Functional Analogous GENE FINDER OVER THE GRID

G.Donvito
INFN Bari

In collaboration with

Enabling Grids for E-science in Europe

www.eu-egee.org

http://grid-it.cnaf.infn.it/

EGEE is a project funded by the European Union under contract IST-2003-508833

Outlook

- Objectives
- Application Schema
- Running over the GRID
- Solution
- Optimization
- Result
- Conclusions

Functional Analogous Finder

Goal: compare gene products according to their described function instead of by the more conventional sequence comparison.

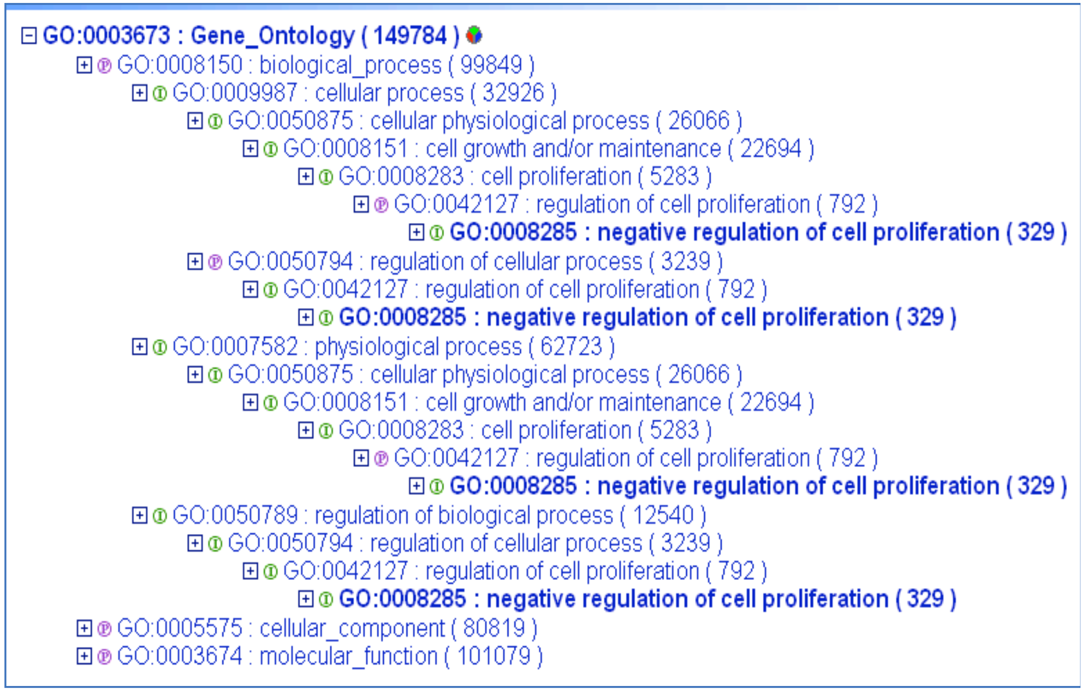
Data source:

Gene Ontology (GO)

Is international standard for gene annotation

A GO term is associated to a gene after experimental observation, sequence similarity, etc.

Is 18800 GO-terms, ~ 1.7M gene products, 7.1M associations



- ⊟ **GO:0003673 : Gene_Ontology (149784)**
  - ⊞ GO:0008150 : biological_process (99849)
    - ⊞ GO:0009987 : cellular process (32926)
      - ⊞ GO:0050875 : cellular physiological process (26066)
        - ⊞ GO:0008151 : cell growth and/or maintenance (22694)
          - ⊞ GO:0008283 : cell proliferation (5283)
            - ⊞ GO:0042127 : regulation of cell proliferation (792)
              - ⊞ **GO:0008285 : negative regulation of cell proliferation (329)**
      - ⊞ GO:0050794 : regulation of cellular process (3239)
        - ⊞ GO:0042127 : regulation of cell proliferation (792)
          - ⊞ **GO:0008285 : negative regulation of cell proliferation (329)**
    - ⊞ GO:0007582 : physiological process (62723)
      - ⊞ GO:0050875 : cellular physiological process (26066)
        - ⊞ GO:0008151 : cell growth and/or maintenance (22694)
          - ⊞ GO:0008283 : cell proliferation (5283)
            - ⊞ GO:0042127 : regulation of cell proliferation (792)
              - ⊞ **GO:0008285 : negative regulation of cell proliferation (329)**
    - ⊞ GO:0050789 : regulation of biological process (12540)
      - ⊞ GO:0050794 : regulation of cellular process (3239)
        - ⊞ GO:0042127 : regulation of cell proliferation (792)
          - ⊞ **GO:0008285 : negative regulation of cell proliferation (329)**
  - ⊞ GO:0005575 : cellular_component (80819)
  - ⊞ GO:0003674 : molecular_function (101079)

