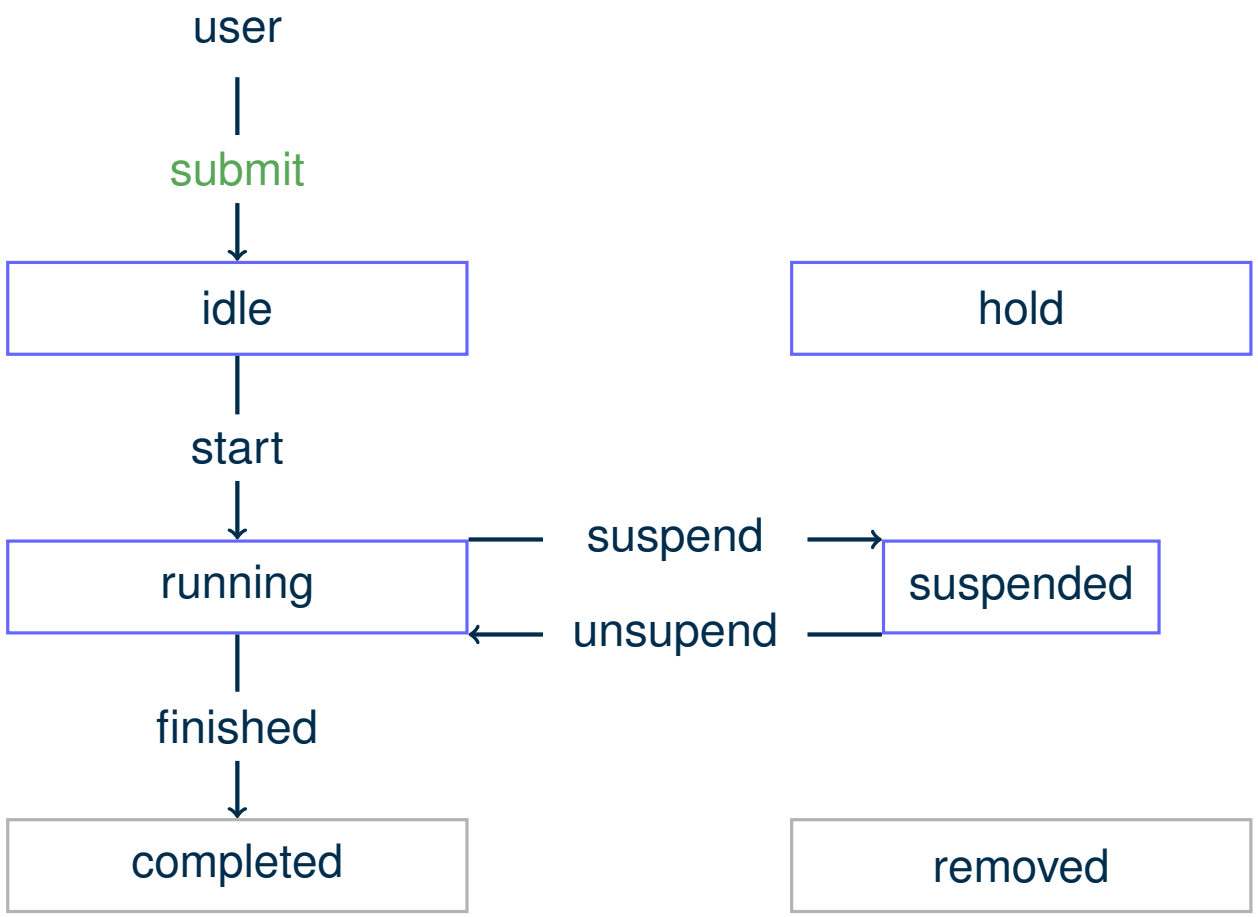


Job Status



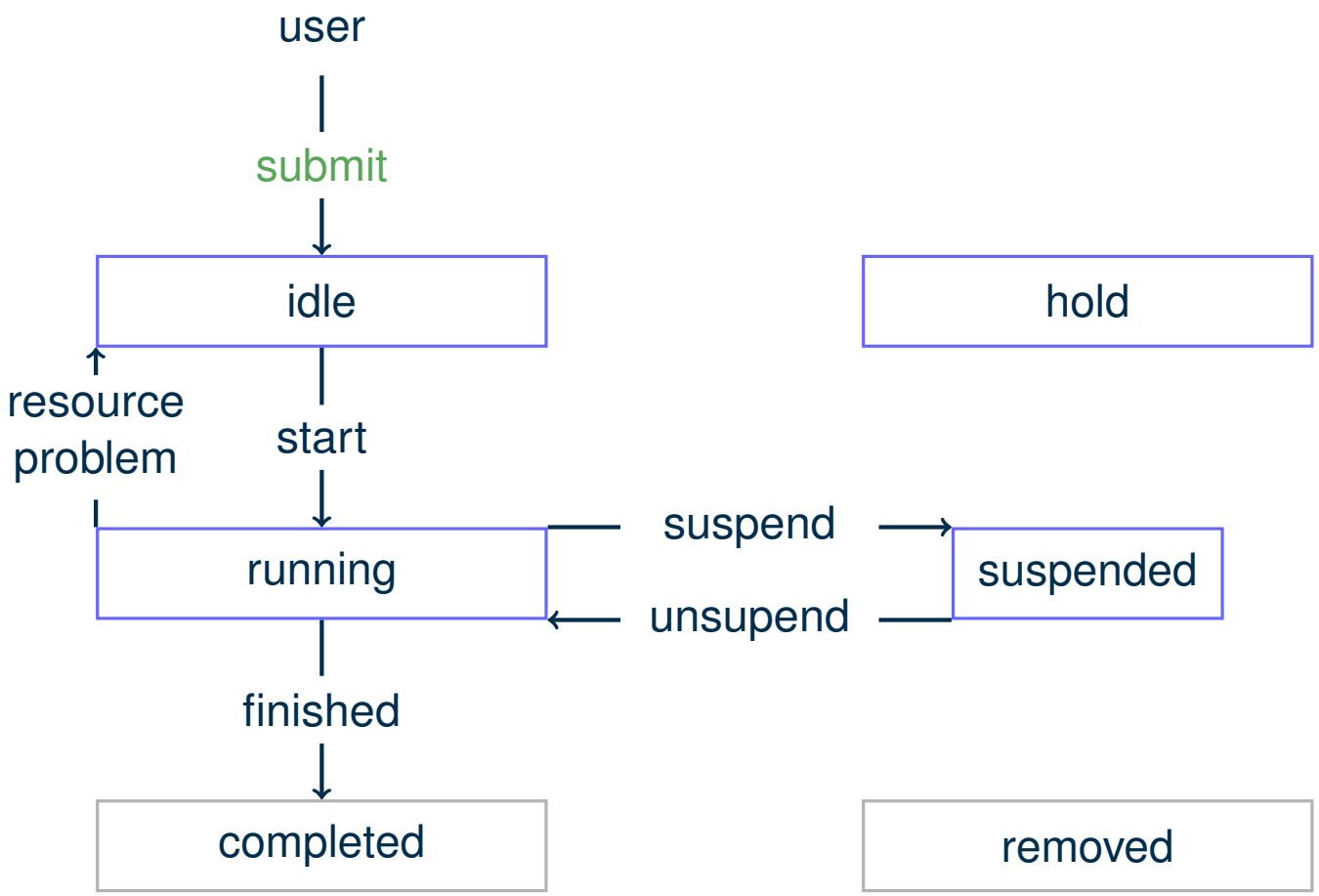
- only green action can be done by users

