COMP Superscalar

COMPSs Sample Applications

Version: 2.0

November 11, 2016

This manual only provides Java, Python and C/C++ sample applications that can be used as a reference. Specifically, it details the applications' code and different ways to execute and analyse them (by using different runcompss flags).

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/` .

Contents

List of Figures

List of Tables

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 Java Sample applications

The first two examples in this section are simple applications developed in COMPSs to easily illustrate how to code, compile and run COMPSs applications. These applications are executed locally and show different ways to take advantage of all the COMPSs features.

The rest of the examples are more elaborated and consider the execution in a cloud platform where the VMs mount a common storage on **/sharedDisk** directory. This is useful in the case of applications that require working with big files, allowing to transfer data only once, at the beginning of the execution, and to enable the application to access the data directly during the rest of the execution.

The Virtual Machine available at our webpage (`http://compss.bsc.es/`) provides a development environment with all the applications listed in the following sections. The codes of all the applications can be found under the */home/compss/workspace_java/* folder.

2.1 Hello World

The Hello Wolrd is a Java application that creates a task and prints a Hello World! message. Its purpose is to clarify that the COMPSs tasks output is redirected to the job files and it is **not** available at the standard output.

Next we provide the important parts of the application's code.

```
// hello.Hello

public static void main(String[] args) throws Exception {
  // Check and get parameters
  if (args.length != 0) {
    usage();
    throw new Exception("[ERROR] Incorrect number of parameters");
  }

  // Hello World from main application
  System.out.println("Hello World! (from main application)");

  // Hello World from a task
  HelloImpl.sayHello();
}
```

As shown in the main code, this application has no input arguments.

```
// hello.HelloImpl

public static void sayHello() {
  System.out.println("Hello World! (from a task)");
}
```

Remember that, to run with COMPSs, java applications must provide an interface. For simplicity, in this example, the content of the interface only declares the task which has no parameters:

```
// hello.HelloItf
```

```
  @Method(declaringClass = "hello.HelloImpl")
  void sayHello(
  );
```

Notice that there is a first Hello World message printed from the main code and, a second one, printed inside a task. When executing sequentially this application users will be able to see both messages at the standard output. However, when executing this application with COMPSs, users will only see the message from the main code at the standard output. The message printed from the task will be stored inside the job log files.

Let's try it. First we proceed to compile the code by running the following instructions:

```
compss@bsc:~$ cd ~/workspace_java/hello/src/main/java/hello/
compss@bsc:~/workspace_java/hello/src/main/java/hello$ javac *.java
compss@bsc:~/workspace_java/hello/src/main/java/hello$ cd ..
compss@bsc:~/workspace_java/hello/src/main/java$ jar cf hello.jar hello
compss@bsc:~/workspace_java/hello/src/main/java$ mv hello.jar ~/workspace_java/hello/jar/
```

Alternatively, this example application is prepared to be compiled with *maven*:

```
compss@bsc:~$ cd ~/workspace_java/hello/
compss@bsc:~/workspace_java/hello$ mvn clean package
```

Once done, we can sequentially execute the application by directly invoking the *jar* file.

```
compss@bsc:~$ cd ~/workspace_java/hello/jar/
compss@bsc:~/workspace_java/hello/jar$ java -cp hello.jar hello.Hello
Hello World! (from main application)
Hello World! (from a task)
```

And we can also execute the application with COMPSs:

```
compss@bsc:~$ cd ~/workspace_java/hello/jar/
compss@bsc:~/workspace_java/hello/jar$ runcompss -d hello.Hello
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing hello.Hello --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Deploying COMPSs Runtime v<version>
[   API]  -  Starting COMPSs Runtime v<version>
[   API]  -  Initializing components
[   API]  -  Ready to process tasks
Hello World! (from main application)
[   API]  -  Creating task from method sayHello in hello.HelloImpl
[   API]  -  There is 0 parameter
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Stop IT reached
[   API]  -  Stopping AP...
[   API]  -  Stopping TD...
[   API]  -  Stopping Comm...
```

```
[   API]  -  Execution Finished

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

Notice that the COMPSs execution is using the *-d* option to allow the job logging. Thus, we can check out the application jobs folder to look for the task output.

```
compss@bsc:~$ cd ~/.COMPSs/hello.Hello_01/jobs/
compss@bsc:~/.COMPSs/hello.Hello_01/jobs$ ls -1
job1_NEW.err
job1_NEW.out
compss@bsc:~/.COMPSs/hello.Hello_01/jobs$ cat job1_NEW.out
compss@bsc:~/.COMPSs/hello.Hello_02/jobs$ cat job1_NEW.out
[DEBUG|integratedtoolkit.Worker|Executor] Creating sandbox for job 1
WORKER - Parameters of execution:
  * Method class: hello.HelloImpl
  * Method name: sayHello
  * Parameter types:
  * Parameter values:
Hello World! (from a task)
[DEBUG|integratedtoolkit.Worker|Executor] Removing sandbox for job 1
```

2.2 Simple

The Simple application is a Java application that increases a counter by means of a task. The counter is stored inside a file that is transferred to the worker when the task is executed. Thus, the tasks inferface is defined as follows:

```
// simple.SimpleItf

@Method(declaringClass = "simple.SimpleImpl")
void increment(
  @Parameter(type = Type.FILE, direction = Direction.INOUT) String file
);
```

Next we also provide the invocation of the task from the main code and the increment's method code.

