Job Submission Tool, web interface and WebDAV data management

Giacinto DONVITO
INFN-BARI
Università degli Studi di Bari

Outline

- The experience with bioinformatics applications
  - Characteristics and issues
- Job Submission Tool:
  - How does it work
  - Main Features
  - Improvements
  - Results
- Conclusions

Introduction

- EGI infrastructure perfectly fits problems that can be subdivided in (many) elementary and independent tasks
- A Job Submission Tool (JST) has been developed (starting from 2007) within 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 fraction of them is expected to fail for any kind of reasons due the distributed grid environment
  - 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

- 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 than 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

- The task status
  - Free: the task is ready to be executed and 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 also 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
- 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.

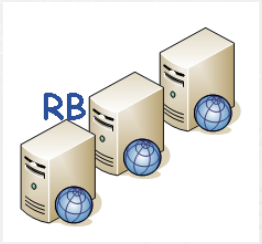
Features II

- Since the tasks are independent and they can be resubmitted if needed, a quite good reliability can be reached and JST can work successfully even if some failure occurs on Grid services
  - More than one WMS is used for jobs submission
  - More than one SE used for the stage-out and stage-in phase
- It is possible to establish tasks dependencies:
  - The 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 each task during a challenge:
  - "Priority" field 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

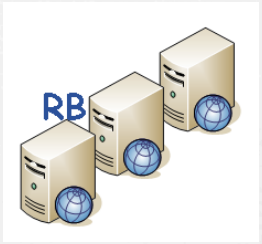
The wrapper

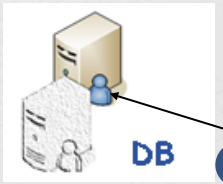
- Requests from the TaskListDB a tasks to be executed
- Retrieves the application executable (it has to be available with on protocols among: https, http, gftp, ftp, xrootd)
- Executes the application code
- Stores the output in one of the configured SEs
  - With one of the configured protocols
- Checks the exit status of the executable and of the stage-out procedure
- Updates the task status into TaskListDB

Workflow



Workflow

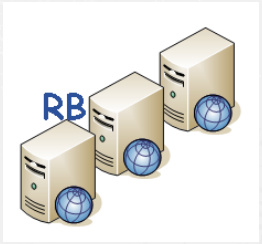




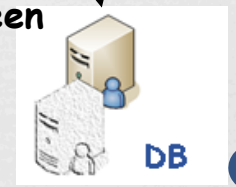
Web Server

The user creates the tasks using the GUI

Workflow



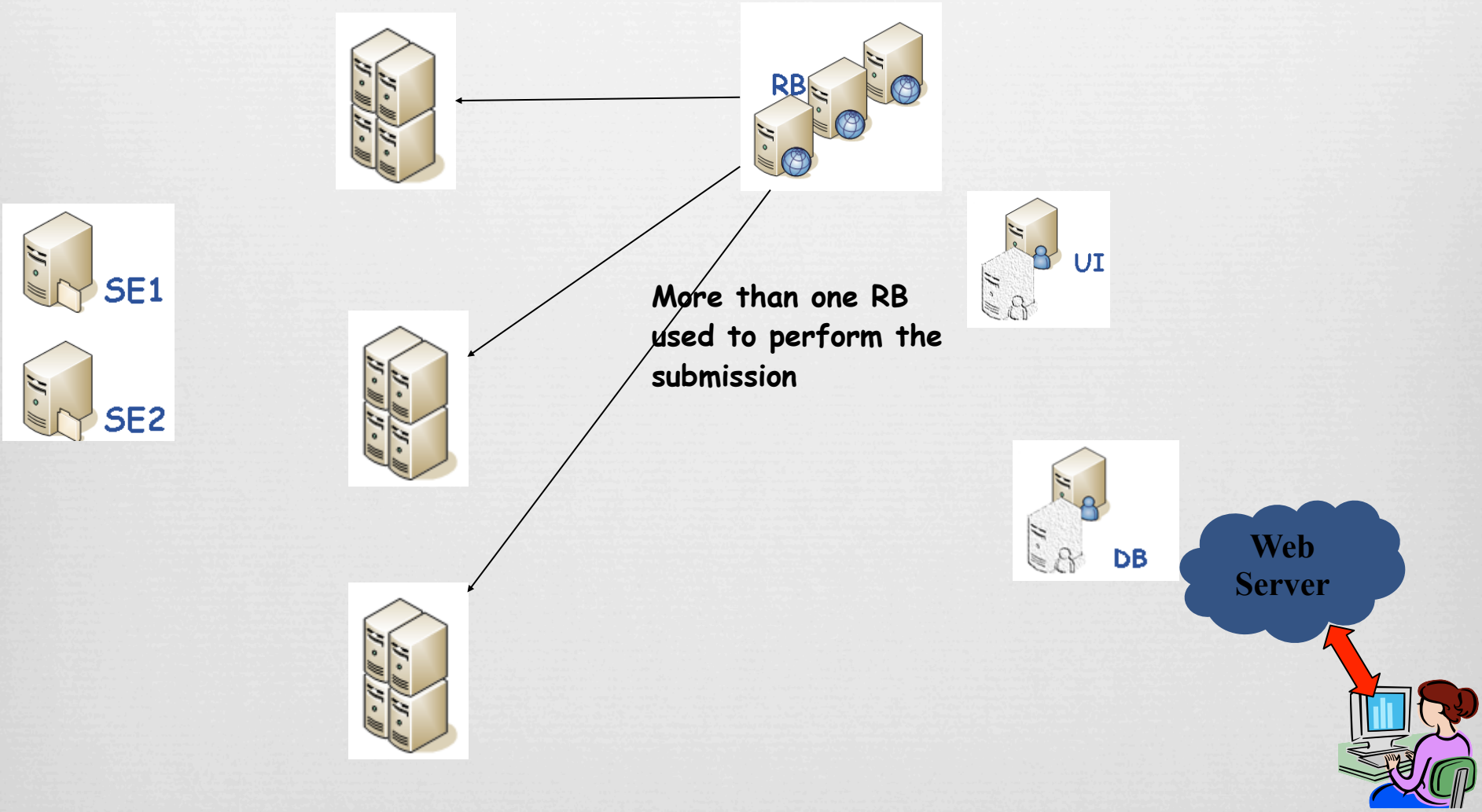
A daemon running on the UI check if any Free task has been created