Job Status



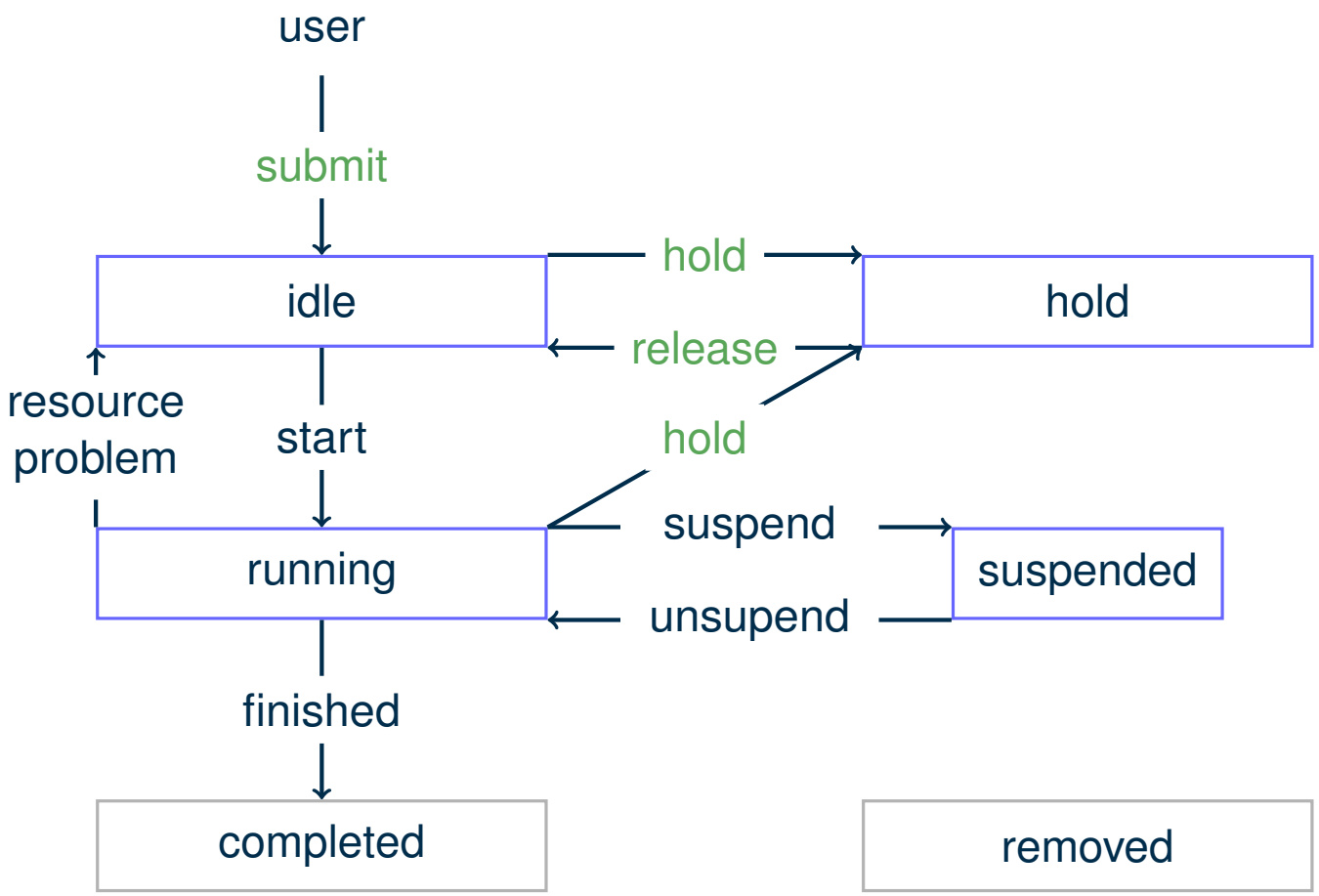
- only green action can be done by users
- only running and suspended jobs are on WNs

