COMP Superscalar

Installation Manual

Version: 2.0
January 18, 2017
This manual only provides information about how to install and configure COMPSs. Specifically, it details the installation process for Debian based distributions and for Red-Hat based distributions, and the steps to configure COMPSs properly.

If you are not wondering to install COMPSs please consider using our already prepared Virtual Machine available at our webpage: 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/releases/compss/latest/docs/COMPSs_User_Manual_App_Development.pdf.
Contents

1 COMP Superscalar (COMPSs) 1

2 Packages description 2
   2.1 Packages structure 2
   2.2 Packages Dependencies 3

3 Debian-based distributions 4
   3.1 Prerequisites 4
   3.2 Package Repository 4
   3.3 Installation 4
   3.4 Post installation 7

4 RedHat-based distributions (zypper) 8
   4.1 Prerequisites 8
   4.2 Package Repository 8
   4.3 Installation 8
   4.4 Post installation 10

5 RedHat-based distributions (yum) 12
   5.1 Prerequisites 12
   5.2 Package Repository 12
   5.3 Installation 12
   5.4 Post installation 14

6 Supercomputers 15
   6.1 Prerequisites 15
   6.2 Installation 15
   6.3 Configuration 16
   6.4 Post installation 17

7 Pip 21
   7.1 Prerequisites 21
   7.2 Installation 22
   7.3 Configuration 22
   7.4 Post installation 23

8 Additional Configuration 24
   8.1 Configure SSH passwordless 24
   8.2 Configure the COMPSs Cloud Connectors 25
     8.2.1 OCCI (Open Cloud Computing Interface) connector 25

9 COMPSs Removal 26
   9.1 How to uninstall or remove COMPSs 26
   9.2 How to clean repositories 26
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 Packages description

2.1 Packages structure

Despite the fact that we recommend users to install the complete COMPSs Framework, we have built different packages to allow users customize as maximum as possible their installation. Figure 1 illustrates the COMPSs packaging structure and its internal dependencies.

Figure 1: COMPSs packaging structure
## 2.2 Packages Dependencies

Next we provide a list of dependencies for each COMPSs package. The exact names may vary depending on the Linux distribution but this list provides a general overview of the COMPSs dependencies. For specific information about your distribution please check the `Depends` section at your package manager (apt, yum, zypper, etc.).

<table>
<thead>
<tr>
<th>COMPSs Framework</th>
<th>Dependencies</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMPSs Framework</td>
<td>compss-runtime, compss-bindings, compss-tools, compss-cloud</td>
</tr>
<tr>
<td>COMPSs Runtime</td>
<td>compss-engine, compss-worker</td>
</tr>
<tr>
<td>COMPSs Engine</td>
<td>openjdk-8-jre, graphviz, xdg-utils</td>
</tr>
<tr>
<td>COMPSs Worker</td>
<td>openjdk-8-jre</td>
</tr>
<tr>
<td>COMPSs Bindings</td>
<td>compss-bindings-common, compss-c-binding, compss-python-binding</td>
</tr>
<tr>
<td>COMPSs Bindings Common</td>
<td>compss-engine, libtool, automake, build-essential</td>
</tr>
<tr>
<td>COMPSs Python Binding</td>
<td>compss-bindings-common, python ((&gt;= \ 2.7)), libpython2.7</td>
</tr>
<tr>
<td>COMPSs C/C++ Binding</td>
<td>compss-binding-common, libboost-serialization-dev, libboost-iostreams-dev, libxml2-dev, csh</td>
</tr>
<tr>
<td>COMPSs Tools</td>
<td>compss-extrae, compss-monitor</td>
</tr>
<tr>
<td>COMPSs Extrae</td>
<td>compss-engine, libxml2 ((&gt;= \ 2.5)), libxml2-dev ((&gt;= \ 2.5)), gfortran, papi</td>
</tr>
<tr>
<td>COMPSs Monitor</td>
<td>compss-engine</td>
</tr>
<tr>
<td>COMPSs Cloud</td>
<td>compss-engine</td>
</tr>
</tbody>
</table>
3 Debian-based distributions

3.1 Prerequisites

The commands described on the following sections require root privileges and Internet connection.

