BioinfoGRID
Bioinformatics Grid Application for life science

Information Society and Media

Databases and Functional Genomics Applications

CNR-ITB, INFN, CNRS, CILEA

Outline

- **Biological Databases in Grid**
- Objectives
  - Provide availability of main Biological Databases in the Grid Environment
  - Optimize user wait times, and storage and transfer costs
  - Keep databases updated
  - Handle versioning
    - co-existence of multiple versions
    - handle naming, name clashes
    - store versions in a cost-efficient manner

Outline

- **Functional Analogous Finder**
- Challenge:
  - find functional analogous gene products
- No sequence comparison for this challenge:
  - Small sequence changes sometimes produce big changes in the gene function
  - Genes compared according to their functional description in GO (Gene Ontology)
    - Statistical method: searched for possible functional analogous gene products by comparing the corresponding associated GO terms and their semantic similarity measure
- Too expensive to be performed locally
  - CPU time: 55 CPU-years

Outline

- **Testing of Biological Database Application in GRID**
- Testing of prototype from D4.1:
  - Testing Availability
    - Resilience: ensure usage independent from failure of the central maintenance node
    - Ensure presence of the databases needed by the biologists
    - Ease of use
  - Testing Performance and Scalability
    - More precise evaluation of costs. Tuning parameters for the update and replication engines
    - Evaluate possible bandwidth optimizations
  - Evaluate /realize improvements

Biological Databases in Grid

- Issue: Maintenance
  - Solution: timer based scripts polling for new versions of maintained databases from FTP sites
- Issue: Scalability / replication
  - 100-1000MB - sized files.
    - High replication costs.
    - However, not possible to use CEs not near to replicas of the needed database (remote download too time consuming)
    - unknown a-priori usage of each database
    - Useless to replicate rarely-used databases

Biological Databases in Grid

- Solution and Evaluation: Adaptive replication engine
  - Database usage monitoring:
    - Moving average over N days (usually 10) for CPU-hours spent on every database
    - Configurable parameter for daily storage cost
    - Configurable parameter for cost of one minute of wait time for the user
  - Dynamic optimal replica number computation
    - math-based simulation of approximate user wait time with (N+1) and (N-1) replicas, based on the current wait time (N replicas)
    - Total (storage + user wait cost) cost computation with (N), (N+1) and (N-1) replicas. Iterative algorithm to find optimal number of r.
  - Constant adaptation of the number of replicas

Biological Databases in Grid

- Problem: versioning
  - Old versions of databases can be meaningful to biologists
    - Need to replicate computations /demonstrate previous results obtained
  - However: high storage costs.
    - Versions updated every few months.
  - In addition: risk of name clashses
    - Need to be able to specify one precise version, unambiguously
    - Co-existence of files with same name

Biological Databases in Grid

- Solution: (versioning)
  - Storage costs: patches
    - Only xdelta patches are stored, and only once (no replication)
    - One patch regresses one version
    - Patches are small, they ideally represent only the differences between two versions of a file.
    - Patches can be applied in sequence to regress to any earlier version
    - Every time a new DB version V+1 is released, the DB version V+1 is uploaded, DB version V is deleted, and a patch V+1toV is uploaded.

Biological Databases in Grid

- Solution: (versioning)
  - Name clashes:
    - Names for older versions are encoded with a specific convention
    - Python script “preparedb.py” needs to be used on the Worker Node for downloading an older version of a required database. The version is given as a date in ISO standard YYYYMMDD. Download of patches and regression to the required version is performed automatically

Biological Databases in Grid Testing

- Preliminary testing and considerations
  Size of databases
  - Max=5.3GB (nt.gz); **Avg=691MB;** $\sigma / \bar{\times} = 1.59$
  - Replication speed (bandwidth testing)
    - Avg=1.91MB/sec; $\sigma = 1.02$; $\sigma / \bar{\times} = 0.53$
  - Replication balance (hours/day n-replicas equilibrium)
    - Up to now we have tuned the parameters so that:
    - 1GB file will get 1 additional replica for each 10 CPU-hours/day of average computation (moving average over 10 days).
    - 2GB files get half that number of replicas, etc.
    - We will perform more exact cost computations/optimization

Biological Databases Use case

- Biologist needs to blast 50,000 sequences against NR version dated 2005-06-15
  - Blastp CPU time against NR: 25 seconds per sequence