Job Status



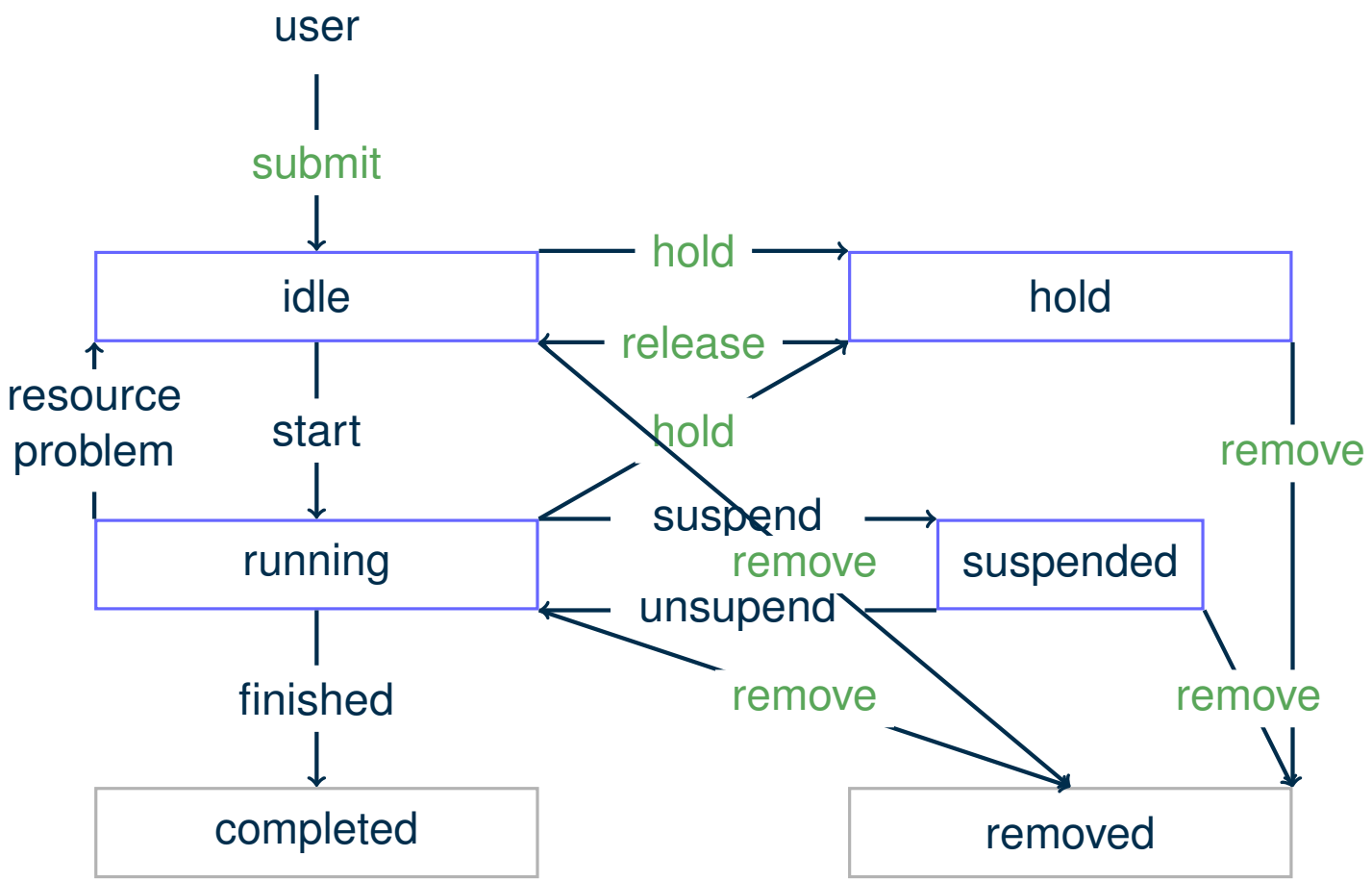
- only green action can be done by users
- only running and suspended jobs are on WNs

