A Guide on Using the Zythos Cluster

Simone Campanoni Nikos Hardavellas

March 17, 2021

Abstract

The purpose of this document is to provide an introduction to the new user of the cluster, and act as a reference guide to active users as they utilize the system. As it may extend in size over time specific sections for reading should be specified to the recipient.

Contents

Chapter 1

The Zythos Cluster

The Zythos cluster has an entry node and a set of computation nodes. You can **only** login to the entry node (*peroni*) unless you follow the protocol described in Section 1.2.

Logins are automatically monitored to make sure that computation nodes are used only through condor. Avoiding direct uses of computation nodes is essential to allow other users to properly perform their experiments. Every time you login to a node, you create workload that might generate perturbation on the benchmarks currently running on that node. These perturbations might change research results we all use in papers. So we must avoid them. Moreover, detecting these performance alterations is not easy. Hence, to help every user of the Zythos cluster, we decided to forbit the direct use of computation nodes that is not *peroni* .

1.1 Nodes

Zythos includes the following nodes:

Table 1.1: Zythos nodes

Name	Processors	Total logical cores	Total phisical cores	Memory	Local hard drive	Co-processors GPUs	Co-processors Others	Year
Login nodes								
peroni	2 Intel(R) Xeon(R) E5-2695 v3 @ 2.30GHz Haswell architecture	56	28	256 GB	71 TB	No	No	2014
Computational nodes								
fix	2 Intel(R) Xeon(R) Gold 6258R @ 2.70GHz Cascade architecture	112	56	512 GB	4 TB	No	No	2020
piraat	2 Intel(R) Xeon(R) Silver 4116 @ 2.10GHz Skylake architecture	48	24	128 GB	1 TB	No	No	2017
allagash	2 Intel(R) Xeon(R) E5-2695 v3 @ 2.30GHz Haswell architecture	56	28	256 GB	2 TB	No	Intel Xeon Phi	2014
boucanier	2 Intel(R) Xeon(R) E5-2420 v2 @ 2.20GHz Ivy Bridge architecture	24	12	32 GB	1 TB	No	No	2014
maudite	2 Intel(R) Xeon(R) X5560 @ 2.80GHz Nehalem architecture	16	8	32 GB	1 TB	No	No	2009
tremens	2 Intel(R) Xeon(R) X5560 @ 2.80GHz Nehalem architecture	16	8	32 GB	1 TB	No	No	2009
guldendraak	2 Intel(R) Xeon(R) X5560 @ 2.80GHz Nehalem architecture	16	8	32 GB	1 TB	No	No	2009

Their network names are:

- *peroni* : `peroni.cs.northwestern.edu`
- *fix* : `fix.cs.northwestern.edu`
- *piraat* : `piraat.cs.northwestern.edu`
- *allagash* : `allagash.cs.northwestern.edu`
- *boucanier* : `boucanier.cs.northwestern.edu`
- *maudite* : `maudite.cs.northwestern.edu`
- *tremens* : `tremens.cs.northwestern.edu`
- *guldendraak* : `guldendraak.cs.northwestern.edu`

1.2 Login to Other Nodes

If you can, please consider to login only on *peroni* . While this is the policy for the Zythos cluster, we understand there are needs to login to other nodes (e.g, debug a program on a specific hardware). Therefore, we created a procedure to follow every time you want to login to another node (e.g., *maudite*). Next are the steps to follow.

1.2.1 Before Login

1. Specify the node you want to login. This is done by modifying the file `software/condor_dummy/job.con` of the Zythos git repository (i.e., `/project/zythos/zythos.git`). You need to write the name of the node at line 7 of that file.
2. Request an exclusive access for the target node. This is done by invoking `make` inside the directory `software/condor_dummy`.
3. Wait to receive an email from condor to your `@eecs.northwestern.edu` email account.
4. As soon as you receive the email, you can login to the specified node.