```
// simple.Simple

public static void main(String[] args) throws Exception {
  // Check and get parameters
  if (args.length != 1) {
    usage();
    throw new Exception("[ERROR] Incorrect number of parameters");
  }
  int initialValue = Integer.parseInt(args[0]);

  // Write value
  FileOutputStream fos = new FileOutputStream(fileName);
  fos.write(initialValue);
  fos.close();
  System.out.println("Initial counter value is " + initialValue);

  //Execute increment
  SimpleImpl.increment(fileName);
```

```
    // Write new value
    FileInputStream fis = new FileInputStream(fileName);
    int finalValue = fis.read();
    fis.close();
    System.out.println("Final counter value is " + finalValue);
  }
```

```
  // simple.SimpleImpl

  public static void increment(String counterFile) throws FileNotFoundException, IOException {
    // Read value
    FileInputStream fis = new FileInputStream(counterFile);
    int count = fis.read();
    fis.close();

    // Write new value
    FileOutputStream fos = new FileOutputStream(counterFile);
    fos.write(++count);
    fos.close();
  }
```

Finally, to compile and execute this application users must run the following commands:

```
compss@bsc:~$ cd ~/workspace_java/simple/src/main/java/simple/
compss@bsc:~/workspace_java/simple/src/main/java/simple$ javac *.java
compss@bsc:~/workspace_java/simple/src/main/java/simple$ cd ..
compss@bsc:~/workspace_java/simple/src/main/java$ jar cf simple.jar simple
compss@bsc:~/workspace_java/simple/src/main/java$ mv simple.jar ~/workspace_java/simple/jar/

compss@bsc:~$ cd ~/workspace_java/simple/jar
compss@bsc:~/workspace_java/simple/jar$ runcompss simple.Simple 1
compss@bsc:~/workspace_java/simple/jar$ runcompss simple.Simple 1
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing simple.Simple --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version>
Initial counter value is 1
Final counter value is 2
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

2.3 Increment

The Increment application is a Java application that increases N times three different counters. Each increase step is developed by a separated task. The purpose of this application is to show parallelism between the three counters.

Next we provide the main code of this application. The code inside the *increment* task is the same than the previous example.

```java
// increment.Increment
public static void main(String[] args) throws Exception {
  // Check and get parameters
  if (args.length != 4) {
    usage();
    throw new Exception("[ERROR] Incorrect number of parameters");
  }
  int N = Integer.parseInt(args[0]);
  int counter1 = Integer.parseInt(args[1]);
  int counter2 = Integer.parseInt(args[2]);
  int counter3 = Integer.parseInt(args[3]);

  // Initialize counter files
  System.out.println("Initial counter values:");
  initializeCounters(counter1, counter2, counter3);

  // Print initial counters state
  printCounterValues();

  // Execute increment tasks
  for (int i = 0; i < N; ++i) {
    IncrementImpl.increment(fileName1);
    IncrementImpl.increment(fileName2);
    IncrementImpl.increment(fileName3);
  }

  // Print final counters state (sync)
  System.out.println("Final counter values:");
  printCounterValues();
}
```

As shown in the main code, this application has 4 parameters that stand for:

1. **N:** Number of times to increase a counter
2. **InitialValue1:** Initial value for counter 1
3. **InitialValue2:** Initial value for counter 2
4. **InitialValue3:** Initial value for counter 3

Next we will compile and run the Increment application with the *-g* option to be able to generate the final graph at the end of the execution.

```
compss@bsc:~$ cd ~/workspace_java/increment/src/main/java/increment/
compss@bsc:~/workspace_java/increment/src/main/java/increment$ javac *.java
compss@bsc:~/workspace_java/increment/src/main/java/increment$ cd ..
compss@bsc:~/workspace_java/increment/src/main/java$ jar cf increment.jar increment
compss@bsc:~/workspace_java/increment/src/main/java$ mv increment.jar ~/workspace_java/increment/jar/

compss@bsc:~$ cd ~/workspace_java/increment/jar
compss@bsc:~/workspace_java/increment/jar$ runcompss -g increment.Increment 10 1 2 3
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing increment.Increment --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version>
Initial counter values:
- Counter1 value is 1
- Counter2 value is 2
- Counter3 value is 3
```

```
Final counter values:
- Counter1 value is 11
- Counter2 value is 12
- Counter3 value is 13
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

By running the *gengraph* command users can obtain the task graph of the above execution. Next we provide the set of commands to obtain the graph show in Figure 1.

