Introduction to HTCondor

GPU Cluster

Faculty MI

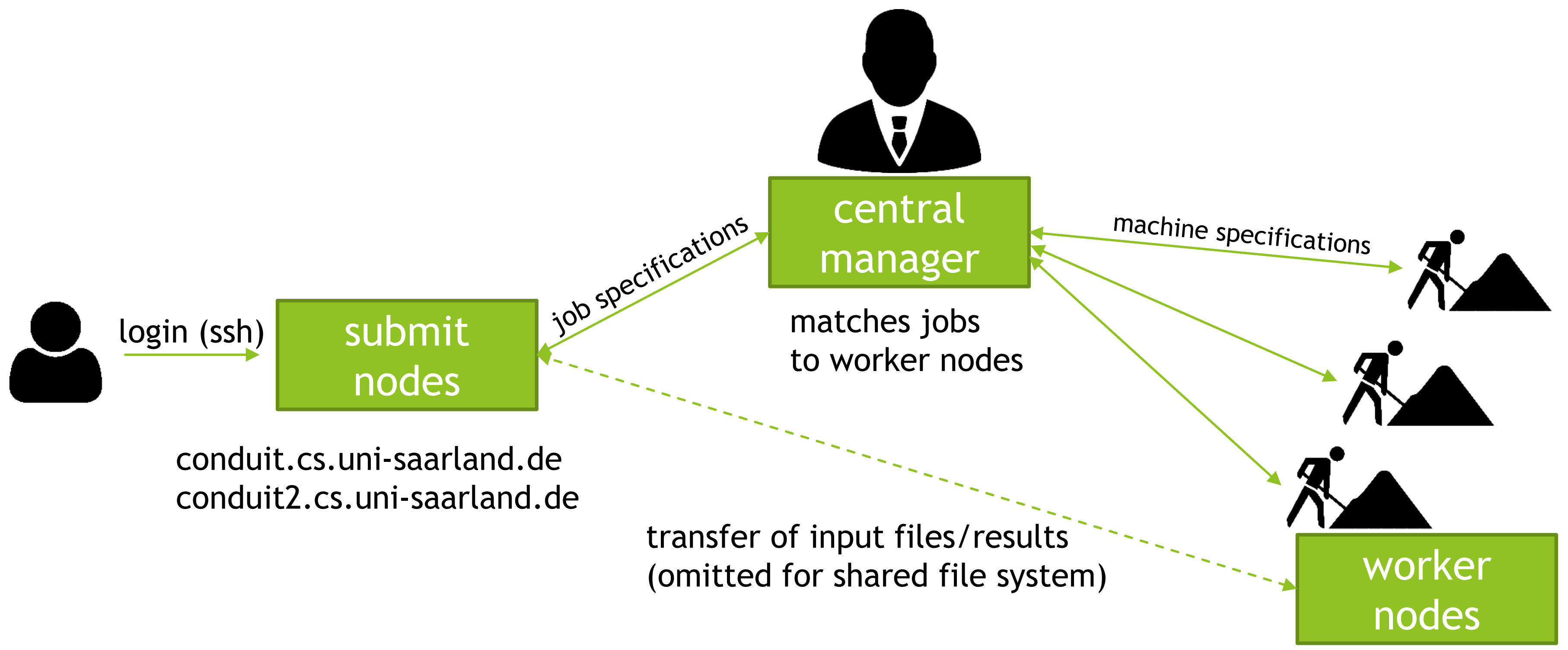
Overview Infrastructure

		CPU	GPU	RAM
Pascal PCIe System	**1**	**2** x Intel Xeon E5-2698 v4 20 cores [2.2 GHz]: **40 cores**	**8** x Nvidia Pascal P100 PCIe Double-Precision 8 x 4.7 = 37.6 teraflops Single-Precision 8 x 9.3 = 74.4 teraflops Half-Precision 8 x 18.7 = 149.6 teraflops	each x 512 GB
Pascal SXM System	**2**	**2** x Intel Xeon E5-2690 v4 28 cores [2.6 GHz]: **56 cores**	**8** x Nvidia Pascal P100 SXM Double-Precision 16 x 5.3 = 84.8 teraflops Single-Precision 16 x 10.6 = 169.6 teraflops Half-Precision 16 x 21.2 = 339.2 teraflops	each x 768 GB
Pascal SXM System	**2**	**2** x Intel Xeon E5-2698 v4 20 cores [2.2 GHz]: **80 cores**	**8** x Nvidia Pascal P100 SXM Double-Precision 16 x 5.3 = 84.8 teraflops Single-Precision 16 x 10.6 = 169.6 teraflops Half-Precision 16 x 21.2 = 339.2 teraflops	each x 512 GB
NVidia DGX-1 Volta SXM System	**2**	**2** x Intel Xeon E5-2698 v4 20 cores [2.2 GHz]: **80 cores**	**8** x Nvidia Volta V100 SXM Double-Precision 16 x 7.8 = 124.8 teraflops Single-Precision 16 x 15.7 = 249.6 teraflops Deep-Learning 16 x 125 = 2000 teraflops	each x 512 GB
NVidia A100 Ampere SXM System	**2**	**2** x AMD EPYC 7742 64 cores [2.2 GHz]: **256 cores**	**8** x Nvidia Ampere SXM Double-Precision 16 x 9.7 = 155.2 teraflops Single-Precision 16 x 19.5 = 312 teraflops Deep-Learning 16 x 312 = 4992 teraflops	each x 1 TB

What is HTCondor?

- Software that runs and schedules tasks on computers
