Hopper Quick Start Guide

Overview of the system

Hopper consists of a total of 168 nodes with 2 E5-2680 v2 processors with 20 cores each. Table 1 shows the hardware details of the new cluster in comparison to the Turing system:

	Turing	Hopper
Total nodes	96/72	144/24
Processor type	Harpertown/Westmere	Ivy bridge
Base Clock Speed	2.5 GHz / 2.27 GHz	2.8 GHz
Cores per node	8/12	20
Total cores	1632	3360
Memory per node (GB)	16/24	144x64/24x256[1]
Memory per core (GB)	2	3.2/12.8
Peak performance (TF)	15.5	75.3
Network	GigE / Infiniband DDR/QDR 2:1	Infiniband FDR10 2:1

Table 1 Hardware overview

Connecting to Hopper

All users having an active VSC account on Turing can connect to the login node of Hopper with the same credentials using the command:

```
$ ssh vsc20XYZ@login.hpc.uantwerpen.be
```

Take into account that the site is only accessible from within UAntwerpen domain, so the page won't load from, e.g., home, unless by using VPN or a proxy.

[1] Currently, 96 nodes with 64 GB RAM are available to all users. The 24 nodes with 256 GB RAM are only available upon special request. Another 48 nodes with 64 GB RAM will become available in the near future.

Accessing your Data

$VSC_HOME and $VSC_DATA are shared by both Turing and Hopper. Every cluster has its own $VSC_SCRATCH space. Table 2 summarizes the available storage areas and their characteristics:

Name	Variable	Type	Access	Backup	Default Quota
/user/antwerpen/20X/vsc20XYZ	$VSC_HOME	GPFS	Global	NO	3 GB
/data/antwerpen/20X/vsc20XYZ	$VSC_DATA	GPFS	Global	NO	25 GB
/scratch/antwerpen/20X/vsc20XYZ	$VSC_SCRATCH $VSC_SCRATCH_SITE	GPFS	Global	NO	25 GB
/tmp	$VSC_SCRATCH_NODE	ext4	local	NO	250 GB

Table 2 Storage areas overview

- **$VSC_HOME:** A regular home directory, which contains all files that a user might need to log on to the system, and small 'utility' scripts/programs/source code/.... The capacity that can be used is restricted by quota. This directory should **NOT** be used for I/O intensive jobs (use $VSC_SCRATCH, otherwise your job might be put on hold).

- **$VSC_DATA:** A data directory which can be used to store programs and their results. This directory should **NOT** be used for I/O intensive jobs (use $VSC_SCRATCH, otherwise your job might be put on hold). There is a default quota of 25 GB, but it can be enlarged upon special request.

- **$VSC_SCRATCH/$VSC_SCRATCH_SITE:** On each cluster you have access to a scratch directory that is shared by all nodes on the cluster. This directory is also accessible from the login nodes, so it is accessible while your jobs run, and after they finish. **This is the preferred place to run your jobs**. The $VSC_SCRATCH file system is specifically designed for high throughput and parallel I/O. Running your job on the other file systems may not only slow it down, it will affect all other jobs as well.

- **$VSC_SCRATCH_NODE:** is a scratch space local to each compute node. Thus, on each node, this directory points to a different physical location and the content is only accessible from that particular compute node, and only during the runtime of your job.

REMARK: Take into account that currently no backups are performed!

Software

All applications on Hopper are installed in a separate directory. Programs compiled for Turing will/might not work on Hopper, and vice versa. Some of the software applications available on Turing have been recompiled and are already available on Hopper, normally the latest version available.

The *modules* software manager tool is also available on Hopper and will show only modules built for Hopper. Note however, that, in order to achieve some degree of uniformity across the different VSC sites, the naming scheme for the software packages has been changed. The new naming scheme is as follows:

PackageName/version-ToolchainName-ToolchainVersion

Where *PackageName* is the official name of the software keeping capital and lower letters.

Attention: The toolchain names have been changed (ictce -> intel, goolf -> foss) on Hopper, so you will need to adjust your job scripts when transferring job scripts from Turing to Hopper or vice versa.

Thus, in some cases the names of the packages have been changed to make them compliant with the new scheme. One example of the change is the Python package:

On Hopper:

```
$ module av Python

-------------- /apps/antwerpen/modules/hopper/2014a/all --------------

Python/2.7.6-foss-2014a
Python/2.7.6-intel-2014a
Python/3.4.1-foss-2014a
Python/3.4.1-intel-2014a
```

On Turing:

```
$ module av python

-------------- /apps/antwerpen/turing/harpertown/modules/all --------------

Python/2.6.2-gimkl-0.5.0
Python/2.6.3-gimkl-0.5.0
Python/2.6.4-ictce-3.2.1.015.u1
Python/2.6.4-ictce-3.2.1.015.u4
Python/2.7.3-ictce-4.0.1
Python/2.7.5-goolf-1.5.9
```

```
Python/2.7.5-ictce-5.5.0
Python/3.2.3-goolf-1.5.9
```

TIP: Revise your job scripts to ensure the appropriate software package name is used! Example:

```
case ${VSC_INSTITUTE_CLUSTER} in
  turing)
    module load Python/3.2.3-goolf-1.5.9
    ;;
  hopper)
    module load Python/3.4.1-intel-2014a
    ;;
esac
```

Table 3 shows a list of some of the available software on Hopper grouped by category:

Biosciences	Chemistry	Mathematics	Others
Bowtie2 NEURON SAMtools	ABINIT ADF elk Gaussian GROMACS LAMMPS OpenMX PLUMED QuantumESPRESSO Siesta VASP	Maple MATLAB METIS R	TELEMAC

Table 3. Available software

To obtain a complete list of the software you can execute:

```
$ module available
```

If you need additional software installed, please send a request to : hpc@uantwerpen.be

Compiling and Running your Code

Several compilers and libraries are available on Hopper, as well as two toolchains flavors “intel” (based on Intel software components) and “foss” (based on free and open source software).

