INTERNATIONAL LABORATORY OF BIOINFORMATICS

A generic Job Submission Tool (JST)

Guido Cuscela
INFN-Bari

Outline

- **The experience with bioinformatics applications**
  - Characteristics and issues
- **Job Submission Tool:**
  - How does it work
  - Features
  - Actual status
  - Future
- **Conclusions**

Introduction

- **EGEE infrastructure perfectly fits problems that can be subdivided in (many) elementary and independent tasks**
- **A Job Submission Tool (JST) has been developed inside LIBI and BioinfoGRID projects in order to exploit Grid power providing a solution for the submission of a large number of jobs to the grid in an unattended way**
- **Several bioinformatics applications have this characteristic:**
  - Often it is needed to submit a large number of small jobs
  - A consistent fractions of them is expected to fail for any kind of reasons due the distributed environment such as Grid one
  - Checking the exit status of a large number of jobs requires too much effort
    - For example it could be not feasible to check all the jobs failed and then resubmit them

How does it work(1)

- **Job Submission Tool**
  - In the first step of the JST workflow, the task list, all the atomic “task” in which the full problem has been subdivided is stored into a central DB (the TaskListDB) server.
  - The TaskListDB is then used to control the assignment of tasks to the jobs and to monitor the jobs execution
  - Tasks: they are the independent activities that need to be executed in order to complete the challenge related to an application
  - Job: It is the process executed on the grid worker nodes that takes care of a specific task execution
  - A single job can take care of more the one task or more jobs may be necessary to execute one task (due for example to failures that may require a job resubmission)
  - On a UI a daemon is always running to check the TaskListDB for new applications to run and new jobs to submit.
  - The same job is submitted every time
    - The differences is only related to the task they have to complete

How does it work(2)

- **The task status**
  - Free: the task is ready to be executed but not assigned yet
  - Running: the task has been assigned to a job that is running on the grid
  - Done: the task has been successfully executed
- **On the WN the task is executed by a wrapper that is responsible of providing information useful for monitoring purpose:**
  - If the task is executed correctly the wrapper changes its status from "Running" to "Done".
  - If a task does not reach the "Done" status in a reasonable amount of time, the task is considered "failed" and can be assigned to a new job.
  - To avoid an infinite loop, a task can be resubmitted only a limited number of times.

Features

- **JST acts on top of the Grid middleware so that users are not required a deep knowledge of the grid technicalities:**
  - It actually submits jobs through WMS, retrieves the jobs outputs and monitors their status
  - Every action is logged in the central DB
- **When the jobs reach the WN they just request to the TaskListDB if there is any task to execute (pull mode). If no, they just exit.**
- **JST tries to use all the computing resources available on the grid (no a priori black or white site lists are necessary). If the environment/configuration found on the WN is not adequate, the job exits.**
- **Since the tasks are independent and they can be resubmitted if needed, a fairly good reliability can be reached and JST can work successfully even if some failure occurs on Grid services**
  - More than one RB is used for jobs submission
  - More than one SE used for the stage-out phase
- **It is possible to establish tasks dependencies:**
  - A task B could start only if the task A has been completed correctly
  - With this technique it is possible to build complex workflow
- **It is possible to change the priority of the tasks in the course of a challenge:**
  - Priority is used to select the task that has to be executed first.
- **A mail is sent to the user when the challenge has been completed:**
  - A web link is provided to retrieve the output

Development

- **So far, application challenge were performed by the JST developers team:**
  - The users had just to provide input files, the executable, the libraries needed …
- **With the experience acquired executing with success several challenge on bioinformatics applications, the JST developers have built a high level service that allows users to exploit the Grid on their own**
- **A web interface has been deployed to provide an easy to use tool for users that do not have a deep knowledge about Grid technology**
  - In order to provide a general and standard tool, the service is based on XML and XSLT transformation
- **The interface is based on few web pages that:**
  - Guide the user creating the tasks
  - Trigger submission of the application to the Grid
    - The UI daemon checks periodically the TaskListDB
  - Monitor the status of the challenge
    - The status of the tasks
    - The number of failures
    - The kind of failures

Gene Analogous Finder (GAF) use case

- GAF performs genomic comparison to find genes with analogous functions
- The algorithm compares the gene descriptions provided by Gene Ontology (GO)
- The comparison is “all against all”
- Each elementary task takes care of the comparison of a limited number of genes descriptions against all the others

Tasks creation

The user decides how to split the input file

Genes descriptions ids

1
2
3
4
.
.
.
100027
100028
100029
100030

Single starting input

1
2
.
.
.
9
10

11
12
.
.
.
19
20

...

...

