COMP Superscalar

COMPSs at MareNostrum Manual

Version: 1.3

August 17, 2015

This manual only provides information about the COMPSs usage at MareNostrum. Specifically, it details the available COMPSs modules, how to load them and how to create and track a COMPSs job.

If you want to install COMPSs at your local machine please refer to the *COMPSs Installation Manual* available at our webpage `http://compss.bsc.es`.

For further information about the application execution please refer to the *COMPSs User Manual: Application execution guide* available at `http://compss.bsc.es` .

For further information about the application development please refer to the *COMPSs User Manual: Application development guide* available at `http://compss.bsc.es/` .

For full COMPSs application examples (codes, execution commands, results, logs, etc.) please refer to the *COMPSs Sample Applications* available at `http://compss.bsc.es/` .

Contents

1 COMP Superscalar (COMPSs)

COMP Superscalar (COMPSs) is a programming model which aims to ease the development of applications for distributed infrastructures, such as Clusters, Grids and Clouds. COMP Superscalar also features a runtime system that exploits the inherent parallelism of applications at execution time.

For the sake of programming productivity, the COMPSs model has four key characteristics:

- **Sequential programming:** COMPSs programmers do not need to deal with the typical duties of parallelization and distribution, such as thread creation and synchronization, data distribution, messaging or fault tolerance. Instead, the model is based on sequential programming, which makes it appealing to users that either lack parallel programming expertise or are looking for better programmability.

- **Infrastructure unaware:** COMPSs offers a model that abstracts the application from the underlying distributed infrastructure. Hence, COMPSs programs do not include any detail that could tie them to a particular platform, like deployment or resource management. This makes applications portable between infrastructures with diverse characteristics.

- **Standard programming languages:** COMPSs is based on the popular programming language Java, but also offers language bindings for Python and C/C++ applications. This facilitates the learning of the model, since programmers can reuse most of their previous knowledge.

- **No APIs:** In the case of COMPSs applications in Java, the model does not require to use any special API call, pragma or construct in the application; everything is pure standard Java syntax and libraries. With regard the Python and C/C++ bindings, a small set of API calls should be used on the COMPSs applications.

2 COMPSs Modules

2.1 Available modules

COMPSs is configured in MareNostrum (MN3) as a Linux Module. Type *module available COMPSs* to list the available COMPSs modules through Linux Module configuration and *module load COMPSs/¡version¿* to load it.

```
$ module available COMPss
---------- /apps/modules/modulefiles/tools ----------
COMPSs/0.0
COMPSs/0.1
COMPSs/0.2_Nested
COMPSs/1.1.2_gpfs
COMPSs/1.1.2_scratch
COMPSs/1.2
COMPSs/1.3

COMPSs/release(default)
COMPSs/trunk


$ module load COMPSs/release
load java/1.7.0u55 (PATH, MANPATH, JAVA_HOME, JAVA_ROOT, JAVA_BINDIR,
                    SDK_HOME, JDK_HOME, JRE_HOME)
load MKL/11.0.1 (LD_LIBRARY_PATH)
load PYTHON/2.7.3 (PATH, MANPATH, LD_LIBRARY_PATH, C_INCLUDE_PATH)
load COMPSs/release (PATH, MANPATH, IT_HOME)
```

The following command can be run to check if the correct COMPSs version has been loaded:

```
$ runcompss --version
COMPSs version 1.3
```

2.2 Configuration

The COMPSs module contains **all** the COMPSs dependencies, including Java, Python and MKL. Modifying any of these dependencies can cause execution failures and thus, we **do not** recomend to change them. Before running any COMPSs job please check your environment and, if needed, comment out any line inside the *.bashrc* file loading custom COMPSs, Java, Python and/or MKL modules.

The COMPSs module needs to be loaded in all the nodes that will run a COMPSs job. Consequently, the *module load* **must** be included in your *.bashrc* file. To do so please run the following command:

```
$ cat "module load COMPSs/release" >> ~/.bashrc
```

Log out and back in again to check that the file has been correctly edited:

```
$ exit
$ ssh USER@mn1.bsc.es
load java/1.7.0u55 (PATH, MANPATH, JAVA_HOME, JAVA_ROOT, JAVA_BINDIR,
                    SDK_HOME, JDK_HOME, JRE_HOME)
load MKL/11.0.1 (LD_LIBRARY_PATH)
load PYTHON/2.7.3 (PATH, MANPATH, LD_LIBRARY_PATH, C_INCLUDE_PATH)
load COMPSs/trunk (PATH, MANPATH, IT_HOME)

$ runcompss --version
COMPSs version 1.3
```

Please remember that COMPSs runs in several nodes and your current enviroment is not exported to them. Thus, all the needed environment variables **must** be loaded through the *.bashrc* file.