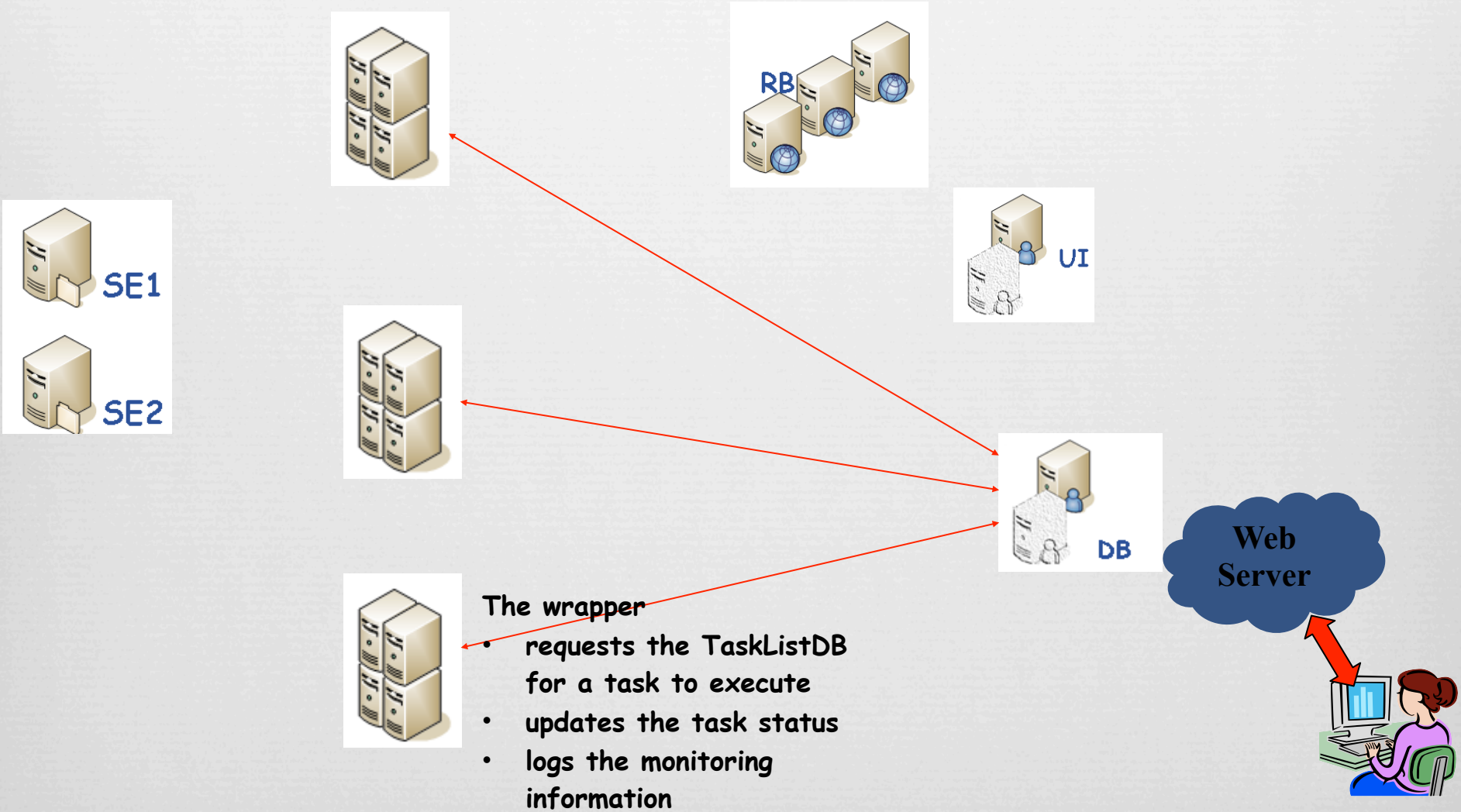
Workflow

The UI daemon submit jobs on the Grid

Workflow


RB
UI
SE1
SE2
DB
More than one RB used to perform the submission
Web Server

Workflow



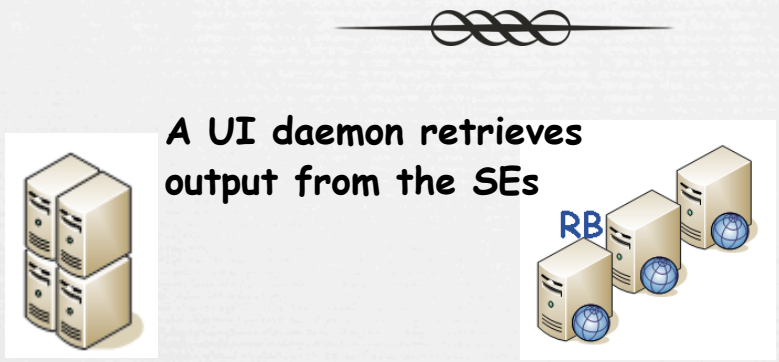
The wrapper

- requests the TaskListDB for a task to execute
- updates the task status
- logs the monitoring information

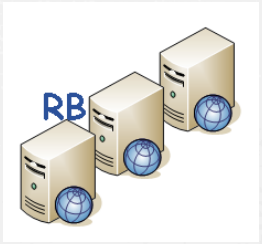
Workflow

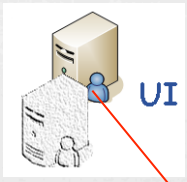
Output registered on a SE chosen in a list of SEs in base to their availability

Workflow


A UI daemon retrieves
output from the SEs
RB
SE1
SE2
UI
DB
Web
Server

Workflow





The daemon send the output to a web server

Development

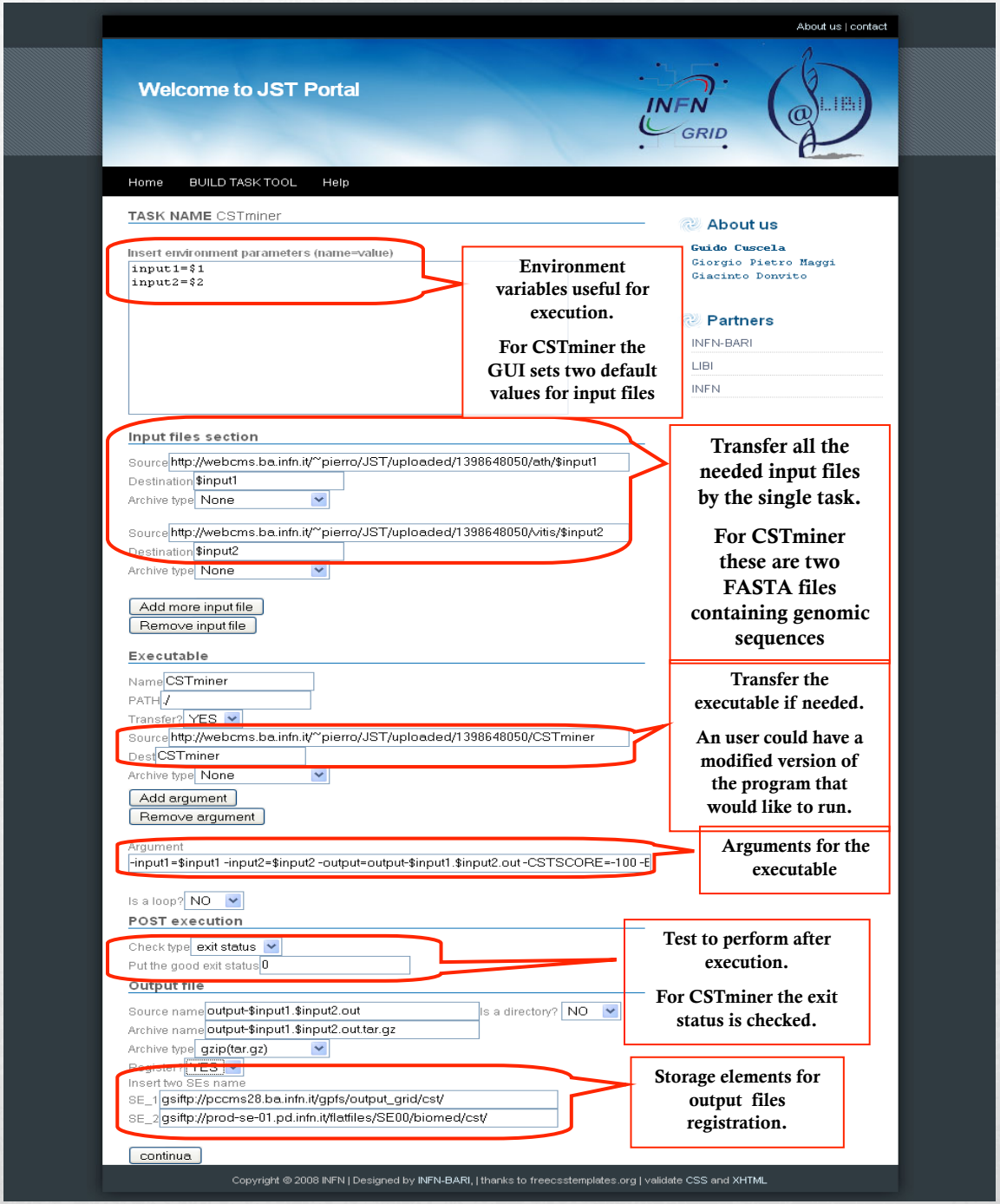
- At the beginning of the project the application challenge were performed by the JST developers team:
  - The users had just to provide input files, the executable, the libraries needed …
- With the experience acquired executing with success several challenge on bioinformatics applications, the JST 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