Once the installation process is finished, please log out and back in again to complete the installation.

3.2 Package Repository

To add the package repository you can easily download our predefined lists by executing the following command:

- **ubuntu x86_64**:
  ```bash
  wget http://compss.bsc.es/releases/repofiles/repo_deb_ubuntu_x86-64.list -O /etc/apt/sources.list.d/compss-framework_x86-64.list
  ```

- **ubuntu noarch**:
  ```bash
  wget http://compss.bsc.es/releases/repofiles/repo_deb_ubuntu_noarch.list -O /etc/apt/sources.list.d/compss-framework_noarch.list
  ```

- **debian x86_64**:
  ```bash
  wget http://compss.bsc.es/releases/repofiles/repo_deb_debian_x86-64.list -O /etc/apt/sources.list.d/compss-framework_x86-64.list
  ```

- **debian noarch**:
  ```bash
  wget http://compss.bsc.es/releases/repofiles/repo_deb_debian_noarch.list -O /etc/apt/sources.list.d/compss-framework_noarch.list
  ```

Next you need to add the repository key by executing:

```bash
wget -qO - http://compss.bsc.es/repo/debs/deb-gpg-bsc-grid.pub.key | apt-key add -
```

And finally, refresh the apt-get repositories:

```bash
apt-get update
```

3.3 Installation

This section describes how to install all the available COMPSs packages. If you are willing to have a full COMPSs installation just follow the COMPSs Framework instructions and skip directly to next section.

- **COMPSs Framework**
  Contains all the COMPSs functionalities including the Runtime, all the bindings, all the tools and the cloud connectors.
  To install this package please run:
apt-get install compss-framework

• COMPSs Runtime
  Contains the COMPSs runtime to support the native functionalities. Install this package if you only need to support Java applications.
  To install this package please run:

  apt-get install compss-runtime

This package is composed of two sub-packages:

  – COMPSs Engine
    Contains the COMPSs Engine, essential to run COMPSs applications as master.
    To install this package please run:

    apt-get install compss-engine

  – COMPSs Worker
    Contains the minimum installation required to run a machine as a COMPSs worker.
    To install this package please run:

    apt-get install compss-worker

• COMPSs Bindings
  Contains all the required bindings for C/C++ and Python applications.
  To install this package please run:

  apt-get install compss-bindings

This package is composed of three sub-packages:

  – COMPSs Bindings Common
    Contains the API required for the communication between any binding and the COMPSs Runtime. It is necessary for any binding installation.
    To install this package please run:
- **COMPSs C/C++ Binding**
  Contains the C/C++ Binding
  To install this package please run:

```
apt-get install compss-bindings-common
```

- **COMPSs Python Binding**
  Contains the Python Binding
  To install this package please run:

```
apt-get install compss-c-binding
```

- **COMPSs Tools**
  Contains all the COMPSs Tools.
  To install this package please run:

```
apt-get install compss-tools
```

This package is composed of three sub-packages:

- **COMPSs Extrae**
  Contains the COMPSs Extrae tool needed to generate and process application traces.
  To install this package please run:

```
apt-get install compss-extrae
```

- **COMPSs Monitor**
  Contains the COMPSs Monitor tool needed to monitor the application execution.
  To install this package please run:

```
apt-get install compss-monitor
```
• COMPSs Cloud
  Contains all the COMPSs Connectors needed to interact with the Cloud.
  To install this package please run:

  `apt-get install compss-cloud`

### 3.4 Post installation

Once your COMPSs package has been installed remember to log out and back in again to end the installation process.

If you need to set up your machine for the first time please take a look at Section 8 for a detailed description of the additional configuration.
4 RedHat-based distributions (zypper)

4.1 Prerequisites
The commands described on the following sections require root privileges and Internet connection.
Once the installation process is finished, please log out and back in again to complete the installation.

