A GRID PLATFORM FOR ITALIAN BIOINFORMATICS

G. Donvito
(INFN-BARI)

EGEE'06 Conference (Geneva)
25-29 September 2006

www.eu-egee.org

Information Society and Media

Outline

- **LIBI project**
- **Bioinformatics application deployed**
  - GENE FINDER
  - CSTminer
  - Massively evolutionary task by blast
  - DNAfan
- **Generic “task queue” and running procedure**

LIBI Project

LIBI (International Laboratory for Bioinformatics) is a Italian project, started from 12 September 2005, supported by the Italian Minister for Research, which collects leading Italian institutes in bioinformatics working together with technological partners with the aim to built a virtual Laboratory.

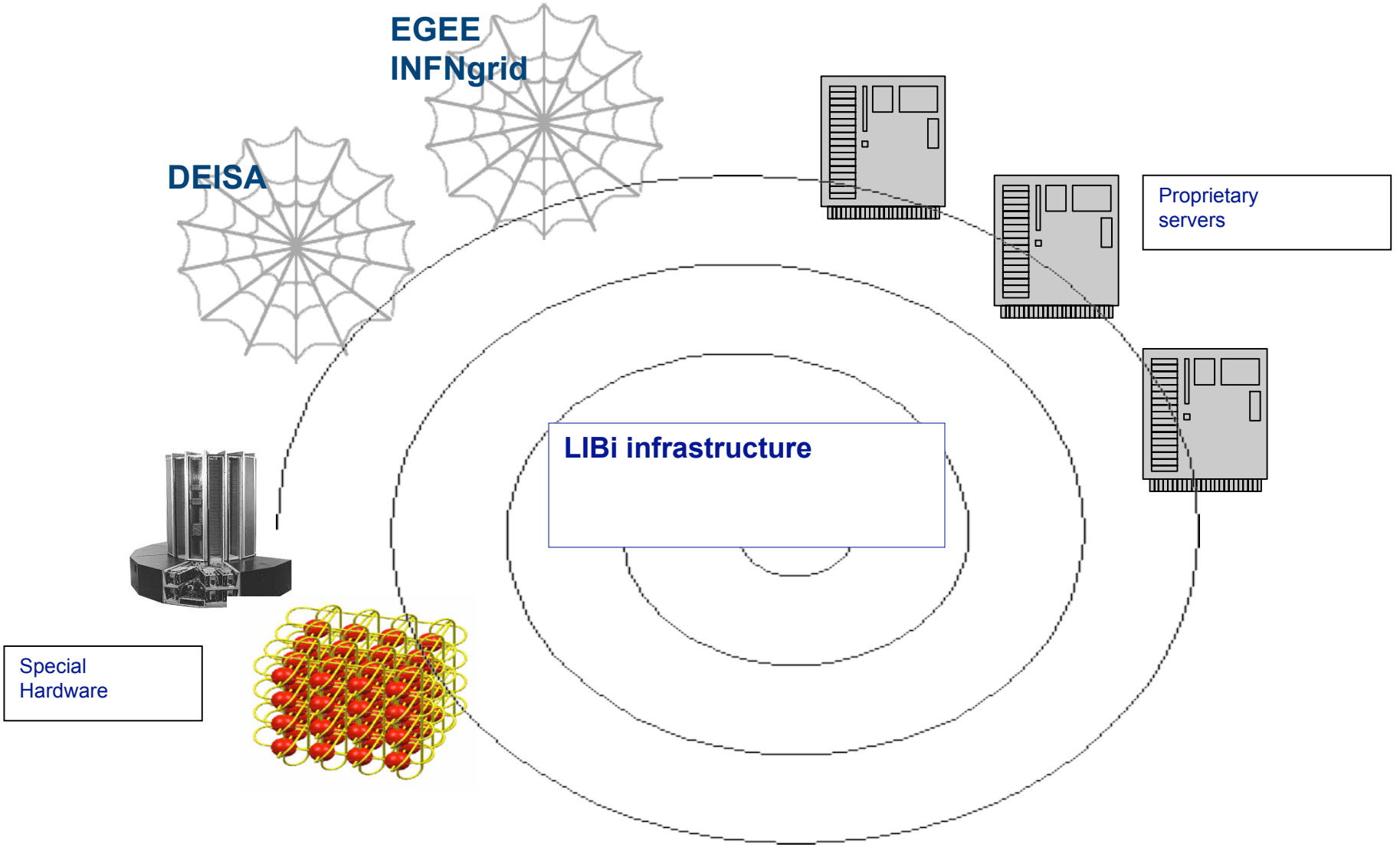
Technological Partners:

CINECA, INFN, SPACI/CACT-ISUFI, IBM

Scientific Partners:

CNRBA, UNIBO, UNIMI, CBMTS

Idea of the project



Organization of LIBI project

- **The goal of the project is to deploy in the virtual laboratory some specific application;**
- **we are working on these applications:**
  - Mr.Bayes: analysis of sequence alignment
  - DNAfan: tools for analysis and extraction of fragments of sequences from large set of sequences
  - PSIBLAST
  - Molecular dinamics software
  - CSTminer
- **A specific technological group deals about technological needs of the laboratory; mainly in the data management aspects of bioinformatics applications**

Bioinformatic Application Deployed

- **GENE FINDER (DONE)**
  - § **Goal**: compare gene products according to their described function instead of by the more conventional sequence comparison.
  - § **Data source**: Gene Ontology (GO). Is international standard for gene annotation. A GO term is associated to a gene after experimental observation, sequence similarity, etc.
  - § **Challenge dimension**:
    - § 1 million gene products against 1 million.
    - § More than 55 CPU years needed.
    - § More than 95000 job submitted.
    - § 64 different farms used.
    - § More then 2400 different hosts used.
    - § 1 month of run on EGEE infrastructure

Bioinformatic Application Deployed (2)

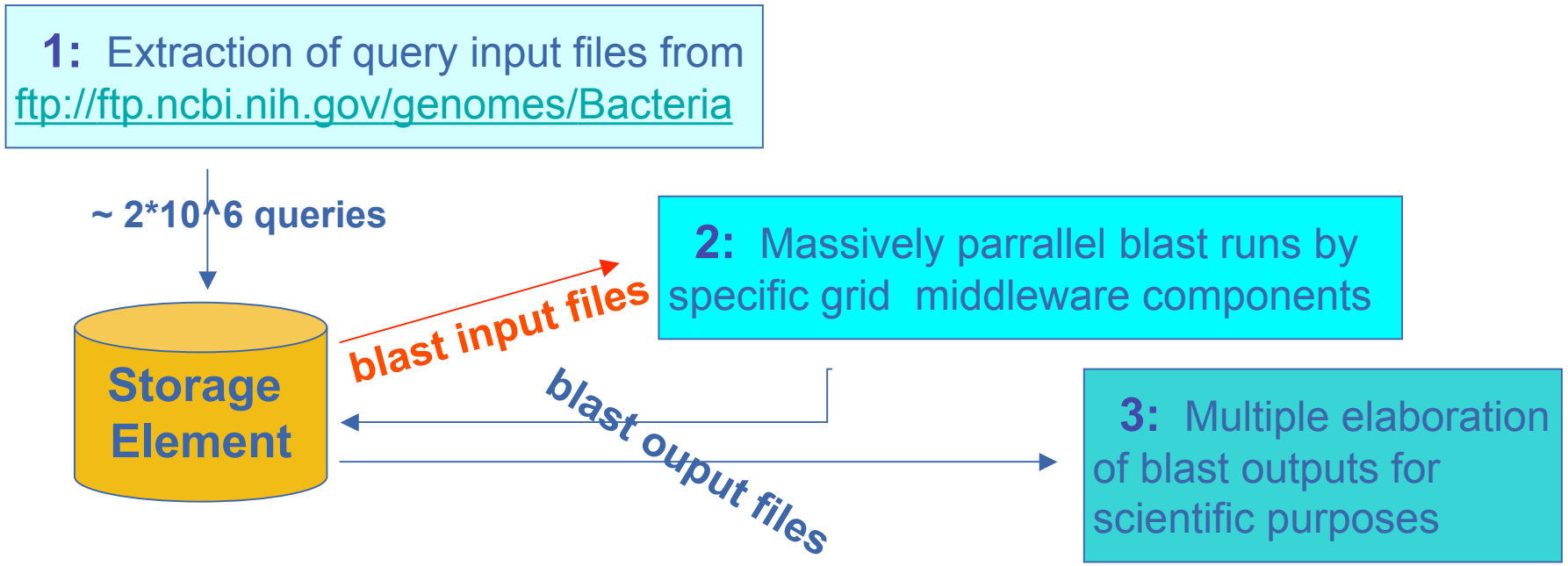
- **CSTminer (finishing)**
  - § **Goal**: compare the entire genome of the Human being against the entire genome of some animals (mouse, dog… ecc)
  - § **First test**: Human against mouse
  - § **Challenge dimension**:
    - § 850 million of comparison (~ 2 sec of CPU for each comparison)
    - § More than 50 CPU years needed.
    - § More than 65000 job submitted.
    - § Up to 2 million of comparison per hour.
    - § 22 different farms used.
    - § More then 900 different hosts used.
    - § 2 month of run on INFN-Grid infrastructure