Development II

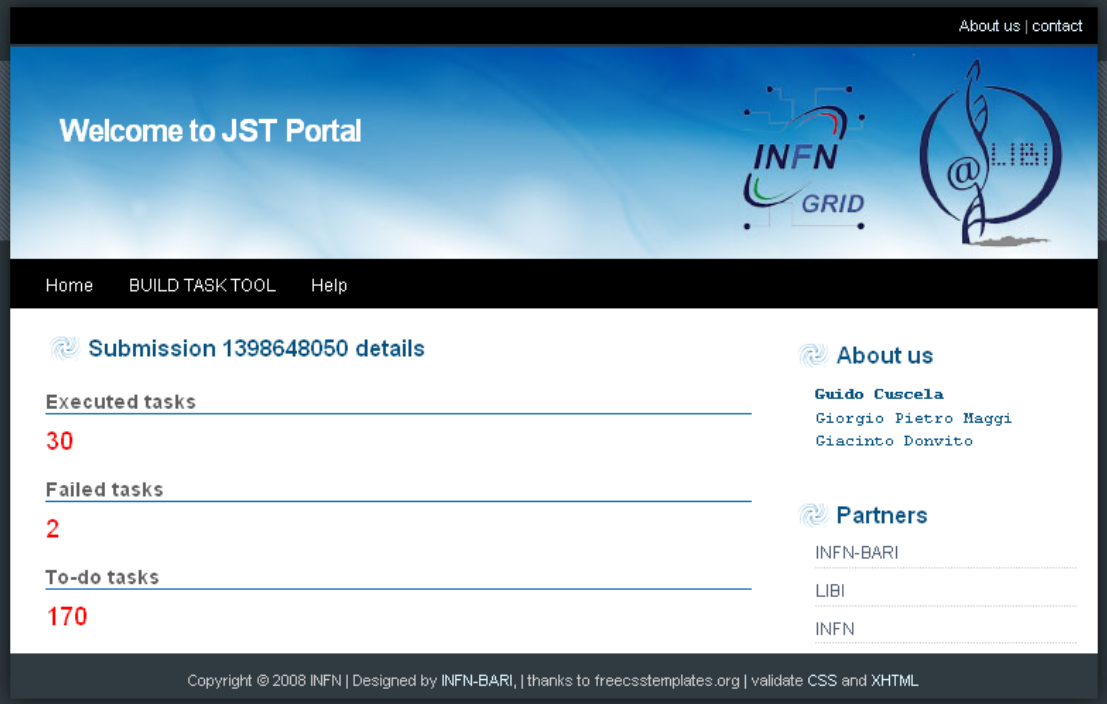
- The interface is based on few web pages that:
  - Guide the user creating the tasks
  - Trigger submission of the application to the Grid
    - The UI daemon checks periodically the TaskListDB
  - Monitor the status of the challenge
    - The status of the tasks
    - The number and the reasons of failures
- Standard username/password login required to the end-user:
  - In order to make the grid experience as much transparent as possible we exploit a Robot Certificate on the server
  - We keep track (into a database) of any submission from each user in order to be able to track any possible misuse
  - This users can submit only well known applications