4.2 Package Repository
To add the package repository you can easily download our predefined lists by executing the following command:

```
x86_64 : zypper addrepo -f http://compss.bsc.es/repo/rpms/stable/suse/x86_64 compss
```

And finally, refresh the repositories:

```
zypper refresh
```

4.3 Installation
This section describes how to install all the available COMPSs packages. If you are willing to have a full COMPSs installation just follow the COMPSs Framework instructions and skip directly to next section.

- **COMPSs Framework**
  Contains the all COMPSs functionalities including the Runtime, all the bindings, all the tools and the cloud connectors.
  To install this package please run:

```
zypper install compss-framework
```

- **COMPSs Runtime**
  Contains the COMPSs runtime to support the native functionalities. Install this package if you only need to support Java applications.
  To install this package please run:

```
zypper install compss-runtime
```
This package is composed of two sub-packages:

- **COMPSs Engine**
  Contains the COMPSs Engine, essential to run COMPSs applications as master.
  To install this package please run:

  ```
  zypper install compss-engine
  ```

- **COMPSs Worker**
  Contains the minimum installation required to run a machine as a COMPSs worker.
  To install this package please run:

  ```
  zypper install compss-worker
  ```

- **COMPSs Bindings**
  Contains all the required bindings for C/C++ and Python applications.
  To install this package please run:

  ```
  zypper install compss-bindings
  ```

This package is composed of three sub-packages:

- **COMPSs Bindings Common**
  Contains the API required for the communication between any binding and the COMPSs Runtime. It is necessary for any binding installation.
  To install this package please run:

  ```
  zypper install compss-bindings-common
  ```

- **COMPSs C/C++ Binding**
  Contains the C/C++ Binding
  To install this package please run:

  ```
  zypper install compss-c-binding
  ```
- **COMPSs Python Binding**
  Contains the Python Binding
  To install this package please run:

```
zypper install compss-python-binding
```

- **COMPSs Tools**
  Contains all the COMPSs Tools.
  To install this package please run:

```
zypper install compss-tools
```

This package is composed of three sub-packages:

- **COMPSs Extrae**
  Contains the COMPSs Extrae tool needed to generate and process application traces.
  To install this package please run:

```
zypper install compss-extrae
```

- **COMPSs Monitor**
  Contains the COMPSs Monitor tool needed to monitor the application execution.
  To install this package please run:

```
zypper install compss-monitor
```

- **COMPSs Cloud**
  Contains all the COMPSs Connectors needed to interact with the Cloud.
  To install this package please run:

```
zypper install compss-cloud
```

### 4.4 Post installation

Once your COMPSs package has been installed remember to log out and back in again to end the installation process.
If you need to set up your machine for the first time please take a look at Section 8 for a detailed description of the additional configuration.
5 RedHat-based distributions (yum)

5.1 Prerequisites
The commands described on the following sections require root privileges and Internet connection.

Once the installation process is finished, please log out and back in again to complete the installation.

5.2 Package Repository
To add the package repository you can easily download our predefined lists by executing the following command:

```bash
wget http://compss.bsc.es/releases/repofiles/repo_rpm_centos_x86-64.repo -O /etc/yum.repos.d/compss-framework_x86-64.repo
```

5.3 Installation
This section describes how to install all the available COMPSs packages. If you are willing to have a full COMPSs installation just follow the COMPSs Framework instructions and skip directly to next section.

- **COMPSs Framework**
  Contains the all COMPSs functionalities including the Runtime, all the bindings, all the tools and the cloud connectors.
  To install this package please run:

```
yum install compss-framework
```

- **COMPSs Runtime**
  Contains the COMPSs runtime to support the native functionalities. Install this package if you only need to support Java applications.
  To install this package please run:

```
yum install compss-runtime
```

This package is composed of two sub-packages:
– **COMPSs Engine**
Contains the COMPSs Engine, essential to run COMPSs applications as master.
To install this package please run:

```bash
yum install compss-engine
```

– **COMPSs Worker**
Contains the minimum installation required to run a machine as a COMPSs worker.
To install this package please run:

```bash
yum install compss-worker
```

