Outline (Feb 20th 2014)

- Session 1 / 9am – 11am: Introduction to COMPSs
- Roundtable: presentation and background of participants
- Programming model
  - Overview
  - Steps
  - Properties
- COMPSs runtime system
  - Overview
  - Features
- Coffee break – 11:00 – 11:30
- Session 2 / 11:30am – 1pm: Application examples
  - Sample codes & Demos
    - Matmul
  - Graphical interface (IDE)
    - Gene Detection
Outline (Feb 20\textsuperscript{th} 2014)

\begin{itemize}
\item Lunch Break 1pm to 2pm
\item Session 3 / 2 pm- 3:30 pm: Hands-on I
  \begin{itemize}
  \item Virtual Machine Setup
  \item BLAST overview
  \item Code modification
    \begin{itemize}
    \item All-to-one and tree-reduction
    \end{itemize}
  \item Compilation and Execution
  \end{itemize}
\item Coffee break: 3:30 – 4:00
\item Session 4 / 4 pm- 6 pm: Hands-on II
  \begin{itemize}
  \item HMMER overview & code modification
  \item Configuration, monitoring, debugging
  \item Overview of tracing, trace analysis
  \item IDE Hands-on
  \item Final notes
  \end{itemize}
\end{itemize}
Overview: Objectives

- Reduce the development complexity of Grid/Cluster/Cloud applications to the minimum
  - Writing an application for a computational distributed infrastructure may be as easy as writing a sequential application

- Target applications: composed of tasks, most of them repetitive
  - Granularity of the tasks or programs
  - Data: files, objects, arrays and primitive types

Programming Model: Steps

1. Identify tasks
   main program {
   taskA(...);
   taskB(...);
   }

2. Select tasks
   task selection interface {
   taskA
   taskB
   }

- Task
- Unit of parallelism
- Asynchrony

Resource 1

Resource 2

... Resource N
Programming Model: Properties (I)

- Based on pure-Java fully-sequential programming
  - No APIs, no threading, no messaging
  - No parallel constructs, no pragmas
  - Sequential consistency

```java
Main Program {
    taskA(data1);
    for (int i=0; i<N; i++)
        taskB(data1, data2);
    if (condition)
        process(data2);
}
```

Programming Model: Dependency detection

- Automatic on-the-fly creation of a task dependency graph

```java
for (int i = 0; i < N; i++) {
    newBWD = random();
    subst(refCFG, newBWD, newCFG);
    dimemas(newCFG, trace, dimOUT);
    extract(newBWD, dimOUT, finalOUT);
    if (i % 2 == 0) display(finalOUT);
}
```
Programming Model: Properties (II)

- Infrastructure unaware

```
public interface SampleItf {
  @Constraints(processorCPUCount = 1, memoryPhysicalSize = 0.5f)
  @Method(declaringClass = "servicess.Example")
  void myMethod(
    @Parameter(direction = INOUT) Reply r
  );

  @Service(namespace = "http://servicess.es/example",
           name = "SampleService",
           port = "SamplePort")
  Reply myServiceOp(
    @Parameter(direction = IN) Query q
  );
}
```
Programming Model: Regular Main program

```java
public class App {
    public static void main(String[] args) {
        Query query = new Query(...);
        Reply reply = myServiceOp(query);

        myMethod(reply);

        reply.printToLog();
    }
}
```

Programming Model: Service Operation

```java
public class ServiceApp {
    @Orchestration
    public static void sampleComposite() {
        Query query = new Query(...);
        Reply reply = myServiceOp(query);

        myMethod(reply);

        reply.printToLog();
    }
}
```
Programming Model: Summary

Sequential Code

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

Parallel Resources

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

(b) Task graph creation based on data dependencies

(c) Scheduling, data transfer, task execution

(d) Task completion, synchronization

Programming Model: Sample Application

```
public static void main(String[] args) {
    String counter1 = args[0], counter2 = args[1],
    counter3 = args[2];
    initializeCounters(counter1, counter2, counter3);
    for (i = 0; i < 3; i++) {
        increment(counter1);
        increment(counter2);
        increment(counter3);
    }
}
```