Task initialization

Task description

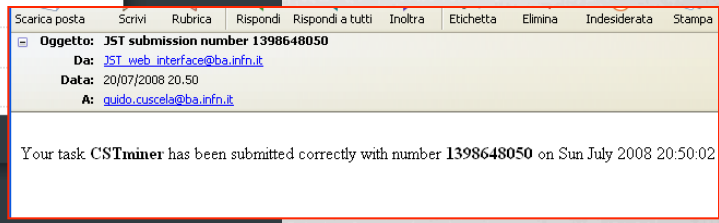


Task Monitoring and retrieving output



It is possible to download the files produced

Every submission has unique id



Tasks creation

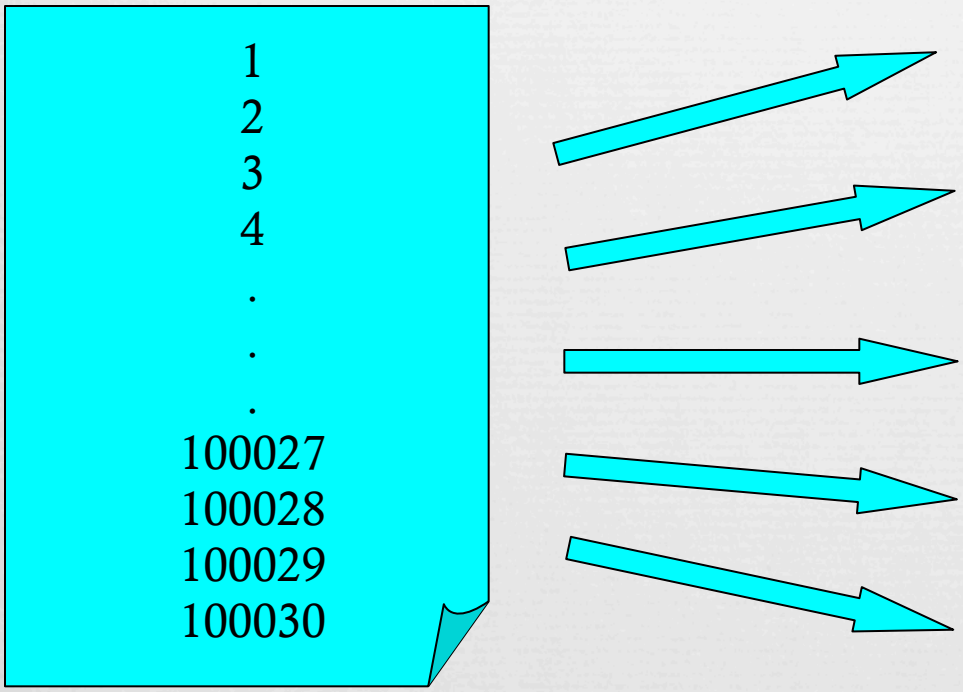
Genes descriptions ids

1
2
3
4
.
.
.
100027
100028
100029
100030

Starting from a single input

Tasks creation

Genes descriptions ids



Starting from a single input

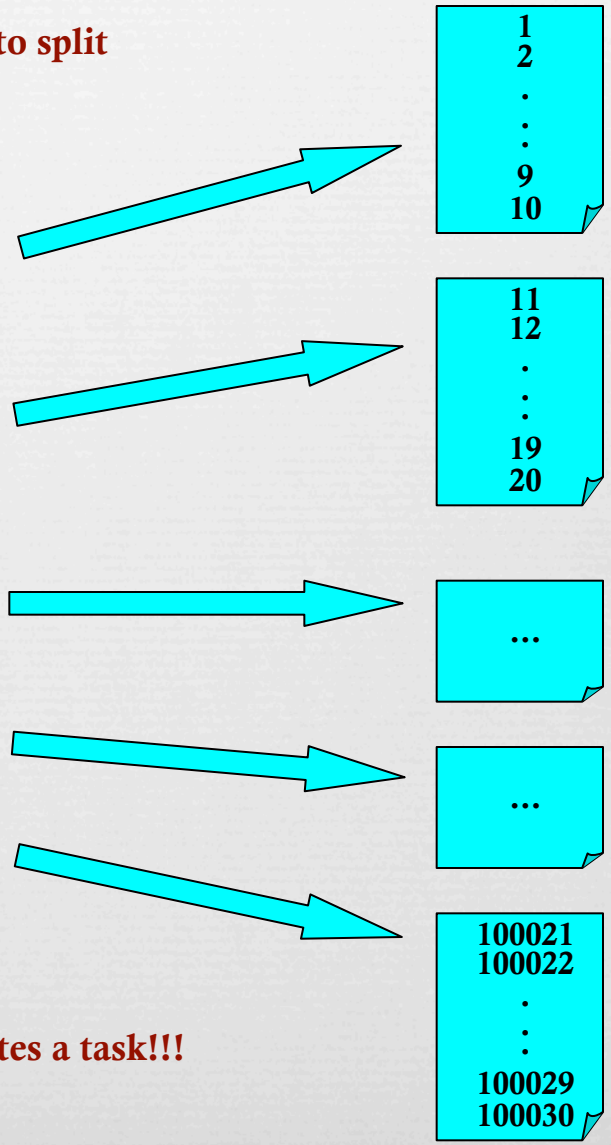
Tasks creation

The user decides how to split the input file

Genes descriptions ids

1
2
3
4
.
.
.
100027
100028
100029
100030

Starting from a single input

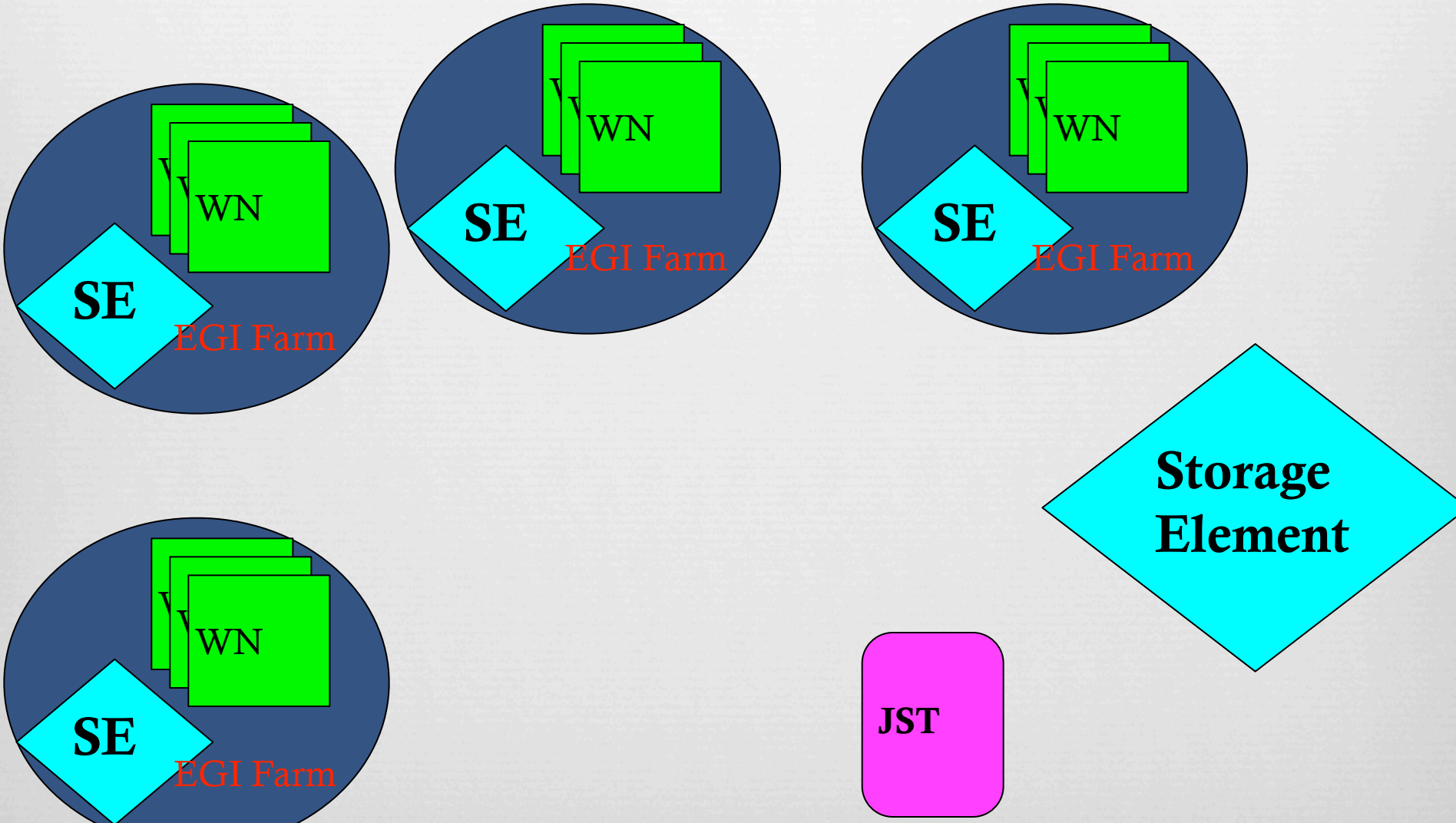


Every input file generates a task!!!

