www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

COMPSs Tutorial

July 2013, Barcelona

Javier Álvarez, Rosa M. Badia, Jorge Ejarque,
Daniele Lezzi, Francesc Lordan,
Roger Rafanell, Raül Sirvent, Enric Tejedor

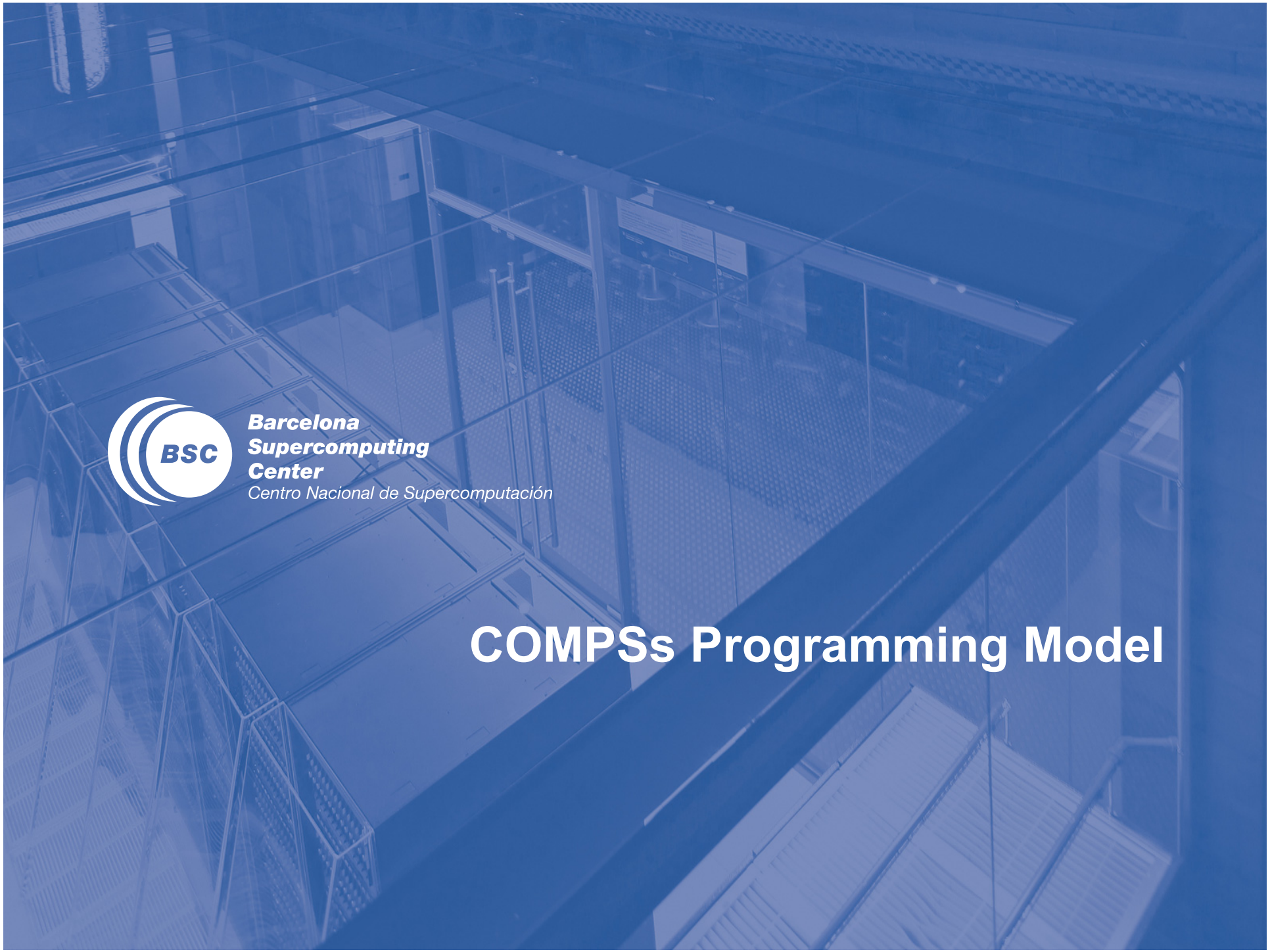
Outline

- **COMPSs Programming Model 11:15 – 11:45**
  - Overview
  - Steps
  - Properties
- **COMPSs Runtime System 11:45– 13:00**
  - Overview
  - Features

LUNCH BREAK 13:00 – 14:00

Outline

- **Hands-on 14:00 – 18:00 (Breaks on demand)**
  - Virtual Machine Setup
  - Application Overview
  - Code modification
  - Configuration, compilation & execution
  - Monitoring, debugging & tracing
  - Final notes


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
COMPSs Programming Model

Overview: Objectives

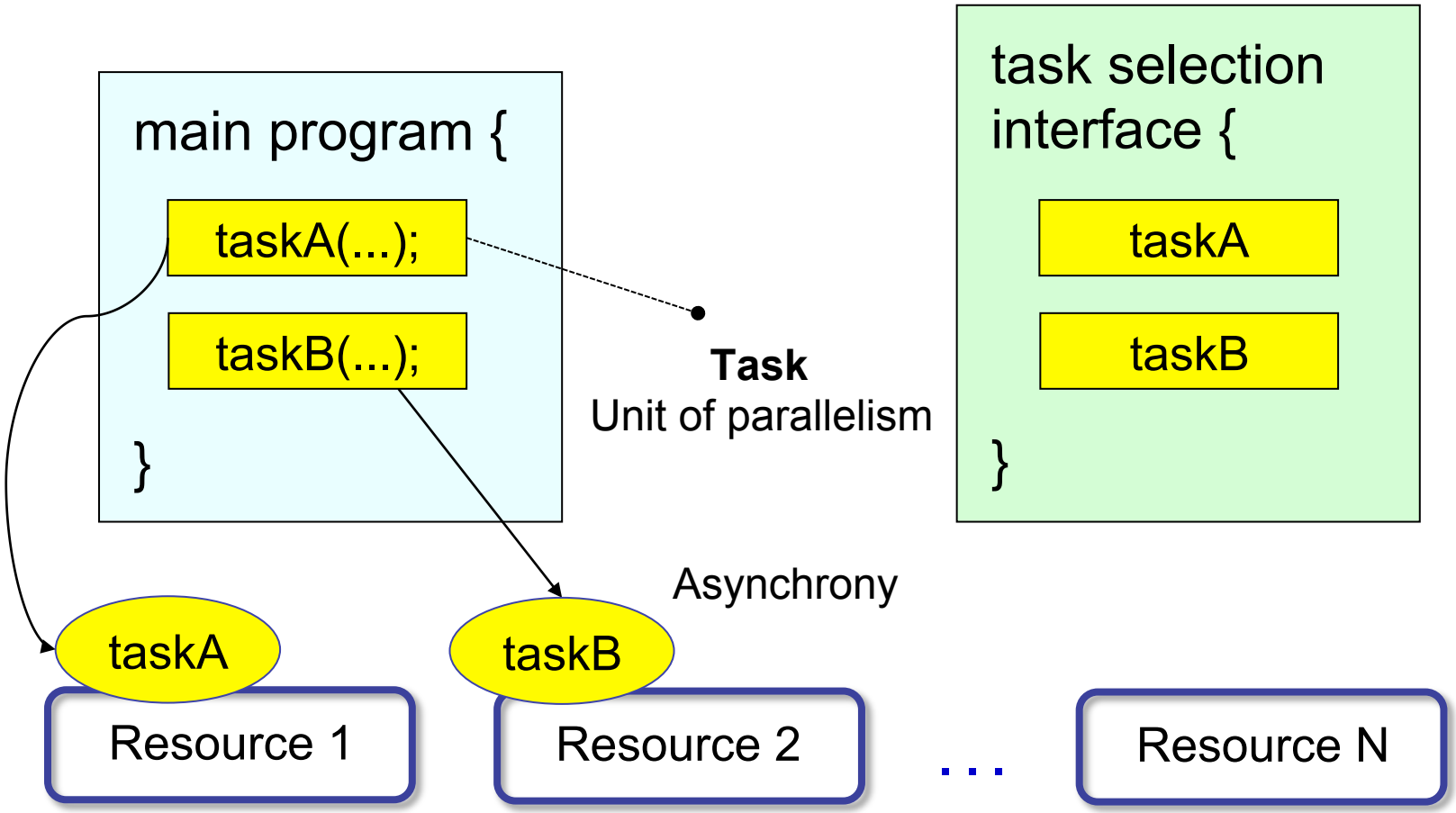
- Reduce the development complexity of Grid/Cluster/ Cloud applications to the minimum
  - Writing an application for a computational distributed infrastructure may be as easy as writing a sequential application

- 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

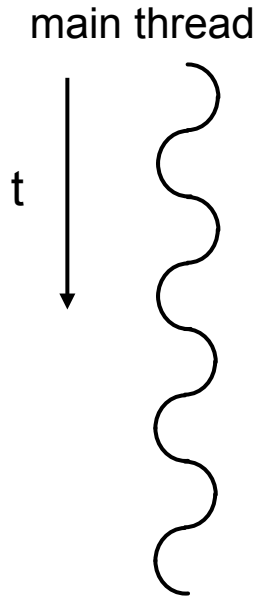
1. Identify tasks

2. Select tasks

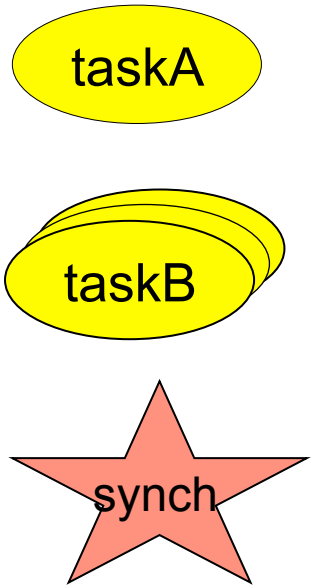


Programming Model: Properties (I)

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




```
Main Program {
   taskA(data1);

   for (int i=0; i< N; i++)
      taskB(data1, data2);

   if (condition)
      process(data2);
}
```

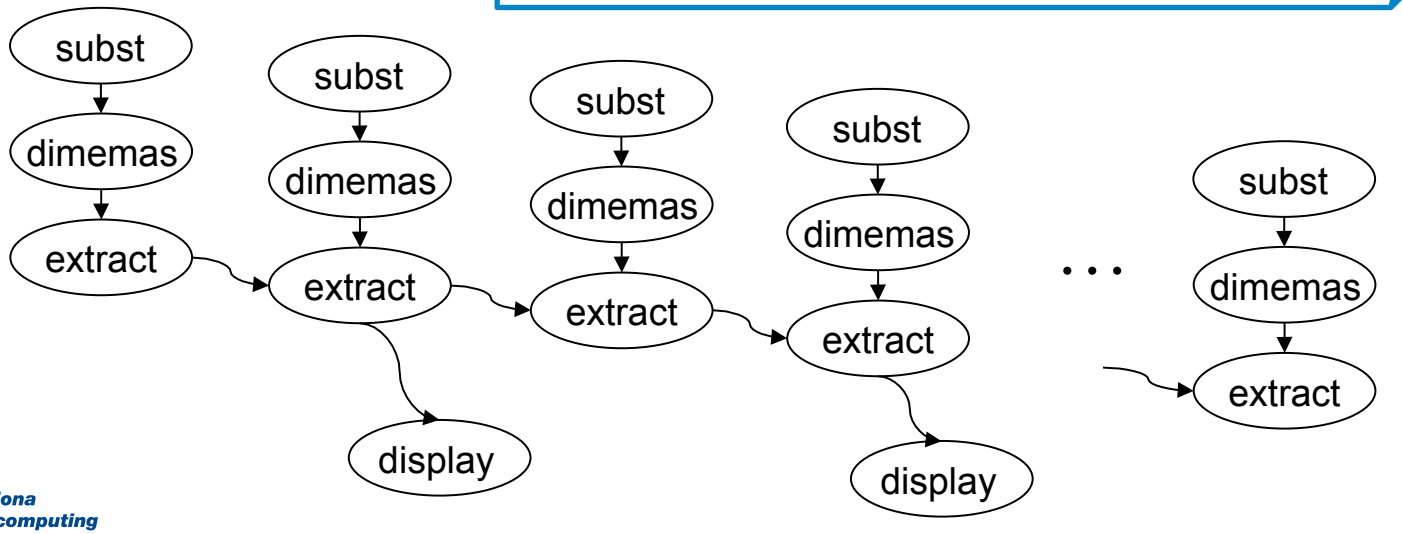


Programming Model: Dependency detection

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

Main Program

```
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

Application

Task Selection Interface

Grid

Cluster

Cloud

Programming Model: Task selection interface