```
compss@bsc:~$ cd ~/.COMPSs/increment.Increment_01/monitor/
compss@bsc:~/.COMPSs/increment.Increment_01/monitor$ gengraph complete_graph.dot
compss@bsc:~/.COMPSs/increment.Increment_01/monitor$ evince complete_graph.pdf
```

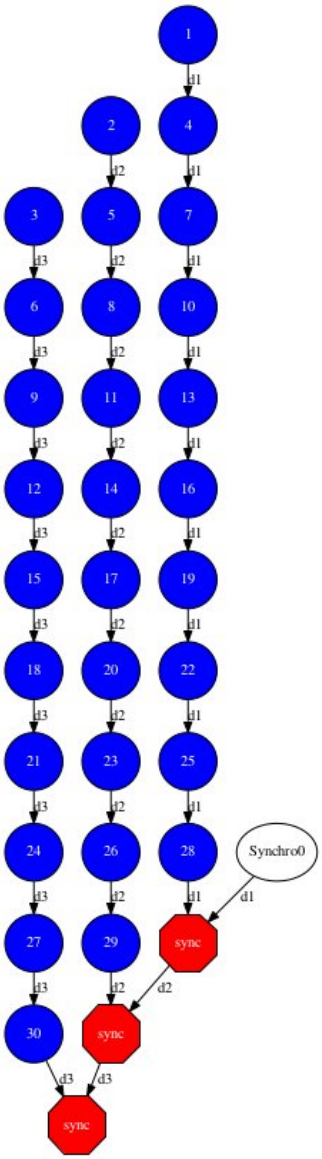

Figure 1: Java increment tasks graph

2.4 Matrix multiplication

The Matrix Multiplication (Matmul) is a pure Java application that multiplies two matrices in a direct way. The application creates 2 matrices of N x N size initialized with values, and multiply the matrices by blocks.

This application provides three different implementations that only differ on the way of storing the matrix:

1. **matmul.objects.Matmul** Matrix stored by means of objects
2. **matmul.files.Matmul** Matrix stored in files
3. **matmul.arrays.Matmul** Matrix represented by an array

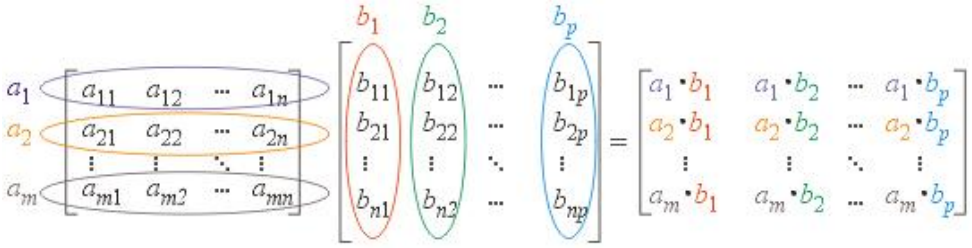

Figure 2: Matrix multiplication

In all the implementations the multiplication is implemented in the multiplyAccumulative method that is thus selected as the task to be executed remotely. As example, we we provide next the task implementation and the tasks interface for the objects implementation.

```
// matmul.objects.Block
public void multiplyAccumulative(Block a, Block b) {
  for (int i = 0; i < M; i++) {
    for (int j = 0; j < M; j++) {
      for (int k = 0; k < M; k++) {
        data[i][j] += a.data[i][k]*b.data[k][j];
      }
    }
  }
}
```

```
// matmul.objects.MatmulItf
@Method(declaringClass = "matmul.objects.Block")
void multiplyAccumulative(
  @Parameter Block a,
  @Parameter Block b
);
```

In order to run the application the matrix dimension (number of blocks) and the dimension of each block have to be supplied. Consequently, any of the implementations must be executed by running the following command.

```
compss@bsc:~$ runcompss matmul.<IMPLEMENTATION_TYPE>.Matmul <matrix_dim> <block_dim>
```

Finally, we provide an example of execution for each implementation.

```
compss@bsc:~$ cd ~/workspace_java/matmul/jar/
compss@bsc:~/workspace_java/matmul/jar$ runcompss matmul.objects.Matmul 8 4
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing matmul.objects.Matmul --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version>
Running with the following parameters:
- N: 8
```

```
- M: 4
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

```
compss@bsc:~$ cd ~/workspace_java/matmul/jar/
compss@bsc:~/workspace_java/matmul/jar$ runcompss matmul.files.Matmul 8 4
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing matmul.files.Matmul --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version>
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

```
compss@bsc:~$ cd ~/workspace_java/matmul/jar/
compss@bsc:~/workspace_java/matmul/jar$ runcompss matmul.arrays.Matmul 8 4
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing matmul.arrays.Matmul --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version>
Running with the following parameters:
- N: 8
- M: 4
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

2.5 Sparse LU decomposition

SparseLU multiplies two matrices using the factorization method of LU decomposition, which factorizes a matrix as a product of a lower triangular matrix and an upper one.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

Figure 3: Sparse LU decomposition

The matrix is divided into N x N blocks on where 4 types of operations will be applied modifying the blocks: **lu0**, **fwd**, **bdiv** and **bmod**. These four operations are implemented

in four methods that are selecetd as the tasks that will be executed remotely. In order to run the application the matrix dimension has to be provided.