    Total: 15 CPU days (Xeon class CPU)
- How is that with the Grid
  - Blast is a widely used application so we support it both through the BGBlast (BioinfoGridBlast) application.
  - In addition there is a more low-level approach which can be used for any application.
  - Through the Gilda portal

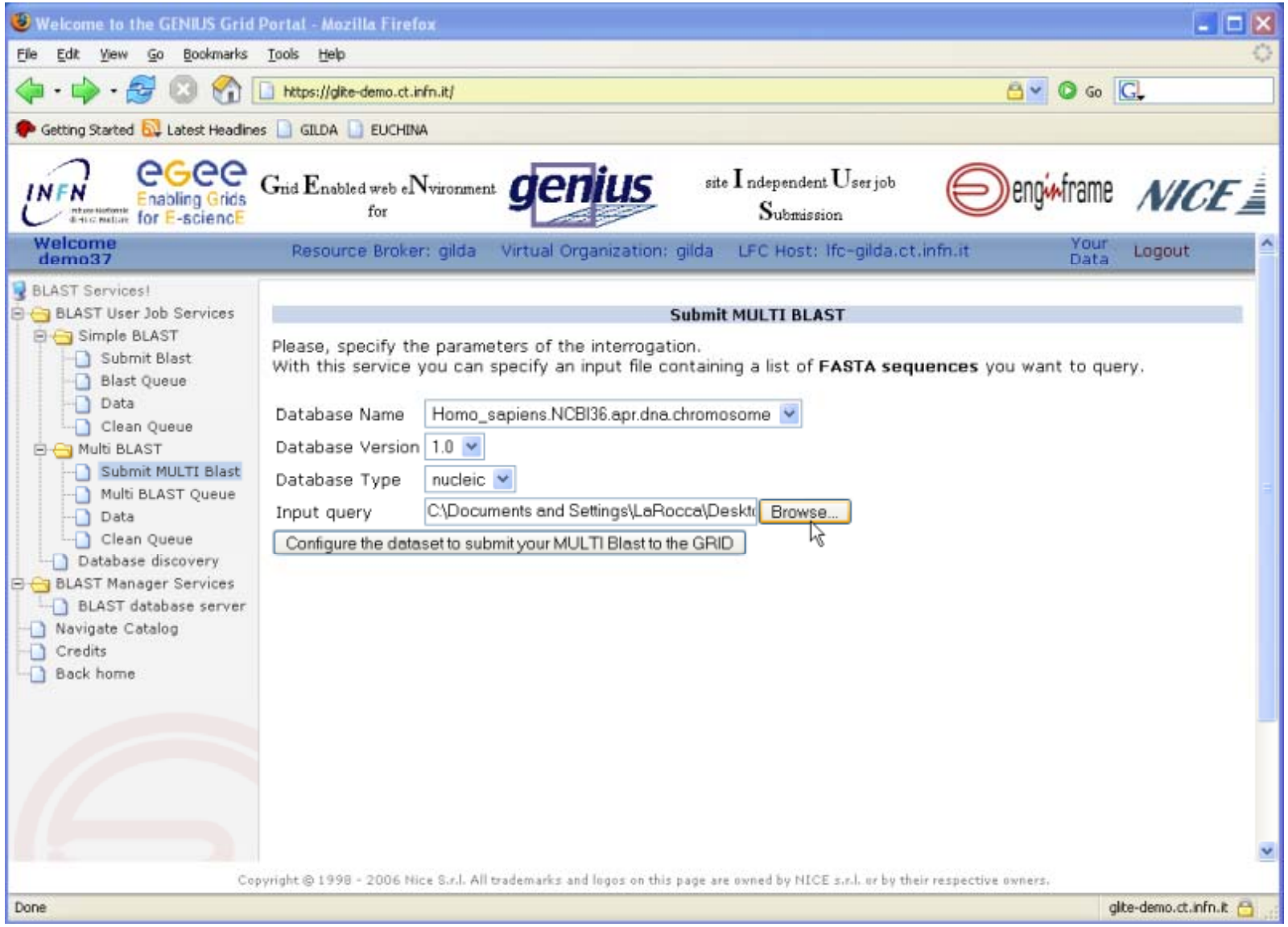
Biological Databases Use case

- Using BGBlast (BioinfoGridBlast)
  - From the commandline on an UI w/ bgblast installed
  - bgblast.pl -p blastp -i input.seq -d nr --version 2005-06-15
  - this will take care of
    - splitting the input.seq reasonably
    - launching parallel jobs
    - regressing the database on the WN to 2005-06-15
    - waiting for the results
  - Needs the commandline: a few users might find uncomfortable
- Using the Bioinfogrid portal
  - This takes care of:
    - Launching Blast (no database regression at this time)
    - Graphically monitoring your jobs' status
    - Intuitively showing the output in a graphical user interface (web)