- Users can submit tasks to a queue from submit nodes
- HTCondor schedules them to run worker nodes
- Worker nodes can have different underlying architectures, e.g. different numbers and types of CPUs, availability of GPUs or not, different RAM size, etc.
- Users must define the requirements for their job (e.g. number of GPUs, needed RAM, …) using a submit file

Overview submit process


login (ssh)
submit nodes
job specifications
central manager
machine specifications
matches jobs to worker nodes
worker nodes
conduit.cs.uni-saarland.de
conduit2.cs.uni-saarland.de
transfer of input files/results (omitted for shared file system)

users of the GPU cluster can only login to the submit nodes

available CPU/GPU worker nodes

	GPU Type	# GPUs	CPU Type	# CPU Cores (incl. Hyper-Threading)	RAM
thor.cs.uni-saarland.de (DGX-1)	Tesla V100-SXM	8	Intel Xeon CPU E5-2698 v4	80	512 GB
loki.cs.uni-saarland.de (DGX-1)	Tesla V100-SXM	8	Intel Xeon CPU E5-2698 v4	80	512 GB
lofn.cs.uni-saarland.de	Tesla P100-SXM	8	Intel Xeon CPU E5-2698 v4	80	512 GB
idun.cs.uni-saarland.de	Tesla P100-SXM	8	Intel Xeon CPU E5-2698 v4	80	512 GB
uller.cs.uni-saarland.de	Tesla P100-PCIE	8	Intel Xeon CPU E5-2698 v4	80	512 GB
tenos.cs.uni-saarland.de	Tesla P100-SXM	8	Intel Xeon CPU E5-2690 v4	56	768 GB
thera.cs.uni-saarland.de	Tesla P100-SXM	7	Intel Xeon CPU E5-2690 v4	56	768 GB
urd.cs.uni-saarland.de (DGX A100)	A-100-SXM	8	AMD EPYC 7742	256	1 TB
vidar.cs.uni-saarland.de (DGX A100)	A-100-SXM	8	AMD EPYC 7742	256	1 TB