```
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
    );
}
```

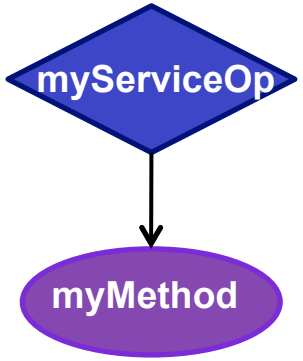
Programming Model: Regular Main program

```
public class App {

   public static void main(String[] args) {
       Query query = new Query(…);
       Reply reply = myServiceOp(query);

       myMethod(reply);

       reply.printToLog();
   }
}
```

Service task call

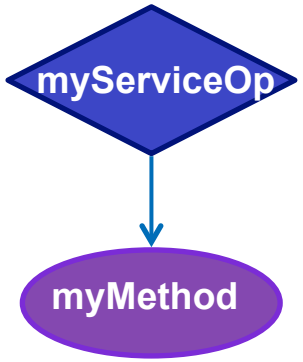
Method task call

Synchronization

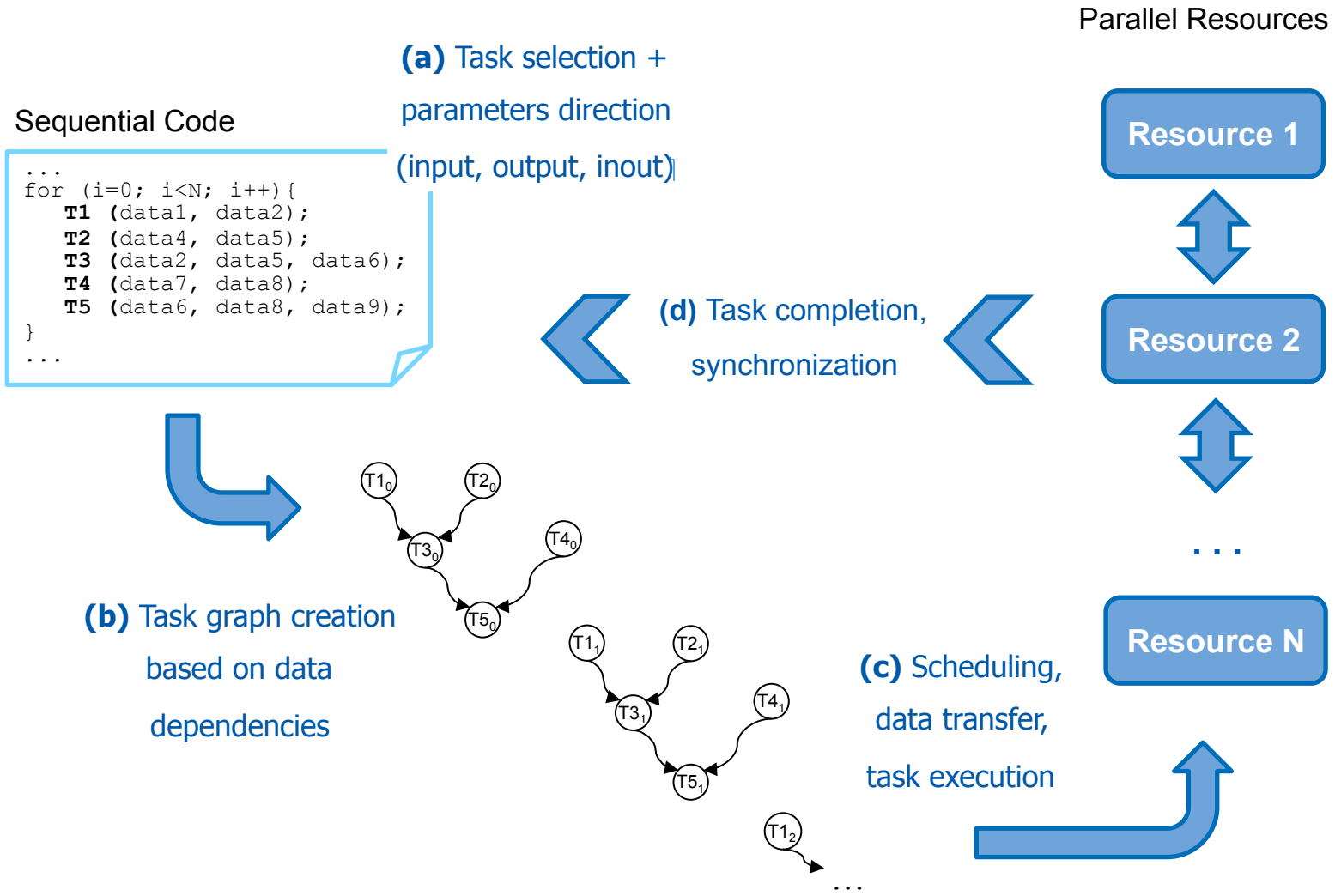


Programming Model: Service Operation

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

      myMethod(reply);

      reply.printToLog();
   }
}
```



Programming Model: Summary



Programming Model: Sample Application

Main program

```
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);
        }
}
```

Subroutine

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

Programming Model: Sample App (Interface)

Task selection interface

```
public interface SimpleItf {

        @Method(declaringClass = "SimpleImpl")
        void increment(
                @Parameter(type = FILE, direction = INOUT)
                String counterFile
        );

}
```

Implementation

Parameter metadata

Programming Model: Final App Code

```
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);
        }
}
```

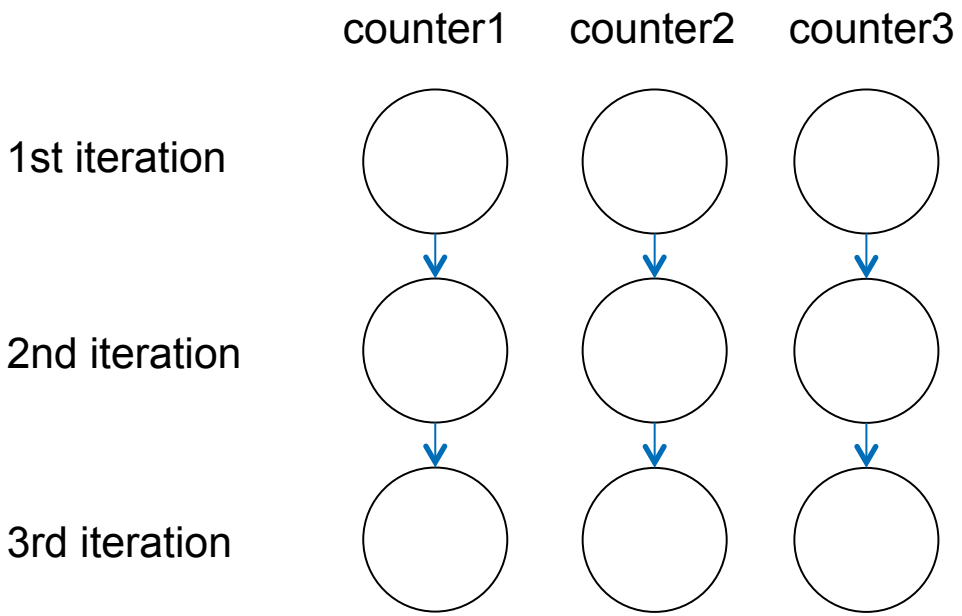
Main program
of the application
NO CHANGES!

Programming Model: Task Graph

Main loop

```
for (i = 0; i < 3; i++) {
        increment(counter1);
        increment(counter2);
        increment(counter3);
}
```

Task graph




BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
COMPSs Runtime System

Runtime System

Application

Task Selection Interface

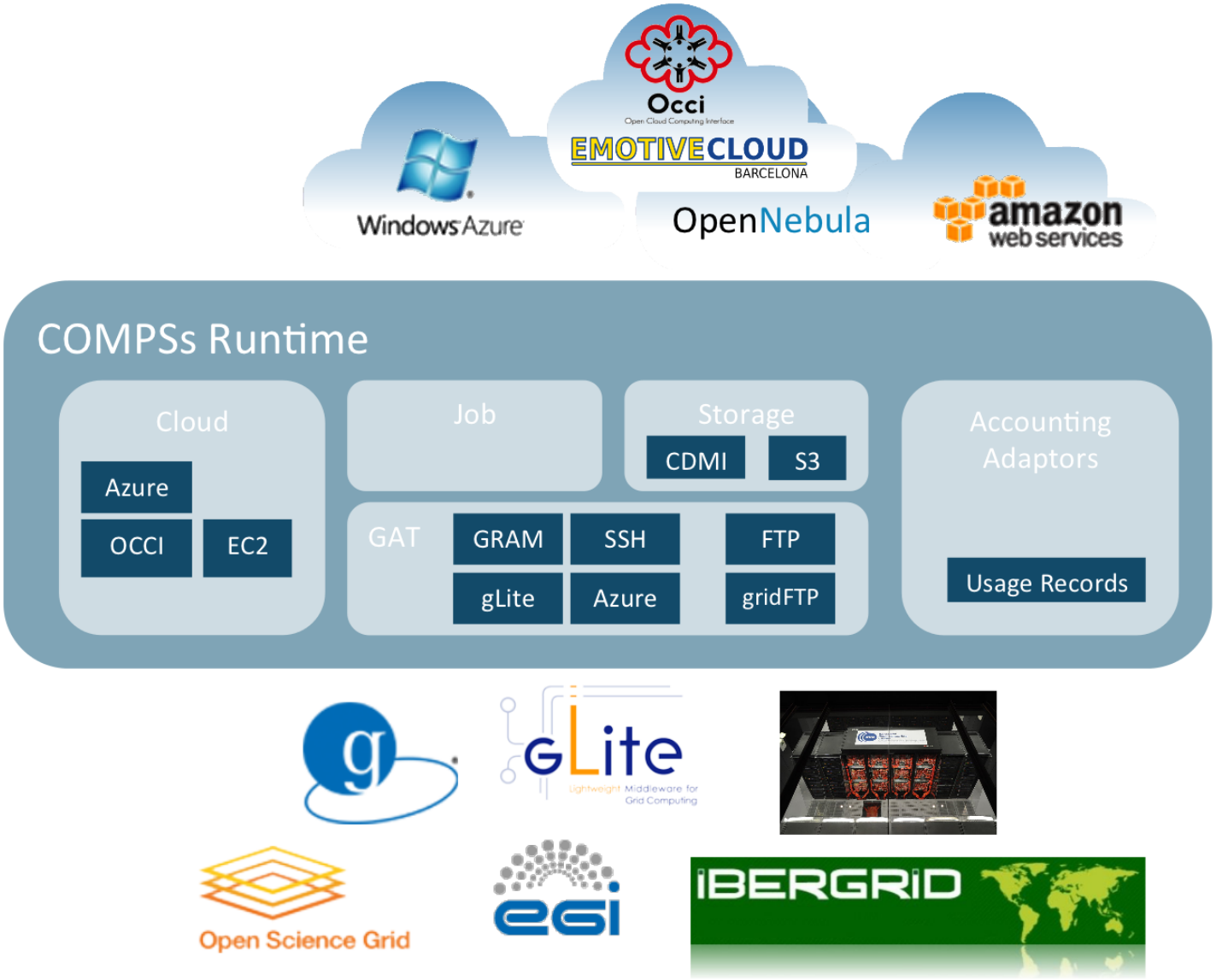
Runtime System

Grid

Cluster

Cloud

Interoperability



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
  - Task nesting


BSC
Barcelona
Supercomputing
Center
Centro Nacional de Supercomputación
HANDS-ON

Hands-On: Overview

- COMPSs Virtual Machine setup
- Applications Overview (HMMER, BLAST & Gene-Detection)
  - Code modification
  - Configuration, compilation & execution
  - Monitoring, debugging & tracing