Biological Databases Use case

- Using the low-level approach (this works with any application, not just Blast or widely known ones)
  - Inside the job, launch: ./preparedb.py nr.tgz 2005-06-15
    - This will downloads the requested (NR) database
    - Additionally, it will regress it to the (optionally) specified date by downloading and applying the needed patches
  - Submit the job to the grid. Bind execution near a replica by using: edg-job-submit --resource someCE . Place preparedb.py in your sandbox
  - Monitoring job completion and fetching results is your responsibility

Biological Databases Use case



Biological Database in GRID

- The following biological databases are currently available in Grid:
  - InterPro databases:
    - PROSITE Patterns (Hofmann, K. et al. 1999),
    - PROSITE profile (Hofmann, K. et al. 1999),
    - PRINTS (Attwood, T. K. et al. 2000),
    - Pfam (Bateman, A. et al. 2000),
    - PRODOM (Corpet, F. et al. 1999),
    - SMART (Schultz, J. et al. 2000),
    - TIGRFAMs (Haft, D.H. et al. 2001),
    - PIRSF,
    - PANTHER,
    - SUPERFAMILY

Biological Databases

- BLAST databases:
  - nr (NCBI),
  - nt (NCBI),
  - pdbaa (NCBI),
  - UCSC_human_chrs (UCSC),
  - human_genomic (NCBI),
  - refseq_protein (NCBI),
  - refseq_rna (NCBI),
  - refseq_genomic (NCBI),
  - ecoli (NCBI),
  - yeast (NCBI),
  - uniprot (UNIPROT),
  - est_human (NCBI),
  - est_mouse (NCBI)

Functional Analogous Finder

Goal

Compare gene products according to their description

AND NOT

according to their sequence similarity.

As description we use the standardised terminology of the Gene Ontology (GO).

Data source:

The Gene Ontology Database (GODB), is a repository of the GO and the associations between the terms and the gene products (GOA).

Currently there are 2M gene products described by 21000 terms producing 9M associations.

Functional Analogous Finder

Approach

- A selection of about 1M “well annotated” gene products are involved in the search.
- A simple chi-square application compares the common and non-common terms between two compared gene products.

Problem

- A comparison of one gene product against the whole 1M gene products occupies 1 CPU for 30 min. on average
- The whole “every gene product against each other” search would occupy 1 CPU for more than 50 years.

Functional Analogous Finder

Solution

- Split the search into a number of small jobs and distribute them together with the DB on as many free WN as possible.

