Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación

GRIDSs / COMPSs Tutorial

Enric Tejedor, Daniele Lezzi, Rosa M. Badia

{enric.tejedor,daniele.lezzi,rosa.m.badia}@bsc.es

RES Users Tutorial
September 20, 2010

Tutorial Outline

Overview of GRIDSs/COMPSs, first example. 12:00 -14:00

1. Objective and overview
2. Programming model: A sample code (Java and C)
3. Configuration, compilation and execution
4. Monitoring and Debugging
5. Migration from GRIDSs to COMPSs

Break 14:00-15:00

Programming examples, Hands-on. 15:00 – 17:00

1. Examples: Matmul, HMMER
2. Tracing and performance analysis
3. Hands-on: IS-ENES

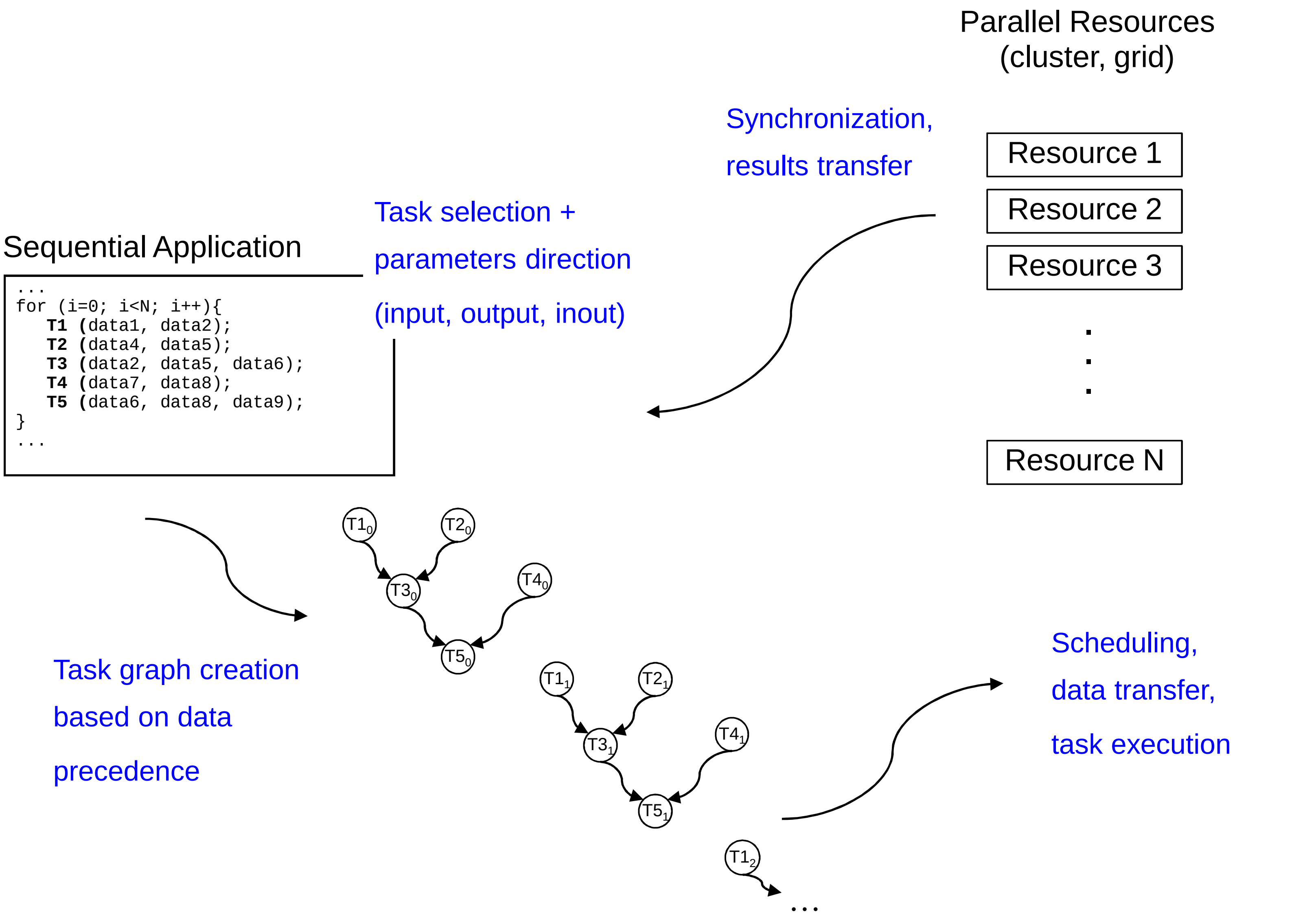
Overview of GRIDSs/COMPSs

1. Objective and overview
2. Programming model: a sample code (Java and C)
3. Configuration, compilation and execution
4. Monitoring and Debugging
5. Migration from GRIDSs to COMPSs