COMPSs development VM Installation

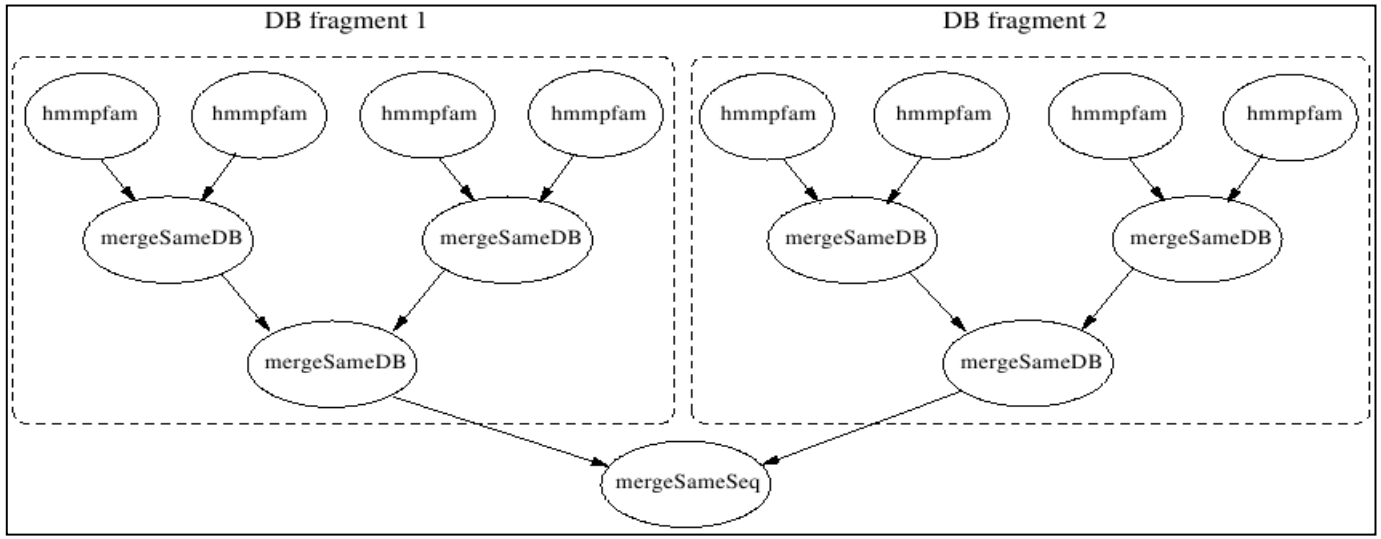
- **Download the COMPSs Development & Test VM (64-bit) OVA**
- **Direct download page:**

  http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/download
- **Import the virtual appliance**

Programming with COMPSs: examples

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

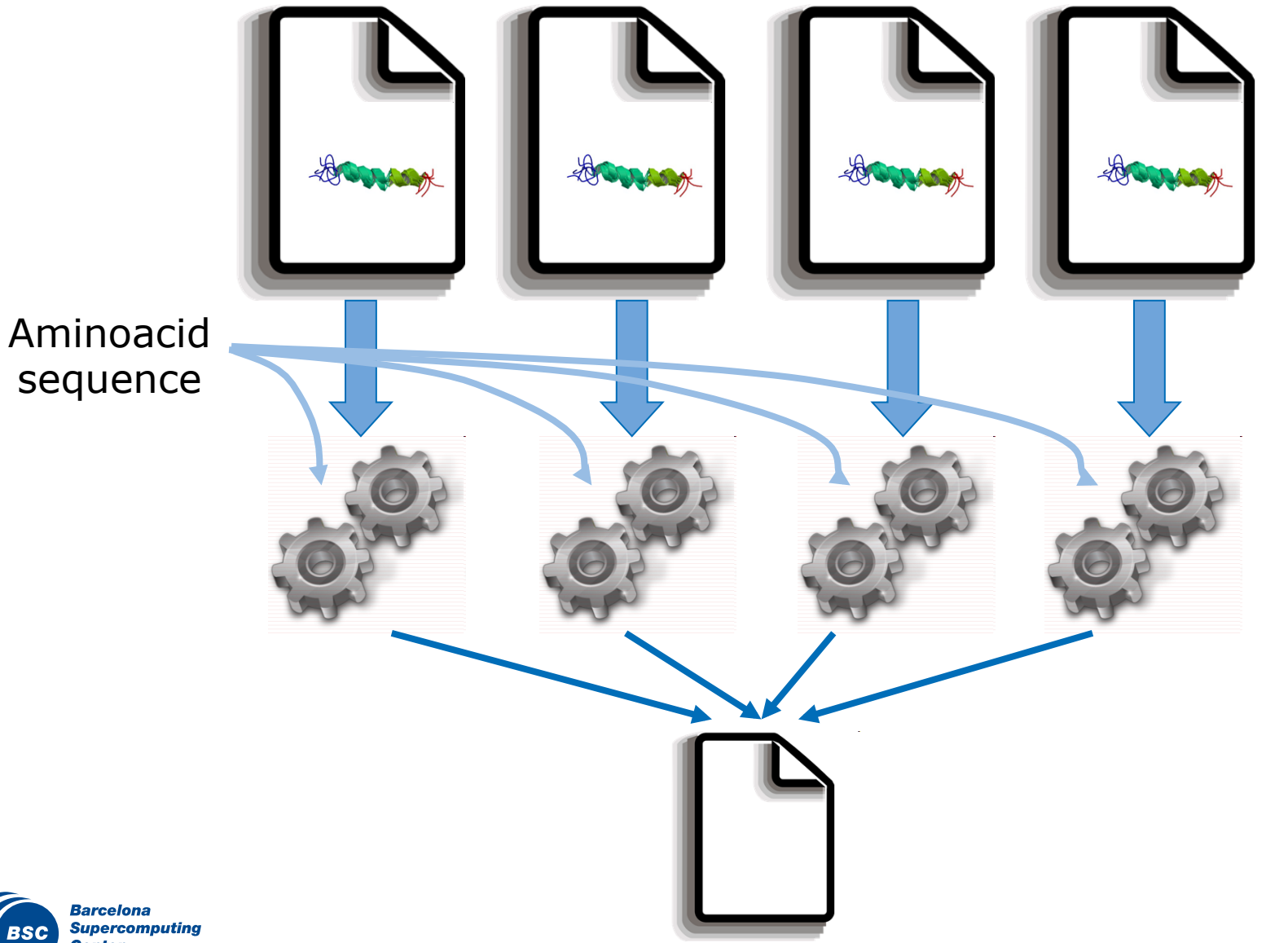
Protein Database

Aminoacid Sequence

IQKKSGKWHTLTDLRA
VNAVIQPMGPLQPGLP
SPAMIPKDWPLIIIDLK
DCFFTIPLAEQDCEKFA
FTIPAINNKEPATRF

Model	Score	E-value	N
IL6_2	-78.5	0.13	1
COLFI_2	-164.5	0.35	1
pgtp_13	-36.3	0.48	1
clf2	-15.6	3.6	1
PKD_9	-24.0	5	1

HMMER example



HMMER example (code)

```
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)

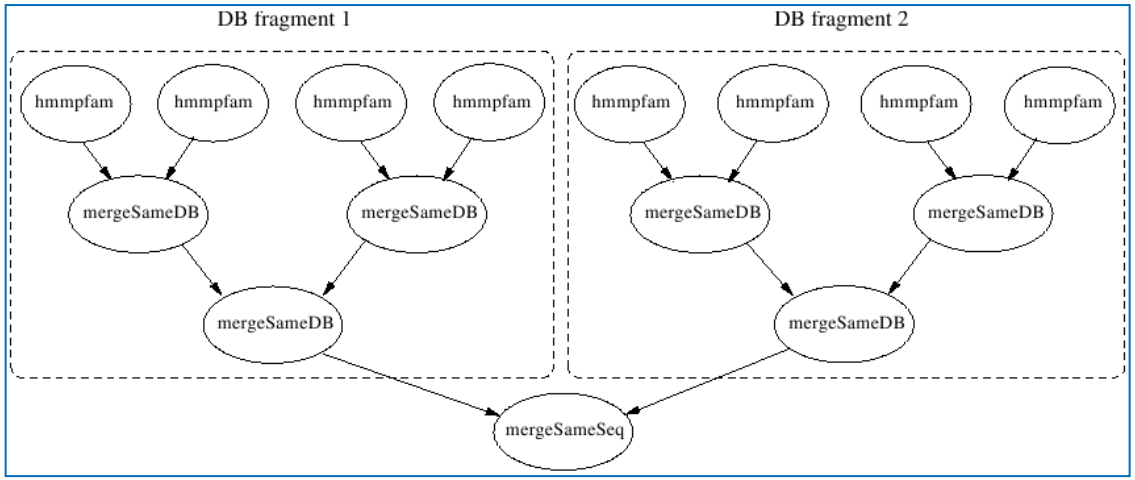
```
public interface HMMPfamItf {