Submitting a Job to the Cluster

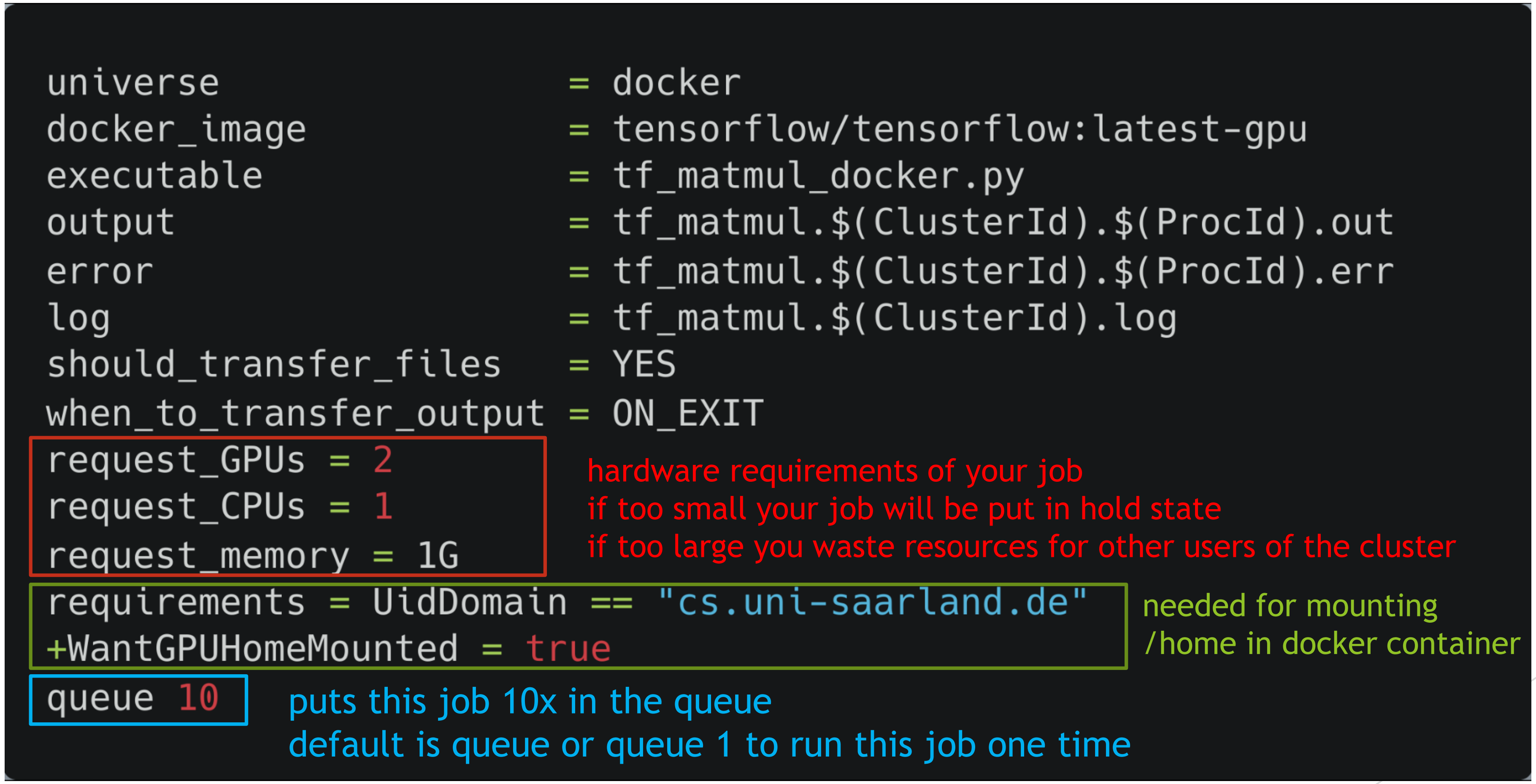
- login on **`{conduit,conduit2}.cs.uni-saarland.de`** with your SIC LDAP credentials using ssh (only reachable from within the university network!)
- you will find a folder **`condor_tutorial`** with some example submit files in your home directory
- **`/home`** is an NFS mount and shared between all nodes in the cluster
- this means that large files don’t have to be copied between submit and worker nodes several times
- we only allow execution of jobs that use the HTCondor docker universe
- when you submit your job, condor will check if a worker node can be matched to run your job
- your job will stay idle as long as there is no worker available for running the job
- your job will be held if something obvious (e.g. missing input files) would prevent it from running successfully

Submitting a Job to the Cluster

- if your job enters the state running, condor creates a job specific scratch directory on the worker node that runs your job
- the scratch directory is mounted in the docker container that runs your job
- you can also prompt condor to mount the `/home` NFS mount into the container with a special flag in the submit file
- Jobs in the docker container are run as your user so that you have access to the files in the NFS share with your UID/GID
- this also means that you have no superuser privileges
- after the job is completed, condor will automatically clean up (destroy the docker container, remove the scratch directory)
- result files are either transferred back to the directory from where you submitted the job or when using the shared filesystem will be in your home where you directed the output