100021
100022
.
.
.
100029
100030

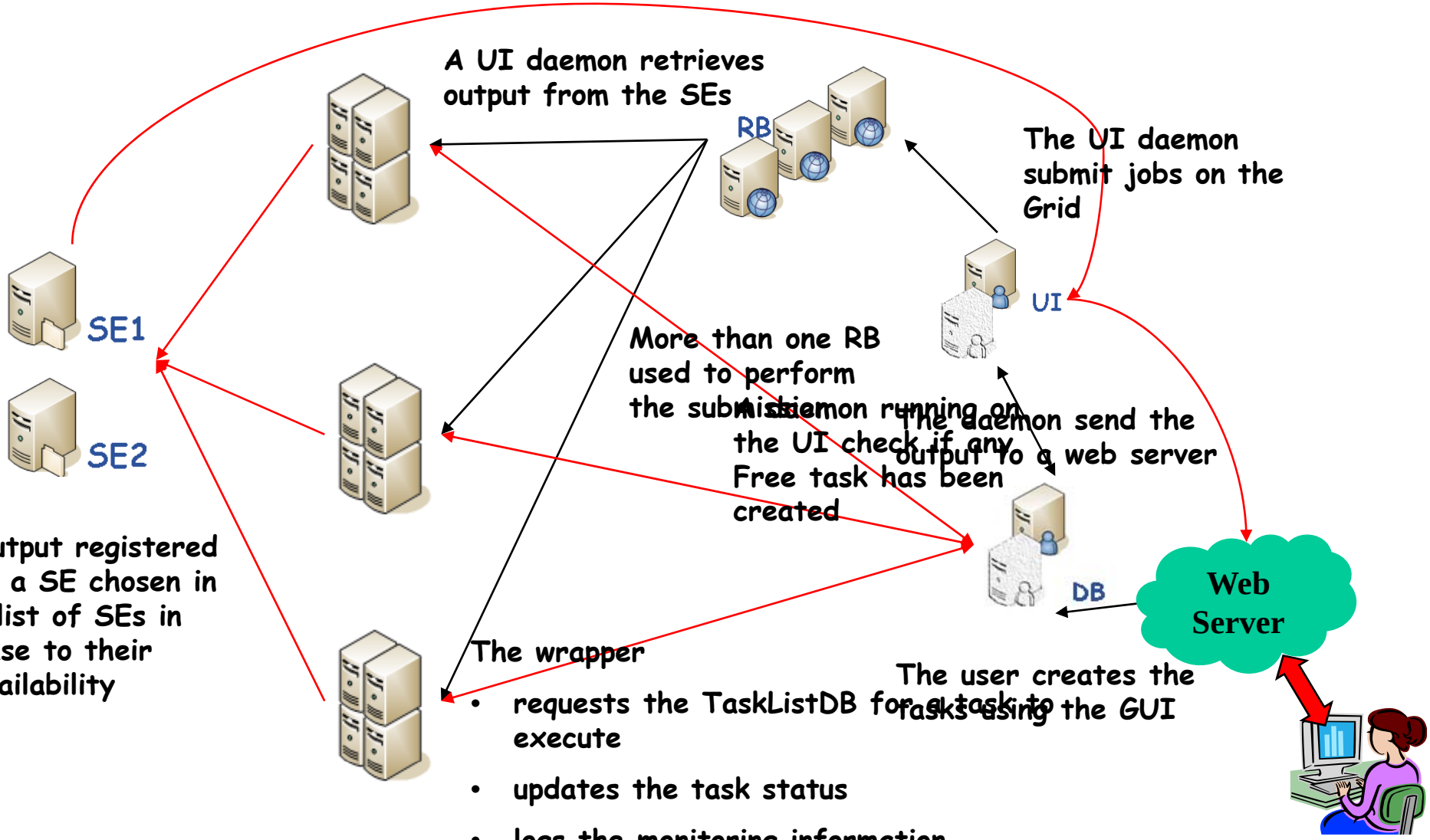
Every input file generates a task!!!

What the user needs

- **Using the GUI the user has mainly to take care about input files**
  - The user has to choose the right number of elementary tasks and in turn the right number of the partial input files in the input directory

- **The way the user decides to subdivide the original input file to create the elementary tasks is fundamental for the entire challenge:**
  - Too small slices can produce a huge number of jobs submitted. Even if the parallelization is increased there could be a low efficiency due to the big overheads introduced by the grid submission (WMS latency and the queuing time in the batch queue) and by the output retrieval phase.
  - Too big slices can decrease the number of jobs submitted but could produce long jobs that can fail for any kind of reasons and reduce the advantage of parallelization