go terms associated to the BCL2_Human gene

GO association are contained in a relational DB (more then 1.7 million of classified genes)

The algorithm

- Only well described gene products are considered (>15 go terms) (≈1 million gene products)

- Search algorithm
  - Compare gene A with gene B with a chi-square test by considering the GO terms common to the two genes and the ones not in common.

	Terms present in A	Terms non present in A
Terms present in B	O_{11}	O_{12}
Terms non present in B	O_{21}	O_{22}

  - The GO terms are weighted with the 1-pvalue to give more importance to the more specific terms.

- Processing: one gene against all others is 15-45 CPU minutes
  - **495000 hours of CPU needed (~55 years)**

- Output: text file with 100 best hits (few kBytes) is compiled as an additional DB associated to the GO DB

Running over the GRID: issues

- DB access:
  - The query, to search for the GO terms associated to “well described” genes, is time-expensive
  - The query is always the same and produces a large amount of data (~ 500 MB)
- A single o few DB server is not the correct way to take advantage of thousands CPU’s available on the grid
  - Do the query only ones and store the result in the WN local memory
  - Reuse the data for more than one gene.
- It is necessary to keep trace of the genes executed, failed or running
- It necessary to run it in a unattended way

Functional Analogous Finder: the grid implementation

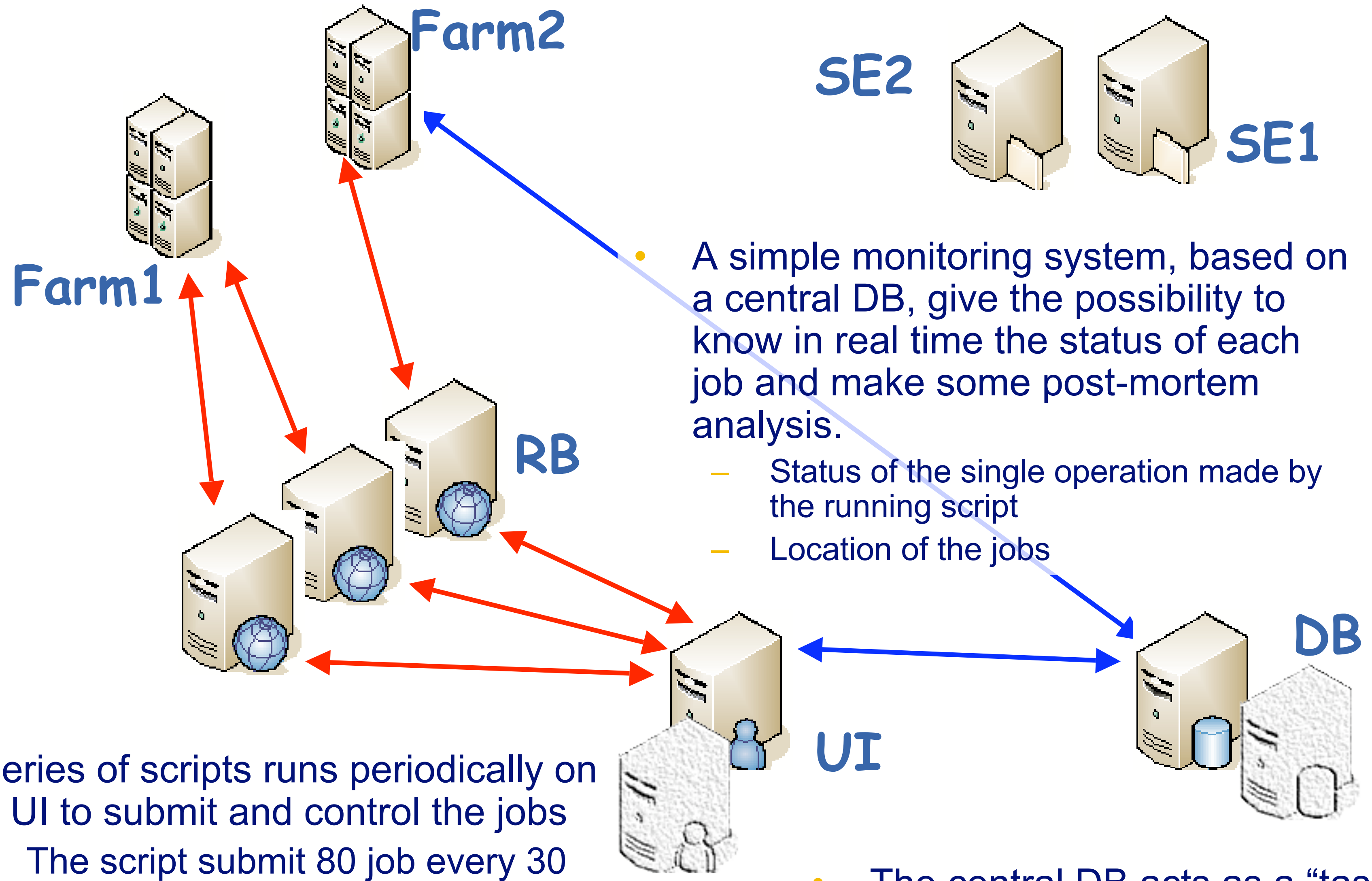
- The gene comparison is performed by a perl script which uses statistic libraries
  - Need to install the perl libraries on each WN (just before running).
  - The libraries are “relocated” to avoid the need for “root” privileges
- The GO terms associated to each “well described” gene are extracted, once for all, from the GO DB and stored in a text file (500 MB)
  - The text file is transferred from one of the available SE’s to the WN in a fail-over schema.
- The list of gene products to examine is stored into a central MySQL DB.
  - The DB keeps track of the completed gene products, the failed and the running ones.
  - The DB acts as “task queue” for automatic job (re)submission.

Functional Analogous Finder: the job submission

- Based on several scripts which runs automatically at fixed time interval.
  - The job submission is made by a script running as a daemon
    - The script submits 80 job every 30 minutes
    - 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 3 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

BioinfoGRID

Functional Analogous Finder: the job submission



- 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 script submit 80 job every 30 minutes

