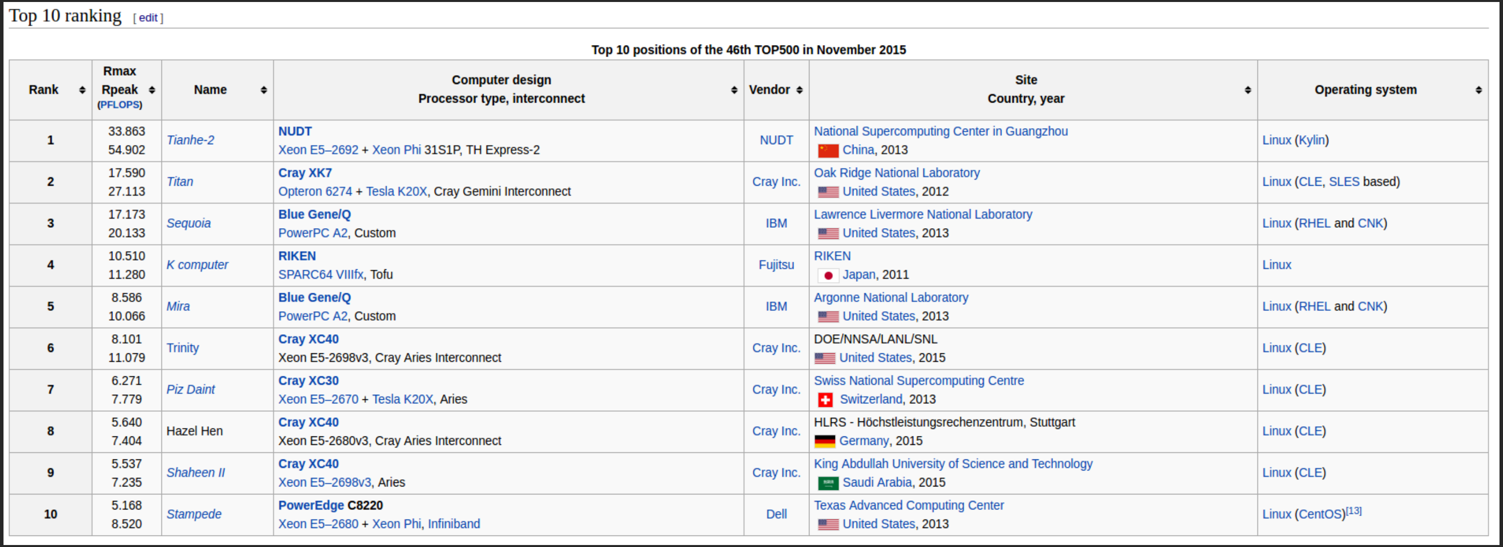
Introduction to ORNL *

TG, MLVF, 03.30.2016

https://github.com/TomaszGolan/ornl_workflow/blob/master/docs/ornl_howto.md

* Oak Ridge National Laboratory

2nd on TOP500 list



Top 10 ranking [edit]

Top 10 positions of the 46th TOP500 in November 2015

Rank	Rmax Rpeak (PFLOPS)	Name	Computer design Processor type, interconnect	Vendor	Site Country, year	Operating system
1	33.863 54.902	*Tianhe-2*	**NUDT** Xeon E5–2692 + Xeon Phi 31S1P, TH Express-2	NUDT	National Supercomputing Center in Guangzhou China, 2013	Linux (Kylin)
2	17.590 27.113	*Titan*	**Cray XK7** Opteron 6274 + Tesla K20X, Cray Gemini Interconnect	Cray Inc.	Oak Ridge National Laboratory United States, 2012	Linux (CLE, SLES based)
3	17.173 20.133	*Sequoia*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Lawrence Livermore National Laboratory United States, 2013	Linux (RHEL and CNK)
4	10.510 11.280	*K computer*	**RIKEN** SPARC64 VIIIfx, Tofu	Fujitsu	RIKEN Japan, 2011	Linux
5	8.586 10.066	*Mira*	**Blue Gene/Q** PowerPC A2, Custom	IBM	Argonne National Laboratory United States, 2013	Linux (RHEL and CNK)
6	8.101 11.079	Trinity	**Cray XC40** Xeon E5-2698v3, Cray Aries Interconnect	Cray Inc.	DOE/NNSA/LANL/SNL United States, 2015	Linux (CLE)
7	6.271 7.779	*Piz Daint*	**Cray XC30** Xeon E5–2670 + Tesla K20X, Aries	Cray Inc.	Swiss National Supercomputing Centre Switzerland, 2013	Linux (CLE)
8	5.640 7.404	Hazel Hen	**Cray XC40** Xeon E5-2680v3, Cray Aries Interconnect	Cray Inc.	HLRS - Höchstleistungsrechenzentrum, Stuttgart Germany, 2015	Linux (CLE)
9	5.537 7.235	*Shaheen II*	**Cray XC40** Xeon E5–2698v3, Aries	Cray Inc.	King Abdullah University of Science and Technology Saudi Arabia, 2015	Linux (CLE)
10	5.168 8.520	*Stampede*	**PowerEdge C8220** Xeon E5–2680 + Xeon Phi, Infiniband	Dell	Texas Advanced Computing Center United States, 2013	Linux (CentOS)[13]