Job Status



- only green action can be done by users
- only running and suspended jobs are on WNs
- hold jobs are removed from the WN, unsaved results are gone

Job Status



- only green action can be done by users
- only running and suspended jobs are on WNs
- hold jobs are removed from the WN, unsaved results are gone
- completed and removed jobs are history and can not be changed

Useful Condor Commands

- *condor_submit*
  - submit job to queue of the batch system
- *condor_q*
  - shows the status of jobs in queue
  - see jobs from all submit nodes use *-g <username>* option
- *condor_rm*
  - remove jobs from queue
  - running jobs get killed and removed
- *condor_history*
  - shows job history (removed and completed jobs)
  - ETP history is long; please use *-limit <number>*

Useful Condor Commands

- *condor_submit*
  - submit job to queue of the batch system
- *condor_q*
  - shows the status of jobs in queue
  - see jobs from all submit nodes use *-g <username>* option
- *condor_rm*
  - remove jobs from queue
  - running jobs get killed and removed
- *condor_history*
  - shows job history (removed and completed jobs)
  - ETP history is long; please use *-limit <number>*
- *condor_release*
  - change jobs status from *hold* to *idle*

- *condor_q*, *condor_rm*, *condor_history*, *condor_q_edit* and *condor_release* has similar selections
  - *<user>*
  - *ClusterID.ProcID* / *ClusterID*
  - *-constraint 'classadd expression'* e.g. *-constraint 'RemoteJob =?= True'*

Useful Condor Commands

- *condor_submit*
  - submit job to queue of the batch system
- *condor_q*
  - shows the status of jobs in queue
  - see jobs from all submit nodes use *-g <username>* option
- *condor_rm*
  - remove jobs from queue
  - running jobs get killed and removed
- *condor_history*
  - shows job history (removed and completed jobs)
  - ETP history is long; please use *-limit <number>*
- *condor_release*
  - change jobs status from *hold* to *idle*

- *condor_q*, *condor_rm*, *condor_history*, *condor_q_edit* and *condor_release* has similar selections
  - *<user>*
  - *ClusterID.ProcID* / *ClusterID*
  - *-constraint 'classadd expression'* e.g. *-constraint 'RemoteJob =?= True'*
- *condor_q_edit*
  - edit job attributes

Useful Condor Commands

- *condor_submit*
  - submit job to queue of the batch system
- *condor_q*
  - shows the status of jobs in queue
  - see jobs from all submit nodes use *-g <username>* option
- *condor_rm*
  - remove jobs from queue
  - running jobs get killed and removed
- *condor_history*
  - shows job history (removed and completed jobs)
  - ETP history is long; please use *-limit <number>*
- *condor_release*
  - change jobs status from *hold* to *idle*