Example Submit File

file: **`$HOME/condor_tutorial/tf_matmul_docker.sub`**



```
universe                = docker
docker_image            = tensorflow/tensorflow:latest-gpu
executable              = tf_matmul_docker.py
output                  = tf_matmul.$(ClusterId).$(ProcId).out
error                   = tf_matmul.$(ClusterId).$(ProcId).err
log                     = tf_matmul.$(ClusterId).log
should_transfer_files   = YES
when_to_transfer_output = ON_EXIT
request_GPUs = 2
request_CPUs = 1
request_memory = 1G
requirements = UidDomain == "cs.uni-saarland.de"
+WantGPUHomeMounted = true
queue 10
```

hardware requirements of your job
if too small your job will be put in hold state
if too large you waste resources for other users of the cluster

needed for mounting
/home in docker container

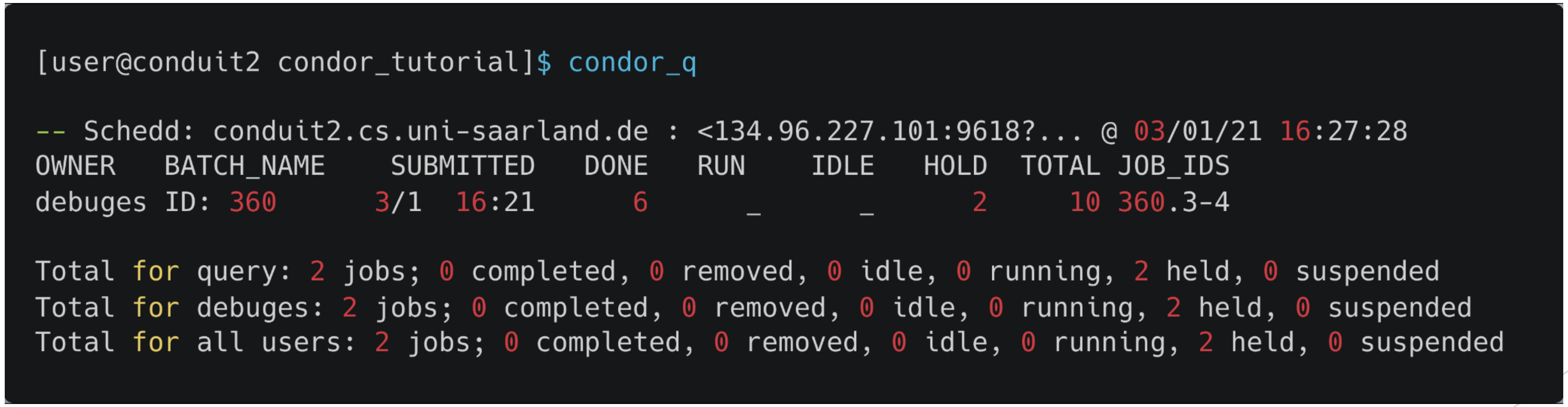
puts this job 10x in the queue
default is queue or queue 1 to run this job one time

Submit File

- universe must be "docker" or your job will never be run on any worker node
- you can use any available docker image that are freely accessible from docker hub, nvidia, etc.
- executable is the script file that will be run upon starting the container
  - if it is a relative path (as in the example) it will be copied by condor to the scratch directory and mounted in the docker container
  - if it is an absolute path, condor is taking it as a path to a file within the container
  - can be left out/commented out if your image defines an entrypoint that runs by default
- output, error, log will contain the messages from stdout, stderr, and the condor log messages for your job
- submit your job to the queue:
  `condor_submit <my_submit_file.sub>`

The condor queue

- **condor_q** allows you to watch the state of your jobs
- idle jobs wait for free resources