Yes, there are more than 2 GPUs *

* there are 18,668 NVIDIA Kepler GPUs

General-purpose system 1/3

home.ccs.ornl.gov

Home is a general purpose system that can be used to log into other OLCF systems that are not directly accessible from outside the OLCF network. For example, running the screen or tmux utility is one common use of Home. Compiling, data transfer, or executing long-running or memory-intensive tasks should never be performed on Home. More information can be found on the The Home Login Host page.

General-purpose system 2/3

dtn.ccs.ornl.gov

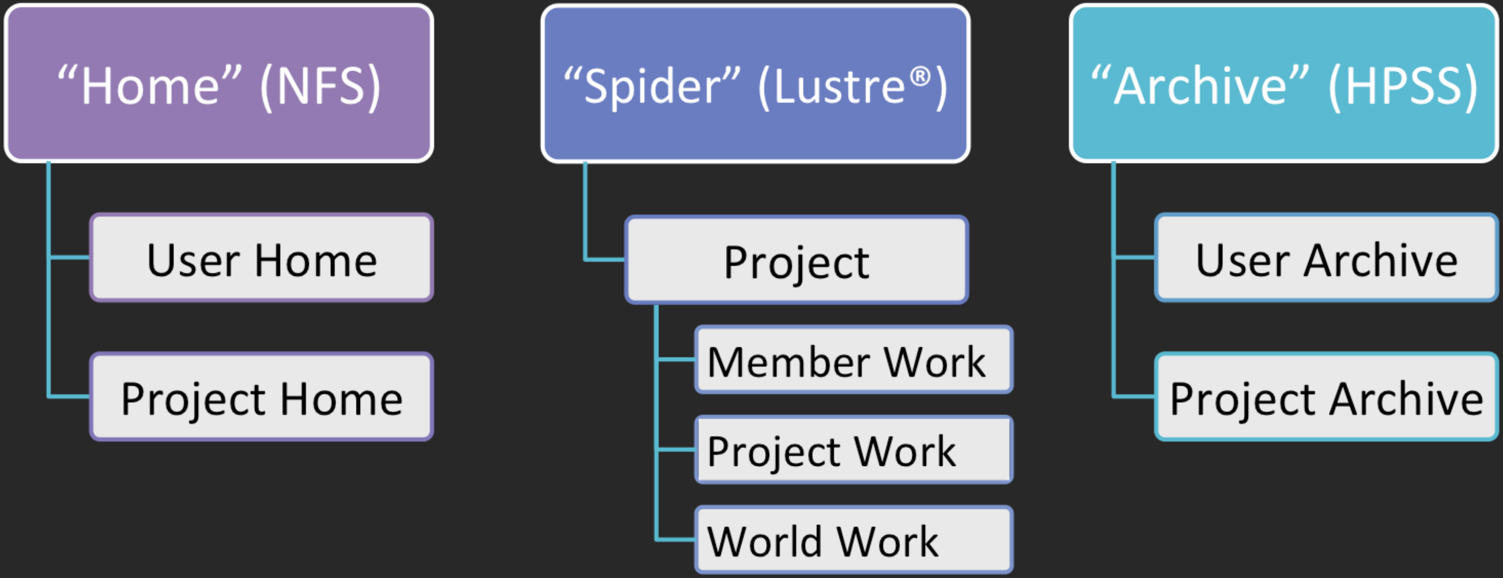
The Data Transfer Nodes are hosts specifically designed to provide optimized data transfer between OLCF systems and systems outside of the OLCF network. More information can be found on the Employing Data Transfer Nodes page.