As the previous application, the sparseLU is provided in three different implementations that only differ on the way of storing the matrix:

1. **sparseLU.objects.SparseLU** Matrix stored by means of objects
2. **sparseLU.files.SparseLU** Matrix stored in files
3. **sparseLU.arrays.SparseLU** Matrix represented by an array

Thus, the commands needed to execute the application is with each implementation are:

```
compss@bsc:~$ cd workspace_java/sparseLU/jar/
compss@bsc:~/workspace_java/sparseLU/jar$ runcompss sparseLU.objects.SparseLU 16 8
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing sparseLU.objects.SparseLU --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version> (build yyyyMMdd-XXXX.XXXX)
[LOG] Running with the following parameters:
[LOG]  - Matrix Size: 16
[LOG]  - Block Size:  8
[LOG] Initializing Matrix
[LOG] Computing SparseLU algorithm on A
[LOG] Main program finished.
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

```
compss@bsc:~$ cd workspace_java/sparseLU/jar/
compss@bsc:~/workspace_java/sparseLU/jar$ runcompss sparseLU.files.SparseLU 4 8
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing sparseLU.files.SparseLU --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version> (build yyyyMMdd-XXXX.XXXX)
[LOG] Running with the following parameters:
[LOG]  - Matrix Size: 4
[LOG]  - Block Size:  8
[LOG] Initializing Matrix
[LOG] Computing SparseLU algorithm on A
[LOG] Main program finished.
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

```
compss@bsc:~$ cd workspace_java/sparseLU/jar/
compss@bsc:~/workspace_java/sparseLU/jar$ runcompss sparseLU.arrays.SparseLU 8 8
```

```
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing sparseLU.arrays.SparseLU --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version> (build yyyyMMdd-XXXX.XXXX)
[LOG] Running with the following parameters:
[LOG]  - Matrix Size: 8
[LOG]  - Block Size:  8
[LOG] Initializing Matrix
[LOG] Computing SparseLU algorithm on A
[LOG] Main program finished.
[   API]  -  No more tasks for app 1
[   API]  -  Getting Result Files 1
[   API]  -  Execution Finished

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

2.6 BLAST Workflow

BLAST is a widely-used bioinformatics tool for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or the nucleotides of DNA sequences with sequence databases, identifying sequences that resemble the query sequence above a certain threshold. The work performed by the COMPSs Blast workflow is computationally intensive and embarrassingly parallel.

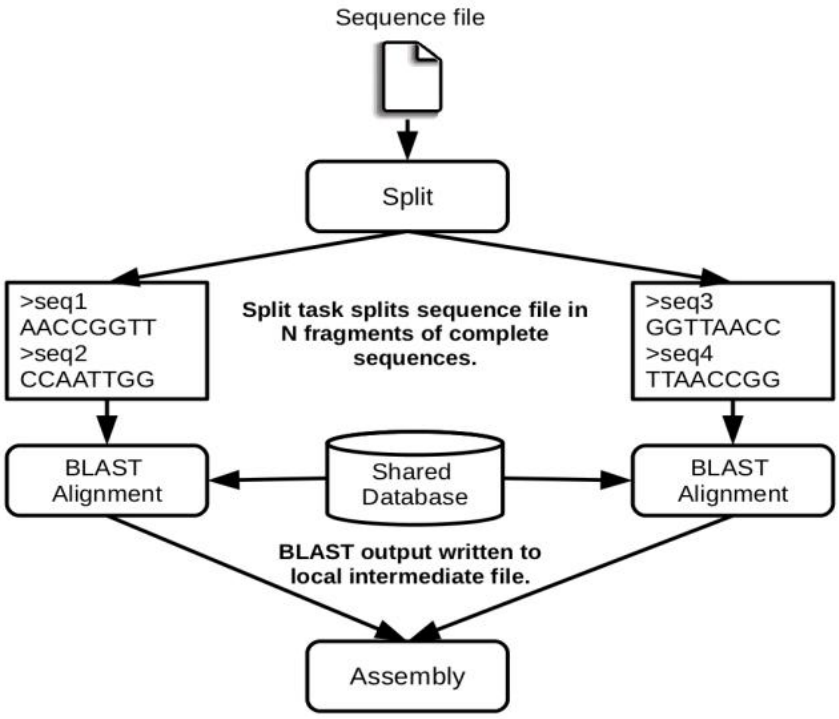

Figure 4: The COMPSs Blast workflow

The workflow describes the three blocks of the workflow implemented in the **Split**, **Align** and **Assembly** methods. The second one is the only method that is chosen to

be executed remotely, so it is the unique method defined in the interface file. The **Split** method chops the query sequences file in N fragments, **Align** compares each sequence fragment against the database by means of the Blast binary, and **Assembly** combines all intermediate files into a single result file.

This application uses a database that will be on the shared disk space avoiding transferring the entire database (which can be large) between the virtual machines.