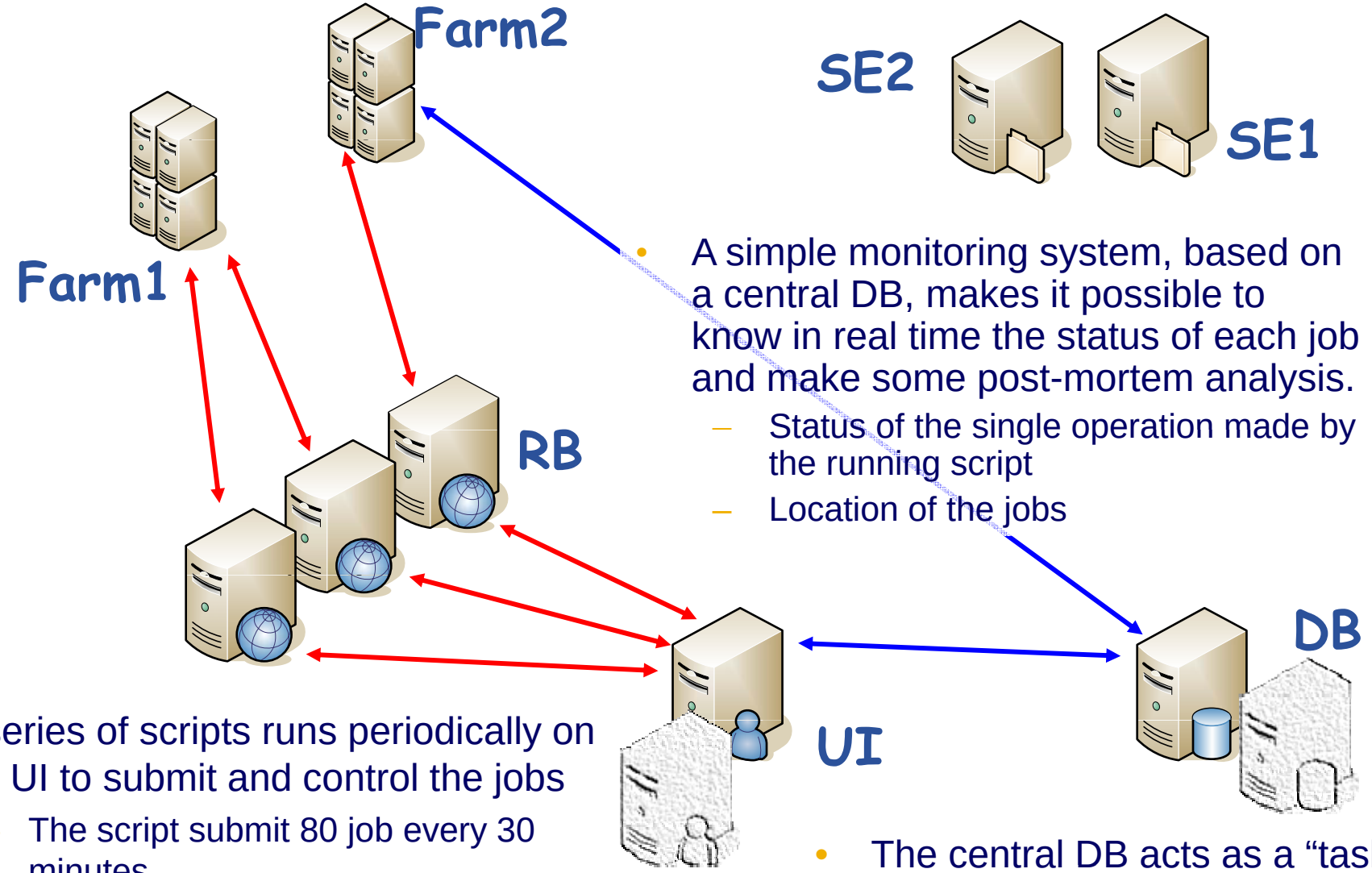
The job submission is made by a script running as a daemon

- The script submits 80 jobs every 30 minutes
- It is possible to run more instances of the submission daemon in order to increase the total number of jobs 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 overloading a single RB and to avoid that the failure of a single RB can stop the submission of jobs

- Retrieve periodically the OutputSandbox of the jobs
- 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 possible 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