1.2.2 After Logout

After you have done using a node you have requested exclusive access to, follow the next steps:

1. Logout from the node you have exclusive access to. Make sure you have no active login to that node.
2. Delete the condor job you have submitted before (e.g., `condor_rm JOBID`).

Chapter 2

Using Condor

This chapter describes the specifics of the condor installation of the Zythos cluster, for example how to submit suspension jobs. Please read the generic documentation about condor (e.g., https://research.cs.wisc.edu/htcondor/quick-start.html) before reading this guide.
[1]

2.1 Short Jobs

Jobs that you know in advance will take less than 1 day can be submitted as short jobs by adding in your condor file the following lines:

```
+IsWholeMachineJob      = false
+IsSuspensionJob        = false
```

2.2 Long Jobs

Jobs that requires more than 1 day (24 hours) to run must classified as long jobs and be scheduled at suspension slots. To do so, add the following lines to your condor file:

```
+IsWholeMachineJob      = false
+IsSuspensionJob        = true
```

2.3 Whole Machine Jobs

Jobs that requires an entire machine to run (e.g., you are measuing the performance/energy of a given program) are classified as whole machine jobs. These jobs must take less than 1 day to complete.

Add the following lines to your condor file to submit a whole machine job:

```
+IsWholeMachineJob      = true
+IsSuspensionJob        = false
```

Chapter 3

Extra software

The extra software is referred to those that can be used by many of us and that their sources do not need to be customized for each of us. This chapter is dedicated to this type of software.

Some of these extra software have been installed without using Red Hat-specific tools. Reasons to install these extra software without using red hat-specific tools are:

- they are not included in Red Hat yet
- Red Hat software versions are too old

3.1 Structure to Follow for Each Extra Software

We would like to follow the next structure for software we manually installed and configured. While not current extra software follow this structure yet (sadly), we aim to reach the point where all of them will.

You can install extra software only if you belong to the group `authors`. If you have doubts about it, please contact us.

A given software, let us called it `X`, is installed under

`/project/extra/X`

To do it, create the above directory making sure to give group permissions to read and write.

Each extra software will have at least two versions installed:

- `/project/extra/X/git`: this is the git version of the software and, therefore, the most up-to-date version.
- `/project/extra/X/Y`: where `Y` is a version number (e.g., `2.1`). This is a stable version of the software.

Each version of each extra software must have an "enable" file that when sourced it enables that version in your environment. Hence, the following files must exist in the examples above:

Software	Maintainer	Example of enable file
LLVM	Simone Campanoni	/project/extra/llvm/9.0.0/enable
PIN	Simone Campanoni	/project/extra/pin/3.13/enable
Boost C++ library	Simone Campanoni	/project/extra/boost/1.72.0/enable
MiniSAT	Simone Campanoni	/project/extra/minisat/2.0/enable
STP	Simone Campanoni	/project/extra/stp/2.3.3/enable
VIM	Simone Campanoni	/project/extra/vim/git/enable
NeoVIM	Simone Campanoni	/project/extra/neovim/git/enable
Google Perf	Simone Campanoni	/project/extra/google/2.7/enable
BurnCPU	Simone Campanoni	/project/extra/burnCPU/enable
GCC	Root	/opt/rh/gcc-toolset-9/enable

Table 3.1: Extra software currently available in Zythos

- `/project/extra/X/git/enable`
- `/project/extra/X/2.1/enable`

Each of these enable files cannot be executable by anyone.

Finally, each extra software needs to have an official maintainer. This is the person that should be contacted in case that software stops working. If you want to install extra software and become, therefore, its maintainer, please contact us so we can add you to the list.

3.2 Extra Software Currently Available

Table 3.1 lists all the extra software currently available.