```
compss@bsc:~$ cp ~/workspace/blast/package/Blast.tar.gz /home/compss/
compss@bsc:~$ tar xzf Blast.tar.gz
```

The command line to execute the workflow:

```
compss@bsc:~$ runcompss blast.Blast <debug>
                                    <bin_location>
                                    <database_file>
                                    <sequences_file>
                                    <frag_number>
                                    <tmpdir>
                                    <output_file>
```

Where:

- **debug**: The debug flag of the application (true or false).
- **bin_location**: Path of the Blast binary.
- **database_file**: Path of database file; the shared disk **/sharedDisk/** is suggested to avoid big data transfers.
- **sequences_file**: Path of sequences file.
- **frag_number**: Number of fragments of the original sequence file, this number determines the number of parallel Align tasks.
- **tmpdir**: Temporary directory (**/home/compss/tmp/**).
- **output_file**: Path of the result file.

Example:

```
compss@bsc:~$ runcompss blast.Blast true
                        /home/compss/workspace_java/blast/binary/blastall
                        /sharedDisk/Blast/databases/swissprot/swissprot
                        /sharedDisk/Blast/sequences/sargasso_test.fasta
                        4
                        /tmp/
                        /home/compss/out.txt
```

3 Python Sample applications

The first two examples in this section are simple applications developed in COMPSs to easily illustrate how to code, compile and run COMPSs applications. These applications are executed locally and show different ways to take advantage of all the COMPSs features.

The rest of the examples are more elaborated and consider the execution in a cloud platform where the VMs mount a common storage on **/sharedDisk** directory. This is useful in the case of applications that require working with big files, allowing to transfer data only once, at the beginning of the execution, and to enable the application to access the data directly during the rest of the execution.

The Virtual Machine available at our webpage (`http://compss.bsc.es/`) provides a development environment with all the applications listed in the following sections. The codes of all the applications can be found under the */home/compss/workspace_python/* folder.

3.1 Simple

The Simple application is a Python application that increases a counter by means of a task. The counter is stored inside a file that is transfered to the worker when the task is executed. Next, we provide the main code and the task declaration:

```
def main_program():
    from pycompss.api.api import compss_open

    # Check and get parameters
    if len(sys.argv) != 2:
        usage()
        exit(-1)
    initialValue = sys.argv[1]
    fileName="counter"

    # Write value
    fos = open(fileName, 'w')
    fos.write(initialValue)
    fos.close()
    print "Initial counter value is " + initialValue

    # Execute increment
    increment(fileName)

    # Write new value
    fis = compss_open(fileName, 'r+')
    finalValue = fis.read()
    fis.close()
    print "Final counter value is " + finalValue
```

```
@task(filePath = FILE_INOUT)
def increment(filePath):
    # Read value
    fis = open(filePath, 'r')
    value = fis.read()
    fis.close()

    # Write value
    fos = open(filePath, 'w')
```

```
    fos.write(str(int(value) + 1))
    fos.close()
```

The simple application can be executed by invoking the runcompss command with the *–lang=python* flag. The following lines provide an example of its execution.

```
compss@bsc:~$ cd ~/workspace_python/simple/
compss@bsc:~/workspace_python/simple$ runcompss --lang=python ~/workspace_python/simple/simple.py 1
Using default location for project file: /opt/COMPSs/Runtime/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/configuration/xml/resources/resources.xml

----------------- Executing simple.py --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Deploying COMPSs Runtime v<version>
[   API]  -  Starting COMPSs Runtime v<version>
Initial counter value is 1
Final counter value is 2
[   API]  -  No more tasks for app 0
[   API]  -  Getting Result Files 0
[   API]  -  Execution Finished

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

3.2 Increment

The Increment application is a Python application that increases N times three different counters. Each increase step is developed by a separated task. The purpose of this application is to show parallelism between the three counters.