- *condor_q*, *condor_rm*, *condor_history*, *condor_q_edit* and *condor_release* has similar selections
  - *<user>*
  - *ClusterID.ProcID* / *ClusterID*
  - *-constraint 'classadd expression'* e.g. *-constraint 'RemoteJob =?= True'*
- *condor_q_edit*
  - edit job attributes
- *condor_status*
  - get information of the machines in the system
  - status of the WNs per default

Useful Condor Commands

- *condor_submit*
  - submit job to queue of the batch system
- *condor_q*
  - shows the status of jobs in queue
  - see jobs from all submit nodes use *-g <username>* option
- *condor_rm*
  - remove jobs from queue
  - running jobs get killed and removed
- *condor_history*
  - shows job history (removed and completed jobs)
  - ETP history is long; please use *-limit <number>*
- *condor_release*
  - change jobs status from *hold* to *idle*

- *condor_q*, *condor_rm*, *condor_history*, *condor_q_edit* and *condor_release* has similar selections
  - *<user>*
  - *ClusterID.ProcID* / *ClusterID*
  - *-constraint 'classadd expression'* e.g. *-constraint 'RemoteJob =?= True'*
- *condor_q_edit*
  - edit job attributes
- *condor_status*
  - get information of the machines in the system
  - status of the WNs per default
- *condor_userprio*
  - shows user prio per accounting group
  - *-allusers* shows prio of all users

Job Information

- HTCondor works with information in form of ClassAds
- ClassAds are key value/expression pairs e.g.
  - *Owner = "mschnepf"*
  - *START = TARGET.Owner=?="mschnepf"*
  - *requirements = (TARGET.Memory >= RequestMemory)*
- every instance in HTCondor has ClassAds: Jobs, WNs, scheduler,...
  - get ClassAds of jobs via *condor_q*/*condor_history*
  - get ClassAds of daemons via *condor_status*
  - *-l* shows all ClassAds of the instance
  - *-af ClassAd* shows only the ClassAd value/expression
- important ClassAds
  - *requirements*: requirements of the job to slot
  - *CloudSite*: group of worker nodes with similar setup
  - *HoldReason*: reason why your job is on hold
  - *RemoteUserCPU*: amount of CPU time the job used on a worker node
- ClassAd syntax
  - keys are not case sensitive, only strings in quotes are case sensitive
  - values can be *undefined*
  - is equal (*=?=*) returns only false or true while == can return false, true, undefined
  - is not equal (*=!=*)
  - and (*&&*), or (||)

Submit a hand full of Jobs

- several jobs with the same executable but different arguments, e.g., input file or seed
- queue command enables to submit a cluster of jobs
  - each job has the same *ClusterID*
  - each job has another *ProcID*
  - Job ID is *ClusterID.ProcID*
  - *ClusterID* and *ProcID* can be used in submit file

```
Universe = container
container_image = /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cverstege/alma9-gridjob:latest


executable = ./executable.sh
arguments = $(ProcID)

request_cpus = 1
request_memory = 2000
request_disk = 500000

log = log_$(ProcID).log
stdout = stdout_$(ProcID).log
stderr = stderr_$(ProcID).log

accounting_group=belle

queue 5
```

Submit a hand full of Jobs

- several jobs with the same executable but different arguments, e.g., input file or seed
- queue command enables to submit a cluster of jobs
  - each job has the same *ClusterID*
  - each job has another *ProcID*
  - Job ID is *ClusterID.ProcID*
  - *ClusterID* and *ProcID* can be used in submit file
  - can read variables from file; one line per job

```
Universe = container
container_image = /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cverstege/alma9-gridjob:latest

executable = ./executable.sh

request_cpus = 1
request_memory = 2000
request_disk = 500000

log = log_$(ProcID).log
stdout = stdout_$(ProcID).log
stderr = stderr_$(ProcID).log

accounting_group=belle

queue arguments from input.csv
```

Interactive Job

- request an interactive job
  - get a job slot for interactive use (no other program is running) with resources set in the submit file
  - add *-interactive* to your submit command
  - ends when you disconnect, e.g., via *exit*
- jump into your job with *condor_ssh_to_job JOBID*
  - see your running processes
  - access to files of your job
  - able to copy files from you job
  - when jobs end, you get kicked out

generated with Microsoft Copilot

Hold State

- you set the job to hold
- your job tried something and failed
  - used more memory/disk than requested
  - restarted more than 5 time
  - log directory does not exist
  - output file of job was not created and can not copy to submit host
- hold jobs are removed from the WN, unsafed data are lost

Stop! Polizei! von iStock

Hold State

- you set the job to hold
- your job tried something and failed
  - used more memory/disk than requested
  - restarted more than 5 time
  - log directory does not exist
  - output file of job was not created and can not copy to submit host
- hold jobs are removed from the WN, unsafed data are lost
- fix your job
  - get hold reason: *condor_q JOBID -hold*
  - adjust ClassAds: *condor_qedit JOBID CLASSAD NEW_VALUE*
  - release job: *condor_release JOBID*

