Karlsruhe Institute of Technology

Institut für Experimentelle Teilchenphysik

Steinbuch Centre for Computing

ETP Batch System Tutorial

Matthias J. Schnepf | 18. August 2022

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 at 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 600 TB 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
    - end of year 2022: CMS: 2 PB
    - currently no storage for Belle II; but end of year 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
    - ...

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-06-00-03

##### start workload (results stored in output.root)
basf2 /ceph/$(whoami)/htcondor/tutorial_ETP/mc_example.py

##### copy output
cp output.root /ceph/$(whoami)/htcondor/tutorial_ETP/01_simple_submit.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

```
#!/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-06-00-03

##### start workload (results stored in output.root)
basf2 /ceph/$(whoami)/htcondor/tutorial_ETP/mc_example.py

##### copy output
cp output.root /ceph/$(whoami)/htcondor/tutorial_ETP/01_simple_submit.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
- develop portable code
- job runtime should be longer than 20 min
- output file size bigger than 1 GB

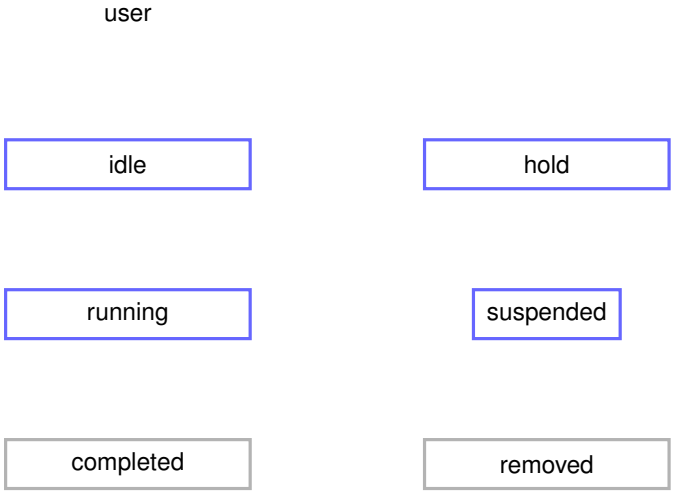
```
#!/bin/bash

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

##### setup environment
source /cvmfs/belle.cern.ch/tools/b2setup release-06-00-03

##### start workload (results stored in output.root)
basf2 /ceph/$(whoami)/htcondor/tutorial_ETP/mc_example.py

##### copy output
cp output.root /ceph/$(whoami)/htcondor/tutorial_ETP/01_simple_submit.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
    - belle (25%)
    - cms (75%)
  - submit (queue)
- submit with *condor_submit submit_file.jdl*