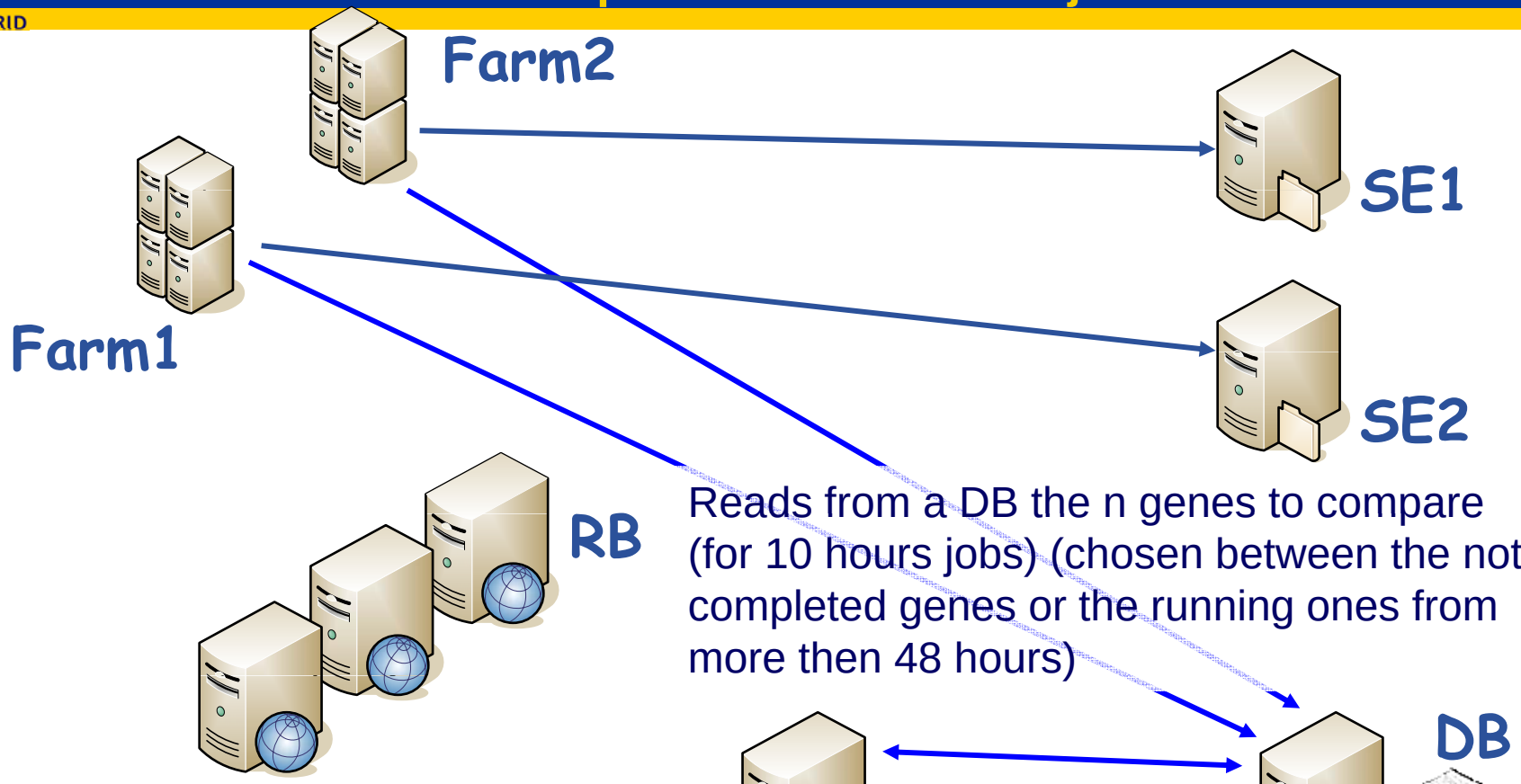
Functional Analogous Finder
the job submission



- A simple monitoring system, based on a central DB, makes it possible 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