    @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
    );
    …
}
```

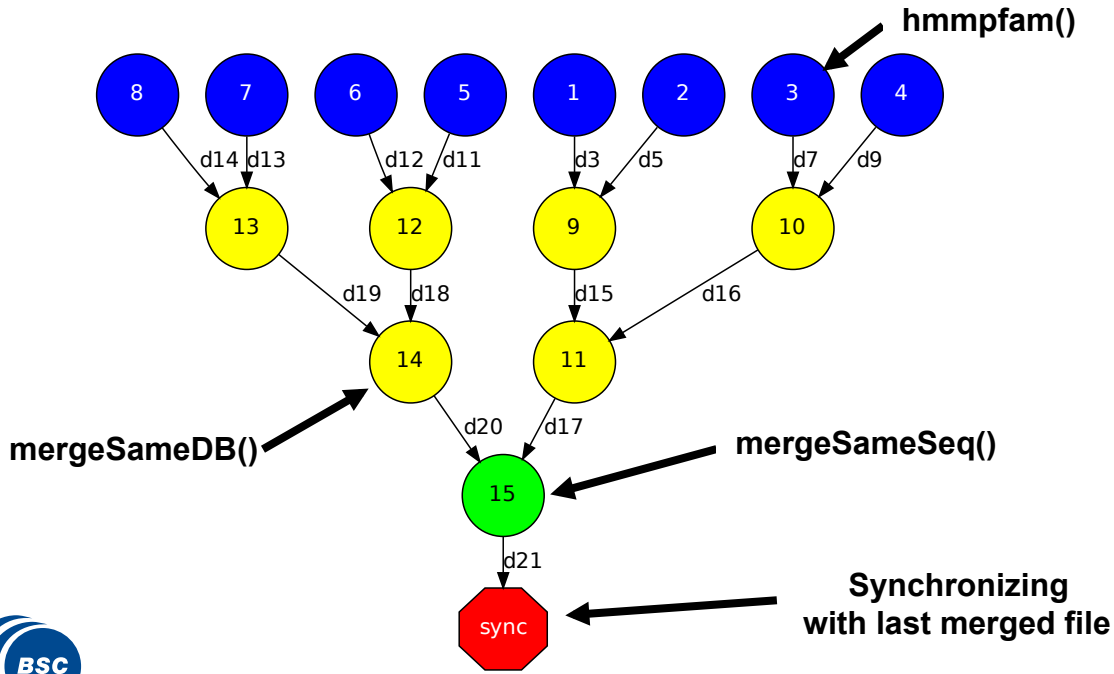
Implementation

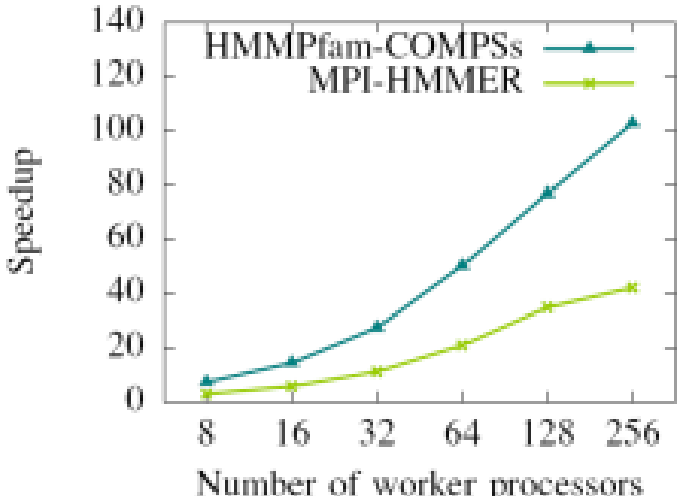
Task constraints

Parameter metadata

HMMER example (workflow)







HMMER: Hands-on

HMMER: Task Selection (work)

- Complete the hmmpfam & mergeSameSeq method interfaces.

HMMER: Configuration, compilation and execution

- 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>
    <!--Description for any physical node-->
    <Resource Name="localhost">
        <Capabilities>
            <Host>
                <TaskCount>0</TaskCount>
                <Queue>short</Queue>
                <Queue/>
            </Host>
            <Processor>
                <Architecture>AMD64</Architecture>
                <Speed>3.0</Speed>
                <CPUCount>2</CPUCount>
            </Processor>
            <OS>
                <OSType>Linux</OSType>
                <MaxProcessesPerUser>32</MaxProcess
            </OS>
            <StorageElement>
                <Size>30</Size>
            </StorageElement>
            …
            …
            <Memory>
                <PhysicalSize>2</PhysicalSize>
                <VirtualSize>8</VirtualSize>
            </Memory>
            <ApplicationSoftware>
                <Software>Java</Software>
            </ApplicationSoftware>
            <Service/>
            <VO/>
            <Cluster/>
            <FileSystem/>
            <NetworkAdaptor/>
            <JobPolicy/>
            <AccessControlPolicy/>
        </Capabilities>
        <Requirements/>
    </Resource>

<ResourceList>
```

HMMER: Configuration, compilation and execution

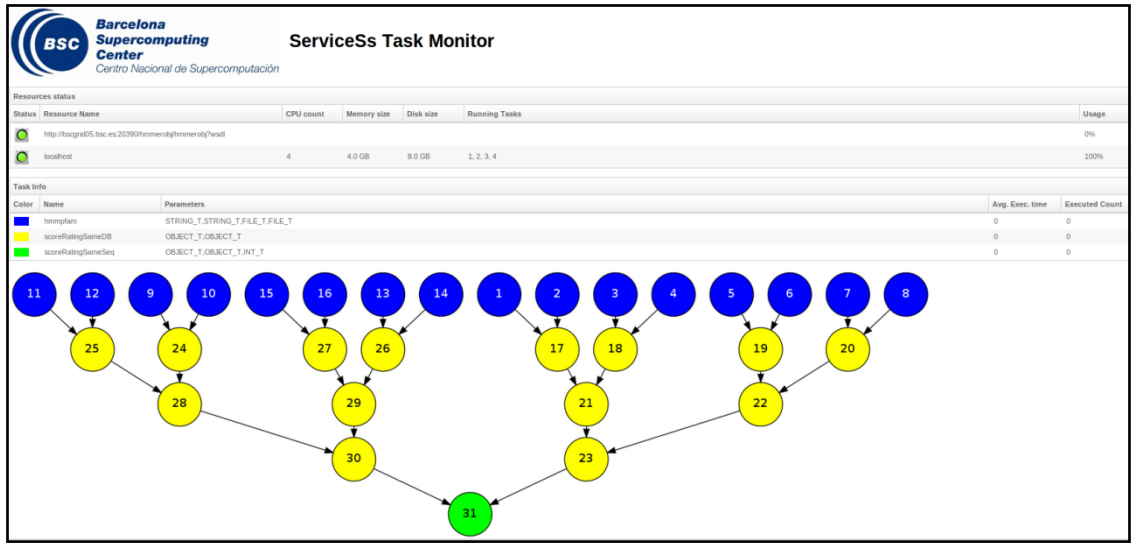
- Compilation (Eclipse IDE)
  - ***Package Explorer -> Project (hmmerobjblanks) -> Export… (Hands-on)***
  - ***Package Explorer -> Project (hmmerobj) -> Export… (Solution)***
- Usage
  - ***runcompss hmmerobj.HMMPfam <database> <sequences> <output> <params>***
- Execution
  - ***cp ~/workspace/hmmerobj/jar/hmmerobj.jar ~***
  - ***export CLASSPATH=$CLASSPATH:/home/user/hmmerobj.jar***
  - ***runcompss hmmerobj.HMMPfam /sharedDisk/Hmmer/smart.HMMs.bin / sharedDisk/Hmmer/256seq /home/user/out.txt 2 8 -A 222***

HMMER: Configuration, compilation and execution

```
user@bsccompss:~$ runcompss hmmerobj.HMMPfam /sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/
      256seq /home/user/out.txt 2 8 -A 222
-e
---------- 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 /tmp/hmmer_frags/dbF0_1 in mode WRITE
[  API] -  Opening file /tmp/hmmer_frags/dbF1 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: Monitoring

- 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**

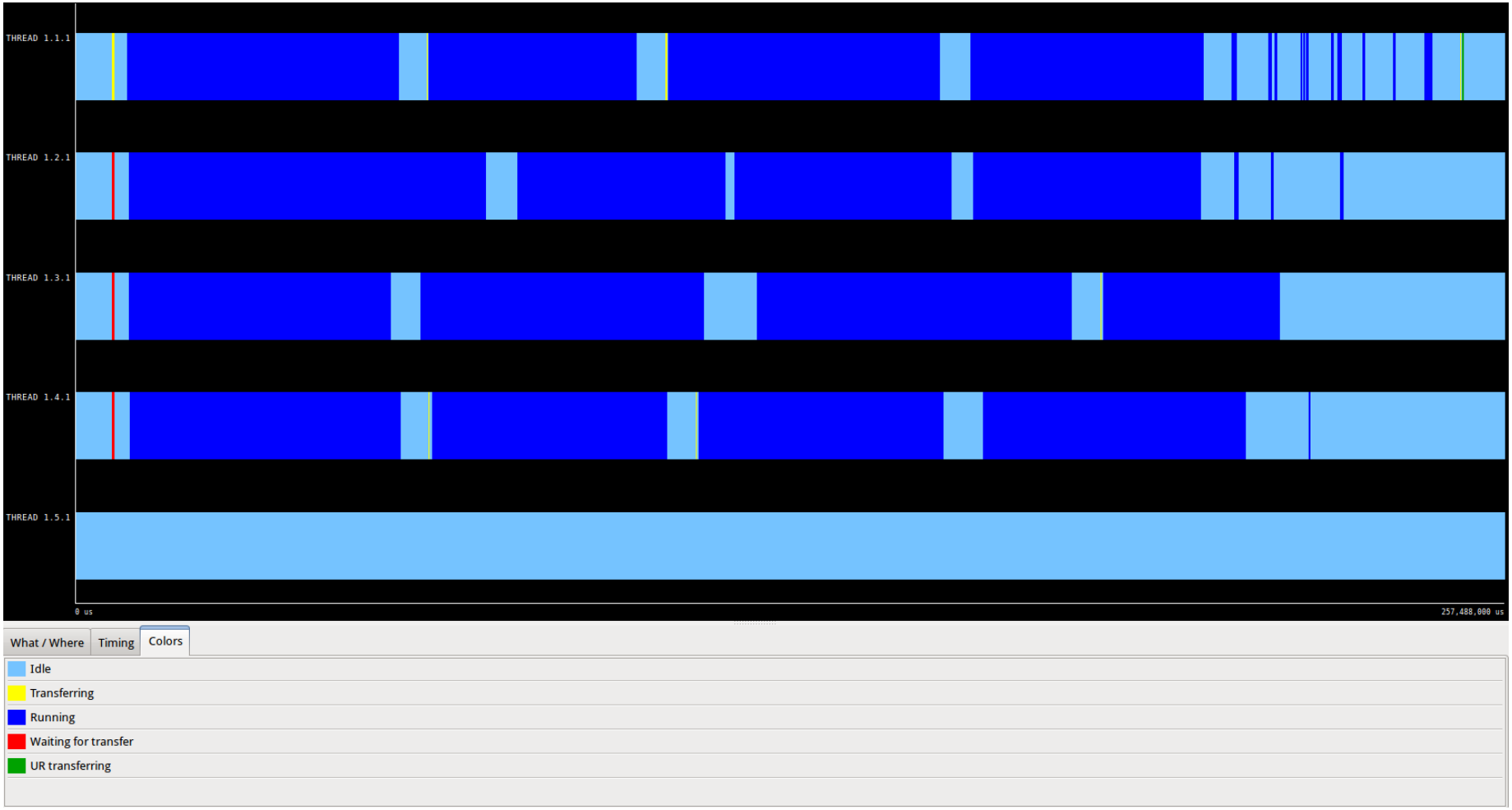
Tracing - Overview

- COMPSs can generate post-execution traces of the distributed execution of the application.
- Useful for analysis and diagnosis.
- How it works:
  - For each task execution and file transfer, an XML file is created to keep track of that event.
  - At the end of the execution, a perl script reads all the XML files and generates a Paraver trace file.
- Traces can be visualized with the Paraver tool
  - **http://www.bsc.es/paraver**

HMMER: Tracing

- The application code has to be instrumented to collect execution data.
  - ***prepare4tracing ~/hmmerobj.jar***
- The runtime is launched with the tracing flag enabled
  - ***export CLASSPATH=$CLASSPATH:/home/user/tracing_hmmerobj.jar***
  - ***runcompssext --app=hmmerobj.HMMPfam --tracing=true --cline_args="/sharedDisk/Hmmer/smart.HMMs.bin /sharedDisk/Hmmer/256seq /home/user/out.txt 2 8 -A 222"***
- Generate the tracing file
  - ***gentrace /home/user/IT/hmmerobj.HMMPfam/***
- Visualize with paraver
  - ***wxparaver /home/user/IT/hmmerobj.HMMPfam/*.prv***

HMMER: Tracing



HMMER: Trace interpretation

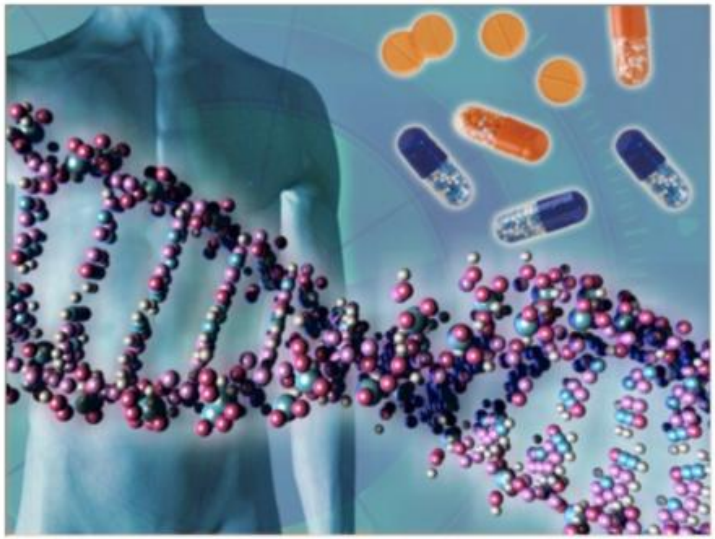
- Lines in the trace:
  - One line for the master
  - N lines for the workers
- Meaning of the colours:
  - Light blue: idle
  - Dark blue: running a task
  - Yellow/green: transferring data
  - Red: waiting for data to be transferred
- Flags (events):
  - Start / end of task
  - Start / end of data transfer

BLAST: Hands-on

Bioinformatics Scenario

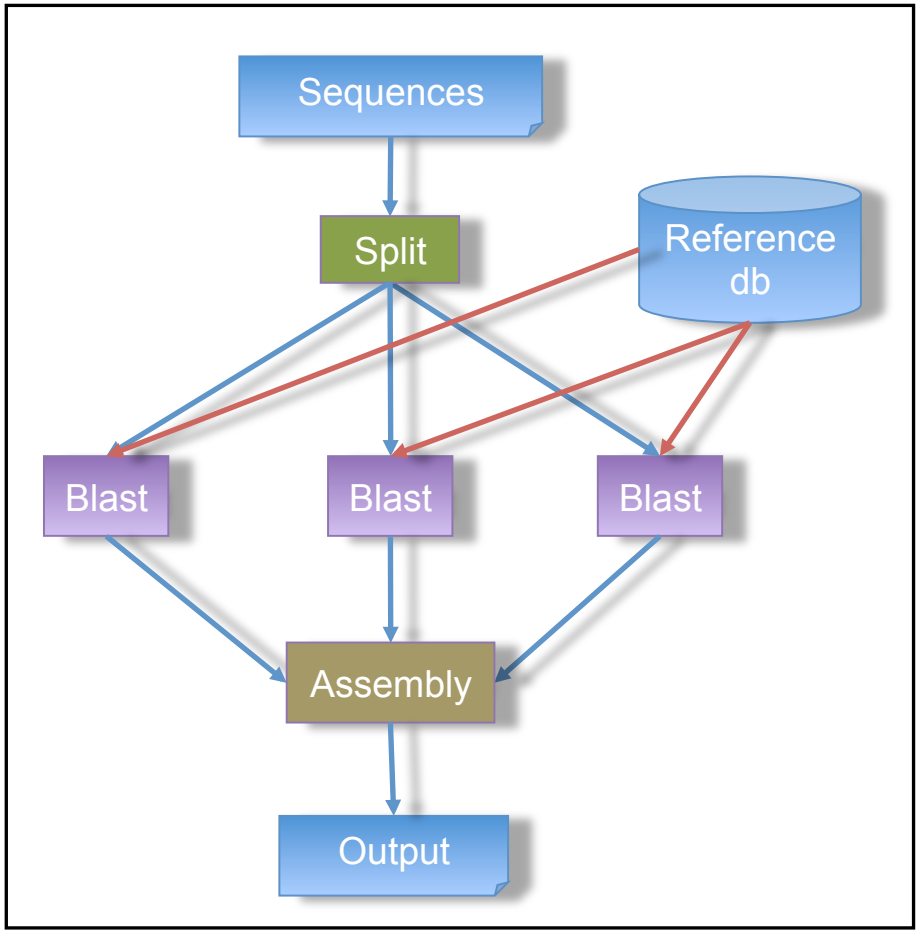
- BLAST (Basic Local Alignment Search Tool) Suite:
  - BLAST: An algorithm for comparing primary biological sequence information, such as the amino-acid sequences of different proteins or nucleotides of DNA sequences.

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.

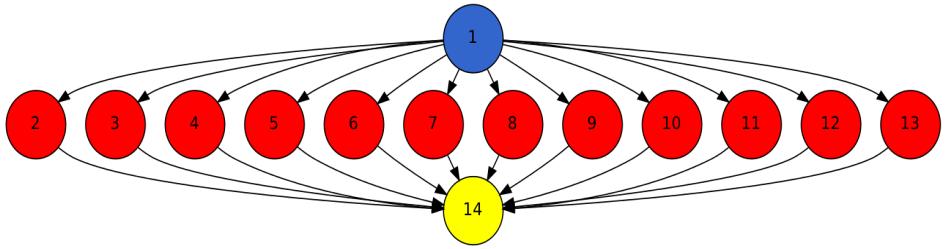


BLAST: Hands-on

- **BLAST**

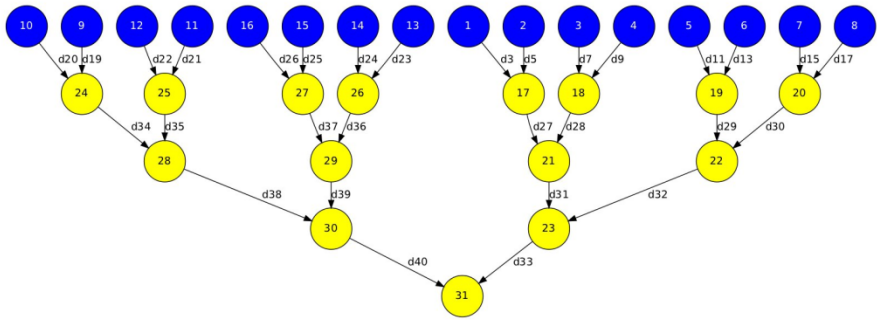


All-to-One Reduction:



OR

Tree-based Reduction:



BLAST: All-to-One reduction

- Main Application (All-to-One):