Stop! Polizei! von iStock

Hold State

- you set the job to hold
- your job tried something and failed
  - used more memory/disk than requested
  - restarted more than 5 time
  - log directory does not exist
  - output file of job was not created and can not copy to submit host
- hold jobs are removed from the WN, unsafed data are lost
- fix your job
  - get hold reason: *condor_q JOBID -hold*
  - adjust ClassAds: *condor_qedit JOBID CLASSAD NEW_VALUE*
  - release job: *condor_release JOBID*
- You can release a job up to 5 times. After that you have to adjust *JobRunCount* via *condor_qedit*

Stop! Polizei! von iStock

Memory Usage

- keep your memory usage low

Memory Usage

- keep your memory usage low
- job is on hold (job did not finish)

Memory Usage

- keep your memory usage low
- job is on hold (job did not finish)
- get reason: *condor_q JOBID -hold*

Memory Usage

- keep your memory usage low
- job is on hold (job did not finish)
- get reason: *condor_q JOBID -hold*
  - HoldReason: *Error from . . . : Docker job has gone over memory limit of 2048 Mb*

bradhouse.dev - How to handle Out of Memory (OOM) in C - 2023

Memory Usage

- keep your memory usage low
- job is on hold (job did not finish)
- get reason: *condor_q JOBID -hold*
  - HoldReason: *Error from . . . : Docker job has gone over memory limit of 2048 Mb*
  - adjust memory request: *condor_qedit JOBID RequestMemory 3000*
  - release job: *condor_release JOBID*

bradhouse.dev - How to handle Out of Memory (OOM) in C - 2023

Memory Usage

- keep your memory usage low
- job is on hold (job did not finish)
- get reason: *condor_q JOBID -hold*
  - HoldReason: *Error from . . . : Docker job has gone over memory limit of 2048 Mb*
  - adjust memory request: *condor_qedit JOBID RequestMemory 3000*
  - release job: *condor_release JOBID*
- automated adjustment in JDL file
  - memory increase by each start: *request_memory = 2000+(1000*NumJobStarts)*
  - automated release after run into memory limit: *periodic_release = (HoldReasonCode == 34)*

bradhouse.dev - How to handle Out of Memory (OOM) in C - 2023

Memory Usage

- keep your memory usage low
- job is on hold (job did not finish)
- get reason: *condor_q JOBID -hold*
  - HoldReason: *Error from . . . : Docker job has gone over memory limit of 2048 Mb*
  - adjust memory request: *condor_qedit JOBID RequestMemory 3000*
  - release job: *condor_release JOBID*
- automated adjustment in JDL file
  - memory increase by each start: *request_memory = 2000+(1000*NumJobStarts)*
  - automated release after run into memory limit: *periodic_release = (HoldReasonCode == 34)*
- more memory ⇒ less resources

bradhouse.dev - How to handle Out of Memory (OOM) in C - 2023

Why does my job not run?

- batch systems has to be full ;-)
- usually jobs has to wait a few minutes before start
- monitoring website

Why does my job not run?

- batch systems has to be full ;-)
- usually jobs has to wait a few minutes before start
- monitoring website
- *condor_q -better-analyse JOBID* provides information about the job-resource matchmaking

Why does my job not run?

- batch systems has to be full ;-)
- usually jobs has to wait a few minutes before start
- monitoring website
- *condor_q -better-analyse JOBID* provides information about the job-resource matchmaking
- see available resources via *condor_status -constraint 'partitionableSlot' -af:t Name CPUs Memory Disk GPUs*

Why does my job not run?

- batch systems has to be full ;-)
- usually jobs has to wait a few minutes before start
- monitoring website
- *condor_q -better-analyse JOBID* provides information about the job-resource matchmaking
- see available resources via *condor_status -constraint 'partitionableSlot' -af:t Name CPUs Memory Disk GPUs*
- match against a specific machine *condor_q -better-analyse JOBID -reverse -machine MACHINE_NAME*

External Resources

- ETP has access to external computing resources via its batch system
  - **T**hroughput **Op**timized **A**nalysis **S**ystem (TOpAS) at GridKa
    - use *condor_status -pool f03-001-140-e.gridka.de* to for TOpAS
  - BWForCluster NEMO2 at Uni. of Freiburg
    - create a NEMO2 account and add an acknowledgment in your thesis, see https://wiki.etp.kit.edu/books/computing-and-infrastructure/page/nemo-cluster
- external resources are only available with *+RemoteJob = True* in the JDL file
- */home*, */work*, or */ceph* from ETP are not mounted on external resources

$\Rightarrow$ transfer data via HTCondor ($<$ 10 MB per job) or Grid protocols (XRootD, WebDAV, . . .)