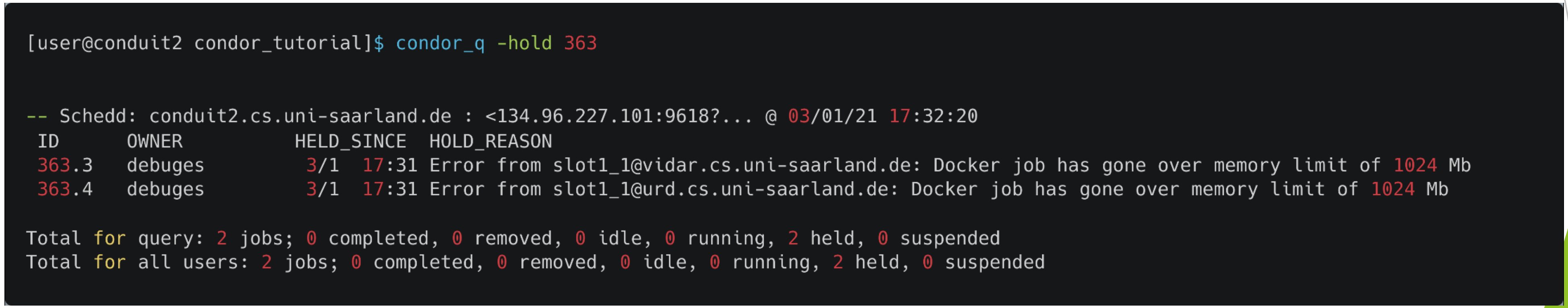
```
[user@conduit2 condor_tutorial]$ condor_q

-- Schedd: conduit2.cs.uni-saarland.de : <134.96.227.101:9618?... @ 03/01/21 16:27:28
OWNER   BATCH_NAME    SUBMITTED   DONE   RUN    IDLE   HOLD  TOTAL JOB_IDS
debuges ID: 360      3/1  16:21      6      _      _      2     10 360.3-4

Total for query: 2 jobs; 0 completed, 0 removed, 0 idle, 0 running, 2 held, 0 suspended
Total for debuges: 2 jobs; 0 completed, 0 removed, 0 idle, 0 running, 2 held, 0 suspended
Total for all users: 2 jobs; 0 completed, 0 removed, 0 idle, 0 running, 2 held, 0 suspended
```

The condor queue

- if a job is put into hold, something was wrong: check it with **`condor_q -hold <jobid>`**

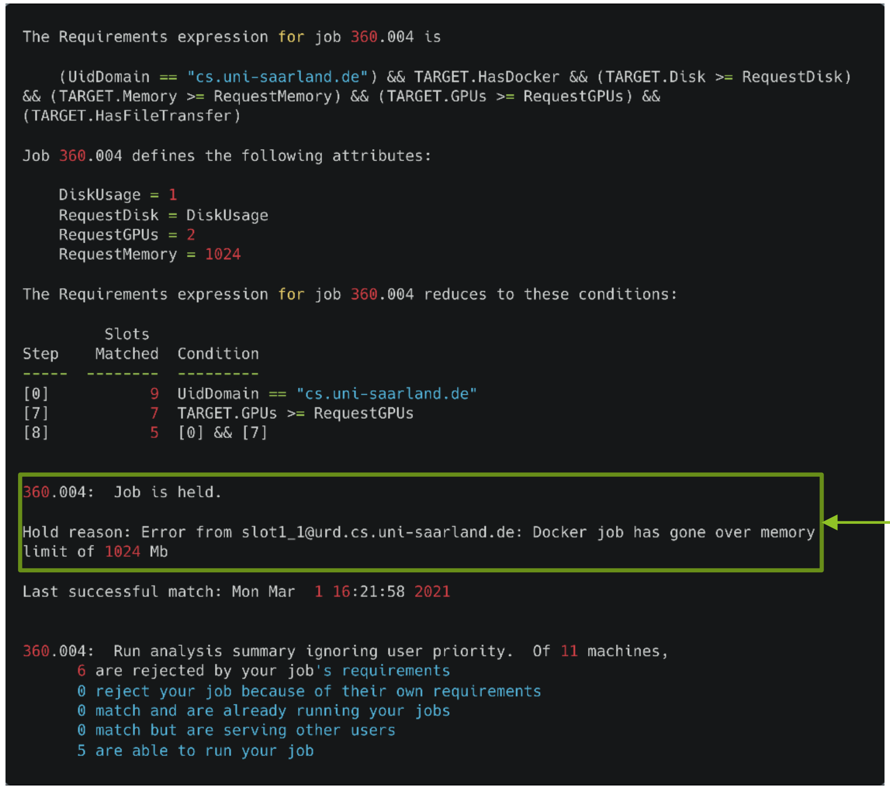


```
[user@conduit2 condor_tutorial]$ condor_q -hold 363


-- Schedd: conduit2.cs.uni-saarland.de : <134.96.227.101:9618?... @ 03/01/21 17:32:20
 ID      OWNER          HELD_SINCE  HOLD_REASON
 363.3   debuges         3/1  17:31 Error from slot1_1@vidar.cs.uni-saarland.de: Docker job has gone over memory limit of 1024 Mb
 363.4   debuges         3/1  17:31 Error from slot1_1@urd.cs.uni-saarland.de: Docker job has gone over memory limit of 1024 Mb

Total for query: 2 jobs; 0 completed, 0 removed, 0 idle, 0 running, 2 held, 0 suspended
Total for all users: 2 jobs; 0 completed, 0 removed, 0 idle, 0 running, 2 held, 0 suspended
```