3 COMPSs Jobs

COMPSs jobs can be easily submited by running the **enqueue_compss** command. This command allows to configure any **runcompss** option and some particular queue options such as the queue system, the number of nodes, the wallclock time, the master working directory, the workers working directory and number of tasks per node.

Next, we provide detailed information about the *enqueue_compss* command:

```
$ enqueue_compss --help
Usage: /apps/COMPSs/1.3/Runtime/scripts/user/enqueue_compss
         [queue_system_options] [COMPSs_options]
         application_name application_arguments

* Options:
  General:
    --help, -h                              Print this help message

  Queue system configuration:
    - --exec_time=<minutes>                 Expected execution time of
                                            the application (in minutes)
                                            Default: 10

    - --num_nodes=<int>                      Number of nodes to use
                                            Default: 2

    - --queue_system=<name>                 Queue system to use:
                                            lsf | pbs | slurm
                                            Default: lsf

    - --tasks_per_node=<int>                Maximum number of simultaneous
                                            tasks running on a node
                                            Default: 16

    - --master_working_dir=<path>           Working directory of the
                                            application
                                            Default: .

    - --worker_working_dir=<name>           Worker directory.
                                            Use: scratch | gpfs
                                            Default: scratch

    - --tasks_in_master=<int>               Maximum number of tasks that
                                            the master node can run as
                                            worker. Cannot exceed
                                            tasks_per_node.
                                            Default: 0

    - --network=<name>                      Communication network for
                                            transfers:
                                            default | infiniband | data.
                                            Default: default

  Runcompss catched parameters:
```

```
  --log_level=<level>, --debug          Set the debug level: off | info
                                        | debug
                                        Default: off

  --tracing=<true|false>                Enable tracing: true | false
                                        Default: false

  --comm=<path>                         Class that implements the
                                        adaptor for communications
                                        Default: integratedtoolkit.nio
                                        .master.NIOAdaptor

  --library_path=<path>                 Non-standard directories to
                                        search for libraries (e.g.
                                        Java JVM library, Python
                                        library, C binding library)
                                        Default: .

  --classpath=<path>                    Path for the application
                                        classes / modules
                                        Default: .
Runcompss delegated parameters:

Runtime configuration options:
  --project=<path>                      Path to the project XML file
                                        Default: /opt/COMPSs/Runtime/
                                        configuration/xml/projects
                                        /project.xml

  --resources=<path>                    Path to the resources XML file
                                        Default: /opt/COMPSs/Runtime/
                                        configuration/xml/resources/
                                        resources.xml

  --lang=<name>                         Language of the application
                                        (java/c/python)
                                        Default: java

  --log_level=<level>, --debug, -d      Set the debug level: off |
                                        info | debug
                                        Default: off

Tools enablers:
  --graph=<bool>, --graph, -g           Generation of the complete
                                        graph (true/false)
                                        When no value is provided it is
                                        set to true
                                        Default: false

  --tracing=<bool>, --tracing, -t       Generation of traces
                                        (true/false)
                                        When no value is provided it is
                                        set to true
                                        Default: false

  --monitoring=<int>, --monitoring, -m  Period between monitoring
```

```
                                        samples (milliseconds)
                                        When no value is provided it is
                                        set to 2000
                                        Default: 0

  Advanced options:
    --comm=<path>                       Class that implements the
                                        adaptor for communications
                                        Default:
                                        integratedtoolkit.nio.master.
                                        NIOAdaptor

    --library_path=<path>               Non-standard directories to
                                        search for libraries (e.g.
                                        Java JVM library, Python
                                        library, C binding library)
                                        Default: .

    --classpath=<path>                  Path for the application
                                        classes / modules
                                        Default: .

    --task_count=<int>                  Only for C/Python Bindings.
                                        Maximum number of different
                                        functions/methods invoked
                                        from the application that
                                        have been selected as tasks
                                        Default: 50

    --uuid=<int>                        Preset an application UUID
                                        Default: Automatic random
                                        generation

* Application name:
    For Java applications:   Fully qualified name of the application
    For C applications:      Path to the master binary
    For Python applications: Path to the .py file containing the main
   program

* Application arguments:
    Command line arguments to pass to the application. Can be empty.
```

Please find more details on the COMPSs framework at

http://compss.bsc.es