Massively evolutionary task by blast on grid

authors: http://www.biocomp.unibo.it & LIBI grid managers

- Genomes of 397 bacteria are compared by blast to find the original sequences in the evolution process
- Each genome corresponds to ~ 5000 sequences: the grid must support ~ 5000*397=1.985.000 blast commands
- One PC ~ 1 year running, on grid < 2 weeks

Task management on grid from an User Interface



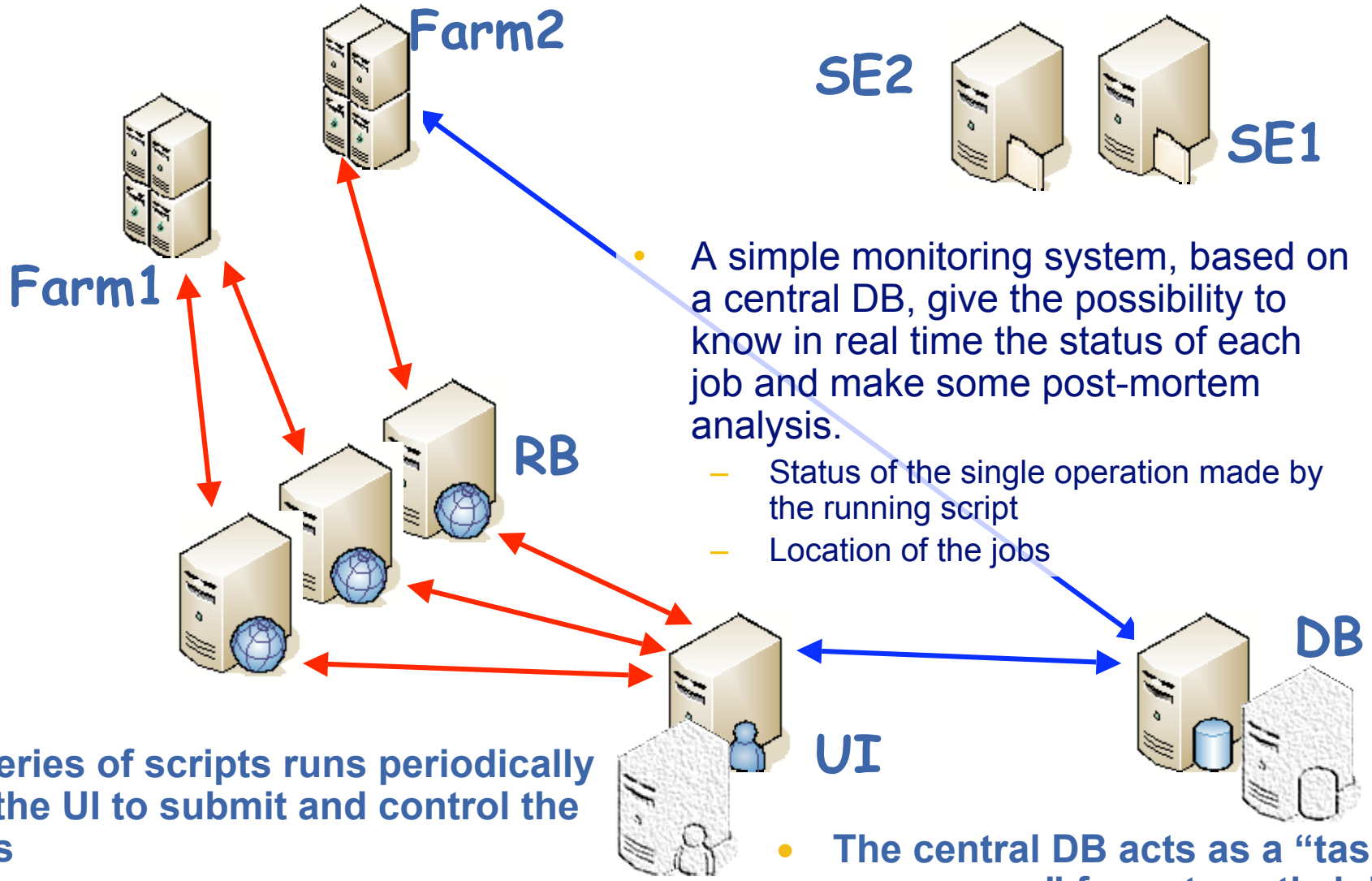
Issue

How to manage the run of this huge amount of "jobs"?