Transfer Data via HTCondor

- transfers via htcondor runs through the scheduler $\Rightarrow$ inefficient and extra load on portal machines

Transfer Data via HTCondor

- transfers via htcondor runs through the scheduler ⇒ inefficient and extra load on portal machines
- transfer files to the WN: *transfer_input_files*
- transfer files from the WN at the end of the job: *transfer_output_files = output.root*
- rename output files: *transfer_output_remaps = ”output.root = output/output_$(Process).root”*

```
Universe = container
container_image = /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cverstege/alma9-gridjob:latest

executable = ./executable.sh

request_cpus = 1
request_memory = 2000
request_disk = 500000

transfer_input_files = /work/$(whoami)/HTCondor_Tutorial/01_simple_submit/mc_example.py

# DO NOT DO THAT FOR FILES BIGGER THAN 10MB
transfer_output_files = output.root
transfer_output_remaps = "output.root = output_$(ProcID).root"


log = log_$(ProcID).log
stdout = stdout_$(ProcID).log
stderr = stderr_$(ProcID).log

accounting_group=belle

queue
```

Data intensive Jobs

- jobs with more than 2 GB input files per 30 min runtime are data intensive

Data intensive Jobs

- jobs with more than 2 GB input files per 30 min runtime are data intensive
- write on local storage on the WN and copy the results at the end of the jobs

Data intensive Jobs

- jobs with more than 2 GB input files per 30 min runtime are data intensive
- write on local storage on the WN and copy the results at the end of the jobs
- for some Grid storage requires a VOMS (**V**irtual **O**rganisation **M**embership **S**ervice) proxy
  - location of the voms-proxy usually at */tmp/x509up_u$(id -u)*
  - location can be set via environment variable *$X509_USER_PROXY*
  - HTCondor copies your proxy to the WN and set *$X509_USER_PROXY*

Data intensive Jobs

- jobs with more than 2 GB input files per 30 min runtime are data intensive
- write on local storage on the WN and copy the results at the end of the jobs
- for some Grid storage requires a VOMS (**V**irtual **O**rganisation **M**embership **S**ervice) proxy
  - location of the voms-proxy usually at */tmp/x509up_u$(id -u)*
  - location can be set via environment variable *$X509_USER_PROXY*
  - HTCondor copies your proxy to the WN and set *$X509_USER_PROXY*

```
Universe = container
container_image = /cvmfs/unpacked.cern.ch/registry.hub.docker.com/cverstege/alma

executable = ./executable.sh

request_cpus = 1
request_memory = 2000
request_disk = 500000

transfer_input_files = /work/$(whoami)/HTCondor_Tutorial/01_simple_submit/mc_ex

# DO NOT DO THAT FOR FILES BIGGER THAN 10MB
transfer_output_files = output.root
transfer_output_remaps = "output.root = output_$(ProcID).root"


log = log_$(ProcID).log
stdout = stdout_$(ProcID).log
stderr = stderr_$(ProcID).log

accounting_group=belle

queue
```

Transfer Data via XRootD

- files can be copied, streamed and written via XRootD
- */ceph/srv*
  - only readable from ETP and TOpAS
  - via *root://ceph-node-j.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-j.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*

Transfer Data via XRootD

- files can be copied, streamed and written via XRootD
- */ceph/srv*
  - only readable from ETP and TOpAS
  - via *root://ceph-node-j.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-j.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*
- NRG at GridKa
  - CMS *root://cmsdcache-kit-disk.gridka.de:1094//store/user/<user>*
  - Belle *root://dcache-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/LOCAL/user/<user>*
  - voms proxy required

Transfer Data via XRootD

- files can be copied, streamed and written via XRootD
- */ceph/srv*
  - only readable from ETP and TOpAS
  - via *root://ceph-node-j.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-j.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*
- NRG at GridKa
  - CMS *root://cmsdcache-kit-disk.gridka.de:1094//store/user/<user>*
  - Belle *root://dcache-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/LOCAL/user/<user>*
  - voms proxy required
- TOpAS and other worker nodes can cache files via XRootD
  - enables fast processing
  - files are identified by filename e.g. */mschnepf/testfile.txt*