The condor queue



```
The Requirements expression for job 360.004 is

    (UidDomain == "cs.uni-saarland.de") && TARGET.HasDocker && (TARGET.Disk >= RequestDisk)
&& (TARGET.Memory >= RequestMemory) && (TARGET.GPUs >= RequestGPUs) &&
(TARGET.HasFileTransfer)

Job 360.004 defines the following attributes:

    DiskUsage = 1
    RequestDisk = DiskUsage
    RequestGPUs = 2
    RequestMemory = 1024

The Requirements expression for job 360.004 reduces to these conditions:

         Slots
Step    Matched  Condition
-----  --------  ---------
[0]           9  UidDomain == "cs.uni-saarland.de"
[7]           7  TARGET.GPUs >= RequestGPUs
[8]           5  [0] && [7]


360.004:  Job is held.

Hold reason: Error from slot1_1@urd.cs.uni-saarland.de: Docker job has gone over memory
limit of 1024 Mb

Last successful match: Mon Mar  1 16:21:58 2021


360.004:  Run analysis summary ignoring user priority.  Of 11 machines,
      6 are rejected by your job's requirements
      0 reject your job because of their own requirements
      0 match and are already running your jobs
      0 match but are serving other users
      5 are able to run your job
```

- get more detailed information about your job matching to available workers with:

  `condor_q -analyze`

  `condor_q -better`

reason for holding the job

condor_qedit / condor_hold / condor_release

- adjust job requirements with **`condor_qedit`** "on-the-fly"
- release held job: **`condor_release <jobid>`**

```
[user@conduit2 condor_tutorial]$ condor_qedit 360 RequestMemory 2024
Set attribute "RequestMemory" for 2 matching jobs.

[user@conduit2 condor_tutorial]$ condor_release 360
All jobs in cluster 360 have been released
```