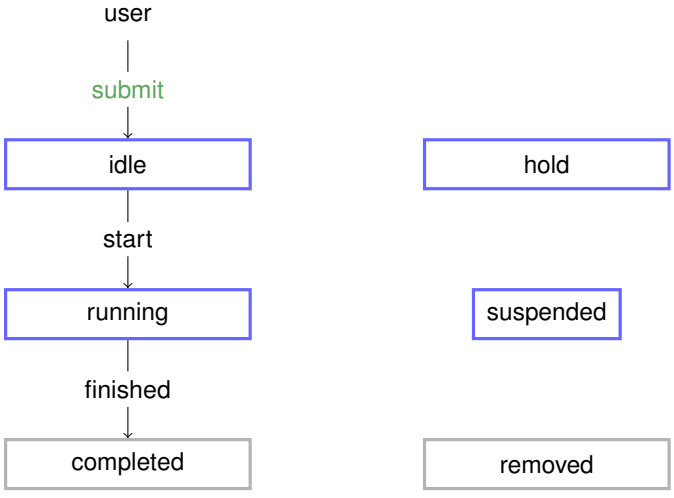
```
Universe = docker
docker_image = mschnepf/slc7-condocker

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



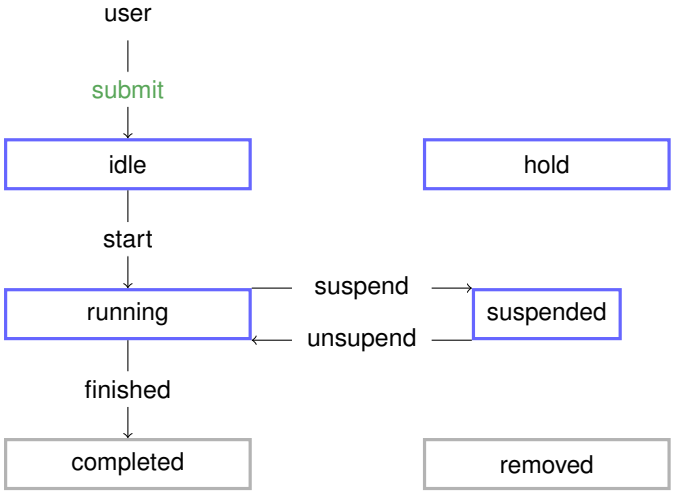
- only idle, running, and hold jobs can be changed

Job Status



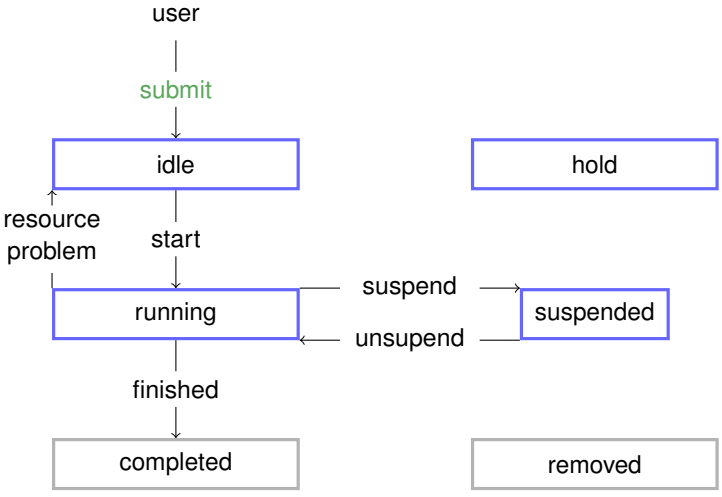
- only idle, running, and hold jobs can be changed
- green action can be done by users
- hold jobs are removed from the WN, unsaved results are gone

Job Status



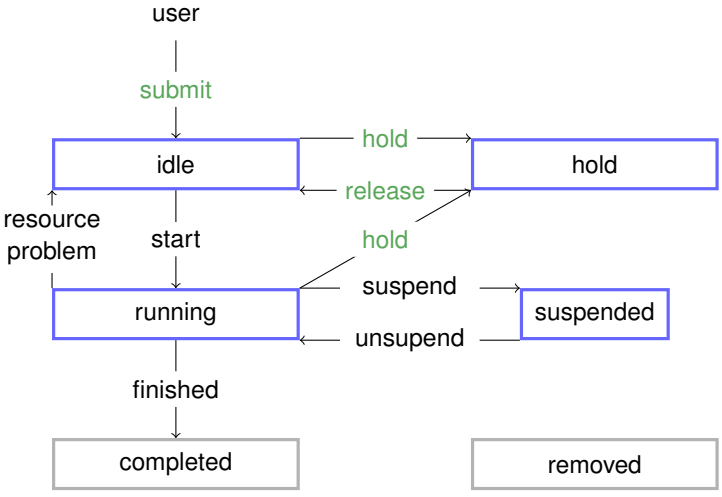
- only idle, running, and hold jobs can be changed
- green action can be done by users
- hold jobs are removed from the WN, unsaved results are gone

Job Status



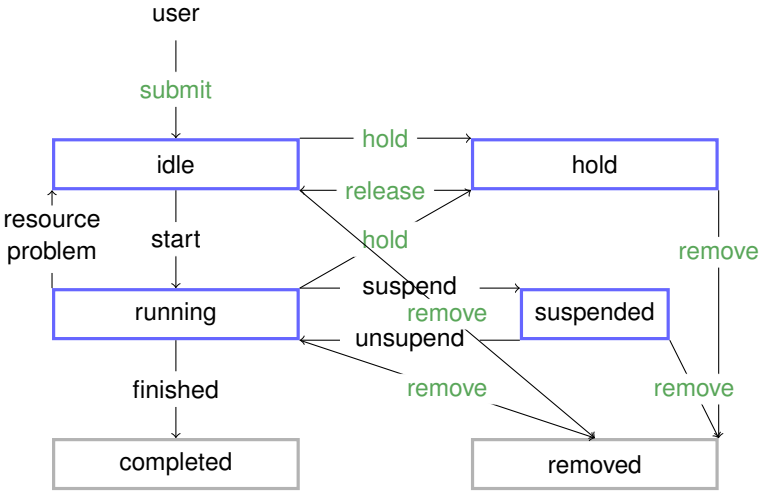
- only idle, running, and hold jobs can be changed
- green action can be done by users
- hold jobs are removed from the WN, unsaved results are gone

Job Status



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

Job Status



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

Useful Condor Commands

- *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_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_edit*
  - edit job attributes

Useful Condor Commands

- *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_edit*
  - edit job attributes

- *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_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_edit*
  - edit job attributes

- *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_submit*
  - submit job to queue of the batch system

Useful Condor Commands

- *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_edit*
  - edit job attributes

- *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_submit*
  - submit job to queue of the batch system
- *condor_status*
  - get information of the machines in the system
  - status of the WNs per default

Useful Condor Commands

- *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_edit*
  - edit job attributes

- *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_submit*
  - submit job to queue of the batch system
- *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 resources
  - *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 (*=!=*)

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 = docker
docker_image = mschnepf/slc7-condocker

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: Arguments from File

- queue command can read variables from file