  $\Rightarrow$ do not overwrite files, the cache does not update

Transfer Data via XRootD

- files can be copied, streamed and written via XRootD
- */ceph/srv*
  - only readable from ETP and TOpAS
  - via *root://ceph-node-j.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-j.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*
- NRG at GridKa
  - CMS *root://cmsdcache-kit-disk.gridka.de:1094//store/user/<user>*
  - Belle *root://dcache-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/LOCAL/user/<user>*
  - voms proxy required
- TOpAS and other worker nodes can cache files via XRootD
  - enables fast processing
  - files are identified by filename e.g. */mschnepf/testfile.txt*

  $\Rightarrow$ do not overwrite files, the cache does not update
- most root installations can open files from remote
  e.g. *myroot.py root://ceph-node-j.etp.kit.edu://mschnepf/test_rootfile.txt*

Send Software Sandbox via Grid Storage

- code can run outside of ETP
  - no absolute paths
  - */work*, */ceph*, */home* is not available
- pack software into one tarball-file (sandbox)
- copy sandbox to Grid storage
- job downloads and unpacks sandbox
- some software needs to linked at new location, e.g. CMSSW

```
##### WN informaton
echo -e "Hostname : $(hostname)"
echo -e "user : $(whoami)"
echo -e "spawndir : $(pwd)"

##### setup environment
## download sandbox
xrdcp root://dcache-kit.gridka.de:1094//pnfs/gridka.de/belle/disk-
tar -xf sandbox_test.tar
## setup sandbox software
source /cvmfs/cms.cern.ch/cmsset_default.sh
cd CMSSW_15_0_10_patch3/src
scram b ProjectRename
eval `scramv1 runtime -sh`

##### start workload at spawndir and create output.root with resul
cd ${spawndir}
which combine

##### copy output from WN
#gfal-copy output.root
root://dcache-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/L
```

GPUs

- only TOpAS provides GPUs $\Rightarrow$ *+RemoteJob=True*
- use the docker image (*container_image = docker://...*)
- request a GPU via *RequestGPU=1* in the JDL
- check if your job can run at TOpAS: *condor_q -better-analyse -pool f03-001-140-e.gridka.de JOBID*
- CUDA is not provided by the system

GPUs

- only TOpAS provides GPUs $\Rightarrow$ *+RemoteJob=True*
- use the docker image (*container_image = docker://...*)
- request a GPU via *RequestGPU=1* in the JDL
- check if your job can run at TOpAS: *condor_q -better-analyse -pool f03-001-140-e.gridka.de JOBID*
- CUDA is not provided by the system
- setup environment
  - source an existing environment, e.g., CMSSW / BASF2 / LCG-Stack
  - sandbox via Grid storage
  - setup your environment via pip or conda
  - build your own container and publish it on docker hub and CVMFS-unpacked

```
#!/bin/bash

echo "hostname $(hostname)"

# Source LCG-Stack (some of them do not work)
source /cvmfs/sft.cern.ch/lcg/views/LCG_107_cuda/x86_64-el9-gcc11
    -opt/setup.sh

env
python3 mnist_training.py

ls -lh
```

GPUs

- only TOpAS provides GPUs $\Rightarrow$ *+RemoteJob=True*
- use the docker image (*container_image = docker://...*)
- request a GPU via *RequestGPU=1* in the JDL
- check if your job can run at TOpAS: *condor_q -better-analyse -pool f03-001-140-e.gridka.de JOBID*
- CUDA is not provided by the system
- setup environment
  - source an existing environment, e.g., CMSSW / BASF2 / LCG-Stack
  - sandbox via Grid storage
  - setup your environment via pip or conda
  - build your own container and publish it on docker hub and CVMFS-unpacked
- request parts of a GPU via *+RequestGPUMemoryMB = 2000*
  - GPU processing power is still shared
  - works only with CUDA software
- DO NOT SET *NVIDIA_VISIBLE_DEVICES*; HTCondor does it

```
#!/bin/bash

echo "hostname $(hostname)"

# Source LCG-Stack (some of them do not work)
source /cvmfs/sft.cern.ch/lcg/views/LCG_107_cuda/x86_64-el9-gcc11
    -opt/setup.sh

env
python3 mnist_training.py

ls -lh
```

Further Information

- ETP Wiki: HTCondor at ETP
- ETP Batch System monitoring
- HTCondor youtube user tutorials
- ReadTheDocs
- ETP mattermost: etp-htcondor
- ETP mattermost: etp-htcondor-gpus

https://www.clker.com/clipart-753080.html

- batch admin
- computing work
  - Bachelorthesis
  - Masterthesis
  - PhD Topics
  - Hiwi

ETP Institute of Experimental Particle Physics

SCC Scientific Computing Center

Backup

CMS Example Analysis Workflow (2017 data)