3.3 Special Notes about GCC

The latest GCC available from Red Hat is installed under

`/opt/rh/gcc-toolset-N`

where `N` is the latest number. At the time we wrote this document, the latest number is `8`. To enable it, run:

```
$ source /opt/rh/gcc-toolset-9/enable
```

3.4 Others

Other software are available. Next is the list of commands to enable them:

```
$ source /project/tmux/tmux-2.7/enable
```

Chapter 4

Using the Xeon Phi cores

The Xeon Phi cores are located at the *allagash* node of the cluster. Due to the special nature of the node a set of specific step are required to use the Xeon Phi Cores.

4.1 Compiling for the Xeon Phi Core

First and foremost, as *allagash* is generally used as a profiling node. **All** compilation and experimentation with your code should be done at *peroni* the entry node of the cluster.

4.1.1 Using the ICC compiler

In order to compiler you application for the Xeon Phi core you will require the Intel Compiler toolchain. Thus, the first step to compile you applications is to extend your *PATH* and *LD_LIBRARY_PATH* variables with the binary and library directories of the intel compiler toolchain.

To do so append the following lines in your *.basrc* file.

```
###
## INTEL COMPILER & PHI
###

export PATH=/opt/intel/bin:/opt/intel/ipp/bin:/opt/intel/tbb/bin:/
    opt/intel/mkl/bin:/opt/intel/mpirt/bin:/opt/intel/ism/bin:/opt/
    intel/itac_latest/bin:/opt/intel/itac_latest/intel64/bin:/opt/
    intel/impi_latest/intel64/bin:/opt/intel/parallel_studio_xe_2015/
    bin:/opt/mpss/3.4.2/sysroots/x86_64-mpsssdk-linux/usr/bin:/opt/
    intel/mic/bin:/opt/intel/itac_latest/mic/bin:/opt/intel/
    impi_latest/mic/bin:/opt/intel/composerxe/debugger/gdb/target/mic/
    bin:/opt/intel/composerxe/debugger/gdb/intel64_mic/bin:${PATH}
```

```
export LD_LIBRARY_PATH=/opt/intel/lib/mic:/opt/intel/ipp/lib/mic:/opt
    /intel/tbb/lib/mic:/opt/intel/mkl/lib/mic:/opt/intel/mpirt/lib/mic
    :/opt/intel/impi_latest/mic/lib:/opt/intel/itac_latest/mic/lib:/
    opt/intel/mic/coi/host-linux-release/lib:/opt/intel/composerxe/
    debugger/gdb/target/mic/lib:${LD_LIBRARY_PATH}
```

Following this you will need to run

```
source .bashrc
```

4.1.2 Make sure you have blocked condor jobs from *allagash*

Due to the fact that allagash is the general node used for profiling applications, any submission from condor should be prohibited while profiling applications withour using condor. This can be done by scheduling a dummy job at allagash that just runs an infinite loop.

Listing 4.1: dummy.sh

```
#!/bin/bash

while true
do
sleep 5
done
```

Listing 4.2: condor.submit

```
Universe = vanilla
Getenv = True
Requirements = (Machine == ``allagash.cs.northwestern.edu'')

Rank = TARGET.Mips
Notification = error
Copy_To_Spool = False
Should_Transfer_Files = yes
When_To_Transfer_Output = ON_EXIT


Output = condor.out
Error = condor.err
Log = condor.log

InitialDir = ./
Executable = ./dummy.sh
Queue
```

After submitting you dummy job. Wait until it starts running on *allagash* before you execute your application.

4.1.3 Mount Scratch

Make sure 'scratch' is mounted at the Xeon Phi core using

```
df -h
```

while logged in the Phi core. You should see some like this:

```
Filesystem                Size      Used Available Use% Mounted on
none                     12.9G     83.9M     12.8G   1% /
none                      7.6G     44.0K      7.6G   0% /dev
host:/scratch             1.6T    634.0G    940.7G  40% /mnt/scratch
```

If you can not locate the */mnt/scratch* entry use the following command while logged in the Phi core to mount the partition

```
mount -o nolock -t nfs host:/scratch /mnt/scratch
```

4.1.4 Shared Libaries Dependencies

You will probably require shared libraries to run your executable. A shared folder utilized by everyone would be a good practice.