GRIDSs/COMPSs Objective

- Reduce the development complexity of Grid/Cluster applications to the minimum
  - Writing an application for a computational Grid may be as easy as writing a sequential application
- Target applications: composed of tasks, most of them repetitive
  - Granularity of the tasks of the level of simulations or programs
  - Data objects are files

GRIDSs/COMPSs – Overview



Parallel Resources
(cluster, grid)

Synchronization,
results transfer

Resource 1

Resource 2

Resource 3

.
.
.

Resource N

Sequential Application

```
...
for (i=0; i<N; i++){
   T1 (data1, data2);
   T2 (data4, data5);
   T3 (data2, data5, data6);
   T4 (data7, data8);
   T5 (data6, data8, data9);
}
...
```

Task selection +
parameters direction
(input, output, inout)

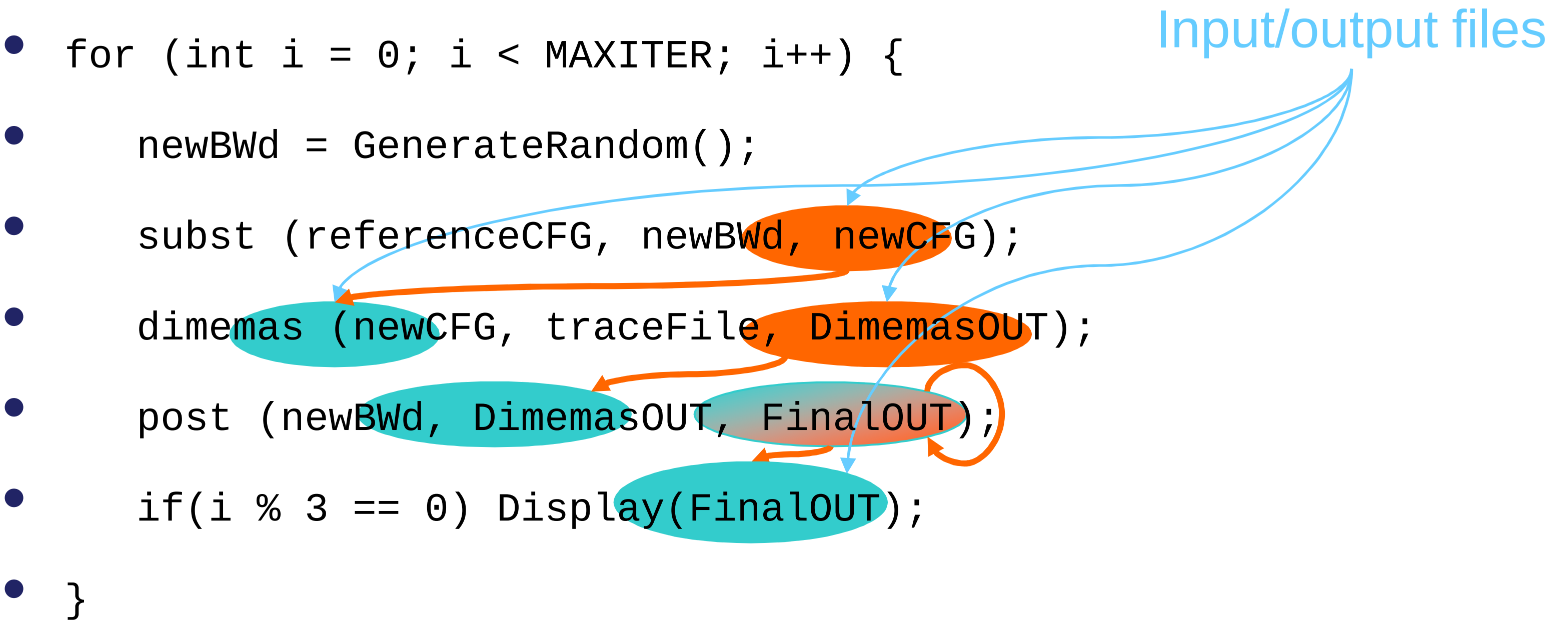
Task graph creation
based on data
precedence