Main program

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

Subroutine
public interface SimpleItf {
    @Method(declaringClass = "SimpleImpl")
    void increment(
        @Parameter(type = FILE, direction = INOUT)
        String counterFile
    );
}

public static void main(String[] args) {
    String counter1 = args[0], counter2 = args[1], counter3 = args[2];
    initializeCounters(counter1, counter2, counter3);
    for (i = 0; i < 3; i++) {
        increment(counter1);
        increment(counter2);
        increment(counter3);
    }
}
for (i = 0; i < 3; i++) {
    increment(counter1);
    increment(counter2);
    increment(counter3);
}

Programming Model: Task Graph

Main loop

Task graph

1st iteration

2nd iteration

3rd iteration

Programming Model: Task Graph

COMPSs Runtime System
Runtime System

- Application
- Task Selection Interface

Basic features

- Supported Features:
  - Data dependency analysis
  - Data renaming
  - Data transfer
  - Task scheduling
  - Resource management
  - Results collection
  - Fault tolerance
  - Shared disks management

- In Progress:
  - Checkpointing
  - Constraint matching
  - Task nesting
Interoperability

Grid/Cluster Configuration: Resources Specification

<?xml version="1.0" encoding="UTF-8"?>
<RootList>
  <!--Description for any physical node-->
  <Resource Name="172.20.200.18">
    <Capabilities>
      <Host>
        <TaskCount>0</TaskCount>
        <Queue>short</Queue>
      </Host>
      <Processor>
        <Architecture>IA32</Architecture>
        <Speed>3.0</Speed>
        <CPUCount>1</CPUCount>
      </Processor>
      <OS>
        <OSType>Linux</OSType>
        <MaxProcessesPerUser>32</MaxProcessesPerUser>
      </OS>
      <StorageElement>
        <Size>30</Size>
      </StorageElement>
    </Capabilities>
    <Requirements/>
  </Resource>
  <Resource Name="172.20.200.19">
    ...
  </Resource>
</RootList>
Grid/Cluster Configuration: Project Specification

Project.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Project>
<!--Description for any physical node-->
<Worker Name="172.20.200.18">
<InstallDir>/opt/COMPSs/Runtime/scripts/</InstallDir>
<WorkingDir>/tmp/</WorkingDir>
<User>user</User>
<LimitOfTasks>1</LimitOfTasks>
</Worker>
<Worker Name="172.20.200.19">
</Worker>
</Project>
```

Cloud Configuration: Resources Specification

Resources.xml

```xml
<ResourceList>
<CloudProvider name="BSCCloud">
<Server>https://bscgrid2.bsc.es:8443/DRP</Server>
<integratedtoolkit.connectors.emotivecloud.DRPSecureClientConnector>
<ImageList>
<Image name="debianbase"/>
</ImageList>
<InstanceTypes>
<Resource Name="bsc.small">
<Capabilities>
<Processor>
<CPUCount>1</CPUCount>
</Processor>
<StorageElement>
<Size>0.5</Size>
</StorageElement>
</Capabilities>
</Resource>
<Resource Name="bsc.medium">
<Capabilities>
<Processor>
<CPUCount>1</CPUCount>
</Processor>
<StorageElement>
<Size>0.5</Size>
</StorageElement>
</Capabilities>
</Resource>
</InstanceTypes>
</CloudProvider>
</ResourceList>
```
Cloud Configuration: Project Specification

Project.xml