```
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(partialOutputs, outputFileName, tempDir, nFrags);
}
```

BLAST: All-to-One reduction

- Remote task implementation:

```
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+ " " +commandArgs;

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

BLAST: All-to-One reduction

Creation of the annotated interface for the selection of remote tasks

```
public interface BlastItf {

    @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 8 /tmp/ /home/user/out.txt***

BLAST: Compilation and 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...
[API] - Opening file /tmp/seqFile1b495168-e913-430a-a347-9894015911e1.sqf in mode APPEND
…

Aligning Sequences:

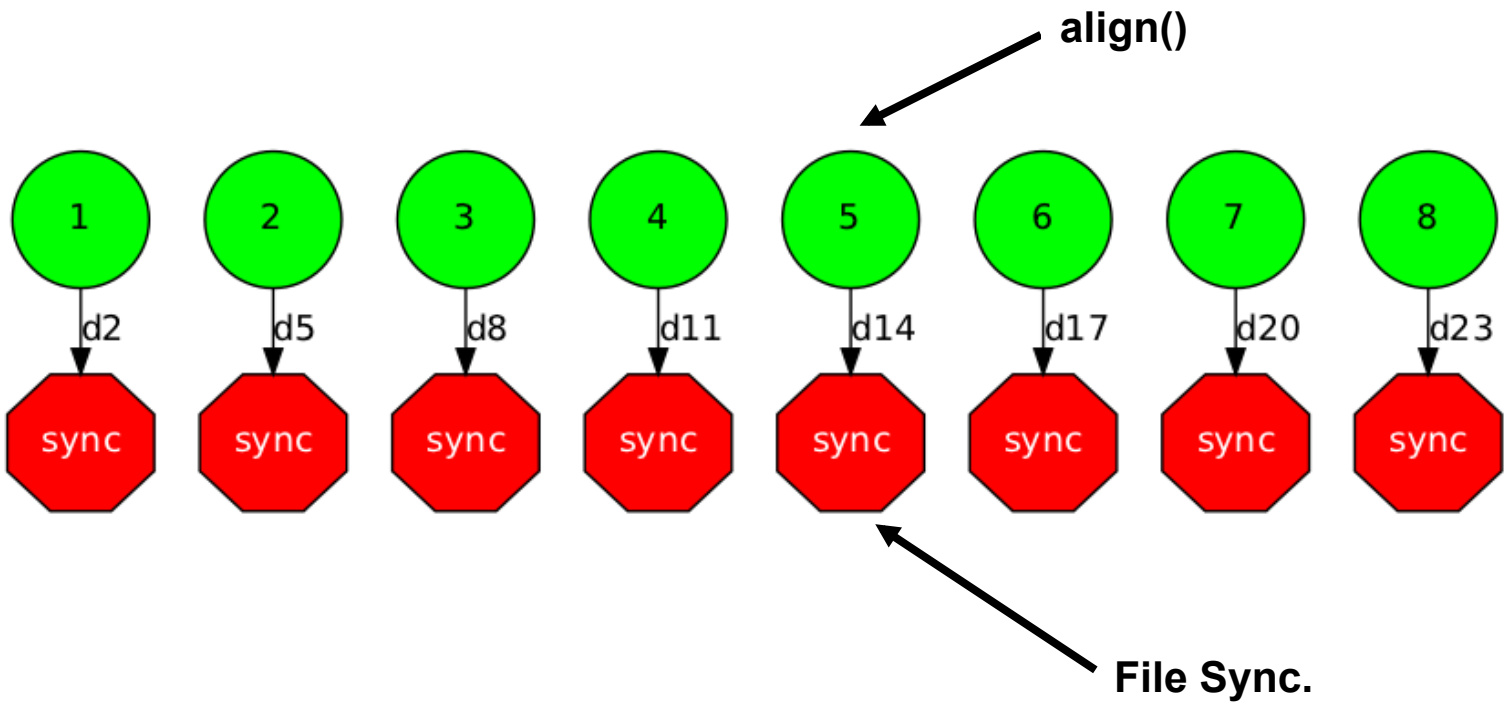
- Number of fragments to assemble -> 8

[API] - Opening file /home/user/IT/blast.Blast/out.txt in mode WRITE
- Assembling partial output -> /tmp/resFile1b495168-e913-430a-a347-9894015911e1.result.txt to final output file -> /home/user/IT/blast.Blast/out.txt
…
- Assembling partial output -> /tmp/resFile270855af-307b-4a1e-bc42-0e0cf22256ae.result.txt to final output file -> /home/user/IT/blast.Blast/out.txt
-Sequences assembled in 184 seconds
…

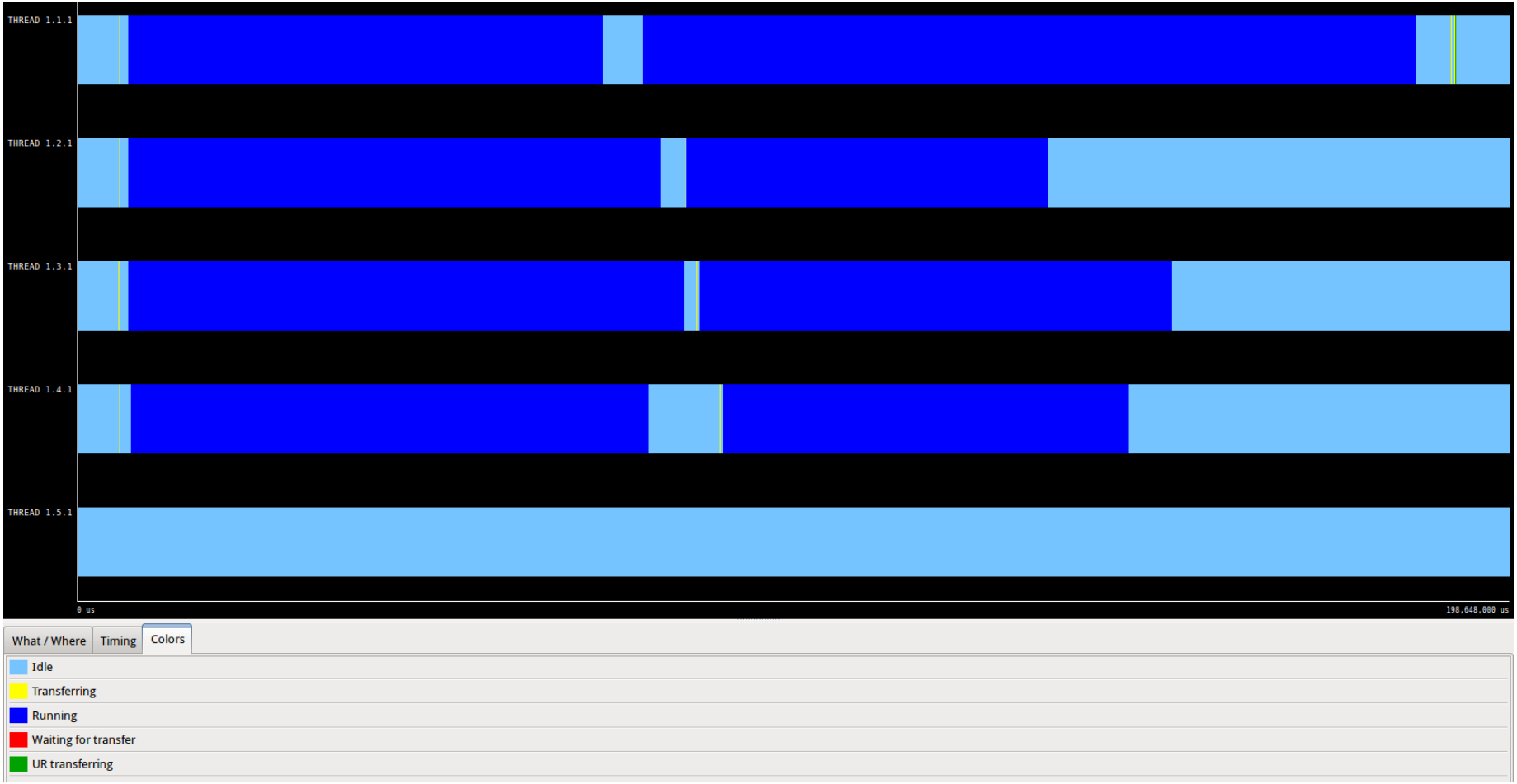
BLAST: All-to-One (work)

- Generate the final graph.
- Launch BLAST (All-to-One) in tracing mode.

BLAST: All-to-One (Graph)



BLAST: All-to-One (Trace)



BLAST: Tree-based reduction (work)

- Code the final reduction and its interface.

BLAST: Tree-based reduction

- Main Application (Tree-based):