Next we provide the main code of this application. The code inside the *increment* task is the same than the previous example.

```
def main_program():
    # Check and get parameters
    if len(sys.argv) != 5:
        usage()
        exit(-1)
    N = int(sys.argv[1])
    counter1 = int(sys.argv[2])
    counter2 = int(sys.argv[3])
    counter3 = int(sys.argv[4])

    # Initialize counter files
    initializeCounters(counter1, counter2, counter3)
    print "Initial counter values:"
    printCounterValues()

    # Execute increment
    for i in range(N):
        increment(FILENAME1)
        increment(FILENAME2)
        increment(FILENAME3)

    # Write final counters state (sync)
    print "Final counter values:"
    printCounterValues()
```

As shown in the main code, this application has 4 parameters that stand for:

1. **N:** Number of times to increase a counter
2. **counter1:** Initial value for counter 1
3. **counter2:** Initial value for counter 2
4. **counter3:** Initial value for counter 3

Next we run the Increment application with the *-g* option to be able to generate the final graph at the end of the execution.

```
compss@bsc:~/workspace_python/increment$ runcompss --lang=python -g ~/workspace_python/increment/increment.
    py 10 1 2 3
Using default location for project file: /opt/COMPSs/Runtime/scripts/configuration/xml/projects/project.xml
Using default location for resources file: /opt/COMPSs/Runtime/scripts/configuration/xml/resources/
    resources.xml

----------------- Executing increment.py --------------------------

WARNING: IT Properties file is null. Setting default values
[   API]  -  Deploying COMPSs Runtime v<version>
[   API]  -  Starting COMPSs Runtime v<version>
Initial counter values:
- Counter1 value is 1
- Counter1 value is 2
- Counter1 value is 3
Final counter values:
- Counter1 value is 11
- Counter1 value is 12
- Counter1 value is 13
[   API]  -  No more tasks for app 0
[   API]  -  Getting Result Files 0
[   API]  -  Execution Finished

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

By running the *gengraph* command users can obtain the task graph of the above execution. Next we provide the set of commands to obtain the graph show in Figure 5.

```
compss@bsc:~$ cd ~/.COMPSs/increment.py_01/monitor/
compss@bsc:~/.COMPSs/increment.py_01/monitor$ gengraph complete_graph.dot
compss@bsc:~/.COMPSs/increment.py_01/monitor$ evince complete_graph.pdf
```

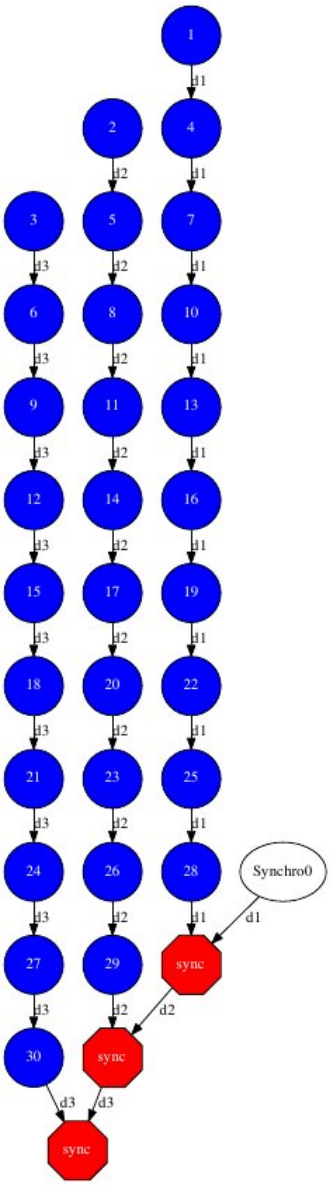

Figure 5: Python increment tasks graph

4 C/C++ Sample applications

The first two examples in this section are simple applications developed in COMPSs to easily illustrate how to code, compile and run COMPSs applications. These applications are executed locally and show different ways to take advantage of all the COMPSs features.

The rest of the examples are more elaborated and consider the execution in a cloud platform where the VMs mount a common storage on **/sharedDisk** directory. This is useful in the case of applications that require working with big files, allowing to transfer data only once, at the beginning of the execution, and to enable the application to access the data directly during the rest of the execution.

The Virtual Machine available at our webpage (`http://compss.bsc.es/`) provides a development environment with all the applications listed in the following sections. The codes of all the applications can be found under the */home/compss/workspace_c/* folder.

4.1 Simple

The Simple application is a C application that increases a counter by means of a task. The counter is stored inside a file that is transfered to the worker when the task is executed. Thus, the tasks inferface is defined as follows:

```
// simple.idl
interface simple {
  void increment(inout File filename);
};
```

Next we also provide the invocation of the task from the main code and the increment's method code.

```
// simple.cc

int main(int argc, char *argv[]) {
    // Check and get parameters
    if (argc != 2) {
  usage();
  return -1;
    }
    string initialValue = argv[1];
    file fileName = strdup(FILE_NAME);

    // Init compss
    compss_on();

    // Write file
    ofstream fos (fileName);
    if (fos.is_open()) {
  fos << initialValue << endl;
  fos.close();
    } else {
  cerr << "[ERROR] Unable to open file" << endl;
  return -1;
    }
    cout << "Initial counter value is " << initialValue << endl;

    // Execute increment
    increment(&fileName);
    compss_wait_on(fileName);
```

```
    // Read new value
    string finalValue;
    ifstream fis (fileName);
    if (fis.is_open()) {
  if (getline(fis, finalValue)) {
      cout << "Final counter value is " << finalValue << endl;
      fis.close();
  } else {
      cerr << "[ERROR] Unable to read final value" << endl;
      fis.close();
      return -1;
  }
    } else {
  cerr << "[ERROR] Unable to open file" << endl;
  return -1;
    }