```xml
<Project>
  <Cloud>
    <InitialVMs>0</InitialVMs>
    <minVMCount>2</minVMCount>
    <maxVMCount>5</maxVMCount>
    <Provider name="BSCCloud">
      <LimitOfVMs>5</LimitOfVMs>
      <Property>
        <Name>Cert</Name>
        <Value>/home/.../cert.p12</Value>
      </Property>
      <Property>
        <Name>Owner</Name>
        <Value>userbsc</Value>
      </Property>
      <Property>
        <Name>JobNameTag</Name>
        <Value>Job</Value>
      </Property>
    </Provider>
  </Cloud>
</Project>
```

COMPSs in a Cluster (interactive)

Typical setup:
- Master node: main program (+ master runtime)
- Worker nodes: tasks (+ worker runtime)

Described by XML files

Workers

![Diagram of COMPSs in a Cluster](image)
COMPSs in a Cluster (queue system)

Execution divided in two phases
- Launch scripts queue a whole COMPSs app execution
- Actual execution starts when reservation is obtained

COMPSs in a Grid
COMPSs in the Cloud

- Runtime integrated in a platform with:
  - Service orientation
  - Virtualization

Cloud: Connector design

- Interaction with:
  - Cloud providers: connectors, SSH
  - Service providers: WS client
Cloud: Elasticity

- Increase/decrease number of VMs depending on task load
- Bursting to Amazon EC2 to face peak load

COMPSs Bindings

Java App

Python App

C/C++ App

Python binding

C/C++ binding

C++ library

Java Runtime

JNI
Matmul example

```java
for (int i = 0; i < MSIZE; i++)
    for (int j = 0; j < MSIZE; j++)
        for (int k = 0; k < MSIZE; k++)
            { 
                long ini, fi;
                ini = System.currentTimeMillis();
                MatmulImpl.mulEplyAccumulaEve(_C[i][j], _A[i][k], _B[k][j]);
                fi = System.currentTimeMillis();
                System.out.println("TASK: " + (fi - ini) / 1000 + " seconds\n");
            }
```

```java
public static void multiplyAccumulative(String f3, String f1, String f2 )
{
    Block a = new Block ( f1 );
    Block b = new Block ( f2 );
    Block c = new Block ( f3 );
    c.multiplyAccumulative( a, b );
    try
        ...
}
```

```java
public void multiplyAccumulative ( Block a, Block b )
{ 
    for( int i = 0; i < this.bRows; i++ ) // rows
        for( int j = 0; j < this.bCols; j++ ) // cols
            this.data[i][j] += a.data[i][k] * b.data[k][j];
```
Matmul interface

```java
package matmul;

import integratedtoolkit.types.annotations.Constraints;
import integratedtoolkit.types.annotations.Method;
import integratedtoolkit.types.annotations.Parameter;
import integratedtoolkit.types.annotations.Parameter.*;

public interface Matmul {
    @Constraints(processorCPUCount = 4, memoryPhysicalSize = 1.5f)
    @Method(declaringClass = "matmul.MatmulImpl")
    void multiplyAccumulative(
        @Parameter(type = Type.FILE, direction = Direction.INOUT)
        String file1,

        @Parameter(type = Type.FILE, direction = Direction.IN)
        String file2,

        @Parameter(type = Type.FILE, direction = Direction.IN)
        String file3
    );
}
```

Matmul: Compiling

- Compiling with command line:
  - cd workspace
  - javac matmul/src/matmul/*.java
  - cd matmul/src/
  - jar cf matmul.jar matmul

- From eclipse:
  - Package Explorer -> Project (matmul) -> Export…
Matmul: Deploying

In this case, in the same machine
- Copy to home directory
- cd
- cp ./matmul/src/matmul.jar .

In remote machines
- Code needs to be transfer to machine that will host main code

Matmul: Executing

Set CLASSPATH
- export CLASSPATH=$CLASSPATH:/home/user/matmul.jar
- runcompss matmul.Matmul 4
Matmul: Monitoring execution

- Browse
  - http://localhost:8080/compss-monitor

Demos: Matmul

- Blocks matrixes multiplication
IDE COMPSs applications as a Service

IDE for implementing and deploying applications

Building & Deployment:
- Generate Packages
- Define hosts & Deploy

Tasks Definition:
- Service Operations (Orchestration)
- Tasks (Core Element)

Demos: Gene Detection Application

Gene Detection algorithm designed by the BSC Life Sciences department
- Automatic Homology-based gene detection and analysis

Combine services with computations
- Example that shows different capabilities of COMPSs
- Implicit Synchronization points
- Different method and service call types
- Objects and files as parameters
Gene Detection

**A. Formal Genome**
- InCompleteGenome
- GenBank
- Complete/Null Genome
- NDB Genome

**B. Similar Sequences**
- Reference Protein
  - BLASTp
  - BLAST Report
  - Pair-BlastDs
  - Local Sequences
  - InferFasta
  - FASTA Seq

**C. Relevant Genes**
- BLASTFromFasta
  - MergeBLAST
  - BLASTx/ie
  - ConnectorsDatabase
  - Gene

**D. Gene report**
- Gene
- MergeGH
- Off Report

**Web services**

**Java methods**

---

**HANDS-ON**
Hands-On: Overview

- COMPSs Virtual Machine setup
- Applications Overview (BLAST, HMMER, …)
  - Code modification
  - Configuration, compilation & execution
  - Monitoring, debugging
  - Overview of tracing and trace performance analysis
  - IDE

COMPSs development VM Installation

- COMPSs Development & Test VM (64-bit) OVA
  - Available from USB
  - Import the virtual appliance in VirtualBox
Bioinformatics Scenario

BLAST (Basic Local Alignment Search Tool) Suite:

- BLAST: An algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or nucleotides of DNA sequences.

BLAST enables a researcher to compare a query sequence with a library or database of sequences, and identify sequences that resemble the query sequence above a certain threshold.
BLAST: Hands-on

• BLAST

BLAST: All-to-One reduction

• Main Application (All-to-One):