```
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);
  }

  //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.assemblyPartitions(partialOutputs.get(result),partialOutputs.get(result+neighbor));
               lastMerge = partialOutputs.get(result);
          }
       }
       neighbor*=2;
    }
}
```

BLAST: Tree-based reduction

Creation the annotated interface for the selection of the remote tasks

```
public interface BlastItf {

    @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.STRING, direction = Direction.IN)
        String commandArgs);

    @Method(declaringClass = "blast.BlastImpl")
    @Constraints(processorCPUCount = 2, memoryPhysicalSize = 2.0f)
    void assemblyPartitions(
        @Parameter(type = Type.FILE, direction = Direction.INOUT)
        String partialFileA,

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

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...
[API] - Opening file /tmp/seqFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.sqf in mode APPEND
…

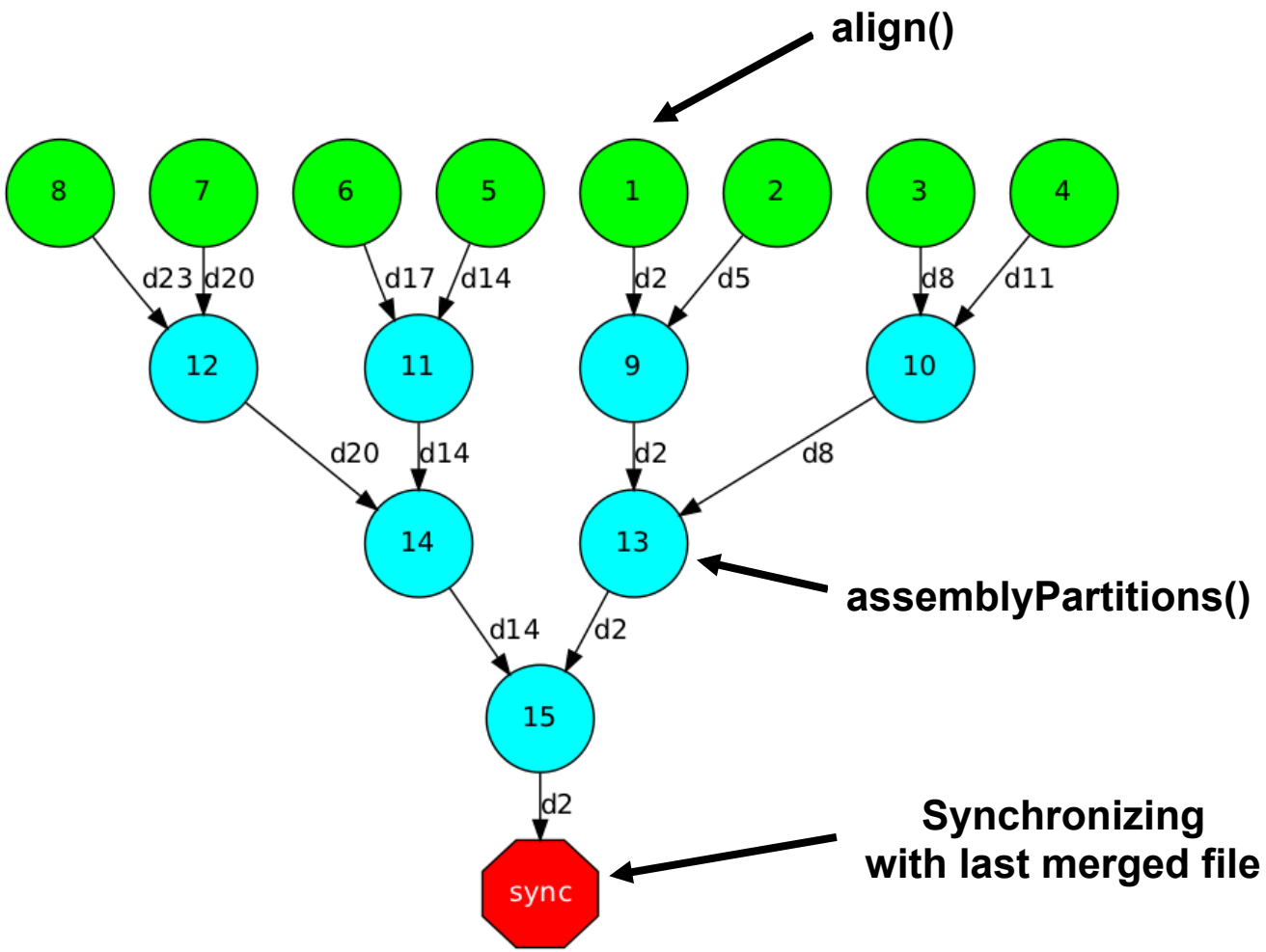
Aligning Sequences:

- Number of fragments to assemble -> 8
- Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFile815b4ff6-a077-422b-bc9b-9c6e10d8a417.result.txt
…
- Merging files -> /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt and /tmp/resFile81605bf8-b0f4-46bc-a521-9f289d219ef3.result.txt

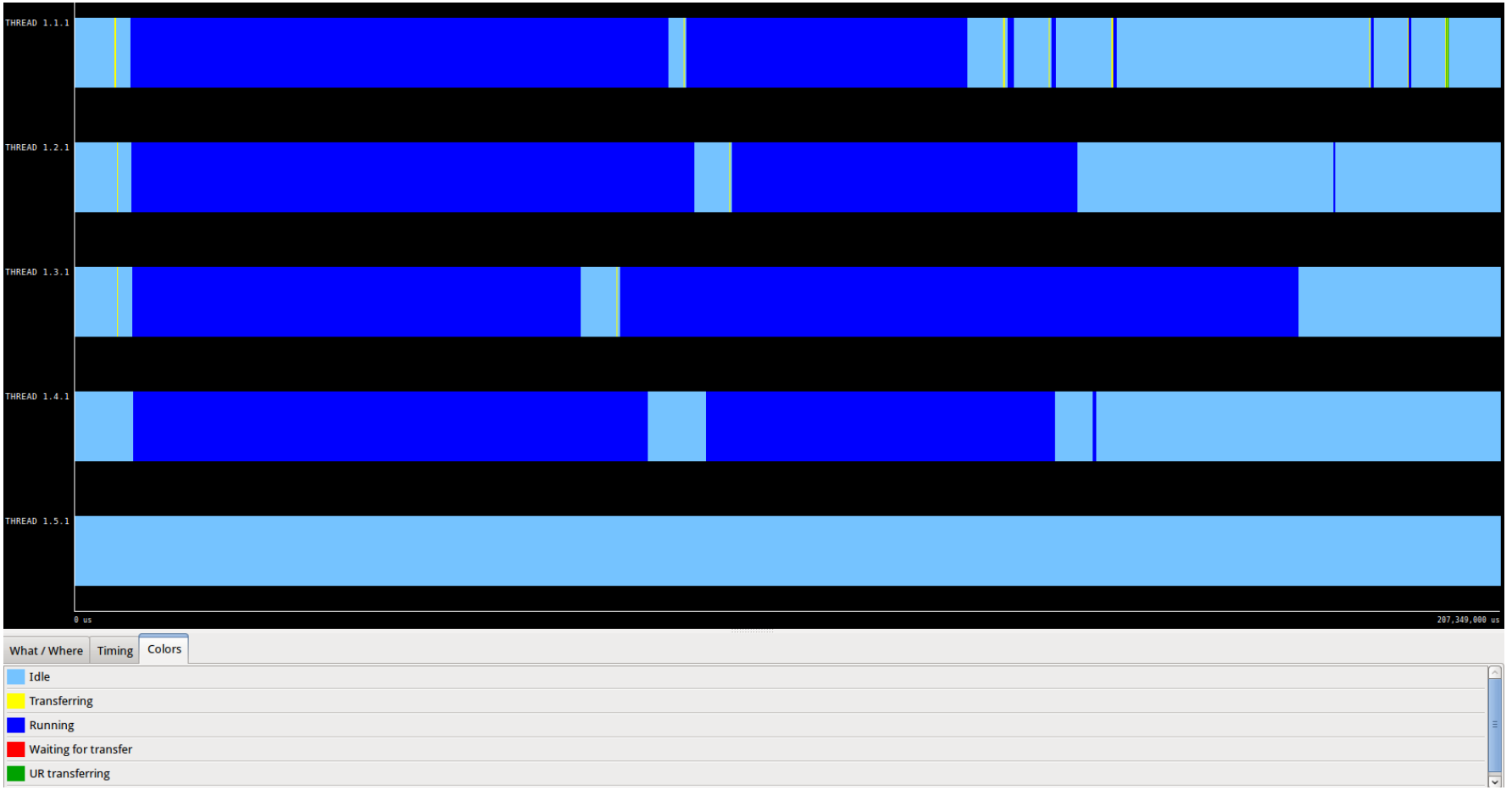
Moving last merged file: /tmp/resFileb0fa2b12-d0f6-42c1-b499-1e207e30ad84.result.txt to /home/user/IT/blast.Blast/out.txt

[API] - Opening file /home/user/IT/blast.Blast/out.txt in mode WRITE
- /sharedDisk/Blast/sequences/sargasso_test.fasta sequences aligned successfully in 193 seconds
…