- It is necessary to keep trace of the tasks executed, failed or running
  - The failed task must be automatically re-submitted
- It is needed a "application level" control about the "status" of a task
- It necessary to run it in a unattended way
- It must be avoided to have a "single-point" of failure that can stop the running

We have developed a general purpose task queue to address these issues

JOB SUBMISSION SCHEMA



- A simple monitoring system, based on a central DB, give the possibility to know in real time the status of each job and make some post-mortem analysis.
  - Status of the single operation made by the running script
  - Location of the jobs

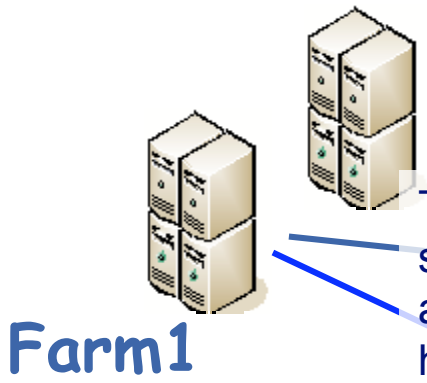
- **A series of scripts runs periodically on the UI to submit and control the jobs**

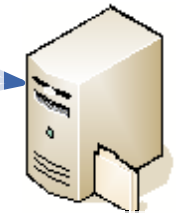
- **The central DB acts as a "task queue" for automatic job submission**

Generic “task-queue”

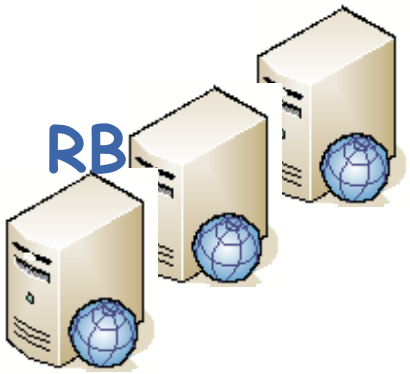
- **It is realized with a MySQL server (version 5.0.x)**
- **It can provide some advanced information about tasks, such as:**
  - Number of failed execution for each task
    - § If there are too many failure on the same task you can avoid to re-run it
  - Job provenance (what job had executed each task)
  - The exact date of the task execution
- **It can manage multiple task dependency:**
  - a task can be run only if all the task from which it depends are done successfully
- **The priority of the task execution can be dynamically changed**
- **The scalability can be increased as needed**
- **The single point of failure can be easily avoided**

Actions performed when the job reaches the WN

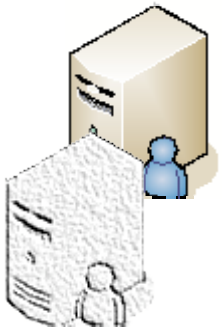


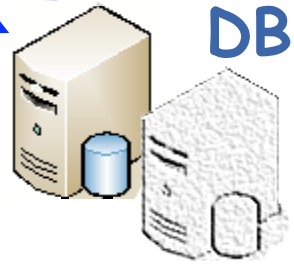


There is a quite standardized procedure to store the output, using more then one SE (with a round-robin procedure), in order to avoid high load on the SE and to guarantee fail-over procedure



When the actual execution is finished the “executable” advice the wrapper about its EXIT_STATUS (i.e. depending on the result of the output stage); then the wrapper can update the status of the task accordingly





A wrapper script updates the monitoring DB with the status of the job
Reads from a DB the first n tasks to be run (chosen between the not completed tasks or the running ones from more then 48 hours) and then starts the real comparison

Generic “task-queue”: result

- **Both gene-finder and CSTminer are executed using this submission system with good success and with a little “human effort”.**

BACKUP SLIDES

Submission procedure

- **Based on several scripts which runs automatically at fixed time interval.**
  - The job submission is made by some script running as a daemon
    - § There is the possibility to run more instances of the submission daemon in order to increase the total number of job submitted in one hour
    - § The multi-process submission improve the speed of submission
  - The submission uses 5 RB in a round robin algorithm in order to avoid the over-load of a single RB and to avoid that the failure of a single RB can stop the submission of jobs
- **A different script retrieves periodically the OutputSandbox of the jobs**
- **Further simple interactive scripts are provided to monitor the status of the production by simply querying the monitoring DB**
  - The user can know the number of processed/running genes
  - The number of the running jobs
  - The location of each job
  - Debug eventual errors in running jobs
- **The software to submit jobs is installed on 2 different machines in order to avoid that a single hardware failure can stop the submission**