```java
public static void main(String[] args) throws Exception {
    sequences[] = splitSequences(inputFile, nFrags);
    for (partition: sequences) {
        BlastImpl.align(database, partition, partitionOutput, blastBinary, commandArgs);
        partitionOutputs.add(partitionOutput);
    }
    assemblyPartitions(partitionOutputs, outputFile, tempDir, nFrags);
}
```
public class BlastImpl{
  public void align(String databasePath, String partitionFile, String partitionOutput, String blastBinary, String commandArgs)
  {
    String cmd = blastBinary + " -p blastx -d " + databasePath + " -i " + partitionFile + " -o " + partitionOutput + " -a " + commandArgs;

    Process simProc = Runtime.getRuntime().exec(cmd);
  }
}

Creation of the annotated interface for the selection of remote tasks

public interface BlastI_
{
  @Method(declaringClass = "blast.BlastImpl")
  @Constraints(processorCPUCount = 4, memoryPhysicalSize = 4.0f)
  void align(
    @Parameter(type = Type.STRING, direction = Direction.IN) String databasePath,
    @Parameter(type = Type.FILE, direction = Direction.IN) String partitionFile,
    @Parameter(type = Type.FILE, direction = Direction.OUT) String partitionOutput,
    @Parameter(type = Type.STRING, direction = Direction.IN) String blastBinary,
    @Parameter(type = Type.STRING, direction = Direction.IN) String commandArgs);
}
BLAST: Compilation and execution

- Compilation (Eclipse IDE)
  - Package Explorer -> Project (blastallone) -> Export...

- Usage
  - runcompss blast.Blast <debug> <binary> <database> <sequences> <#fragments> <tmpdir> <output>

- Execution
  - cp ~/workspace/blastallone/jar/blast.jar ~
  - export CLASSPATH=$CLASSPATH:/home/user/blast.jar
  - runcompss blast.Blast true /home/user/workspace/blast/binary/blastall /sharedDisk/Blast/databases/swissprot/swissprot /sharedDisk/Blast/sequences/sargasso_test.fasta 4 /tmp/ /home/user/out.txt

BLAST Sequence Alignment Tool

Parameters:
- Debug Enabled
- Blast binary: /home/user/workspace/blastallone/blastall
- Number of expected fragments: 8
- Database Name with Path: /sharedDisk/Blast/databases/swissprot/swissprot
- Database Name: swissprot
- Input Sequences File: /sharedDisk/Blast/sequences/sargasso_test.fasta
- Temporary Directory: /tmp/
- Output File: /home/user/IT/blast.Blast.out.txt
- Command Line Arguments:
  - The total number of sequences is: 20
  - The total number of sequences of a fragment is: 3
  - Splitting sequences among fragment files...
  - Aligning Sequences:
    - Number of fragments to assemble is 8
    - Sequences assembled in 184 seconds
BLAST: All-to-One (work)

- Generate the final graph
- Launch BLAST (All-to-One)

BLAST: All-to-One (Graph)