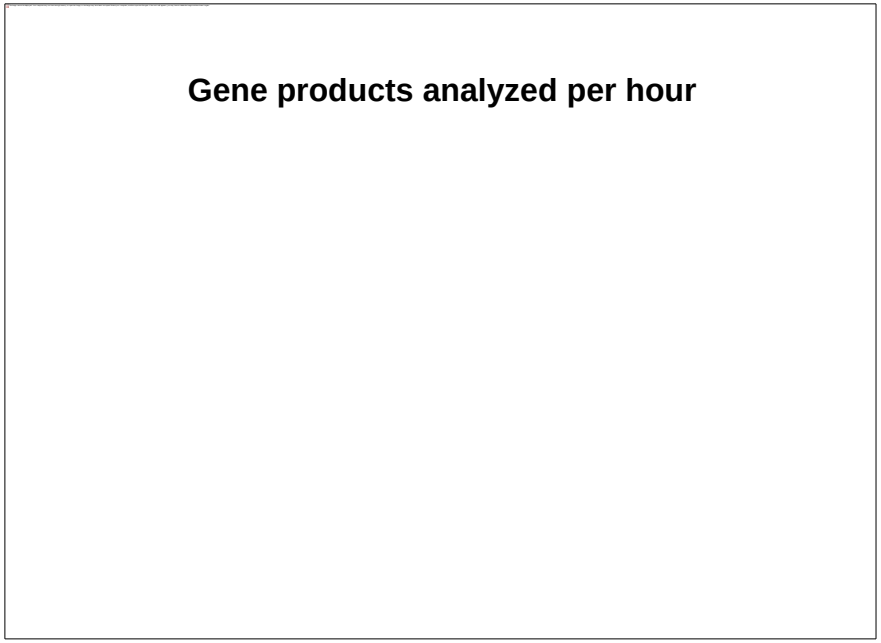


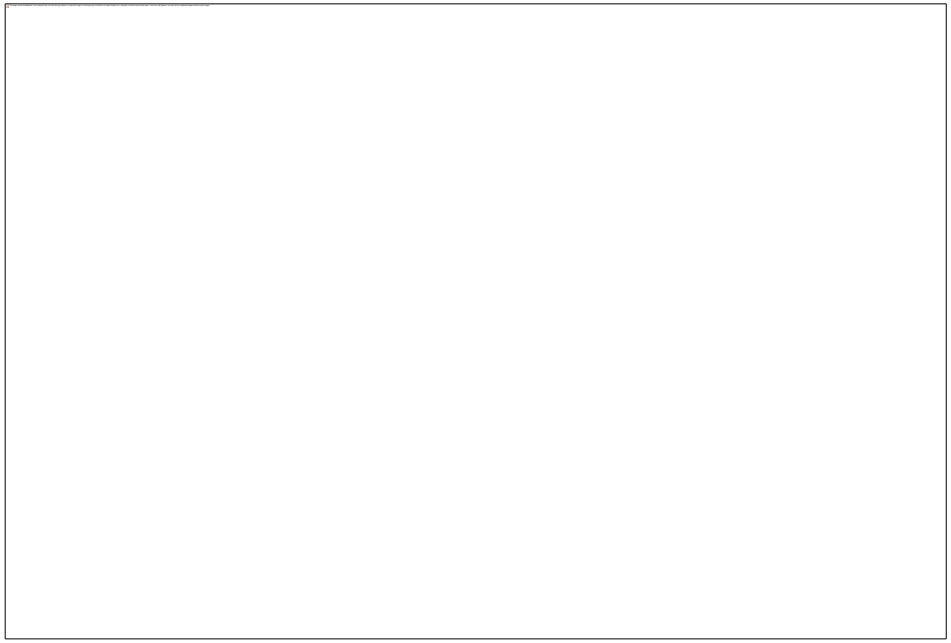
- Downloads the input files (from one of three available SE)
- Decompresses them
- Installs the perl libraries

Functional Analogous Finder

Results

- All 1M gene products processed in less than one month
- **Different Farms used: 64**
  - **Different Hosts used: 2446**
  - Total Submitted jobs: 95041
  - Total started jobs: 66313
  - Total of successful jobs (from application point of view): 42992
  - Total Failed job (for input staging problem): 3209





Functional Analogous Finder

Conclusion

- Good efficiency
- Easy to manage
- A quite general procedure for the submission of jobs
  - A general procedure to keep track of to-do/done tasks
- 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 analysis grows linearly with the number of nodes that are available
  - We don’t see any bottleneck 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 fail always)
  - A static fix for black-holes has been made on the running script
  - There is the possibility to exclude farm from submission by modifying the jdl file
    - An automatic procedure is needed to create a dynamic black-list of farms

Functional Analogous Finder

Future

- We will set up a database (best hits DB) publicly accessible to query the search results and downloading the data.
- The GODB is frequently updated and therefore the functional analogous search needs to be re-run after every update.

  The full search lasted only 30 days, however, it is unthinkable to repeat the search at every updates of the GODB.

  Therefore we are currently working on setting up a procedure that, in case of an update of the GODB, will limit the search to only those genes affected by the update, and makes the necessary adjustments in the “best hits DB” in order to reflect the update in the GODB.
- **The procedure will 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**

Papers

- Gabriele Trombetti, Alessandro Orro, Ivan Merelli, Luciano Milanesi - **“BGBlast: A Blast Grid Implementation with Database self-Updating and Adaptive Replication”** - *NETTAB2006 proceedings*

- L.Milanesi et al. -**“BioinfoGRID: Bioinformatics Grid Application for life science”** - *BITS2006 proceedings*

- Alessandro Orro, Ivan Merelli, Gabriele Trombetti, Luciano Milanesi **- “Enabling Post Processing Data Extraction in Grid Environment”** - *NETTAB2006 proceedings*

- L. Milanesi et al. -**“BioinfoGRID: Bioinformatics GRID based applications overview”** - *NETTAB2006 proceedings*

Acknowledgments

- Gabriele Trombetti,
- Alessandro Orro,
- Ivan Merelli,
- Andreas Gisel
- Giacinto Donvito
- Giorgio Maggi,
- Claudio Arlandini
- Luciano Milanesi
- Giuseppe La Rocca
- Roberto Barbera