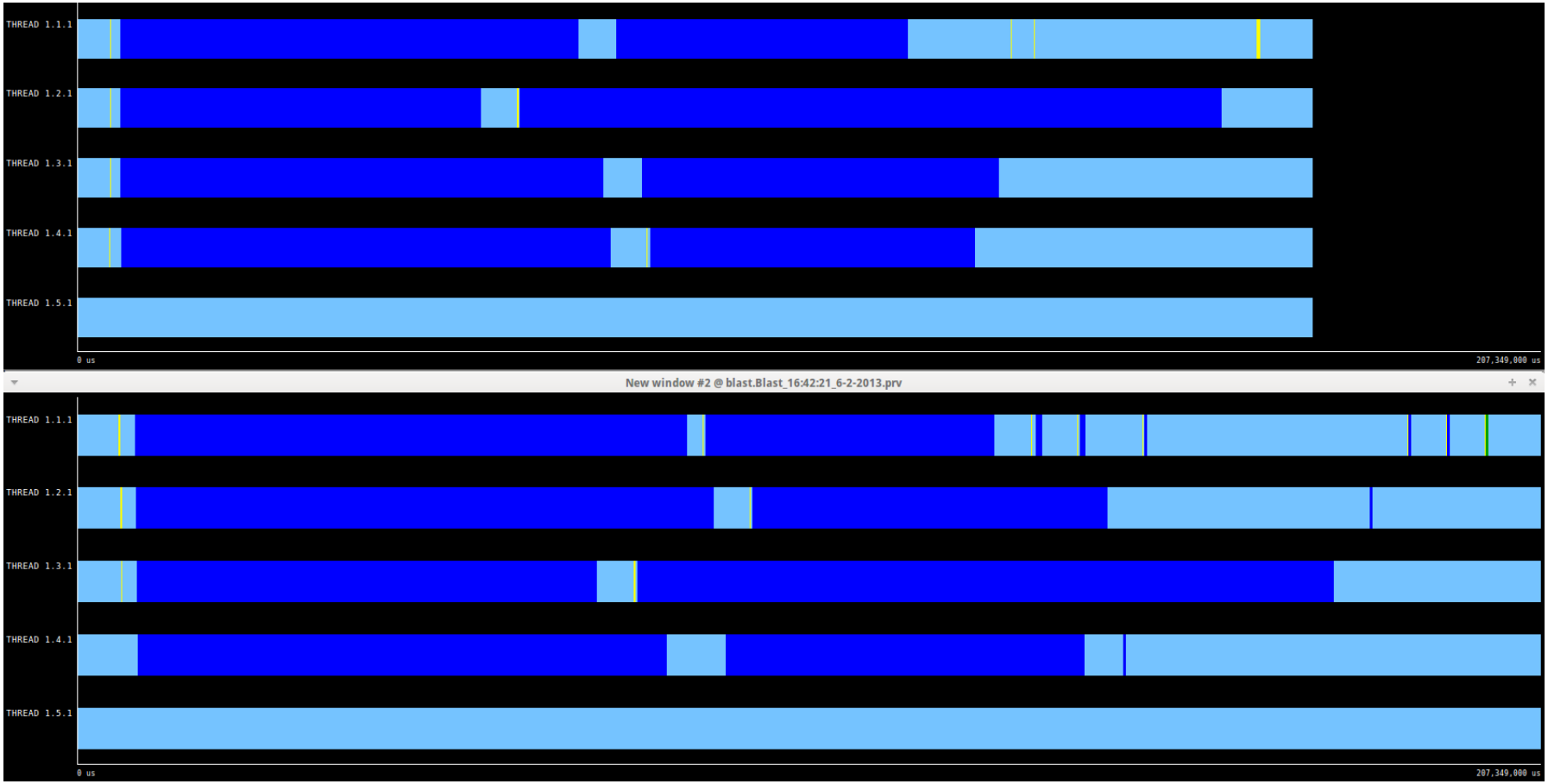
BLAST: Tree-based (Graph)



BLAST: Tree-based (Trace)

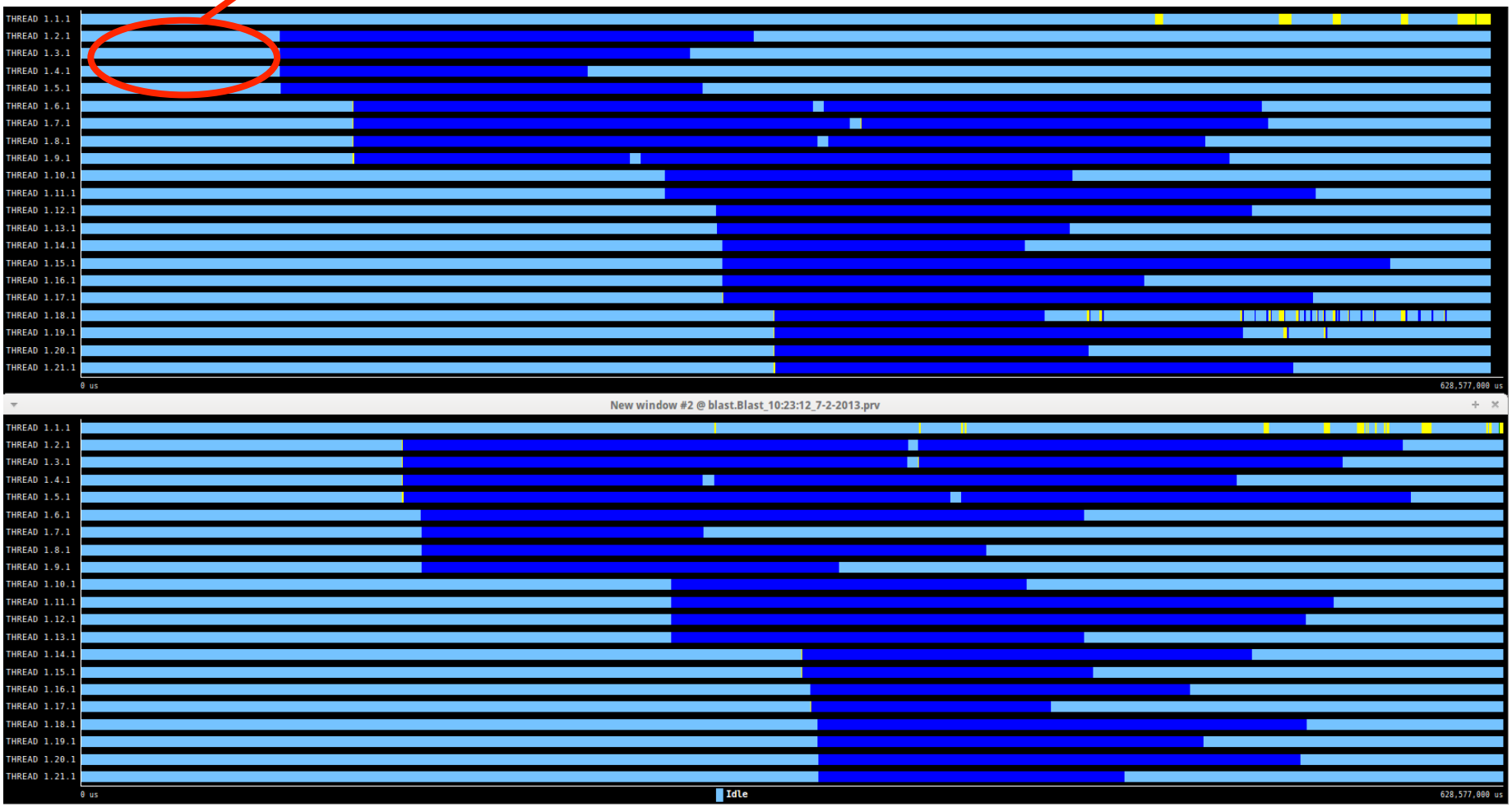


BLAST: All-to-One vs Tree-based (Local)

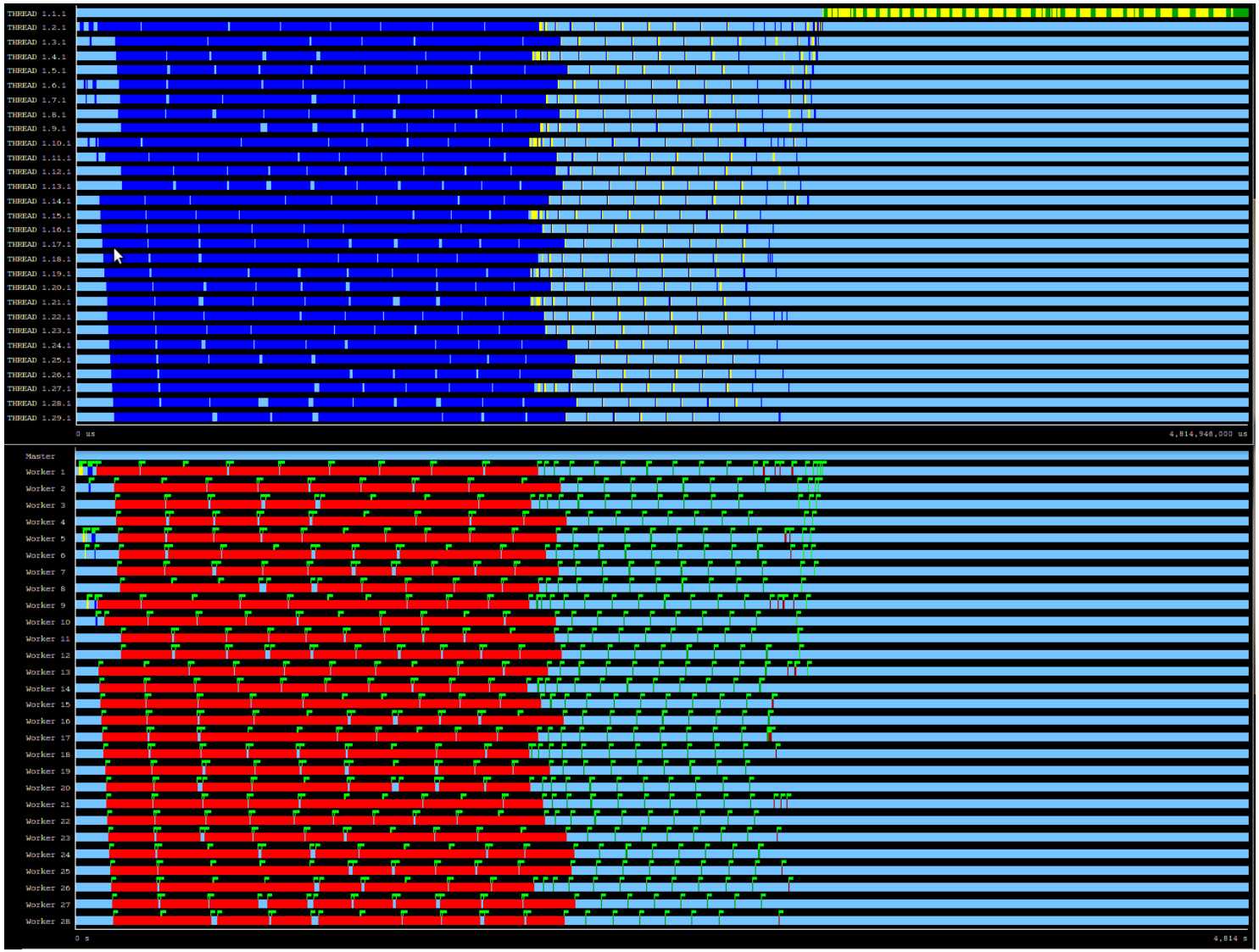


BLAST: All-to-One vs Tree-based (Cloud)