```plaintext
align()
```

1 2 3 4 5 6 7 8
q2 q5 q8 q11 q14 q17 q20 q23
sync sync sync sync sync sync sync sync

File Sync.
• Code the final reduction and its interface.

• Main Application (Tree-based):

```java
public static void main(String args[]) throws Exception {
    sequences[] = splitSequences(inputFile, nFragments);
    for (partition: sequences) {
        BlastImpl.align(database, partition, partitionOutput, blastBinary, commandArgs);
        partitionOutputs.add(partitionOutput);
    }
    // Final Assembly process -> Merge 2 by 2
    int neighbour=1;
    while (neighbour<partialOutputs.size()) {
        for (int result=0; result<partialOutputs.size(); result+=2*neighbour) {
            if (result+neighbour < partialOutputs.size()) {
                BlastImpl.assemblyParitions(partialOutputs.get(result), partialOutputs.get(result+neighbour));
                lastMerge = partialOutputs.get(result);
            }
        }
        neighbour*=2;
    }
}
```
Creation the annotated interface for the selection of the remote tasks

```java
public interface BlastIF {
    @Method(declaringClass = "blast.BlastImpl")
    @Constraints(processorCPUCount = 4, memoryPhysicalSize = 4.0f)
    void align(
        @Parameter(type = Type.STRING, direction = Direction.IN)
        String databasePath,
        @Parameter(type = Type.FILE, direction = Direction.IN)
        String partitionFile,
        ...
    );

    @Parameters(type = Type.STRING, direction = Direction.IN)
    String commandArgs);

    @Method(declaringClass = "blast.BlastImpl")
    @Constraints(processorCPUCount = 2, memoryPhysicalSize = 2.0f)
    void assemblyParEons(
        @Parameter(type = Type.FILE, direction = Direction.INOUT)
        String parEalFileA,
        @Parameter(type = Type.FILE, direction = Direction.IN)
        String parEalFileB);
}
```

BLAST: Tree-based execution

---------- Executing blast.Blast in IT mode total----------

BLAST Sequence Alignment Tool

Parameters:
- Debug Enabled
- Blast binary: /home/user/workspace/blastAllOne/binary/blastall
- Number of expected fragments: 8
- Database Name with Path: /sharedDisk/Blast/databases/swissprot/swissprot
- Database Name: swissprot
- Input Sequences File: /sharedDisk/Blast/sequences/sargasso_test.fasta
- Temporary Directory: tmp
- Output File: /home/user/IT/blast.Blast/out.txt
- Command Line Arguments:
- The total number of sequences is: 20
- The total number of sequences of a fragment is: 3
- Splitting sequences among fragment files...
- Alining Sequences:
- Number of fragments to assemble -> 8
- Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt
- Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt
- Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt
- Merging last merged file: /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt to /home/user/IT/blast.Blast/out.txt
- Opening file /home/user/IT/blast.Blast/out.txt in mode WRITE
- /sharedDisk/Blast/sequences/sargasso_test.fasta sequences aligned successfully in 193 seconds
----------
BLAST: Tree-based (Graph)

align()

assemblyPartitions()

Synchronizing with last merged file

HMMER: Hands-on
HMMER Hands On

Application: HMMER suite (hmmpfam)

- hmmpfam is part of the HMMER suite: set of tools for protein sequence analysis
  - Reads a sequences file and compares each sequence in it against a database of HMMs
  - HMM (Hidden Markov Model): statistical figure that represents a protein family
- Goal: create an hmmpfam efficient service
  - Starting point: sequential version of the hmmpfam tool
- With the COMPSs: hmmpfam becomes parallel
  - Phase 1: Split both input sequences and database
  - Phase 2: Process them in parallel (speed up execution)
  - Phase 3: Reduction of results

HMMER example

**HMMER**

**Protein Database**

**Aminoacid Sequence**

<table>
<thead>
<tr>
<th>Model</th>
<th>Score</th>
<th>E-value</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>IL6_2</td>
<td>-78.5</td>
<td>0.13</td>
<td>1</td>
</tr>
<tr>
<td>COLFI_2</td>
<td>-164.5</td>
<td>0.35</td>
<td>1</td>
</tr>
<tr>
<td>pgtp_13</td>
<td>-36.3</td>
<td>0.48</td>
<td>1</td>
</tr>
<tr>
<td>clf2</td>
<td>-15.6</td>
<td>3.6</td>
<td>1</td>
</tr>
<tr>
<td>PKD_9</td>
<td>-24.0</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
Aminoacid sequence

HMMER example