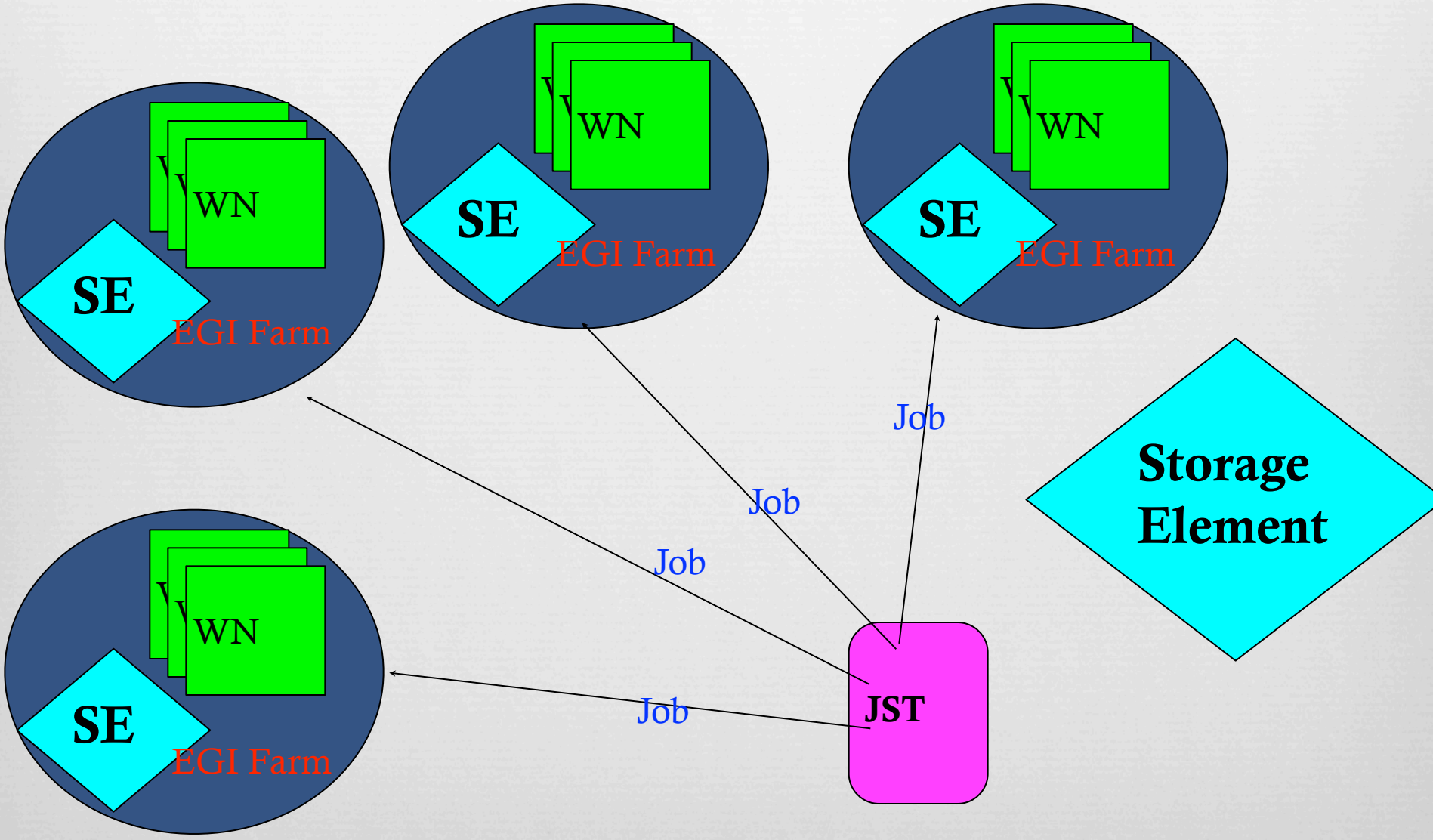
Data-Management and Bioinformatics applications

- Many of the bioinformatics applications run over a given Database for each task
  - This means that a lot of jobs around the grid is reading the same small set of files (~few GBytes)
  - This could easily become a bottleneck
  - We do not know in advance where the jobs will be executed…
  - … but we would try to exploit every computing resources that we could reach over the grid
- We are using an automatic procedure to distribute files around the grid as the jobs runs
  - This procedure is completely based on the EGI data-management tools (lcg-*, LFC, SRM Storage)

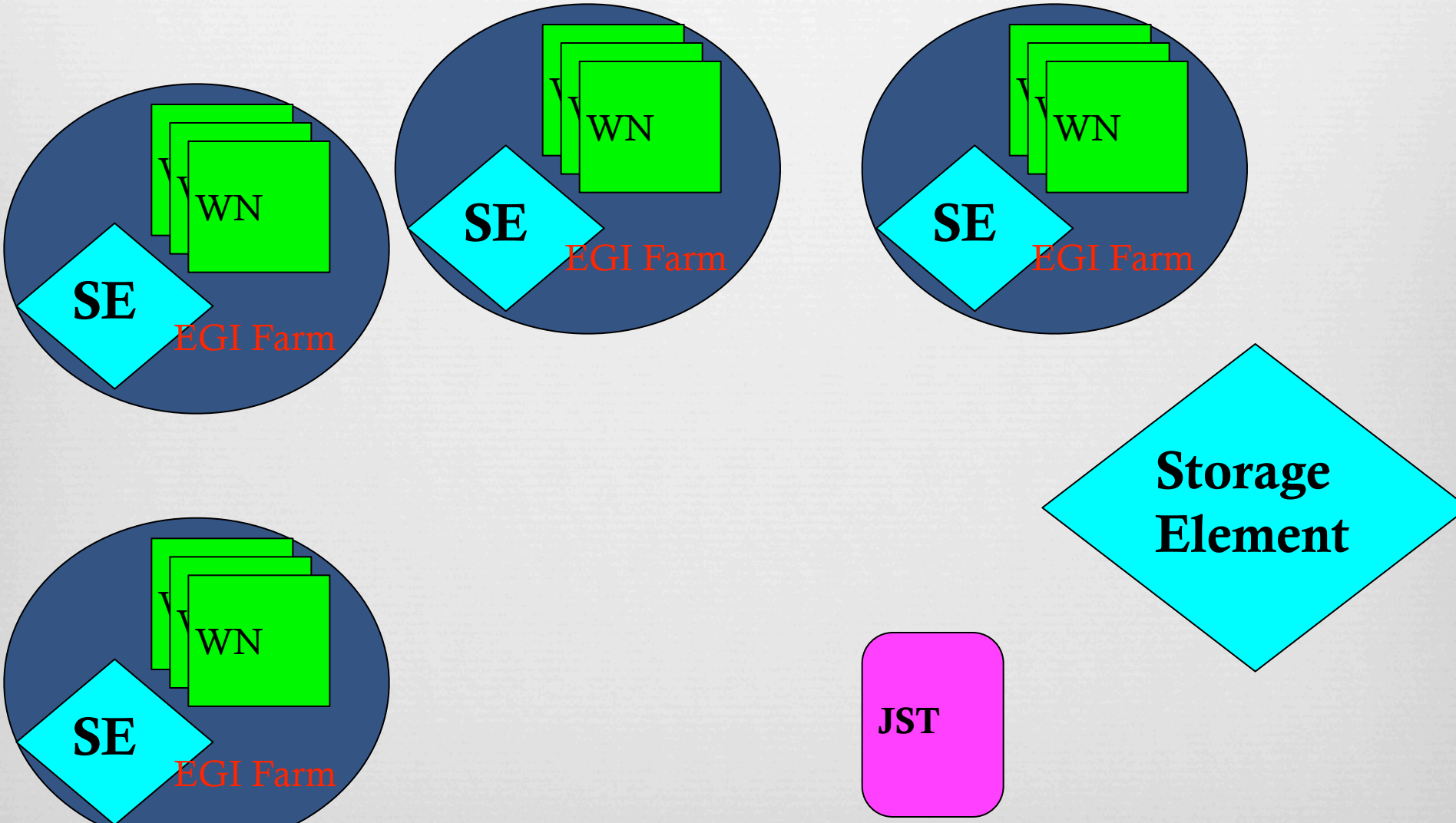
Data-Management and Bioinformatics applications