A toolchain is a collection of tools to build (HPC) software consistently. It consists of:

- compilers for C/C++ and Fortran
- a communications library (MPI)
- mathematical libraries (linear algebra, FFT).

Toolchains are versioned and refreshed twice a year. *Currently, software is not rebuilt when a new version of a toolchain is defined.* Version numbers consist of the year of their definition, followed by either a or b, e.g., 2014a. Note that the software components are not necessarily the most recent releases; rather they are selected for stability and reliability. Table 4 summarizes the toolchains available on Hopper and their components:

	Intel compilers	Open source
Name	intel	foss
version	2014a, 2014b	2014a, 2014b
Compilers	Intel compilers icc, icpc, ifort	GNU compilers gcc, g++, gfortran
MPI Library	Intel MPI	OpenMPI
Math libraries	Intel MKL	OpenBLAS, LAPACK FFTW ScaLAPACK

Table 4. Toolchains on Hopper

To know which modules are loaded by foss/2014a, just enter the following command.

```
$ module show foss/2014a

-------------------------------------------------------------------

/apps/antwerpen/modules/hopper/2014a/all/foss/2014a:

module-whatis      Description: GNU Compiler Collection (GCC) based compiler
toolchain, including
 OpenMPI for MPI support, OpenBLAS (BLAS and LAPACK support), FFTW and
ScaLAPACK. - Homepage: (none)

conflict     foss
module             load GCC/4.8.2
module             load OpenMPI/1.6.5-GCC-4.8.2
module             load OpenBLAS/0.2.8-gompi-2014a-LAPACK-3.5.0
module             load FFTW/3.3.3-gompi-2014a
module             load ScaLAPACK/2.0.2-gompi-2014a-OpenBLAS-0.2.8-LAPACK-
3.5.0
setenv             EBROOTFOSS /apps/antwerpen/ivybridge/sl6/foss/2014a
```

```
setenv              EBVERSIONFOSS 2014a
setenv              EBDEVELFOSS
/apps/antwerpen/ivybridge/sl6/foss/2014a/easybuild/foss-2014a-easybuild-devel

------------------------------------------------------------------
```

The “conflict” line says it’s not allowed to have two foss modules loaded at the same time.

TIP: Recompile your codes for using them on Hopper, check the results of the recompiled codes before starting production runs and use the available toolchains for compiling whenever possible.

The current default toolchain version is 2014a. To load the module ABINIT/7.8.2-intel-2014a, you write

```
module load ABINIT/7.8.2-intel-2014a
```

When the default toolchain becomes, say, 2015a, and you would like to load that specific ABINIT version again, you will have to load the 2014a environment first, since otherwise, the module cannot be found:

```
module load hopper/2014a
module load ABINIT/7.8.2-intel-2014a
```

Running Jobs

Torque/Moab is used for scheduling jobs on Hopper, so the same commands and scripts used on Turing will work on Hopper. The single user policy is still active.

None of the available queues on Turing are available on Hopper. It is strongly recommended that you use the PBS –l option to request resources for cputime and memory. (This setup also works on Turing.)

Some useful are –l options are:

Resources usage

- -l walltime=4:30:00 (job will not run longer than 4h 30 min)
- -l nodes=2:ppn=20 (job needs 2 nodes and 20 cores per node)
- -l mem=40gb (jobs request 40 GB of memory, sum for all processes)
- -l pmem=4gb (job request 4 GB of memory per core)

Other node features: processor type, accelerators, etc.

- -l nodes=1:feature

An example of a job submitted using resource request could be:

```
$ qsub -l nodes=10:ppn=20:ivybridge,walltime=1:00:00,pmem=2gb myprogram
```

This job requests 10 nodes with 20 Ivy Bridge cores each. It needs a wall time of 1 hour and 2 GB of memory per core.

Features available on Turing:

- ib, gbe: choose Infiniband or gigabit ethernet as interconnect
- harpertown, westmere: harpertown or westmere as processor

Features available on Hopper:

- ib: always Infiniband; feature available for consistency
- ivybridge: ivybridge processor

When you're only running jobs on Hopper, features are not required since it's a homogenous system. However, if you're running jobs on Turing and Hopper, it might be useful to specify the ib feature to keep the job script generic.

TIP: Revise your batch scripts to remove references to the queue names of Turing and modify them to be based on resources instead on queue names.

REMARK: To run MPI programs on Hopper, you no longer need to "module load mpiexec" (moreover, it is not even installed). Use "mpirun" instead of mpiexec.

Questions or remarks are welcome at hpc@uantwerpen.be

Last update: 2014-11-25