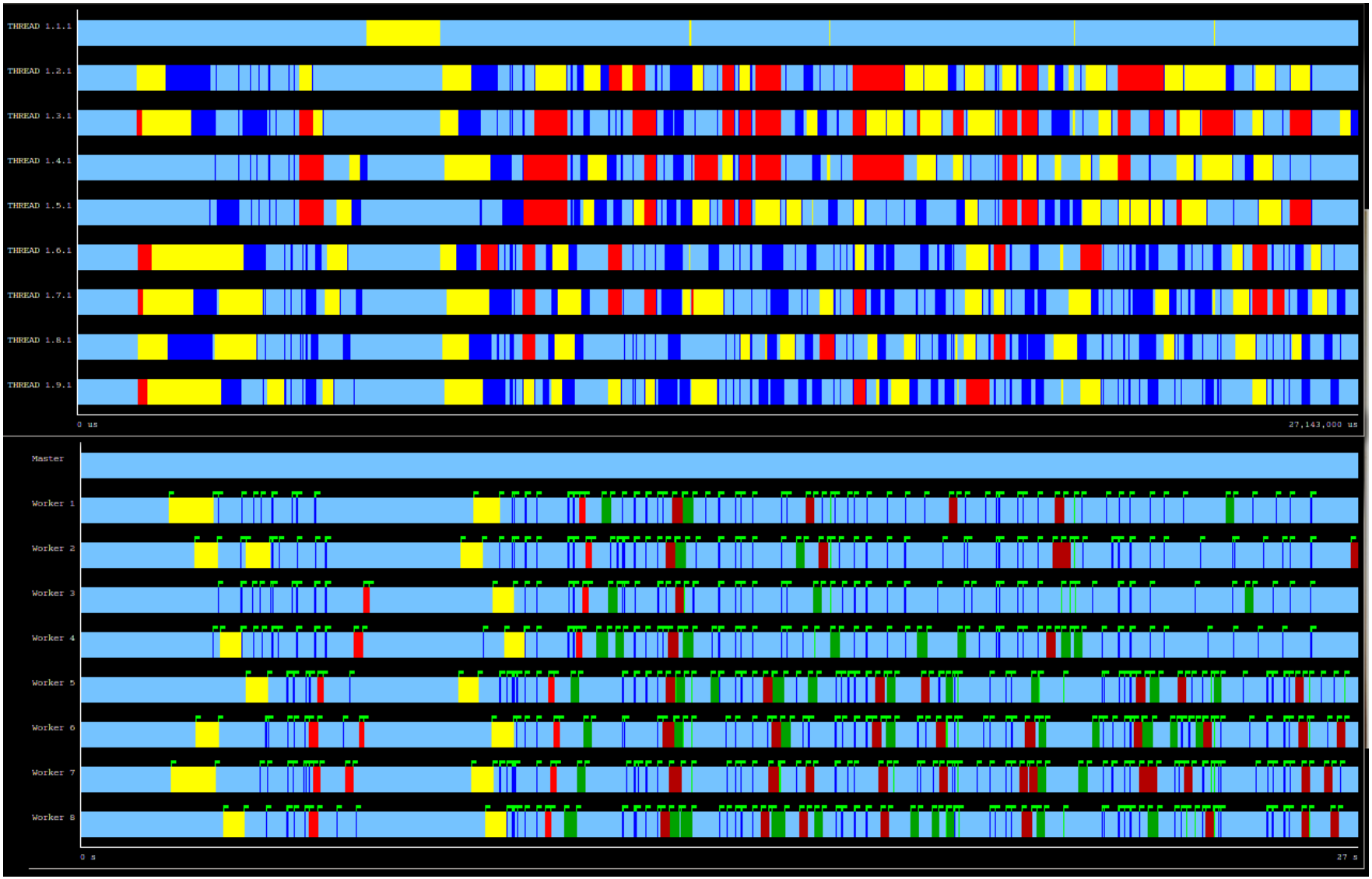
Virtual Machine Creation



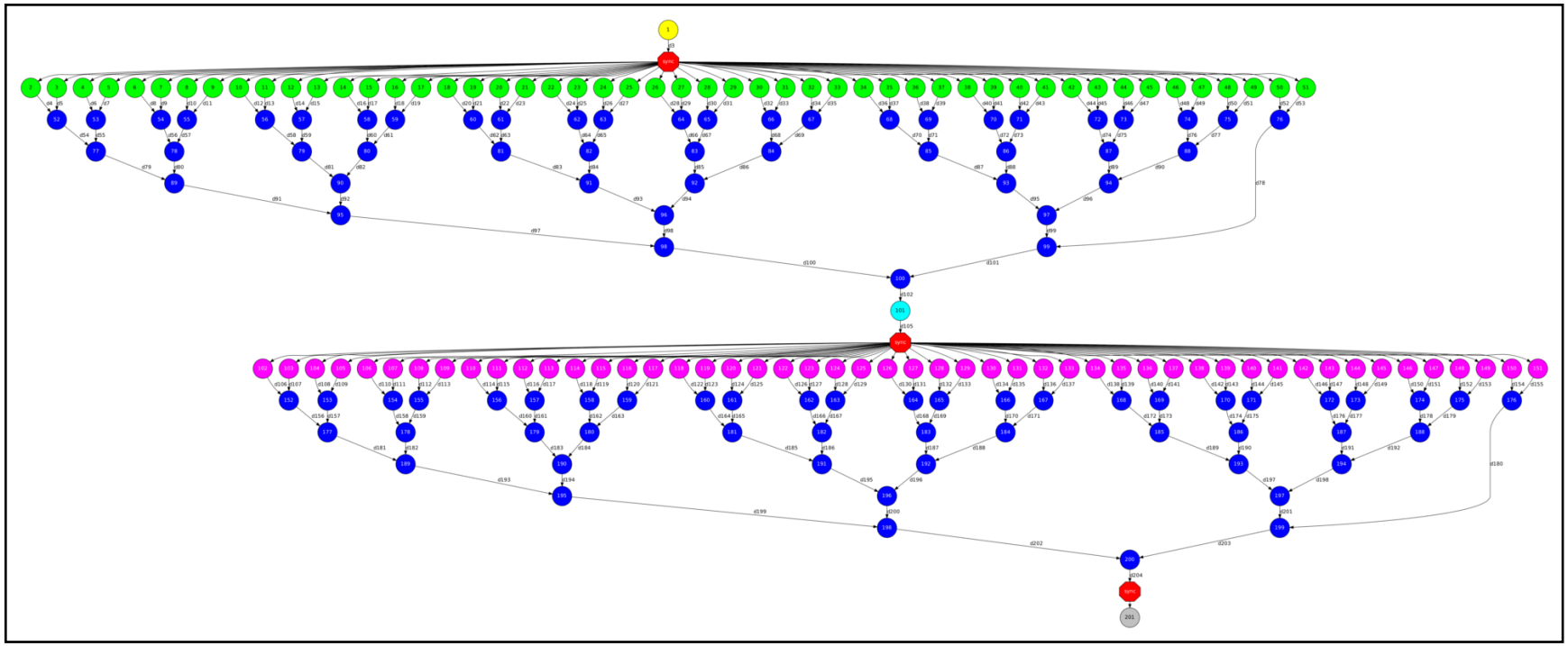
Other tracing examples

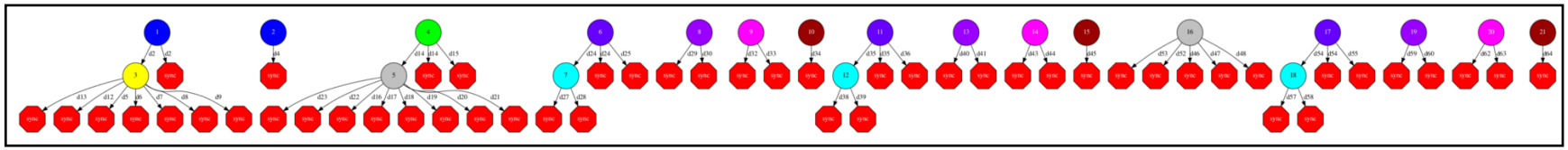


Other tracing examples

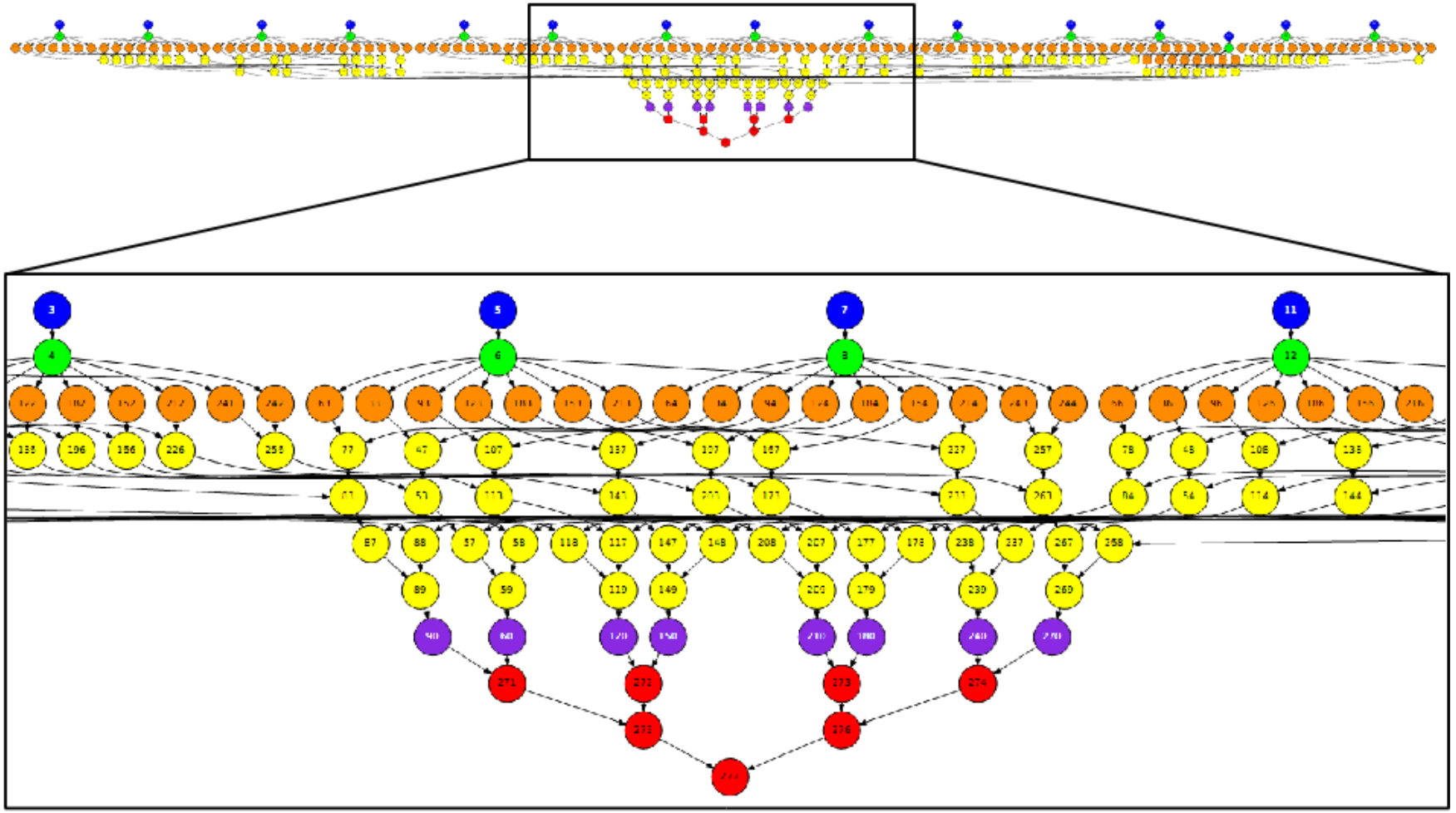


Complex workflow examples (1)





Complex workflow examples (2)



Demo: BLAST

Conclusions

- 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**
- **Direct downloads page:**

  http://www.bsc.es/computer-sciences/grid-computing/comp-superscalar/download
  - *Sample applications & development virtual appliances*
  - *Tutorials*
  - *Red-Hat & Debian based installation packages*
  - *…*

www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thank you!

For further information please contact

support-compss@bsc.es