• **COMPSs Bindings**
Contains all the required bindings for C/C++ and Python applications.
To install this package please run:

```bash
yum install compss-bindings
```

This package is composed of three sub-packages:

– **COMPSs Bindings Common**
Contains the API required for the communication between any binding and the COMPSs Runtime. It is necessary for any binding installation.
To install this package please run:

```bash
yum install compss-bindings-common
```

– **COMPSs C/C++ Binding**
Contains the C/C++ Binding
To install this package please run:

```bash
yum install compss-c-binding
```

– **COMPSs Python Binding**
Contains the Python Binding
To install this package please run:
• **COMPSs Tools**  
  Contains all the COMPSs Tools.  
  To install this package please run:

```
yum install compss-tools
```

This package is composed of three sub-packages:

- **COMPSs Extrae**  
  Contains the COMPSs Extrae tool needed to generate and process application traces.  
  To install this package please run:

```
yum install compss-extrae
```

- **COMPSs Monitor**  
  Contains the COMPSs Monitor tool needed to monitor the application execution.  
  To install this package please run:

```
yum install compss-monitor
```

• **COMPSs Cloud**  
  Contains all the COMPSs Connectors needed to interact with the Cloud.  
  To install this package please run:

```
yum install compss-cloud
```

### 5.4 Post installation

Once your COMPSs package has been installed remember to log out and back in again to end the installation process.

If you need to set up your machine for the first time please take a look at Section 8 for a detailed description of the additional configuration.
6 Supercomputers

The COMPSs Framework can be installed in any Supercomputer by installing its packages as in a normal distribution. The packages are ready to be reallocated so the administrators can choose the right location for the COMPSs installation.

However, if the administrators are not willing to install COMPSs through the packaging system, we also provide a COMPSs zipped file containing a pre-build script to easily install COMPSs. Next subsections provide further information about this process.

6.1 Prerequisites

In order to successfully run the installation script some dependencies must be present on the target machine. Administrators must provide the correct installation and environment of the following software:

- Autotools
- BOOST
- Java 8 JRE

The following environment variables must be defined:

- \texttt{JAVA\_HOME}
- \texttt{BOOST\_CPPFLAGS}

The tracing system can be enhanced with:

- PAPI, which provides support for hardware counters
- MPI, which speeds up the tracing merge (and enables it for huge traces)

6.2 Installation

To perform the COMPSs Framework installation please execute the following commands:

```bash
# Check out the last COMPSs release

# Unpackage COMPSs
$ tar -xvzf COMPSs_<version>.tar.gz

# Install COMPSs at your preferred target location
$ cd COMPSs
$ ./install <targetDir>

# Clean downloaded files
$ rm -r COMPSs
$ rm COMPSs_<version>.tar.gz
```
The installation script will create a COMPSs folder inside the given `<targetDir>` so the final COMPSs installation will be placed under the `<targetDir>/COMPSs` folder. Please note that if the folder already exists it will be automatically erased.

After completing the previous steps, administrators must ensure that the nodes have passwordless ssh access. If it is not the case, please contact the COMPSs team at support-compss@bsc.es.

The COMPSs package also provides a `compssenv` file that loads the required environment to allow users work more easily with COMPSs. Thus, after the installation process we recommend to source the `<targetDir>/COMPSs/compssenv` into the users `.bashrc`.

Once done, remember to log out and back in again to end the installation process.

### 6.3 Configuration

For queue system executions, COMPSs has a pre-build structure (see Figure 2) to execute applications in SuperComputers. For this purpose, users must use the `enqueue_compss` script provided in the COMPSs installation. This script has several parameters (see `enqueue_compss -h`) that allow users to customize their executions for any SuperComputer.

![Figure 2: Structure of COMPSs queue scripts. In Blue user scripts, in Green queue scripts and in Orange system dependant scripts](image)

To make this structure works, the administrators must define a configuration file for the queue system (under `<targetDir>/COMPSs/scripts/queues/cfgs/QUEUE/QUEUE.cfg`) and a configuration file for the specific SuperComputer parameters (under `<targetDir>/COMPSs/scripts/queues/cfgs/SC_NAME.cfg`). The COMPSs installation already provides queue configurations for `LSF` and `SLURM` and several examples for Super-Computer configurations.