Scheduling,
data transfer,
task execution

GRIDSs/COMPSs – Overview - Runtime features

- Common features:
  - Data dependency analysis
  - Data renaming
  - Data transfer
  - Task scheduling
  - Resource management
  - Results collection
  - Fault tolerance
- Other
  - Shared disks management
  - Checkpointing

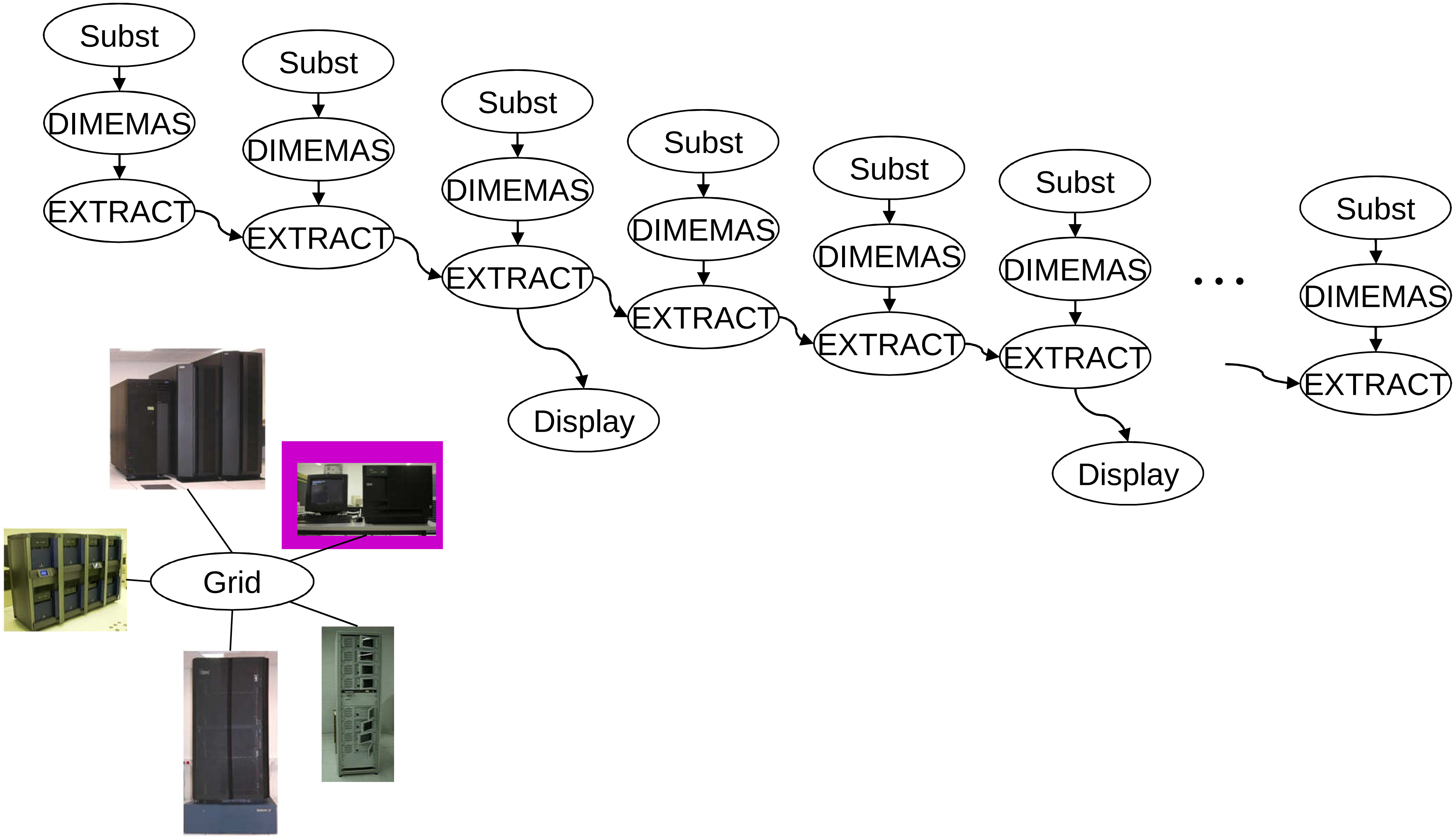
GRIDSs/COMPSs – Overview - Behaviour



Input/output files

```
for (int i = 0; i < MAXITER; i++) {
    newBWd = GenerateRandom();
    subst (referenceCFG, newBWd, newCFG);
    dimemas (newCFG, traceFile, DimemasOUT);
    post (newBWd, DimemasOUT, FinalOUT);
    if(i % 3 == 0) Display(FinalOUT);
}
```

GRIDSs/COMPSs – Overview - Behaviour



Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Display

Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Display

. . .