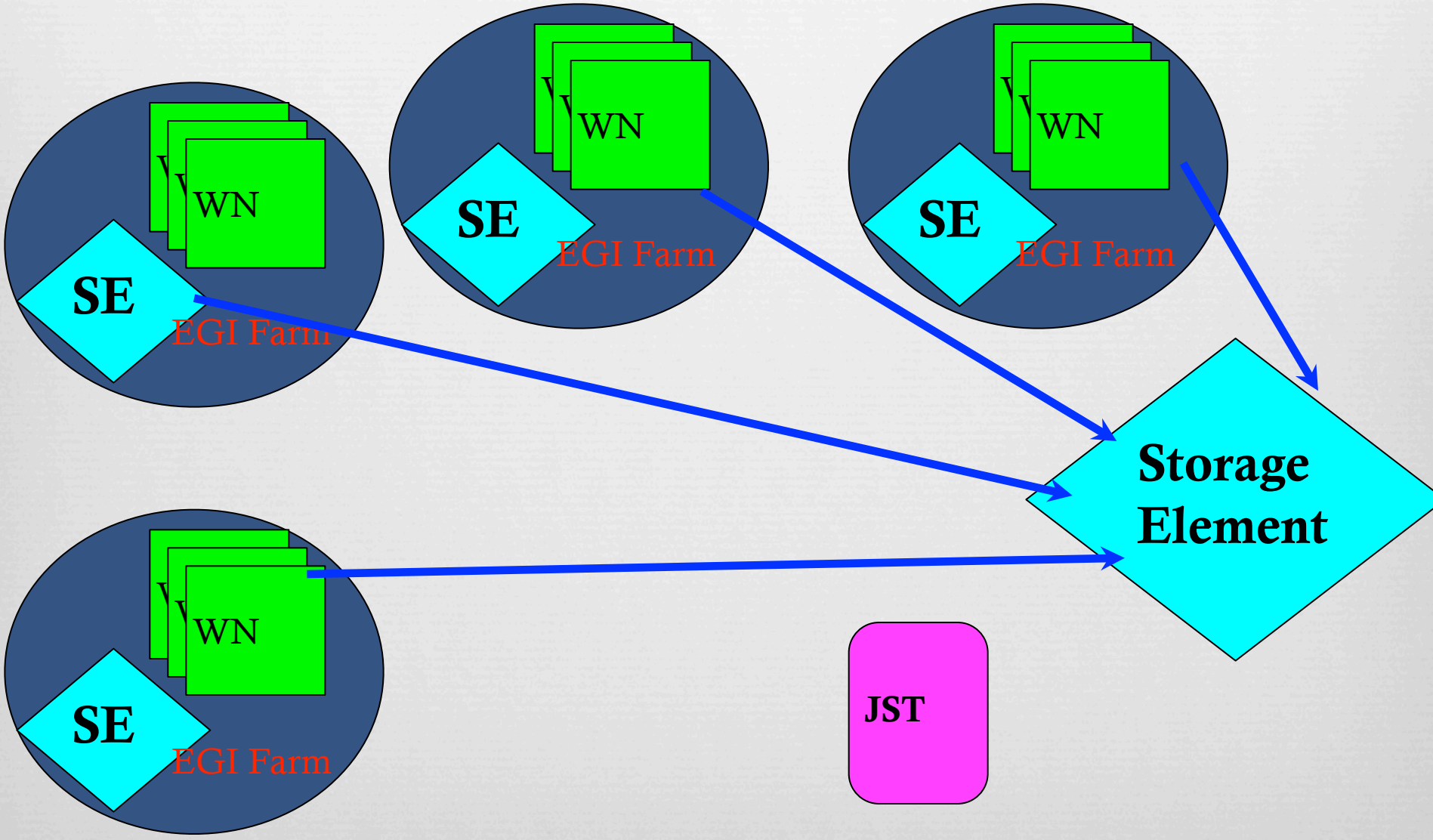
Data-Management and Bioinformatics applications



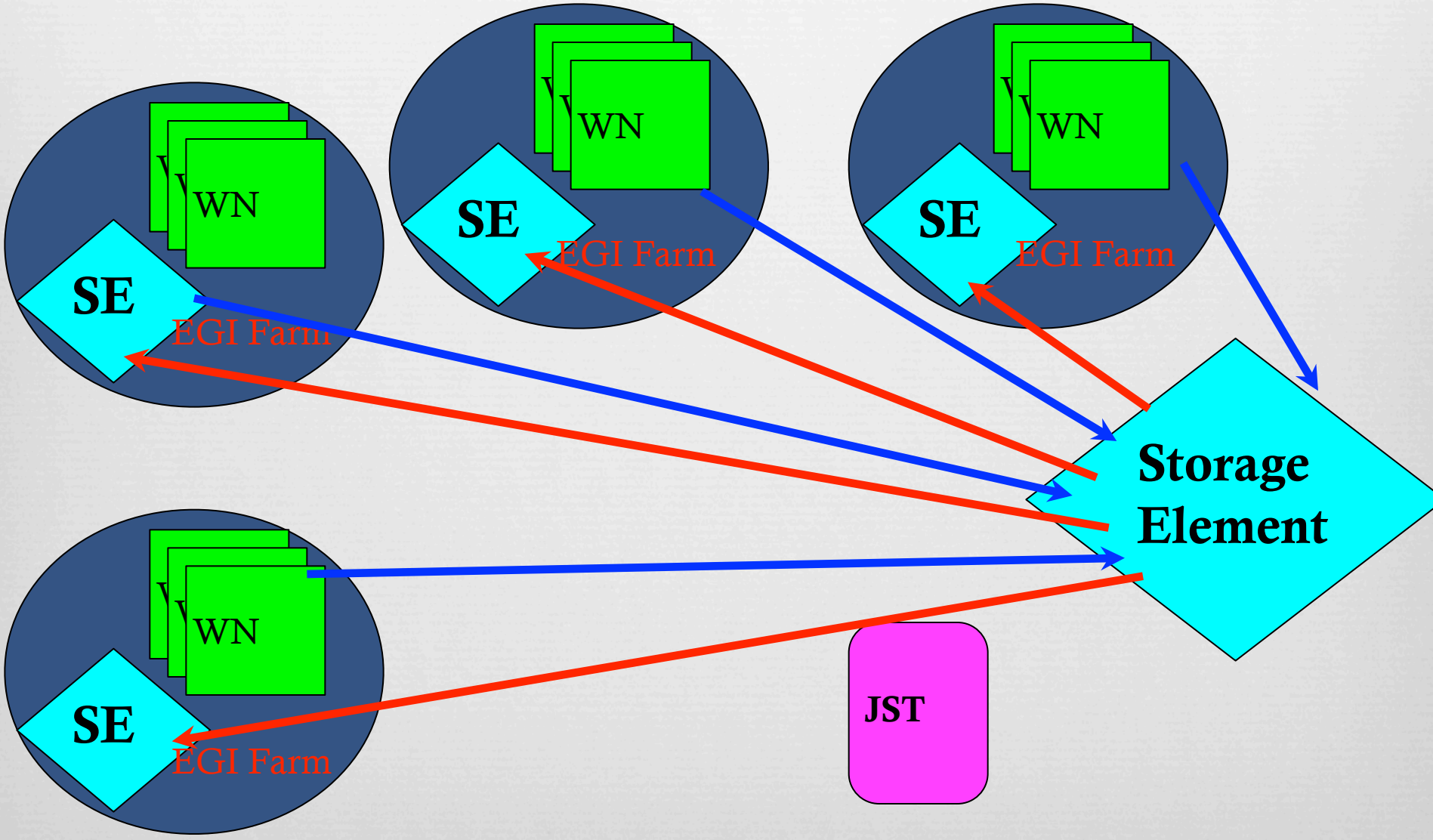
Data-Management and Bioinformatics applications