To create a new configuration we recommend to use one of the configurations provided by COMPSs (such as the configuration for the *MareNostrum III* SuperComputer) or to contact us at support-compss@bsc.es.
6.4 Post installation

To check that COMPSs Framework has been successfully installed you may run:

```bash
# Check the COMPSs version
$ runcompss -v
COMPSs version <version>
```

For queue system executions, COMPSs provides several prebuild queue scripts that can be accessible through the `enqueue_compss` command. Users can check the available options by running:

```
enqueue_compss -h
Usage: 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
      --num_switches=<int>         Maximum number of different switches.
                                      Select 0 for no restrictions.
                                      Maximum nodes per switch: 18
                                      Only available for at least 4 nodes.
                                      Default: 0
      --tasks_per_node=<int>       Maximum number of simultaneous tasks running on a node
                                      Default: 16
      --node_memory=<MB>           Maximum node memory: disabled | <int> (MB)
                                      Default: 28
      --network=<name>             Communication network for transfers:
                                      default | ethernet | infiniband | data.
                                      Default: ethernet
      --sc_cfg=<name>              SuperComputer configuration file to use.
                                      Must exist inside queues/cfgs/
                                      Default: default
      --queue=<name>               Queue name to submit the job. Depends on the queue system.
                                      For example (MN3): bsc_cs | bsc_debug | debug | interactive
                                      Default: default
      --reservation=<name>         Reservation to use when submitting the job.
                                      Default: disabled
      --job_dependency=<jobID>     Postpone job execution until the job dependency has ended.
                                      Default: None
      --master_working_dir=<path>  Working directory of the application
                                      Default: .
      --worker_working_dir=<name | path> Worker directory. Use: scratch | gpfs | <path>
                                      Default: scratch
      --worker_in_master_tasks=<int> Maximum number of tasks that the master node
```

17
can run as worker. Cannot exceed tasks_per_node. Default: 0

`--worker_in_master_memory=<int> MB` Maximum memory in master node assigned to the worker. Cannot exceed the node_memory. Mandatory if worker_in_master_tasks is specified. Default: disabled

Extra options for the JVM of the COMPSs Worker in the Master Node. Each option separated by "," and without blank spaces (Notice the quotes) Default:

Task execution under COMPSs or Storage. Default: compss

Path to the storage configuration file

Name of the storage platform dataClay or Hecuba.

Tools enablers:

Generation of the complete graph (true/false) When no value is provided it is set to true Default: false

Set generation of traces and/or tracing level ( [ true | basic ] | advanced | false) True and basic levels will produce the same traces. When no value is provided it is set to true Default: false

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

Enables external debugger connection on the specified port (or 9999 if empty) Default: false

Task execution under COMPSs or Storage. Default: compss

Path to the storage configuration file Default: None

Path to the project XML file Default: /opt/COMPSs/Runtime/configuration/xml/projects/default_project.xml

Path to the resources XML file Default: /opt/COMPSs/Runtime/configuration/xml/resources/default_resources.xml

Language of the application (java/c/python) Default: java

Set the debug level: off | info | debug Default: off

Class that implements the adaptor for communications Supported adaptors:

Integrate toolkit.nio.master.NIOAAdaptor
Integrate toolkit.gat.master.GATAdaptor
Default: integratedtoolkit.nio.master.NIOAAdaptor
**--scheduler=**<path>

Class that implements the Scheduler for COMPSs

Supported schedulers:
- integratedtoolkit.components.impl.TaskScheduler
- integratedtoolkit.scheduler.readyscheduler.ReadyScheduler

Default: integratedtoolkit.scheduler.readyscheduler.