    // Close COMPSs and end
    compss_off();
    return 0;
}
```

```
//simple-functions.cc

void increment(file *fileName) {
    cout << "INIT TASK" << endl;
    cout << "Param: " << *fileName << endl;
    // Read value
    char initialValue;
    ifstream fis (*fileName);
    if (fis.is_open()) {
  if (fis >> initialValue) {
      fis.close();
  } else {
      cerr << "[ERROR] Unable to read final value" << endl;
      fis.close();
  }
  fis.close();
    } else {
  cerr << "[ERROR] Unable to open file" << endl;
    }

    // Increment
    cout << "INIT VALUE: " << initialValue << endl;
    int finalValue = ((int)(initialValue) - (int)('0')) + 1;
    cout << "FINAL VALUE: " << finalValue << endl;

    // Write new value
    ofstream fos (*fileName);
    if (fos.is_open()) {
  fos << finalValue << endl;
  fos.close();
    } else {
  cerr << "[ERROR] Unable to open file" << endl;
    }
    cout << "END TASK" << endl;
}
```

Finally, to compile and execute this application users must run the following commands:

```
compss@bsc:~$ cd ~/workspace_c/simple/
```

```
compss@bsc:~/workspace_c/simple$ buildapp simple
compss@bsc:~/workspace_c/simple$ runcompss --lang=c --project=./xml/project.xml --resources=./xml/resources
    .xml ~/workspace_c/simple/master/simple 1

----------------- Executing simple --------------------------

JVM_OPTION_FILE: /tmp/tmp.X3MVgWY1L1
IT_HOME: /opt/COMPSs/Runtime/
Args: 1

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version> (build yyyyMMdd-XXXX.XXXX)
Initial counter value is 1
[   BINDING]  -  @GS_register  -  Ref: 0x7ffe027514c8
[   BINDING]  -  @GS_register  -  ENTRY ADDED
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: counter
[   BINDING]  -  @compss_wait_on  -  Entry.type: 0
[   BINDING]  -  @compss_wait_on  -  Entry.classname: File
[   BINDING]  -  @compss_wait_on  -  Entry.filename: counter
[   BINDING]  -  @compss_wait_on  -  Runtime filename: /home/compss/.COMPSs/simple_01/tmpFiles/
    d1v2_1446817611279.IT
[   BINDING]  -  @compss_wait_on  -  File renaming: /home/compss/.COMPSs/simple_01/tmpFiles/
    d1v2_1446817611279.IT to counter
Final counter value is 2
[   API]  -  No more tasks for app 0
[   API]  -  Getting Result Files 0
[   API]  -  Execution Finished

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

4.2 Increment

The Increment application is a C application that increases N times three different counters. Each increase step is developed by a separated task. The purpose of this application is to show parallelism between the three counters.

Next we provide the main code of this application. The code inside the *increment* task is the same than the previous example.

```
// increment.cc

int main(int argc, char *argv[]) {
    // Check and get parameters
    if (argc != 5) {
  usage();
  return -1;
    }
    int N = atoi( argv[1] );
    string counter1 = argv[2];
    string counter2 = argv[3];
    string counter3 = argv[4];

    // Init COMPSs
    compss_on();

    // Initialize counter files
    file fileName1 = strdup(FILE_NAME1);
    file fileName2 = strdup(FILE_NAME2);
    file fileName3 = strdup(FILE_NAME3);
    initializeCounters(counter1, counter2, counter3, fileName1, fileName2, fileName3);

    // Print initial counters state
```

```
    cout << "Initial counter values: " << endl;
    printCounterValues(fileName1, fileName2, fileName3);

    // Execute increment tasks
    for (int i = 0; i < N; ++i) {
  increment(&fileName1);
  increment(&fileName2);
  increment(&fileName3);
    }

    // Sync master
    compss_wait_on(fileName1);
    compss_wait_on(fileName2);
    compss_wait_on(fileName3);

    // Print final state
    cout << "Final counter values: " << endl;
    printCounterValues(fileName1, fileName2, fileName3);

    // Stop COMPSs
    compss_off();

    return 0;
}
```

As shown in the main code, this application has 4 parameters that stand for:

1. **N:** Number of times to increase a counter
2. **counter1:** Initial value for counter 1
3. **counter2:** Initial value for counter 2
4. **counter3:** Initial value for counter 3