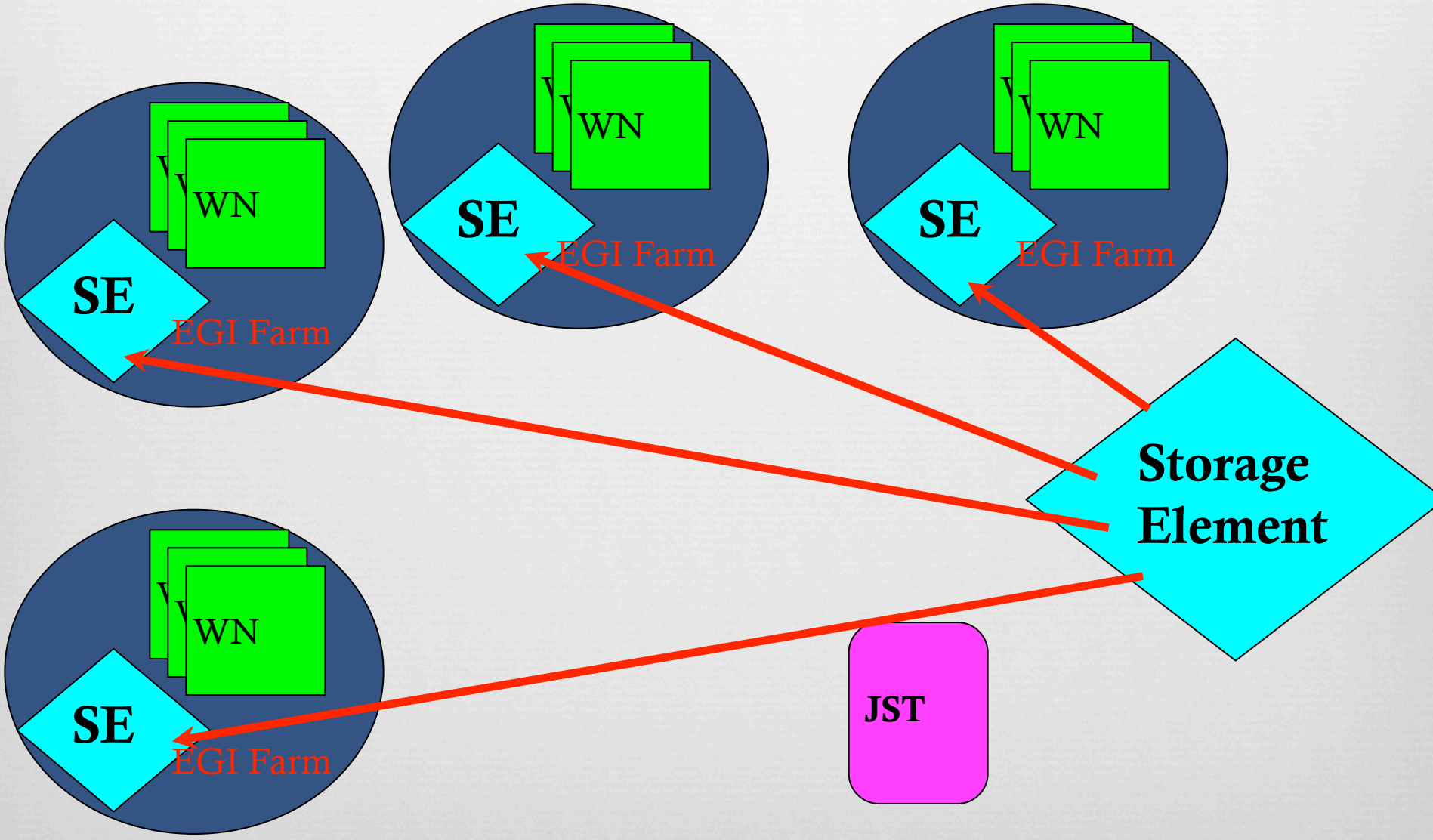
Data-Management and Bioinformatics applications



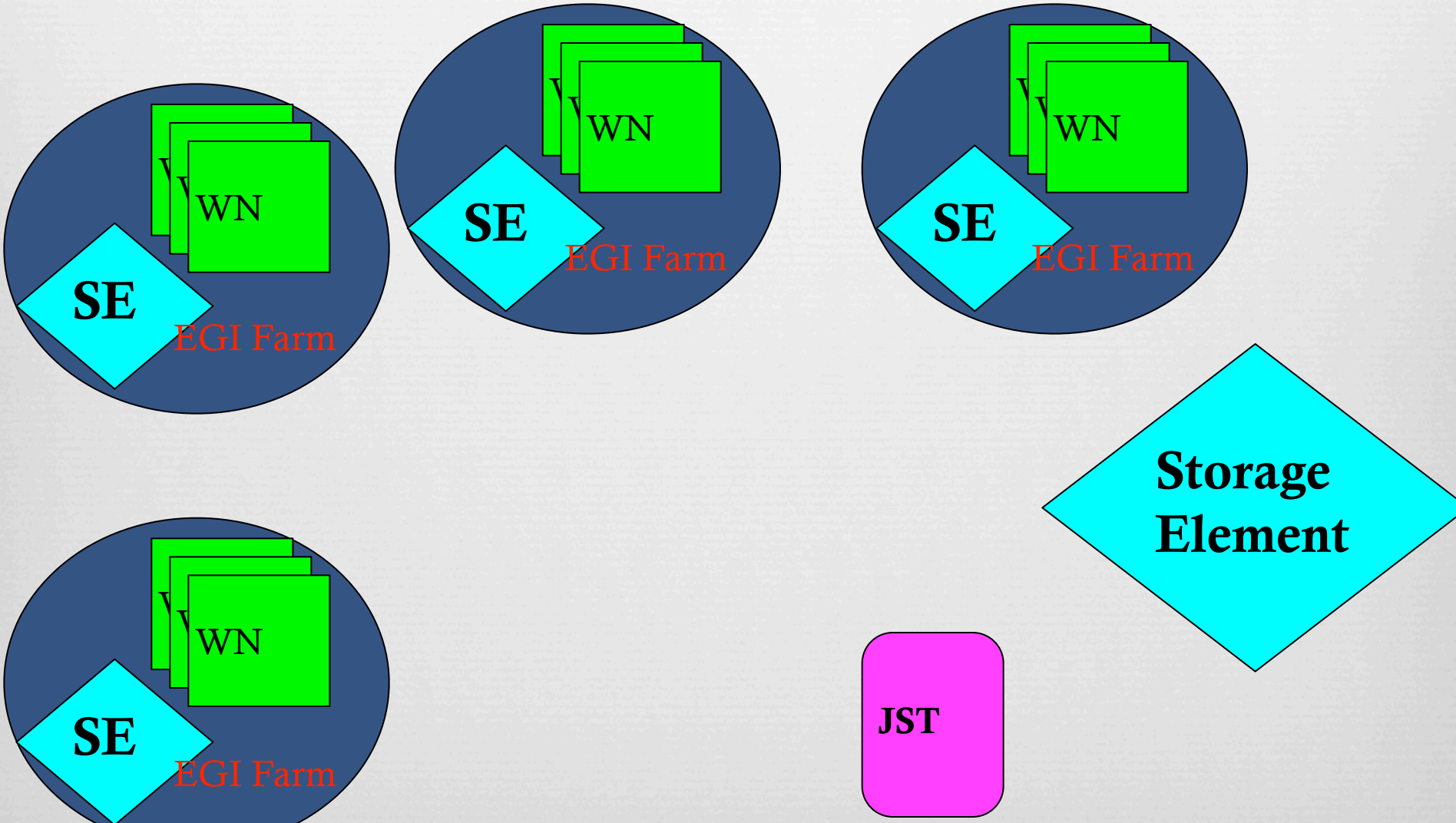
Data-Management and Bioinformatics applications