General-purpose system 3/3

HPSS

The High Performance Storage System (HPSS) provides tape storage for large amounts of data created on OLCF systems. The HPSS can be accessed from any OLCF system through the hsi utility. More information can be found on the HPSS page.

Storage


"Home" (NFS)
User Home
Project Home
"Spider" (Lustre®)
Project
Member Work
Project Work
World Work
"Archive" (HPSS)
User Archive
Project Archive

What	Where	Path
Long-term data for routine access that is unrelated to a project	*User Home*	`$HOME`
Long-term data for archival access that is unrelated to a project	*User Archive*	`/home/$USER`
Long-term project data for routine access that's shared with other project members	*Project Home*	`/ccs/proj/[projid]`
Short-term project data for fast, batch-job access that you don't want to share	*Member Work*	`$MEMBERWORK/[projid]`
Short-term project data for fast, batch-job access that's shared with other project members	*Project Work*	`$PROJWORK/[projid]`
Short-term project data for fast, batch-job access that's shared with those outside your project	*World Work*	`$WORLDWORK/[projid]`
Long-term project data for archival access that's shared with other project members	*Project Archive*	`/proj/[projid]`

Area	Path	Type	Permissions	Quota	Backups	Purged	Retention
User Home	`$HOME`	NFS	User-controlled	10 GB	Yes	No	90 days
User Archive	`/home/$USER`	HPSS	User-controlled	2 TB	No	No	90 days
Project Home	`/ccs/proj/[projid]`	NFS	770	50 GB	Yes	No	90 days
Member Work	`$MEMBERWORK/[projid]`	Lustre®	700	10 TB	No	14 days	N/A
Project Work	`$PROJWORK/[projid]`	Lustre®	770	100 TB	No	90 days	N/A
World Work	`$WORLDWORK/[projid]`	Lustre®	775	10 TB	No	90 days	N/A
Project Archive	`/proj/[projid]`	HPSS	770	100 TB	No	No	90 days

- **Purged** - Period of time, post-file-access, after which a file will be marked as eligible for permanent deletion.
- **Retention** - Period of time, post-account-deactivation or post-project-end, after which data will be marked as eligible for permanent deletion.

Handling HPSS

- Access to archive (`/home/$USER` or `/css/proj/hep105`) only using `hsi` at dtn!!!

Copying file to HPSS

```
ssh dtn.ccs.ornl.gov # use only this node to handle HPSS
hsi put myFile
```

Retrieve file from HPSS

```
ssh dtn.ccs.ornl.gov # use only this node to handle HPSS
hsi get myFile
```

More on hsi

- call `hsi` [nothing] to manage archive in "interactive" mode
- call many hsi commands at a time, e.g.

```
touch myFile
hsi "mkdir myDir; cd myDir; put myFile"
```

- `htar -cf myTarball.htar myDir` creates a htarball in HPSS and corresponding `idx` file
- `htar -xf myTarball.htar` get a htarball from HPSS and extract in current directory
- `htar -xf myTarball.htar myFile` extract only myFile from myTarball.htar into current directory
- `htar -cf myDir/myTarball.htar myDir` works too

What you can't do, but could try...

- archive directly to a folder

```
hsi put myFile some/path/in/archive/
```

must cd first

- unhtar directly to some folder

```
htar -cf data.htar my_folder
```

it will extract only my_folder from data.htar

Example workflow

- call dtn job to get your data from archive
- call titan job to do your calculations
- call dtn job to put your result to archive

HelloWorld.sh

```
#!/bin/bash

date >> HelloWorld.dat
pwd >> HelloWorld.dat
ls >> HelloWorld.dat
```

- run using (start with dtn!!!)

```
qsub -q dtn HelloWorld_get.pbs
```

- HelloWorld_get.pbs will run next jobs

HelloWorld_get.pbs

```
#PBS -A hep105
#PBS -l walltime=00:00:30
#PBS -l nodes=1
#PBS -j oe
#PBS -o HelloWorld_get.out

cd $MEMBERWORK/hep105/HelloWorld/
hsi get HelloWorld/HelloWorld.dat
hsi get HelloWorld/HelloWorld.sh

qsub -q titan $HOME/HelloWorld/HelloWorld_do.pbs
```

HelloWorld_do.pbs

```
#PBS -A hep105
#PBS -l walltime=00:00:30
#PBS -l nodes=1
#PBS -j oe
#PBS -o HelloWorld_do.out

cd $MEMBERWORK/hep105/HelloWorld/
aprun ./HelloWorld.sh

qsub -q dtn $HOME/HelloWorld/HelloWorld_put.pbs
```