```
Universe = docker
docker_image = mschnepf/slc7-condocker

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
```

Memory Usage

- try to keep your memory usage low

Memory Usage

- try to keep your memory usage low
- job is on hold (job did not finished)

Memory Usage

- try to keep your memory usage low
- job is on hold (job did not finished)
- reason: *condor_q JOBID -hold*

Memory Usage

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

Memory Usage

- try to keep your memory usage low
- job is on hold (job did not finished)
- 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*

Memory Usage

- try to keep your memory usage low
- job is on hold (job did not finished)
- 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)*

Memory Usage

- try to keep your memory usage low
- job is on hold (job did not finished)
- 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 $\Rightarrow$ less resources

Memory Usage

- try to keep your memory usage low
- job is on hold (job did not finished)
- 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 $\Rightarrow$ less resources
- You can release a job up to 5 times. After that you have to adjust *JobRunCount* via *condor_qedit*

Can My Job run?

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

Can My Job run?

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

External Resources

- ETP has access to external computing resources via its batch system
  - **T**hroughput **Op**timized **A**nalysis **S**ystem (TOpAS) at GridKa
  - BWForCluster NEMO at Uni. of Freiburg
- external resources are available with *+RemoteJob = True* in the JDL file
- external computing resources have not mounted */home*, */work*, or */ceph* from ETP

$\Rightarrow$ transfer data via HTCondor (small files) or Grid protocols (SRM, XRootD,...)

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 $\Rightarrow$ 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 = docker
docker_image = mschnepf/slc7-condocker

executable = ./executable.sh

request_cpus = 1
request_memory = 2000
request_disk = 500000

transfer_input_files = /ceph/mschnepf/htcondor/tutorial_ETP/mc_example.py


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
- please select systems that support IO-intensive jobs
  - add *requirements = (TARGET.ProvidesIO =?= True)* in your JDL

Data intensive Jobs

- jobs with more than 2 GB input files per 30 min runtime are data intensive
- please select systems that support IO-intensive jobs
  - add *requirements = (TARGET.ProvidesIO =?= True)* in your JDL
- 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
- please select systems that support IO-intensive jobs
  - add *requirements = (TARGET.ProvidesIO =?= True)* in your JDL
- 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/x509_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 = docker
docker_image = mschnepf/slc7-condocker

executable = ./executable.sh

request_cpus = 1
request_memory = 2000
request_disk = 500000

transfer_input_files = my_python_script.py

transfer_output_files = ouput.txt


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

accounting_group=belle

requirements = TARGET.ProvidesIO=?=True
voms_proxy = /tmp/x509_u12089

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-a.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-a.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-a.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-a.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*
- NRG at GridKa
  - CMS *root://cmsxrootd-redirectors.gridka.de:1094//store/user/<user>*
  - Belle *root://dcachexrootd-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/LOCAL/<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-a.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-a.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*
- NRG at GridKa
  - CMS *root://cmsxrootd-redirectors.gridka.de:1094//store/user/<user>*
  - Belle *root://dcachexrootd-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/LOCAL/<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-a.etp.kit.edu://<user>*
    e.g. *xrdcp root://ceph-node-a.etp.kit.edu://mschnepf/testfile.txt file:///tmp/testfile.txt*
- NRG at GridKa
  - CMS *root://cmsxrootd-redirectors.gridka.de:1094//store/user/<user>*
  - Belle *root://dcachexrootd-kit.gridka.de:1094://pnfs/gridka.de/belle/disk-only/LOCAL/<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-a.etp.kit.edu://mschnepf/test_rootfile.txt*

GPUs

- only TOpAS provides GPUs $\Rightarrow$ *+RemoteJob=True*
- Request a GPU via *RequestGPU=1* in the JDL
- check if your job can run at TOpAS use: *condor_q -better-analyse -pool f03-001-140-e.gridka.de JOBID*

GPUs

- only TOpAS provides GPUs $\Rightarrow$ *+RemoteJob=True*
- Request a GPU via *RequestGPU=1* in the JDL
- check if your job can run at TOpAS use: *condor_q -better-analyse -pool f03-001-140-e.gridka.de JOBID*
- setup environment
  - choose docker container with cuda
  - source a basic environment, e.g., CMSSW / BASF2
  - setup your environment via pip or conda (in backup)
  - send and install your additional software into your python environment