**--library_path=**<path>

Non-standard directories to search for libraries
(e.g. Java JVM library, Python library, C binding library)

Default: Working Directory

**--classpath=**<path>

Path for the application classes / modules

Default: Working Directory

**--base_log_dir=**<path>

Base directory to store COMPSs log files
(a .COMPSs/ folder will be created inside this location)

Default: User home

**--specific_log_dir=**<path>

Use a specific directory to store COMPSs log files
(the folder MUST exist and no sandbox is created)

Warning: Overwrites --base_log_dir option

Default: Disabled

**--uuid=**<int>

Preset an application UUID

Default: Automatic random generation

**--master_port=**<int>

Port to run the COMPSs master communications.

Only for NIO adaptor

Default: 43000

**--jvm_master_opts=**"<string>"

Extra options for the COMPSs Master JVM.

Each option separated by "," and without blank spaces (Notice the quotes)

Default:

**--jvm_workers_opts=**"<string>"

Extra options for the COMPSs Workers JVMs.

Each option separated by "," and without blank spaces (Notice the quotes)

Default: -Xms1024m,-Xmx1024m,-Xmn400m

**--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

**--pythonpath=**<path>

Additional folders or paths to add to the PYTHONPATH

Default: /home/cramonco/svn/compss/framework/trunk/compss

**--PyObject_serialize=**<bool>

Only for Python Binding. Enable the object serialization to string when possible (true/false).

Default: false

* **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.

If none of the pre-build queue configurations adapts to your infrastructure (lsf, pbs, slurm, etc.) please contact the COMPSs team at support – compss@bsc.es to find out a solution.
If you are willing to test the COMPSs Framework installation you can run any of the applications available at our application repository [https://compss.bsc.es/projects/bar](https://compss.bsc.es/projects/bar). We suggest to run the java simple application following the steps listed inside its README file.

For further information about either the installation or the usage please check the README file inside the COMPSs package.
7 Pip

7.1 Prerequisites

In order to be able to install COMPSs and PyCOMPSs with Pip the following requirements must be met:

1. Have all the dependencies (excluding the COMPSs packages) mentioned in the section 2.2 satisfied and Python pip. As an example for some distributions:

   **Fedora 25** dependencies installation command:

```
sudo dnf install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel graphviz xdg-utils libtool automake python python-libs python-pip python-devel python2-decorator boost-devel boost-serialization boost-iostreams libxml2 libxml2-devel gcc gcc-c++ gcc-gfortran tcsh @development-tools redhat-rpm-config papi
```

   # If the libxml softlink is not created during the installation of libxml2, the COMPSs installation may fail.
   # In this case, that softlink has to be created manually with the following command:
   sudo ln -s /usr/include/libxml2/libxml/ /usr/include/libxml

   **Ubuntu 16.04** dependencies installation command:

```
sudo apt-get install -y openjdk-8-jdk graphviz xdg-utils libtool automake build-essential python2.7 libpython2.7 libboost-serialization-dev libboost-iostreams-dev libxml2 libxml2-dev csh gfortran python-pip papi
```

   **OpenSuse 42.2** dependencies installation command:

```
sudo zypper install --type pattern -y devel_basis
sudo zypper install -y java-1_8_0-openjdk-headless java-1_8_0-openjdk java-1_8_0-openjdk-devel graphviz xdg-utils python python-devel libpython2_7-1_0 python-decorator libtool automake boost-devel libboost_serialization1_54_0 libboost_iostreams1_54_0 libxml2 libxml2-devel tcsh gcc-gfortran python-pip papi libpapi
```

   **Debian 8** dependencies installation command:

```
su -
echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee /etc/apt/sources.list.d/webupd8team-java.list
echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee -a /etc/apt/sources.list.d/webupd8team-java.list
apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
apt-get update
apt-get install oracle-java8-installer
apt-get install graphviz xdg-utils libtool automake build-essential python python-decorator python-pip python-dev libboost-serialization1.55.0 libboost-iostreams1.55.0 libxml2 libxml2-dev libboost-dev csh gfortran papi tools
```
CentOS 7 dependencies installation command:

```bash
sudo yum -y update
sudo yum install java-1.8.0-openjdk java-1.8.0-openjdk-devel graphviz xdg-utils libtool
automake python python-libs python-pip python-devel python2-decorator boost-devel boost-
serialization boost-iostreams libxml2 libxml2-devel gcc gcc-c++ gcc-gfortran tcsh @development-
tools redhat-rpm-config papi
sudo pip install decorator
```