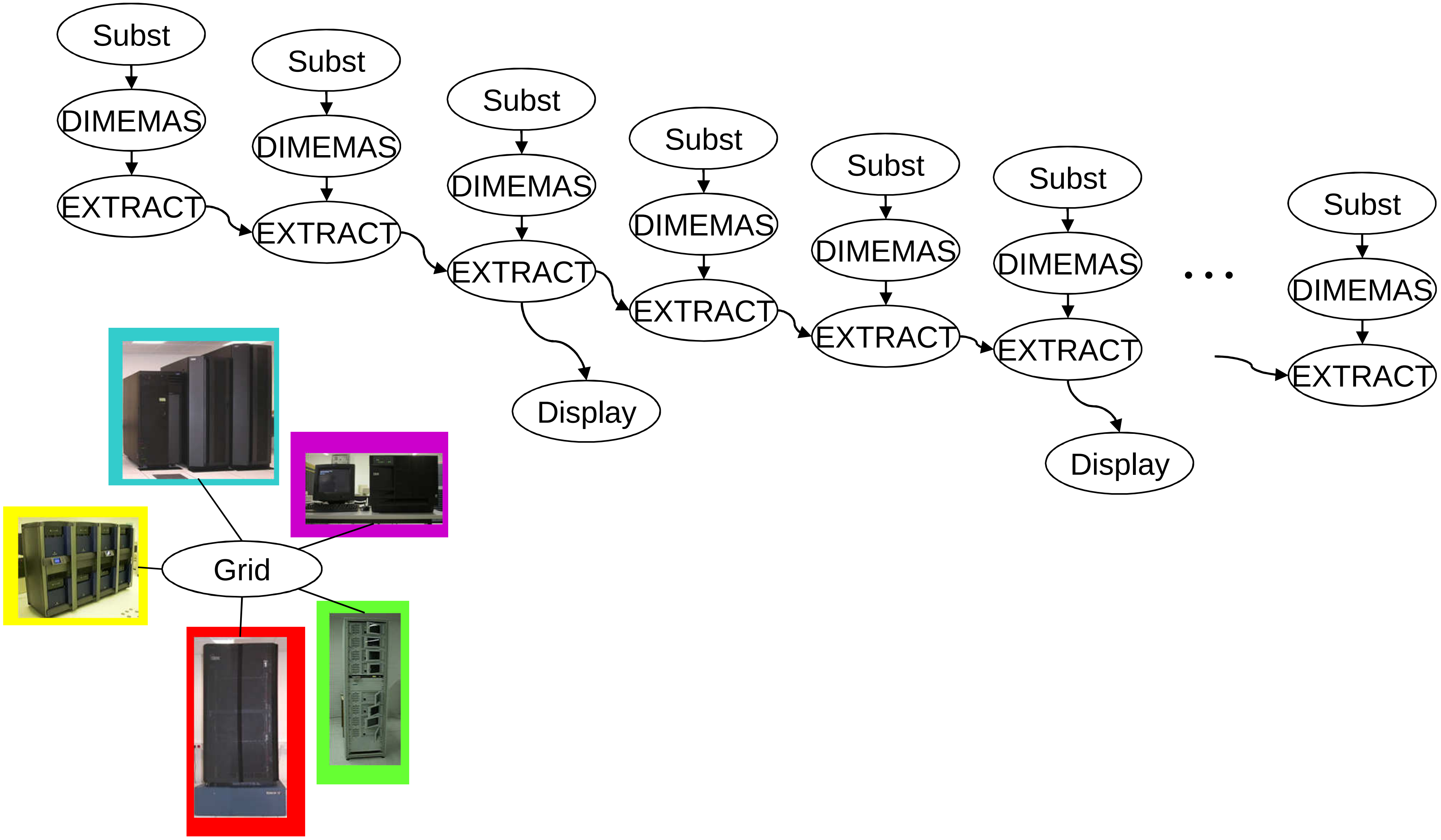
Subst

DIMEMAS

EXTRACT

Grid

GRIDSs/COMPSs – Overview - Behaviour



Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Display

Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Subst

DIMEMAS

EXTRACT

Display

. . .

Subst

DIMEMAS

EXTRACT

Grid

Programming model – Java – Sequential code

Java application

```
public static void main(String[] args) {
    String counterFile = args[0];
    int initialValue = args[1];

    initializeCounter(counter, initialValue);

    SimpleImpl.increment(counterFile);

    printCounter(counter);
}
```

Subroutine

```
public static void increment(String counterFile) {
    int value = readCounter(counterFile);
    value++;
    writeCounter(counterFile, value);
}
```

Programming model – Java – Task selection

Java interface

```
public interface SimpleItf {

    @ClassName(“SimpleImpl")
    void increment(
      @ParamMetadata(type = Type.FILE, direction = Direction.INOUT)
      String counterFile
    );

}
```

Implementation

Parameter metadata

Programming model – Java – Final app code

```
public static void main(String[] args) {
    String counterFile = args[0];
    int initialValue = Integer.parseInt(args[1]);

    initializeCounter(counter, initialValue);

    SimpleImpl.increment(counterFile);

    printCounter(counter);
}
```

Java application
NO CHANGES!

Programming model – C – Sequential code

```
int main(int argc, char **argv) {
        char *counter_file = argv[1];
        int init_value = atoi(argv[2]);
        initialize_counter(counter_file, init_value);

        increment(counter_file);

        print_counter(counter_file);
        return 0;
}
```

C main application code

```
void increment(char *counter_file) {
        int value = read_counter(counter_file);
        value++;
        write_counter(counter_file);
}
```

Subroutine

Programming model – Java – Task selection

IDL file

```
interface SIMPLE {

    void increment(inout File counter_file);

};
```

Parameter metadata

Programming model – C – Final code

C application
+ API calls

```
int main(int argc, char **argv) {
        char *counter_file = argv[1];
        int init_value = atoi(argv[2]);

        initialize_counter(counter_file, init_value);

        GS_On(PRJ_FILE, RES_FILE, MASTER_DIR, APPNAME);
        increment(counter_file);
        GS_Off(0);

        print_counter(counter_file);

        return 0;
}
```