```java
String[] outputs = new String[numDBFrags];

//Process
for (String dbFrag : dbFrags) {
    outputs[dbNum]= HMMPfamImpl.hmmpfam(sequence, dbFrag);
}

//Merge all DB outputs of the same DB fragment
int neighbour = 1;
while (neighbour < numDBFrags) {
    for (int db = 0; db < numDBFrags; db += 2 * neighbour) {
        if (db + neighbour < numDBFrags) {
            HMMPfamImpl.mergeSameDB(outputs[db], outputs[db + neighbour]);
        }
    }
    neighbour *= 2;
}
```

HMMER example (code)
public interface HMMPfamItf {
  @Method(declaringClass = "worker.hmmerobj.HMMPfamImpl")
  @Constraints(storageElemSize = 1.5f)
  String hmmpfam(
    @Parameter(type = Type.FILE, direction = Direction.IN)
    String seqFile,
    @Parameter(type = Type.FILE, direction = Direction.IN)
    String dbFile,
    ...
  );

  @Method(declaringClass = "worker.hmmerobj.HMMPfamImpl")
  void mergeSameDB(
    @Parameter(type = Type.OBJECT, direction = Direction.IN)
    String resultFile1,
    @Parameter(type = Type.OBJECT, direction = Direction.IN)
    String resultFile2
  );
  ...
}
• Complete the hmmpfam & mergeSameSeq method interfaces.

Project.xml: /opt/COMPSs/Runtime/xml/projects/project.xml

<?xml version="1.0" encoding="UTF-8"?>
<Project>
    <!--Description for any physical node-->
    <Worker Name="localhost">
        <InstallDir>/opt/COMPSs/Runtime/scripts/</InstallDir>
        <WorkingDir>/tmp/</WorkingDir>
        <User>user</User>
        <LimitOfTasks>2</LimitOfTasks>
    </Worker>
</Project>
HMMER: Configuration, compilation and execution

- Configuration: /opt/COMPSs/Runtime/xml/resources/resources.xml