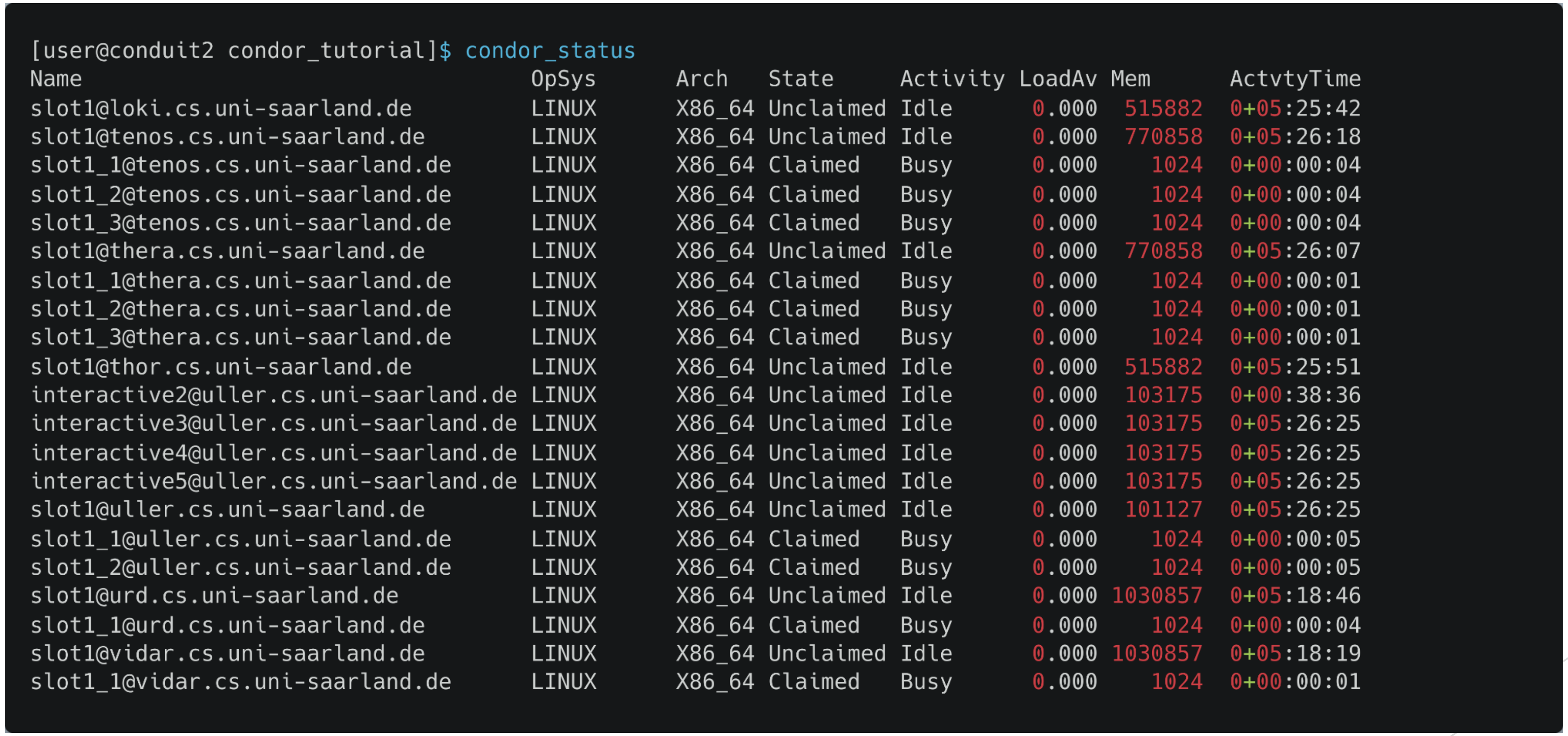
- you can also put your jobs into hold with **`condor_hold <jobid>`**
- when a job is released it gets put back into idle state and the matchmaking takes place again

condor_rm

- if you need to delete your jobs from the queue, you can use:

```
condor_rm <jobid>    //only remove job with jobid
condor_rm -a         //delete all your jobs from the queue
```