Please copy you shared libraries at */scratch/phi/shared_libs*

An example commmand that copies some frequently used libraries is the follwing:

```
cp /opt/intel/composer_xe_2015/mkl/lib/mic/libmkl_intel_lp64.so /scratch/phi/shared_libs
cp /opt/intel/composer_xe_2015/mkl/lib/mic/libmkl_intel_thread.so /scratch/phi/shared_libs
cp /opt/intel/composer_xe_2015/mkl/lib/mic/libmkl_core.so /scratch/phi/shared_libs
cp /opt/intel/composer_xe_2015/lib/mic/libiomp5.so /scratch/phi/shared_libs
cp /opt/intel/composer_xe_2015/lib/mic/libcilkrts.so.5 /scratch/phi/shared_libs
```

4.1.5 LD_LIBRARY_PATH

The *LD_LIBRARY_PATH* is not maintained so it is good to append it before the execution command of your application. For example:

```
LD_LIBRARY_PATH=/mnt/scratch/phi/shared_libs/:${LD_LIBRARY_PATH} ./my
    /binary
```

Following the steps up to here should allow you to run application at the Phi core

4.2 Profiling your application at the Xeon Phi Cores

In order to profile your application at the Xeon Phi Core you will need to go through the step of section 4.1.

After you made sure you are able to compile and run you applications, in order to profile it you will need to go through the following steps.

4.2.1 Vtune profiler's access to the Xeon Phi Cores

The Vtune profiler by default tries to connect to the Xeon Phi core using the username of the running user. Thus, until we find a way around that. All users should get accounts at the phi core.

4.2.2 Absolute Paths

Paths to your executable and input output files should preferably be absolute and not relative to the expected directory of execution.

An example profiling command (run from allagash) is the following:

```
amplxe-cl -verbose -target-system=mic-native:0 -c advanced-hotspots -
    search-dir=. -- LD_LIBRARY_PATH=/mnt/scratch/phi/shared_libs/ /mnt
    /scratch/phi/georgios/matrix/matrix.mic
```

by default the above command should generate a directory named *r000ah* at the current working directory.

To open the profiling information use the following command (make sure you have enabled X forwarding)

```
amplxe-gui ./r000ah
```

4.2.3 After Profiling Clean Up

Due to some bug the *amplxe-gui* applications does not exit after you close its GUI. Thus, you will need to manually kill the processes using the *kill* command.

Chapter 5

Filesystem Structure

In order to avoid Chaos and improve efficiency please follow the suggested filesystem structure and conventions.

5.1 Partitions

Zythos has the following partitions:

- `/` : this is local to each node. This partition includes binaries and headers installed locally in that machine.
- `/home` : this is a NFS partition. All nodes in the Zythos cluster can access it. This partition includes home directories.
- `/project` : this is a NFS partition. All nodes in the Zythos cluster can access it. This partition includes files related to projects that need to be backed up.
- `/nfs-scratch` : this is a NFS partition. All nodes in the Zythos cluster can access it. This partition includes temporary files. This partition is not backed up.

5.2 Accessing Partitions from your Laptop

Zythos partitions can be accessed via sshfs from your laptop. To do it, you need to mount each partition to an empty directory in your laptop.

Let us assume you have an empty directory called `~/remote/home`. Also, let us assume your username in Zythos is `Y`. To mount your home directory of Zythos to your local `~/remote/home` directory, run:

```
$ sshfs Y@peroni.cs.northwestern.edu:/home/Y ~/remote/home
-onoappledouble,volname=project
```

Bibliography

[1] John Bent. *Data-Driven Batch Scheduling.* PhD thesis, University of Wisconsin, Madison, May 2005.