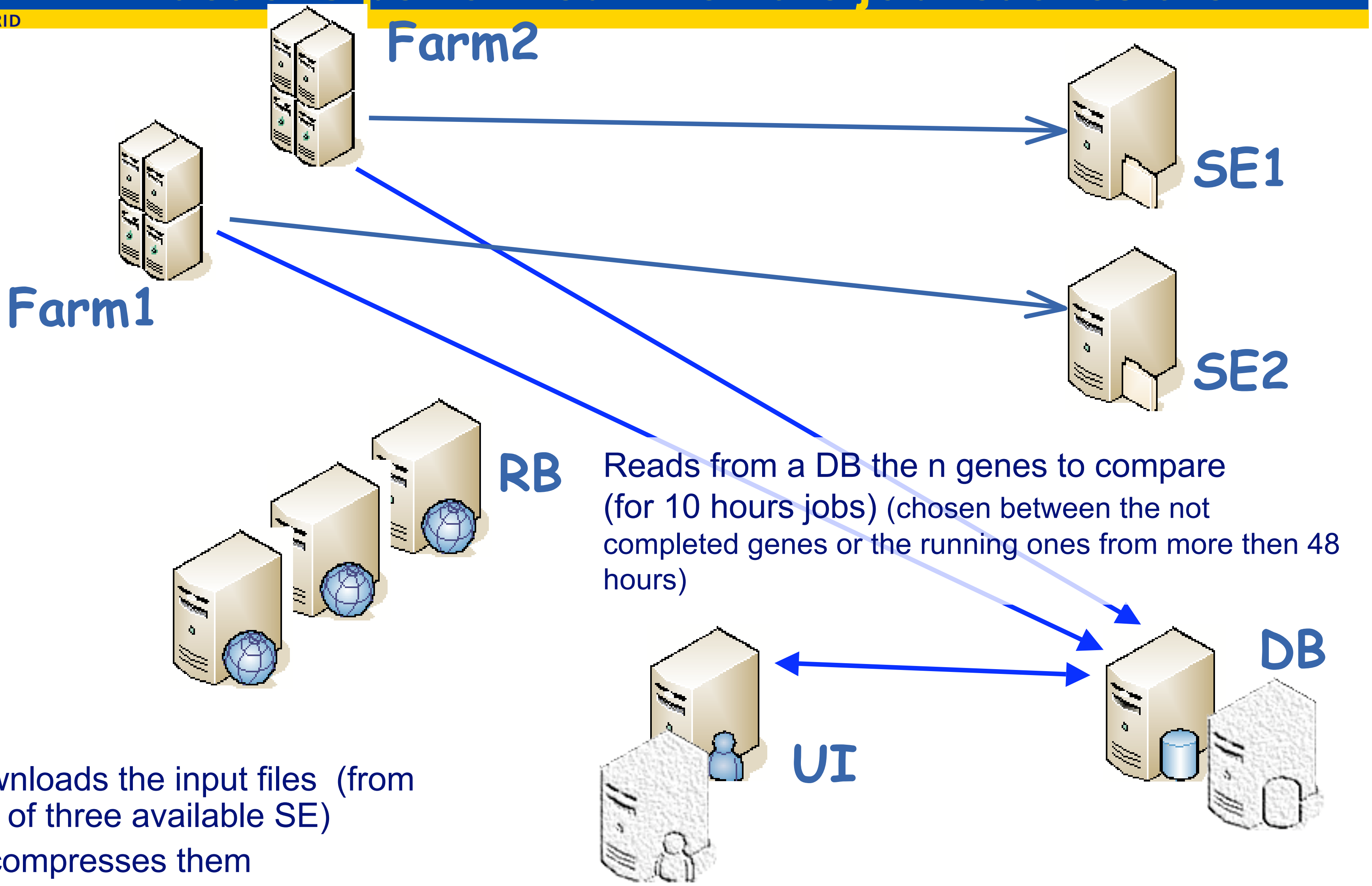
- The central DB acts as a “task queue” for automatic job submission

Functional Analogous Finder: actions performed when the job reaches the WN

- When a job arrives on a WN:
  - Downloads the input files (from one of three available SE)
  - Decompress them
  - Installs perl lib
  - Downloads from a DB the first “n” genes choosing them from the “free” genes or from “running” ones that are in “running” state from more than 48 hours
    - § The number of the genes the had to be executed on a WN depends on the CPU power
  - Starts the perl script
    - § In the perl script there is a loop to calculate all the selected genes without re-reading the input files
    - § When a gene is completed and the output files is written the file is copied to one of the two SE available and the gene is put in status “done” on the DB

This behavior give the possibility to submit always the same job without worry about failed jobs