HelloWorld_put.pbs

```
#PBS -A hep105
#PBS -l walltime=00:00:30
#PBS -l nodes=1
#PBS -j oe
#PBS -o HelloWorld_put.out

cd $MEMBERWORK/hep105/HelloWorld/
hsi put HelloWorld.dat
hsi mv HelloWorld.dat HelloWorld/
```

More on nodes 1/3

Login nodes

Login nodes are designed to facilitate ssh access into the overall system, and to handle simple tasks. When you first log in, you are placed on a login node. Login nodes are shared by all users of a system, and should only be used for basic tasks such as file editing, code compilation, data backup, and job submission. Login nodes should not be used for memory-intensive nor processing-intensive tasks. Users should also limit the number of simultaneous tasks performed on login nodes. For example, a user should not run ten simultaneous tar processes.

More on nodes 2/3

Service nodes

Memory-intensive tasks, processor-intensive tasks, and any production-type work should be submitted to the machine's batch system (e.g. to Torque/MOAB via qsub). When a job is submitted to the batch system, the job submission script is first executed on a service node. Any job submitted to the batch system is handled in this way, including interactive batch jobs (e.g. via `qsub -I`). Often users are under the (false) impression that they are executing commands on compute nodes while typing commands in an interactive batch job. On Cray machines, this is not the case.

More on nodes 3/3

On Cray machines, when the aprun command is issued within a job script (or on the command line within an interactive batch job), the binary passed to aprun is copied to and executed in parallel on a set of compute nodes. Compute nodes run a Linux microkernel for reduced overhead and improved performance.

Only User Work (Lustre®) and Project Work (Lustre®) storage areas are available to compute nodes on OLCF Cray systems. Other storage spaces (User Home, User Archive, Project Home, and Project Archive) are not mounted on compute nodes.

Walltime vs nodes (titan)

Bin	Min Nodes	Max Nodes	Max Walltime (Hours)	Aging Boost (Days)
1	11,250	--	24.0	15
2	3,750	11,249	24.0	5
3	313	3,749	12.0	0
4	126	312	6.0	0
5	1	125	2.0	0

Workflow generator

https://github.com/TomaszGolan/ornl_workflow

```
usage: ./generate_workflow.py <opts>

GENERATE WORKFLOW @ TITAN

optional arguments:
  -h, --help            show this help message and exit

REQUIRED ARGUMENTS:
  --framework [caffe, theano, ...]
                        which framework will be used (default: None)
  --command [./scriptName.py [options]]
                        main command with options (default: None)

WORKING DIRS SETTINGS:
  --framework_dir [PATH]
                        path to store framework (default: $PROJWORK/hep105/software/)
  --software_dir [PATH]
                        path to store software (default: $MEMBERWORK/hep105/software/)
  --data_dir [PATH]     path to store data (default: $PROJWORK/hep105/data/)
  --log_dir [PATH]      path to store logs (default: $MEMBERWORK/hep105/logs/)
  --output_dir [PATH]   path to store output (default: $MEMBERWORK/hep105/output/)

INPUT FILES / SOFTWARE:
  --input_data [file1 file2 ...]
                        list of files to get from /proj/hep105/data (default:
                        )
  --software_list [/path1/software.list]
                        path to a file contains all required software
                        (default: `pwd`/software.list)

EXTRA OPTIONS:
  --tag [tag]           tag used for logs and output files names (default:
                        $USER_YYYY-MM-DD)
  --force_framework     get framework even if it exists already (default:
                        false)
  --force_data          get data files even if they exist already (default:
                        false)
  --no_archive          do not save files in HPSS after job is done
```

```bash
#!/bin/bash

NEPOCHS=1
LRATE=0.0025
L2REG=0.0001

TAG="lasagne_first_test_small_betaprime"

DATAFILENAME="minosmatch_fuel_me1Bmc_small.hdf5"
SAVEMODELNAME="$MEMBERWORK/hep105/output/$TAG/$TAG.npz"
PYTHONPROG="./minerva_triamese_betaprime.py"

COMMAND="$PYTHONPROG -l \
         -n $NEPOCHS \
         -r $LRATE \
         -g $L2REG \
         -s $SAVEMODELNAME \
         -d $PROJWORK/hep105/data/$DATAFILENAME"

./generate_workflow.py \
  --framework theano \
  --input_data "theano/$DATAFILENAME" \
  --tag $TAG \
  --command "$COMMAND"
```