```
<xml version="1.0" encoding="UTF-8">
  <ResourceList>
    <Resource Name="localhost">
      <Capabilities>
        <Host>
          <TaskCount>0</TaskCount>
          <Queue>short</Queue>
        </Host>
        <Processor>
          <Architecture>AMD64</Architecture>
          <Speed>3.0</Speed>
          <CPUCount>2</CPUCount>
        </Processor>
        <OS>
          <OSType>Linux</OSType>
          <MaxProcessesPerUser>32</MaxProcessesPerUser>
        </OS>
        <StorageElement>
          <Size>30</Size>
        </StorageElement>
        <Memory>
          <PhysicalSize>2</PhysicalSize>
          <VirtualSize>8</VirtualSize>
        </Memory>
      </Capabilities>
    </Resource>
    <ResourceList>
```

HMMER: Configuration, compilation and execution

- Compilation (Eclipse IDE)
  - Package Explorer -> Project (hmmerobjblanks) -> Export... (Hands-on)
  - Package Explorer -> Project (hmmerobj) -> Export... (Solution)

- Usage
  - runcompss hmmerobj.HMMPfam <database> <sequences> <output> <params>

- Execution
  - cp ~/workspace/hmmerobj/jar/hmmerobj.jar ~
  - export CLASSPATH=$CLASSPATH:/home/user/hmmerobj.jar
  - runcompss hmmerobj.HMMPfam /sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq /home/user/out.txt 2 & A 222
user@bsccompss:~$ runcompss hmmerobj.HMMPfam /sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq/home/user/out.txt 2 8 -A 222

----------------- Executing hmmerobj.HMMPfam in IT mode total --------------------------

  [ API ]  -  Deploying the Integrated Toolkit
  [ API ]  -  Starting the Integrated Toolkit
  [ API ]  -  Initializing components
  [ API ]  -  Ready to process tasks
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF0_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF1_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF2_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF3_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF4_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF5_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF6_1 in mode WRITE
  [ API ]  -  Opening file /tmp/hmmer_frags/seqF7_1 in mode WRITE
  [ API ]  -  Opening file /home/user/out.txt in mode WRITE
  [ API ]  -  No more tasks for app 1
  [ API ]  -  Stopping IT
  [ API ]  -  Cleaning
  [ API ]  -  Integrated Toolkit stopped

HMMER: Configuration, compilation and execution

The runtime of COMPSs provides some information at execution time so the user can follow the progress of the application:

- Real-time monitoring information (http://localhost:8080/compss-monitor/)
  - # tasks
  - Resources usage information
  - Execution time per task
  - Real-time execution graph
  - Etc.
HMMER: Debugging

• COMPSs can be run in debug mode showing more information about the execution allowing to detect possible problems
  • Log level configurable at: /opt/COMPSs/Runtime/log/it-log4j
  • The user can check the execution of its application by reading:
    • The output/errors of the main application (console stdout)
    • The output/error of a task # N
      • ~/IT/[APP_NAME]/jobs/jobN_[out|err]
    • Messages from the runtime COMPSs
      • ~/it.log
    • Task to resources allocation:
      • ~/resources.log
  • The user can verify the correct structure of the parallel application generating a complete post-mortem application graph
    • gengraph $HOME/APP_NAME.dot

Tracing: Overview

COMPSe can generate post-execution traces of the distributed execution of the application

– Useful for performance analysis and diagnosis

How it works?

– Task execution and file transfers are application events
– An XML file is created at workers to keep track of these events
– At the end of the execution all the XML files are merged to get the final trace file
– Instrumentation and Visualization tools from BSC are needed.
Tracing: Instrumentation

COMPSs uses Extrae tool to dynamically instrument the application

- In a worker:
  - Extrae keeps track of the events in an intermediate file

- In the master:
  - Extrae merges the intermediate files to get the final trace file

- For more information about Extrae visit:
  - http://www.bsc.es/computer-sciences/extrae

---

Tracing: Instrumentation

Executing hmmerobj.HMMPfam

[ API] - Deploying the Integrated Toolkit
[ API] - Starting the Integrated Toolkit
[ API] - Initializing components

Welcome to Extrae 2.4.3rc4 (revision 311 based on framework/trunk/files/extrae)
Extrae: Generating intermediate files for Paraver traces.
Extrae: Intermediate files will be stored in /home/user/IT/hmmerobj.HMMPfam
Extrae: Tracing mode is set to: Detail.
Extrae: Successfully initiated with 1 tasks

[ API] - Ready to process tasks
...
...

COMPSs runtime starts

Extrae keeps tracing events in background

Extrae starts before the user application execution
**Tracing: Instrumentation**

- [ API] - No more tasks for app 1
- [ API] - Stopping IT
- [ API] - Cleaning