Configuration – Java and C

- Environment variables
  - export JAVA_HOME=/opt/ibm/java-ppc-60
  - export IT_HOME=/gpfs/apps/COMPSs
  - export CLASSPATH=.:$IT_HOME/integratedtoolkit/lib/IT.jar
  - export GS_HOME=$IT_HOME/bindinglib
  - export PATH=$PATH:$GS_HOME/bin
  - export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$GS_HOME/lib:$JAVA_HOME/jre/lib/ppc64/classic

Compilation – Java

- *user@node:~/app_dir>* $JAVA_HOME/bin/javac *.java
  - Main app code: Simple.java
  - Annotated interface: SimpleItf.java
  - Subroutine implementation: SimpleImpl.java

Compilation – C

- *user@node:~/app_dir>* gsbuild build all simple
  - Main app code: simple.cc
  - IDL file: simple.idl
  - Subroutine implementation: simple-functions.c

Execution – Java and C

- *user@node:~/app_dir*> $JAVA_HOME/bin/javac *.java
  - Main app code: Simple.java
  - Annotated interface: SimpleItf.java
  - Subroutine implementation: SimpleImpl.java

Monitoring and debugging

- It.log
- Tasks log

Migration GRIDSs -> COMPSs

- TODO: DANIELE

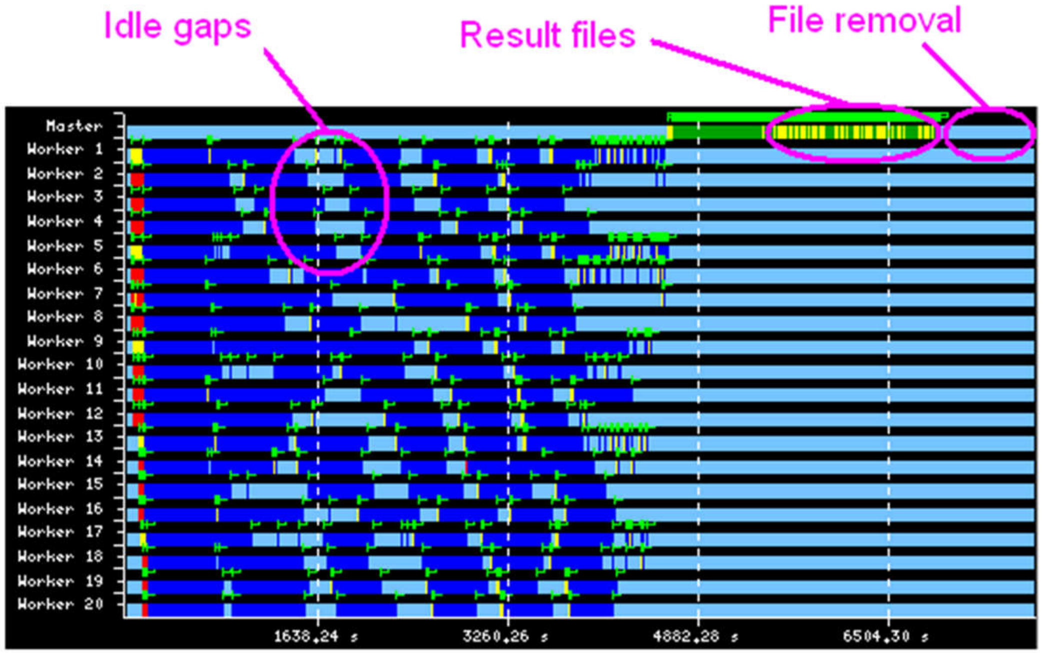
BREAK

Hands-on: Programming examples

- Matmul: C
- HMMER: Java

Tracing and performance analysis

- COMPSs can generate post-mortem traces of the distributed execution of the application
  - Master + workers, tasks + file transfers
- Useful for analysis and diagnosis



Thank you!

Questions?