Functional Analogous Finder: actions performed when the job reaches the WN

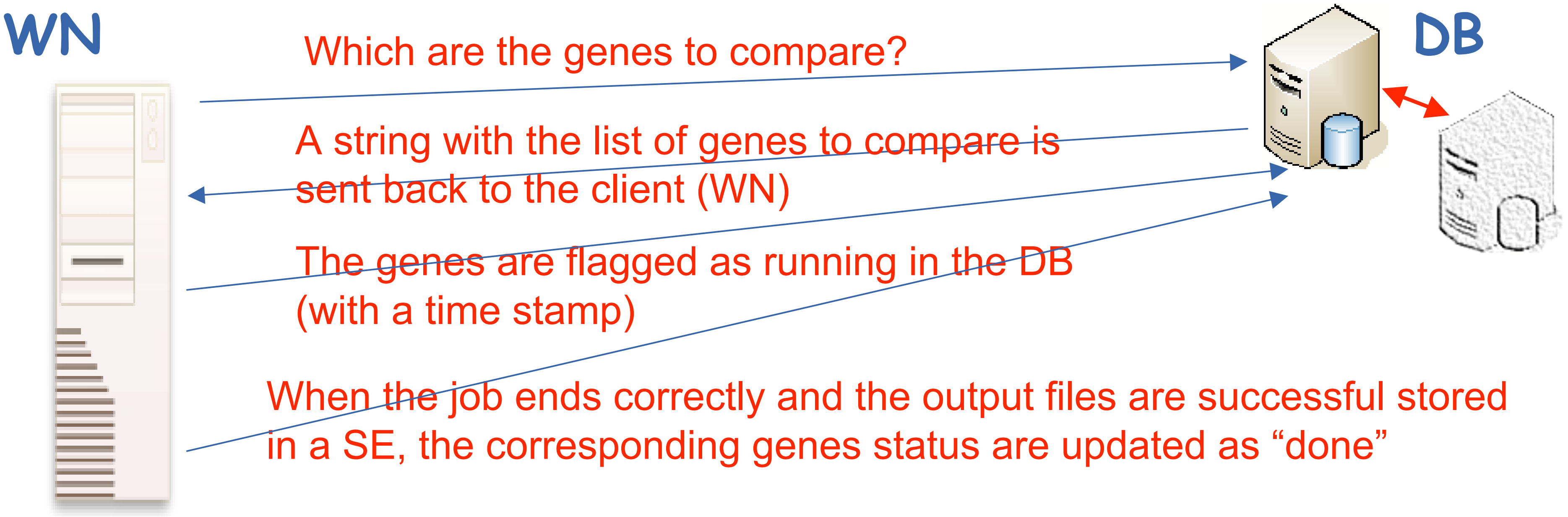


Reads from a DB the n genes to compare (for 10 hours jobs) (chosen between the not completed genes or the running ones from more then 48 hours)

- Downloads the input files (from one of three available SE)
- Decompresses them
- Installs the perl libraries

- Start the perl script and the comparison

The TaskQueue



Which are the genes to compare?

A string with the list of genes to compare is sent back to the client (WN)

The genes are flagged as running in the DB (with a time stamp)