Extrae: Application has ended. Tracing has been terminated.

... 

merger: Output trace format is: Paraver
merger: Extrae 2.4.3rc4 (revision 311 based on framework/trunk/files/extrae)

... 

[ API] - Integrated Toolkit stopped

... 

mpi2prv: Selected output trace format is Paraver
mpi2prv: Parsing intermediate files

mpi2prv: Generating tracefile (intermediate buffers of 1342156 events)

mpi2prv: Congratulations! hmmerobj.HMMPfam_compss_trace_1392736225.prv has been generated.

---

**Tracing: Visualization**

- Paraver is the BSC tool for trace visualization

  - Trace events are encoding in Paraver (.prv) format by Extrae
  - Paraver is a powerful tool for trace visualization.
  - An experimented user could obtain many different views of the trace events.
  - For more information about Paraver visit:
Tracing: Hands-on

• Compilation (Eclipse IDE)
  • Package Explorer -> Project (hmmerobj) -> Export...

• Execution
  • `cp ~/workspace/hmmerobj/jar/hmmerobj.jar ~`
  • `export CLASSPATH=$CLASSPATH:/home/user/hmmerobj.jar`
  • `runcompssexec --app=hmmerobj.HMMPfam --tracing=true --cline_args="/sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq /home/user/out.txt 2 8 -A 222"`
  • `wxparaver /home/user/IT/hmmerobj.HMMPfam/*.prv`

Tracing: Hands-on

• COMPSs provides a configuration file to automatically obtain the view of the trace
  • `File / Load Configuration...`
  • `/opt/COMPSs/paraver/cfgs/tasks.cfg`

• Some small adjustments must be done in order to view the trace correctly
Tracing: Hands-on

• Fit window
  • Right click on the trace window
  • Fit Semantic Scale / Fit Both

• View Event Flags
  • Right click on the trace window
  • View / Event Flags

Tracing: Hands-on

• Show Info Panel
  • Right click on the trace window
  • Check Info Panel option
  • Select Colors tab of the panel
Tracing: Hands-on

- Zoom to see details
  - Select a region in the trace window to see in detail
  - And repeat the process till the needed zoom level
  - The undo zoom option is in the right click panel

Previous task ends  Processor is idle  New task starts

Tracing: Hands-on

Summarizing:
- Lines in the trace:
  - One line for the master
  - N lines for the workers

- Meaning of the colours:
  - Light blue: idle
  - Other colors: task running, see the color legend

- Flags (events):
  - Start / end of task
Tracing: BLAST, All-to-One vs Tree-based (Cloud)

Virtual Machine Creation

Tracing: Other examples
Tracing: Other examples

Integrated Development Environment
HANDS-ON
IDE Hands On – Create a COMPSs Project

1. Menu File->New -> Project…
2. Select New Application Project
3. Introduce Project Details
   (Also available CompSs ->Implementation->Create Application Project)

IDE Hands On – Create an Orchestration Class

1. Click New… in Orchestration Classes section of Application Editor
2. Introduce the class name and type (Standard)
   (Also available CompSs ->Implementation->Create Orchestration Class)
**IDE Hands On – Create an Orchestration Element**

1. Click **New...** in **Orchestration Elements** sec. of the **Application Editor**
2. Introduce the method name and parameters

(Also available: CompSs ->Implementation->Add Orchestration Element)

**IDE Hands On – Add an Core Element from JAR**

1. Click **New...** in **Core Elements** section of the **Application Editor**
2. Select **New method core element from existing class**
3. Select method

- Select the jar file `/home/user/ide_workspace/Conversor/jars/Conversor.jar`
- Select the Conversor class
- Select the convertToWords method

(Also available: CompSs ->Implementation->Add Core Element)
IDE Hands On – Add an Core Element from scratch

1. Click New… in Core Elements section of the Application Editor
2. Select New method core element from scratch
3. Add class and method names
4. Add return type and params
5. Add method code

IDE Hands On – Introduce the OE code

Include the OE Code to call the CE methods
IDE Hands On – Add conversor dependency to OE

1.- Click Add…

2.- Select the jar library: /home/user/ide_workspace/Conversor/jars/Conversor.jar

IDE Hands On – Deploy Locally

Include location: /home/user/ide_workspace/numConversor/
IDE Hands On – Deploy Grid

Import resources: /home/user/ide_workspace/resources.xml
Select Master resource
Select worker resources
For each selected resource define username, install and working folders

Final Notes

- Sequential programming approach
- Parallelization at task level
- Transparent data management and remote execution
- Can operate on different infrastructures:
  - Cluster
  - Grid
  - Cloud (Public/Private)
    - PaaS
    - IaaS
  - Web services
Final notes

- Project page: [http://www.bsc.es/compss](http://www.bsc.es/compss)
- Direct downloads page:
  - Sample applications & development virtual appliances
  - Tutorials
  - Red-Hat & Debian based installation packages
  - ...

Thank you!
For further information please contact
support-compss@bsc.es