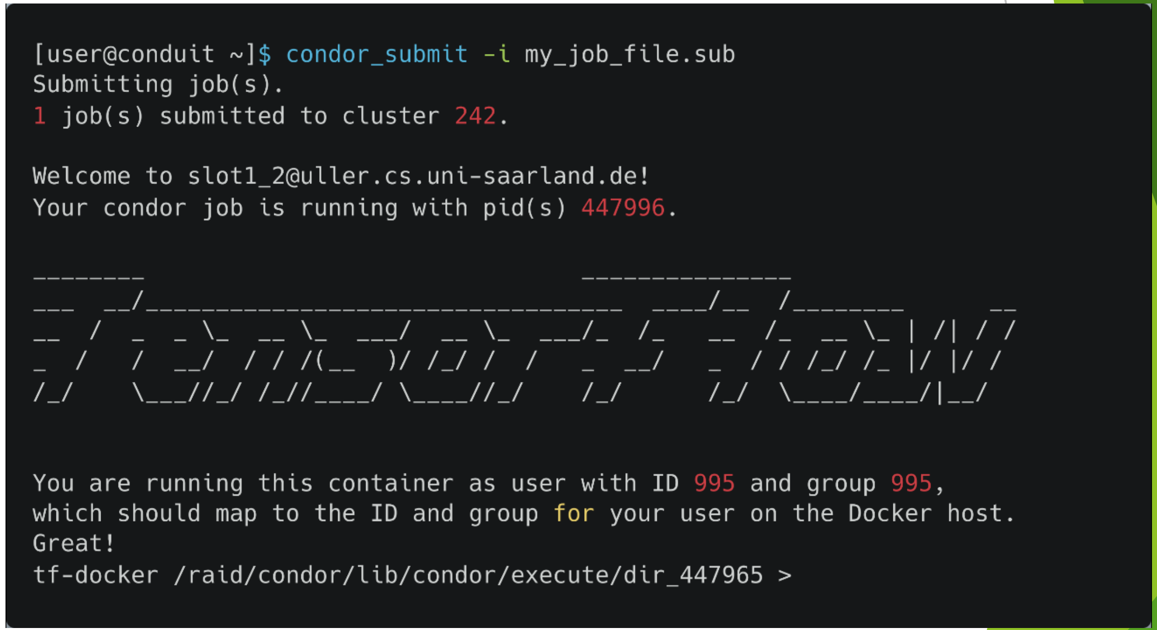
condor_status - overview of claimed and idle worker nodes



```
[user@conduit2 condor_tutorial]$ condor_status
Name                                OpSys      Arch   State     Activity LoadAv Mem      ActvtyTime
slot1@loki.cs.uni-saarland.de       LINUX      X86_64 Unclaimed Idle      0.000  515882  0+05:25:42
slot1@tenos.cs.uni-saarland.de      LINUX      X86_64 Unclaimed Idle      0.000  770858  0+05:26:18
slot1_1@tenos.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:04
slot1_2@tenos.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:04
slot1_3@tenos.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:04
slot1@thera.cs.uni-saarland.de      LINUX      X86_64 Unclaimed Idle      0.000  770858  0+05:26:07
slot1_1@thera.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:01
slot1_2@thera.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:01
slot1_3@thera.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:01
slot1@thor.cs.uni-saarland.de       LINUX      X86_64 Unclaimed Idle      0.000  515882  0+05:25:51
interactive2@uller.cs.uni-saarland.de LINUX    X86_64 Unclaimed Idle      0.000  103175  0+00:38:36
interactive3@uller.cs.uni-saarland.de LINUX    X86_64 Unclaimed Idle      0.000  103175  0+05:26:25
interactive4@uller.cs.uni-saarland.de LINUX    X86_64 Unclaimed Idle      0.000  103175  0+05:26:25
interactive5@uller.cs.uni-saarland.de LINUX    X86_64 Unclaimed Idle      0.000  103175  0+05:26:25
slot1@uller.cs.uni-saarland.de      LINUX      X86_64 Unclaimed Idle      0.000  101127  0+05:26:25
slot1_1@uller.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:05
slot1_2@uller.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:05
slot1@urd.cs.uni-saarland.de        LINUX      X86_64 Unclaimed Idle      0.000 1030857  0+05:18:46
slot1_1@urd.cs.uni-saarland.de      LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:04
slot1@vidar.cs.uni-saarland.de      LINUX      X86_64 Unclaimed Idle      0.000 1030857  0+05:18:19
slot1_1@vidar.cs.uni-saarland.de    LINUX      X86_64 Claimed   Busy      0.000    1024  0+00:00:01
```

Interactive Jobs

- you can use interactive jobs to debug your code before submitting large/many jobs to the queue
- the number of parallel running interactive jobs are limited to 4
- only one of our worker nodes allows interactive jobs
- interactive jobs are **killed** automatically after one hour to allow other users to get an interactive slot
- same syntax as normal submit, just add `'-i'` as additional parameter
- number of GPUs/CPUs is restricted to 1 for an interactive slot
- Please note that trying to use a docker image with a predefined entrypoint (e.g. hello-world) might not work as interactive job (exits the container after the script runs)



```
[user@conduit ~]$ condor_submit -i my_job_file.sub
Submitting job(s).
1 job(s) submitted to cluster 242.

Welcome to slot1_2@uller.cs.uni-saarland.de!
Your condor job is running with pid(s) 447996.

________                               _______________
___  __/__________________________________  ____/__  /________      __
__  /  _  _ \_  __ \_  ___/  __ \_  ___/_  /_   __  /_  __ \_ | /| / /
_  /   /  __/  / / /(__  )/ /_/ /  /   _  __/   _  / / /_/ /_ |/ |/ /
/_/    \___//_/ /_//____/ \____//_/    /_/      /_/  \____/____/|__/


You are running this container as user with ID 995 and group 995,
which should map to the ID and group for your user on the Docker host. Great!
tf-docker /raid/condor/lib/condor/execute/dir_447965 >
```

Further reading/tutorials

This was just a very superficial glimpse on the options that HTCondor provides. You can find more detailed information in the official HTCondor docs:

- https://htcondor.readthedocs.io/en/latest/
- Submitting a job: https://htcondor.readthedocs.io/en/latest/users-manual/submitting-a-job.html
- Managing a job: https://htcondor.readthedocs.io/en/latest/users-manual/managing-a-job.html