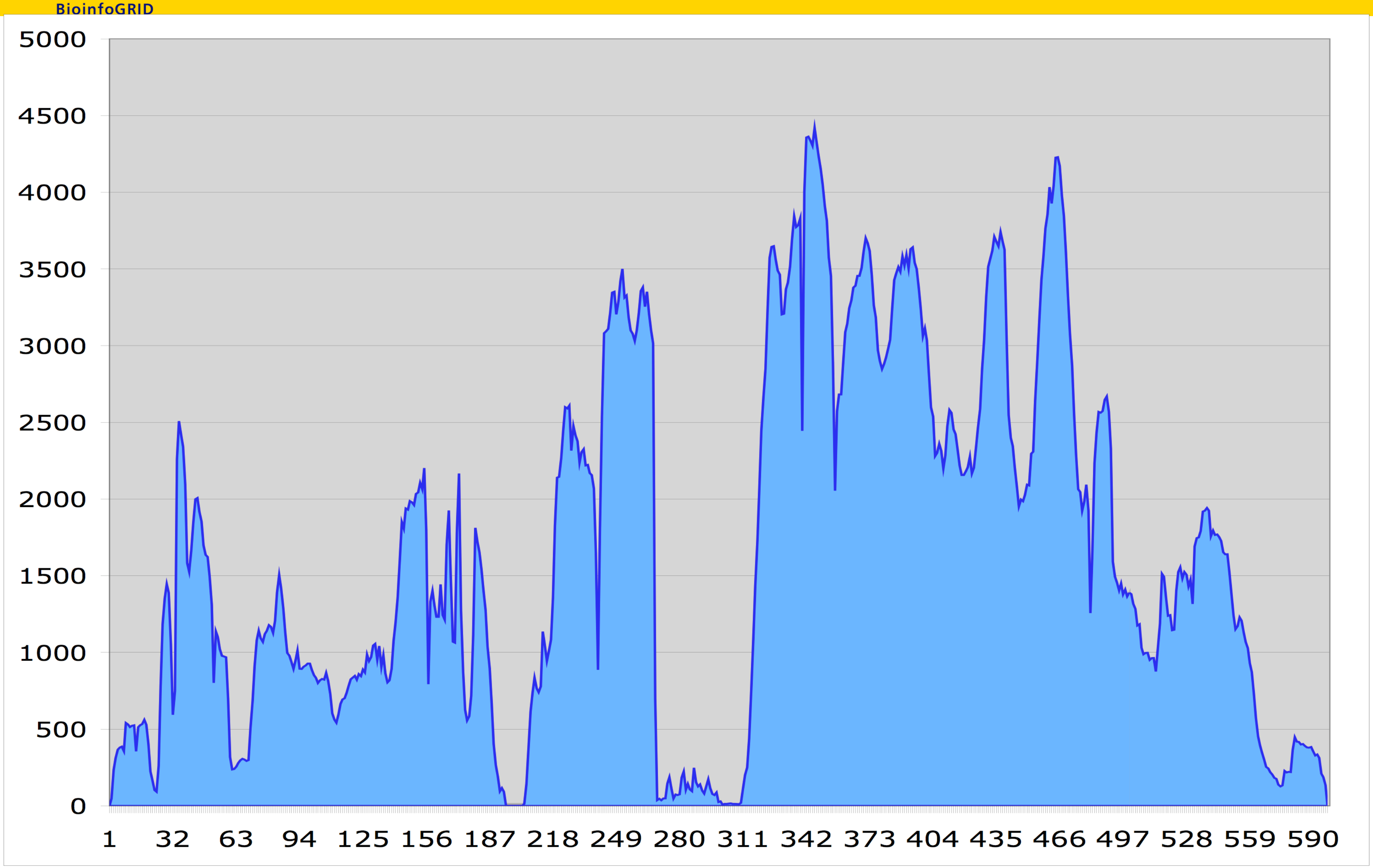
When the job ends correctly and the output files are successful stored in a SE, the corresponding genes status are updated as “done”

- A new gene is chosen from the genes in status “free” or from genes in status “running” from more than a configurable time
- The description can be everything (also a real bash script, in order to run different application with the same wrapper script)
- The genes can be ordered with an arbitrary priority also at run-time
- The central database can be easily backed-up in order to guarantee continuity of service

Functional Analogous Finder: Result (1)