```
#!/bin/bash

# Source base environment
source /cvmfs/sft.cern.ch/lcg/views/LCG_100cuda/x86_64-
       centos7-gcc8-opt/setup.sh

# create own python environment
python3 -m venv venv
source venv/bin/activate
python3 -m pip install pip --upgrade


# install torch 1.9.0
python3 -m pip install torch==1.9.0+cu111 torchvision
       ==0.10.0+cu111 torchaudio===0.9.0 -f https://
       download.pytorch.org/whl/torch_stable.html

# install older torch version
#python3 -m pip install torch torchvision torchaudio

env
python3 mnist_training.py

ls -lh
```

Submission Tools

- jdlcreator
  - build argument list and submit jobs in python
  - \+ simple
  - \+ no job management or environment support
- grid control
  - submit and job management tool
  - \+ supports different batch systems
  - \+ supports complex argument list
  - \+ jobs handling with automatic resubmission
  - \+ support CMSSW (transportation and setup)
- b2luigi
  - workflow management
    - \+ complex argument lists are possible
    - \+ manage dependencies of jobs
    - \+ automated resubmission
  - \+ support different batch systems and gbasf2
  - \- problem with external resources
- luigi + law
  - workflow management
    - \+ complex argument lists are possible
    - \+ manage dependencies of jobs
    - \+ automated resubmission
  - \+ support different batch systems
  - \+ support Grid transfer protocols
  - \- complex

Further Information

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

Backup

GPUs (Conda)

- only TOpAS provides GPUs $\Rightarrow$ *+RemoteJob=True*
- Request a GPU via *RequestGPU=1* in the JDL
- check if your job can run at TOpAS use: *condor_q -better-analyse -pool f03-001-140-e.gridka.de JOBID*
- setup environment
  - choose docker container with cuda
  - install miniconda
  - setup your conda environment
  - send and install your additional software into your python environment

```
#!/bin/bash

# Source base environment
wget https://repo.anaconda.com/miniconda/Miniconda3-
        latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh -b -p $(pwd)/
        miniconda
source miniconda/bin/activate

# Install torch
conda install pytorch torchvision torchaudio cudatoolkit
        =11.3 -c pytorch

env
python3 mnist_training.py

ls -lh
```

CMS Example Analysis Workflow (2017 data)