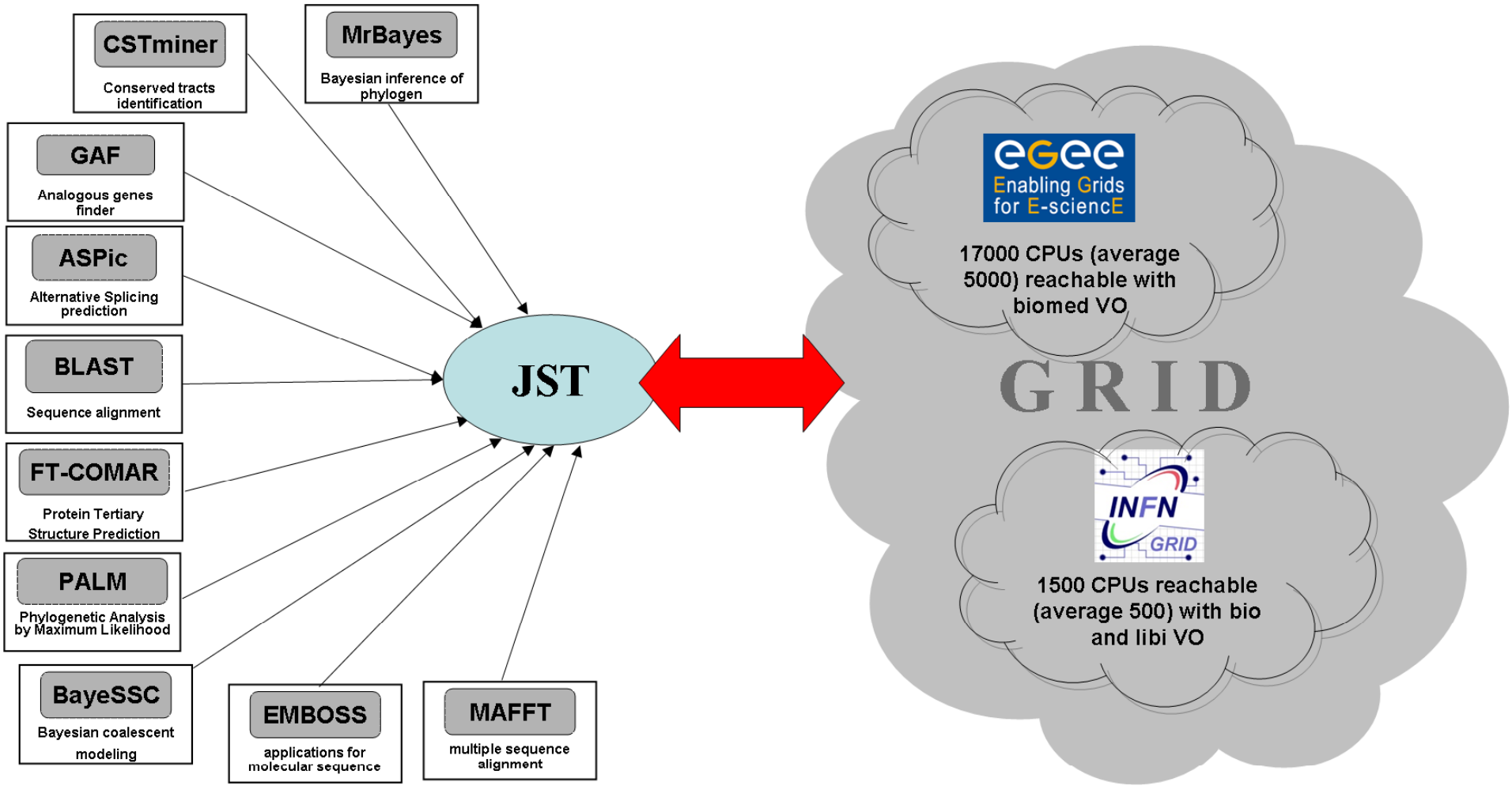
Workflow


A UI daemon retrieves output from the SEs
RB
The UI daemon submit jobs on the Grid
SE1
SE2
UI
More than one RB used to perform the submission
A daemon running on the UI check if any Free task has been created
The daemon send the output to a web server
Output registered on a SE chosen in a list of SEs in base to their availability
DB
Web Server
The wrapper
• requests the TaskListDB for a task to execute
• updates the task status
• logs the monitoring information
The user creates the tasks using the GUI

The wrapper

- Requests from the TaskListDB the tasks to be executed
- Retrieves the application executable (it has to be available in some way: https, http, globus-url-copy …)
- Executes the application code
- Stores the output in one of the configured SEs
- Checks the exit status of the executable and of the stage-out procedure
- Updates the task status into TaskListDB

The executed challenges on EGEE and INFNGrid



Future

- **JST services can be used from every high level services portal to improve the process of integration of new applications to the Grid:**
  - Since JST is responsible of the grid Job submission, the user does not need to be a grid expert
- **JST is under test and evaluation in GRB, the official LIBI project portal (https://sara.unile.it/cgi-bin/libi/enter)**
- **The GENIUS portal (https://genius.ct.infn.it/) already uses JST for MrBayes submission on the Grid**
- **The developers are working to make available new applications in the web interface.**

Conclusions

- **It is very easy to port new applications on the grid using JST, as matter of fact, we have already used it to run on the grid several bioinformatics applications: MrBayes, Aspic, CSTminer, Blast, PAML, Muscle, Biopython, FT Comar …**
- **We used this tool in order to run several challenges with the mentioned applications.**
- **Roughly we have executed jobs for more than 300 CPU/years**
- **Not only bioinformatics applications could be run on the Grid using JST**
  - If your application could be split in single independent tasks it perfectly fits JST features

Link & references

- **JST description:**
  - http://webcms.ba.infn.it/cms-software/index.html/index.php/Main/JobSubmissionTool

- **The web interface (registration needed):**
  - http://webcms.ba.infn.it/~pierro/JST/index.php

guido.cuscela@ba.infn.it
giacinto.donvito@ba.infn.it