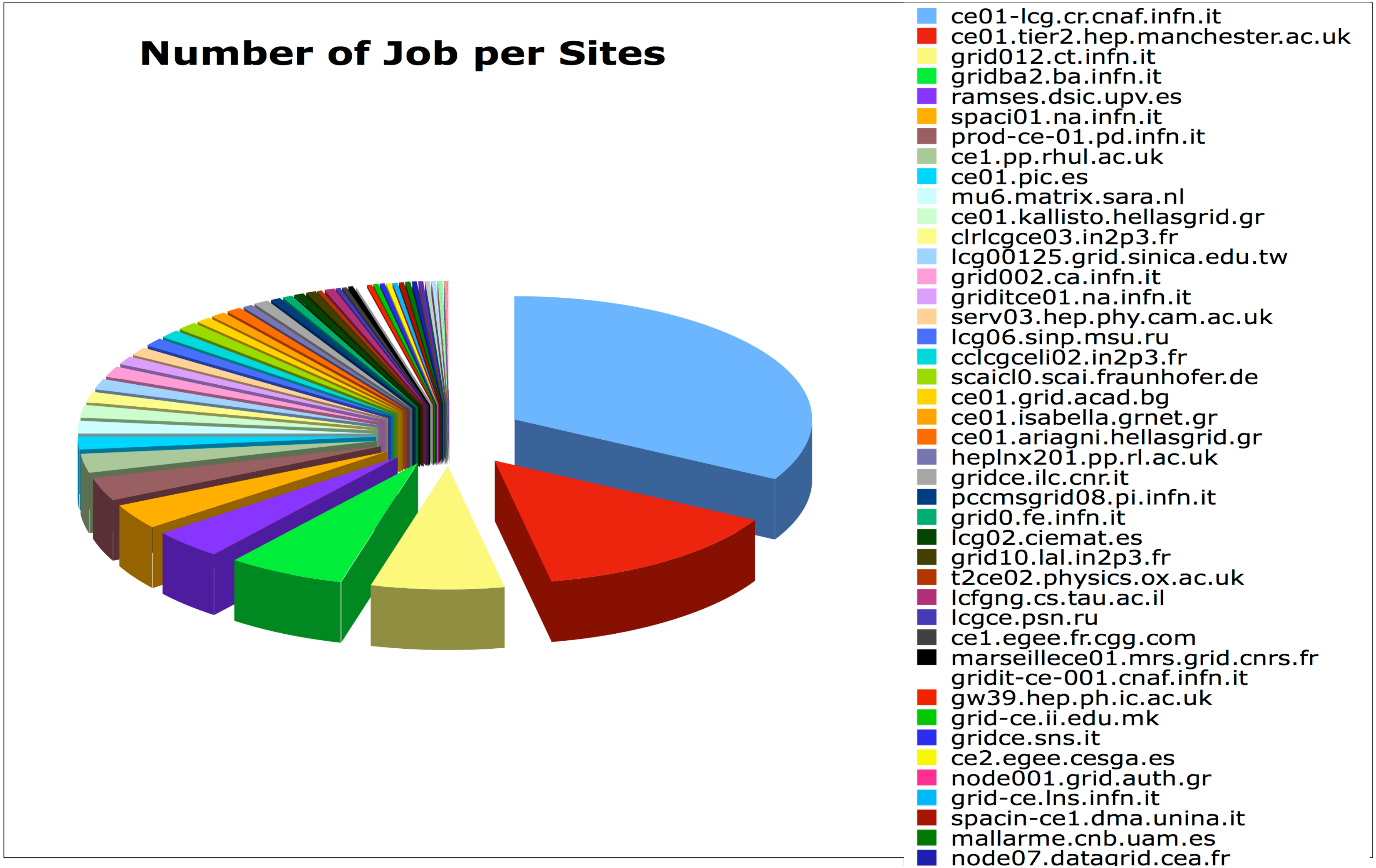
- First test run (80.000 gene products) done over last Christmas
- After optimization of the script and of the procedure
  - a new test run was started
  - It was run using the VO “bio” on the Italian INFN-GRID infrastructure
  - And “biomed” on whole EGEE
- We have started submission at: 27 February 2006
- All genes processed in less than one month
- More data:
  - **Different Farm used: 64**
  - **Different Host used: 2446**
  - Total Submitted jobs: 95041
  - Total started jobs: 66313
  - Completely successful job (from application point of view): 42992
  - Total Failed job (for input staging problem): 3209

Functional Analogous Finder: Result (2)
genes analyzed per hour



BioinfoGRID

Functional Analogous Finder: Result (3)



Conclusion

- Good efficiency
- Easy to manage
- A quite general procedure of submission of jobs
  - A general procedure to take trace of to-do/done task
- Grid submission success rate is not so important
- The procedure is thought in order to run for a long period in an unattended way
- The rate of analyzing genes grows linearly with the number of nodes that are available
  - We don’t see any bottle-neck at this level (the task queue does scale on the thousand concurrent jobs)
- It is important to avoid black-holes (WN or farms that make job to fails always)
  - A patch to do this is made on the running script
  - There is the possibility to exclude farm from submission modifying the jdl file
    - It is needed an automatic procedure to create a dynamic black-list of farms
- **Now this procedure can be improved using the features of the new gLite WMS**
  - **Pre and Post running script (shallow and deep re-submission)**
  - **File picking**
  - **Bulk (parametric) submission**