Next we will compile and run the Increment application with the *-g* option to be able to generate the final graph at the end of the execution.

```
compss@bsc:~$ cd ~/workspace_c/increment/
compss@bsc:~/workspace_c/increment$ buildapp increment
compss@bsc:~/workspace_c/increment$ runcompss --lang=c -g --project=./xml/project.xml --resources=./xml/
    resources.xml ~/workspace_c/increment/master/increment 10 1 2 3

----------------- Executing increment --------------------------

JVM_OPTION_FILE: /tmp/tmp.TsZ1Y3j5bM
IT_HOME: /opt/COMPSs/Runtime
Args: 10 1 2 3

WARNING: IT Properties file is null. Setting default values
[   API]  -  Starting COMPSs Runtime v<version> (build yyyyMMdd-XXXX.XXXX)
Initial counter values:
- Counter1 value is 1
- Counter2 value is 2
- Counter3 value is 3
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY ADDED
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY ADDED
[   BINDING]  -  @GS_register  -  Entry.type: 0
```

```
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY ADDED
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
```

```
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file1.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596b8
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file2.txt
[   BINDING]  -  @GS_register  -  Ref: 0x7ffdd12596c0
[   BINDING]  -  @GS_register  -  ENTRY FOUND
[   BINDING]  -  @GS_register  -  Entry.type: 0
```

```
[   BINDING]  -  @GS_register  -  Entry.classname: File
[   BINDING]  -  @GS_register  -  Entry.filename: file3.txt
[   BINDING]  -  @compss_wait_on  -  Entry.type: 0
[   BINDING]  -  @compss_wait_on  -  Entry.classname: File
[   BINDING]  -  @compss_wait_on  -  Entry.filename: file1.txt
[   BINDING]  -  @compss_wait_on  -  Runtime filename: /home/compss/.COMPSs/increment_01/tmpFiles/
    d1v11_1446817729367.IT
[   BINDING]  -  @compss_wait_on  -  File renaming: /home/compss/.COMPSs/increment_01/tmpFiles/
    d1v11_1446817729367.IT to file1.txt
[   BINDING]  -  @compss_wait_on  -  Entry.type: 0
[   BINDING]  -  @compss_wait_on  -  Entry.classname: File
[   BINDING]  -  @compss_wait_on  -  Entry.filename: file2.txt
[   BINDING]  -  @compss_wait_on  -  Runtime filename: /home/compss/.COMPSs/increment_01/tmpFiles/
    d2v11_1446817729367.IT
[   BINDING]  -  @compss_wait_on  -  File renaming: /home/compss/.COMPSs/increment_01/tmpFiles/
    d2v11_1446817729367.IT to file2.txt
[   BINDING]  -  @compss_wait_on  -  Entry.type: 0
[   BINDING]  -  @compss_wait_on  -  Entry.classname: File
[   BINDING]  -  @compss_wait_on  -  Entry.filename: file3.txt
[   BINDING]  -  @compss_wait_on  -  Runtime filename: /home/compss/.COMPSs/increment_01/tmpFiles/
    d3v11_1446817729367.IT
[   BINDING]  -  @compss_wait_on  -  File renaming: /home/compss/.COMPSs/increment_01/tmpFiles/
    d3v11_1446817729367.IT to file3.txt
Final counter values:
- Counter1 value is 2
- Counter2 value is 3
- Counter3 value is 4
[   API]  -  No more tasks for app 0
[   API]  -  Getting Result Files 0
[   API]  -  Execution Finished

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

By running the *gengraph* command users can obtain the task graph of the above execution. Next we provide the set of commands to obtain the graph show in Figure 6.

```
compss@bsc:~$ cd ~/.COMPSs/increment_01/monitor/
compss@bsc:~/.COMPSs/increment_01/monitor$ gengraph complete_graph.dot
compss@bsc:~/.COMPSs/increment_01/monitor$ evince complete_graph.pdf
```

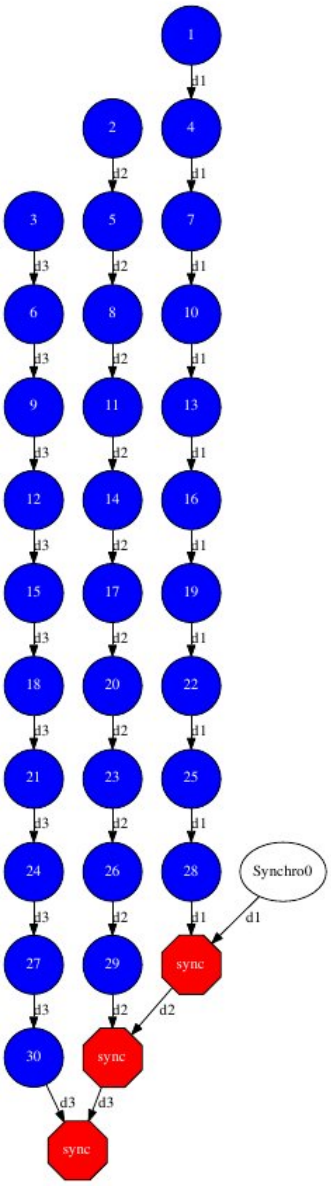

Figure 6: C increment tasks graph

Please find more details on the COMPSs framework at

http://compss.bsc.es