2. Have a proper `JAVA_HOME` environment variable definition. This variable must contain a valid path to a Java JDK (as a remark, it must point to a JDK, not JRE). A possible value is the following:

```
user@machine:~> echo $JAVA_HOME
/usr/lib64/jvm/java-openjdk/
```

### 7.2 Installation

Depending on the machine, the installation command may vary. It must be assured that the Python2.7 pip is being executed. Some of the possible scenarios and their proper installation command are:

1. There is more than one Python version installed:

   ```
sudo -E python2.7 -m pip install compss -v
   ```

2. There is only Python2.7 installed:

   ```
sudo -E pip install compss -v
   ```

It is recommended to restart the user session once the installation process has finished. Alternatively, the following command sets all the COMPSs environment.

```
source /etc/profile.d/compss.sh
```

However, this command should be executed in every different terminal during the current user session.

### 7.3 Configuration

The steps mentioned at section 8.1 must be done in order to have a functional COMPSs and pyCOMPSs installation.
7.4 Post installation

As mentioned in section 7.2 it is recommended to restart the user session once the installation process has finished.
8 Additional Configuration

8.1 Configure SSH passwordless

By default, COMPSs uses SSH libraries for communication between nodes. Consequently, after COMPSs is installed on a set of machines, the SSH keys must be configured on those machines so that COMPSs can establish passwordless connections between them. This requires to install the OpenSSH package (if not present already) and follow these steps in each machine:

1. Generate an SSH key pair

   $ ssh-keygen -t dsa

2. Distribute the public key to all the other machines and configure it as authorized

   For every other available machine (MACHINE):
   $ scp ~/.ssh/id_dsa.pub MACHINE:./myDSA.pub
   $ ssh MACHINE "cat .myDSA.pub >> ~/.ssh/authorized_keys; rm .myDSA.pub"

3. Check that passwordless SSH connections are working fine

   For every other available machine (MACHINE):
   $ ssh MACHINE

   For example, considering the cluster shown in Figure 3, users will have to execute the following commands to grant free ssh access between any pair of machines:
8.2 Configure the COMPSs Cloud Connectors

This section provides information about the additional configuration needed for some Cloud Connectors.

8.2.1 OCCI (Open Cloud Computing Interface) connector

In order to execute a COMPSs application using cloud resources, the rOCCI (Ruby OCCI) connector has to be configured properly. The connector uses the rOCCI CLI client (upper versions from 4.2.5) which has to be installed in the node where the COMPSs main application runs. The client can be installed following the instructions detailed at [http://appdb.egi.eu/store/software/rocci.cli](http://appdb.egi.eu/store/software/rocci.cli)
9 COMPSs Removal

9.1 How to uninstall or remove COMPSs

COMPSs can be easily uninstalled via the Linux Packaging Tools by running the following commands:

Debian: apt-get remove compss-framework
RedHat (zypper): zypper remove compss-framework
RedHat (yum): yum remove compss-framework

Notice that some of the COMPSs packages are meta-packages and, thus, you will need to manually uninstall all the COMPSs packages or use the autoremove tools:

Debian: apt-get autoremove
RedHat (zypper): zypper remove --clean-deps compss-framework
RedHat (yum): yum autoremove

In Debian based distributions uninstalling COMPSs will not erase your configuration files. If you are willing to completely remove COMPSs please remember to use the purge option:

Debian: apt-get purge compss-framework

9.2 How to clean repositories

During the installation process you may have added the COMPSs repository. If you want to clean your repository list please erase the compss list by executing the following commands:

Debian:
$ rm -f /etc/apt/sources.list.d/compss-framework_*.list
$ apt-get update

RedHat (zypper):
$ zypper removerepo compss
$ zypper refresh

RedHat (yum):
$ rm -f /etc/yum.repos.d/compss-framework_*.repo
Please find more details on the COMPSs framework at

http://compss.bsc.es