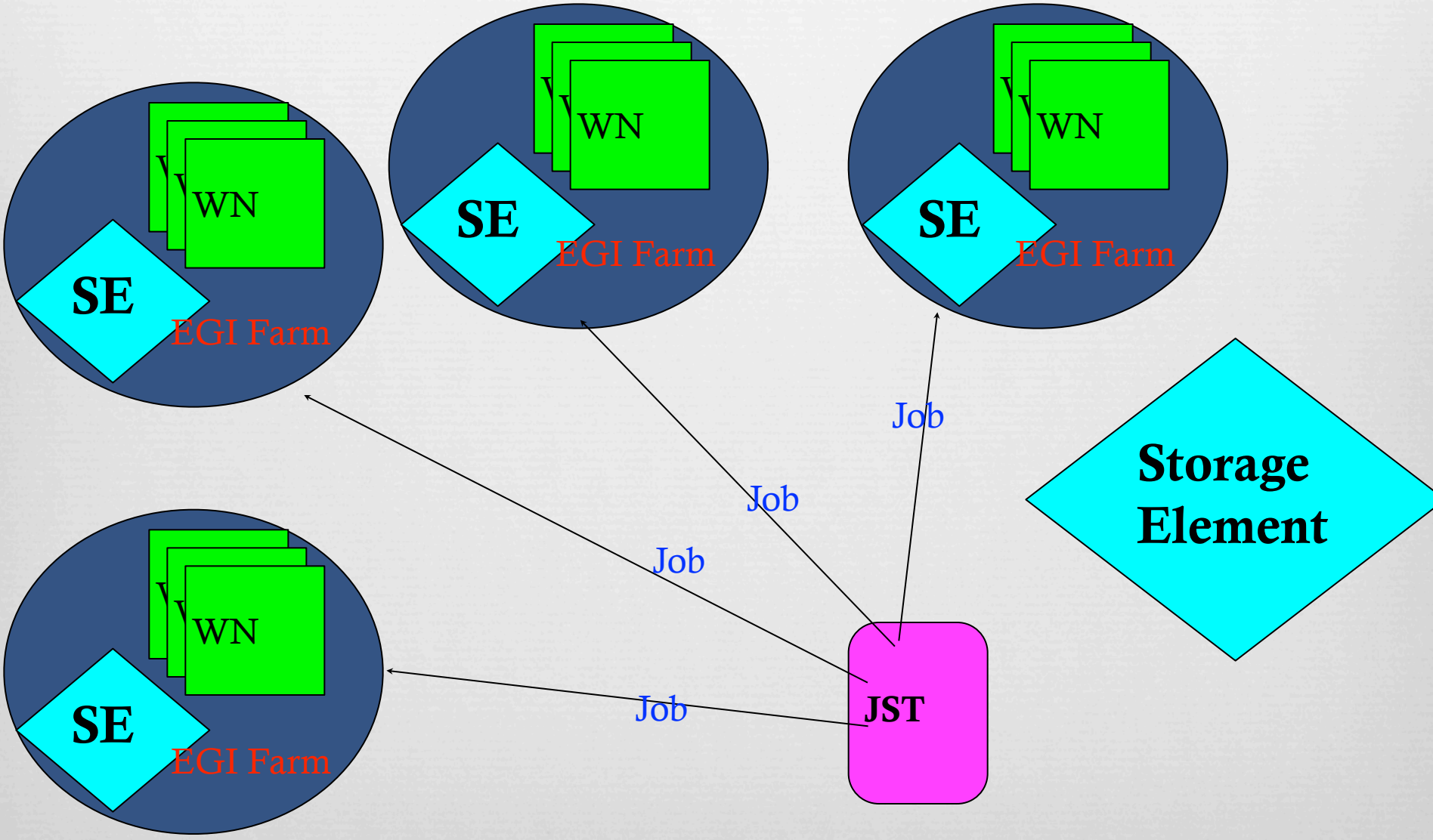
Data-Management and Bioinformatics applications



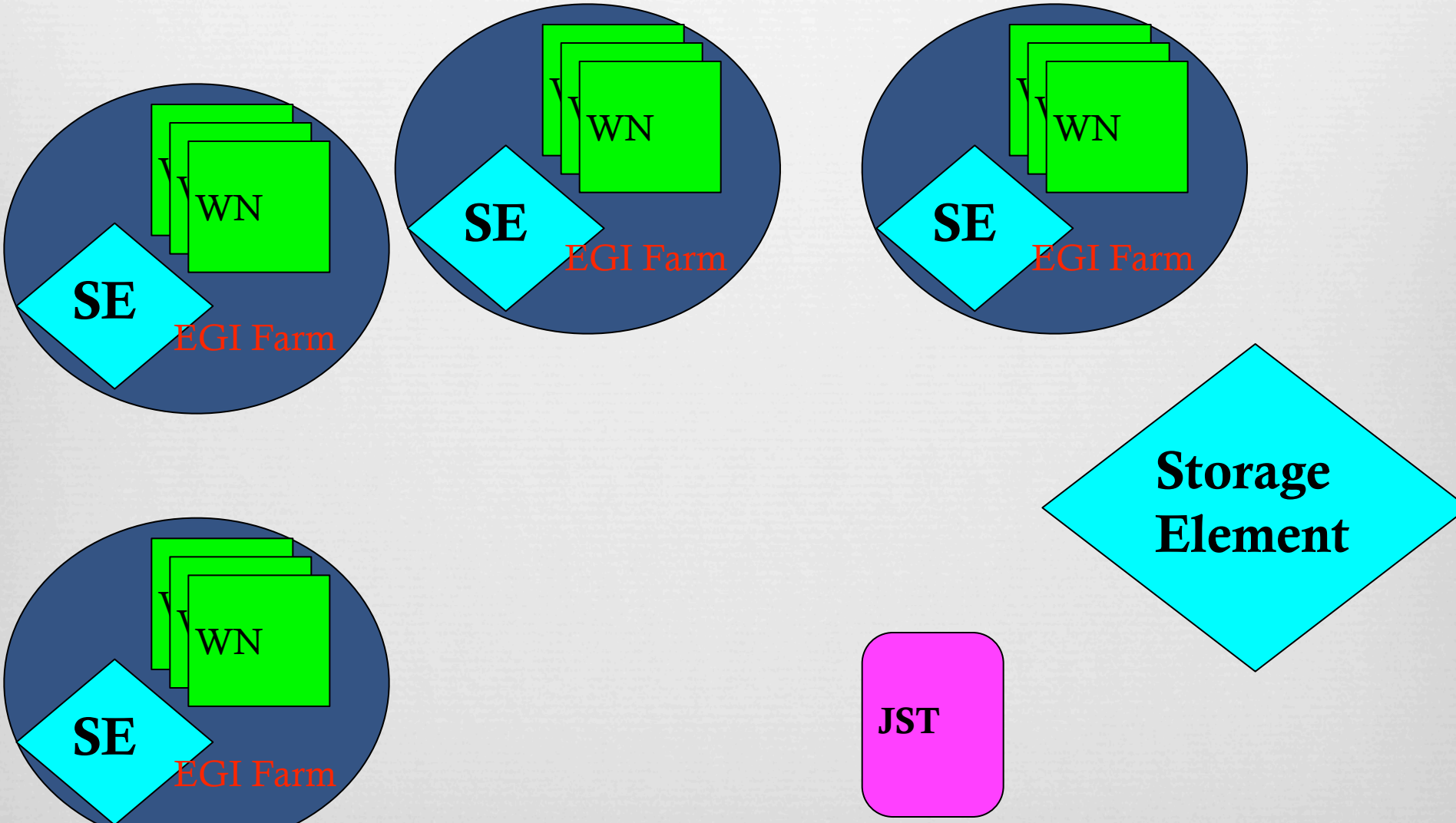
Data-Management and Bioinformatics applications



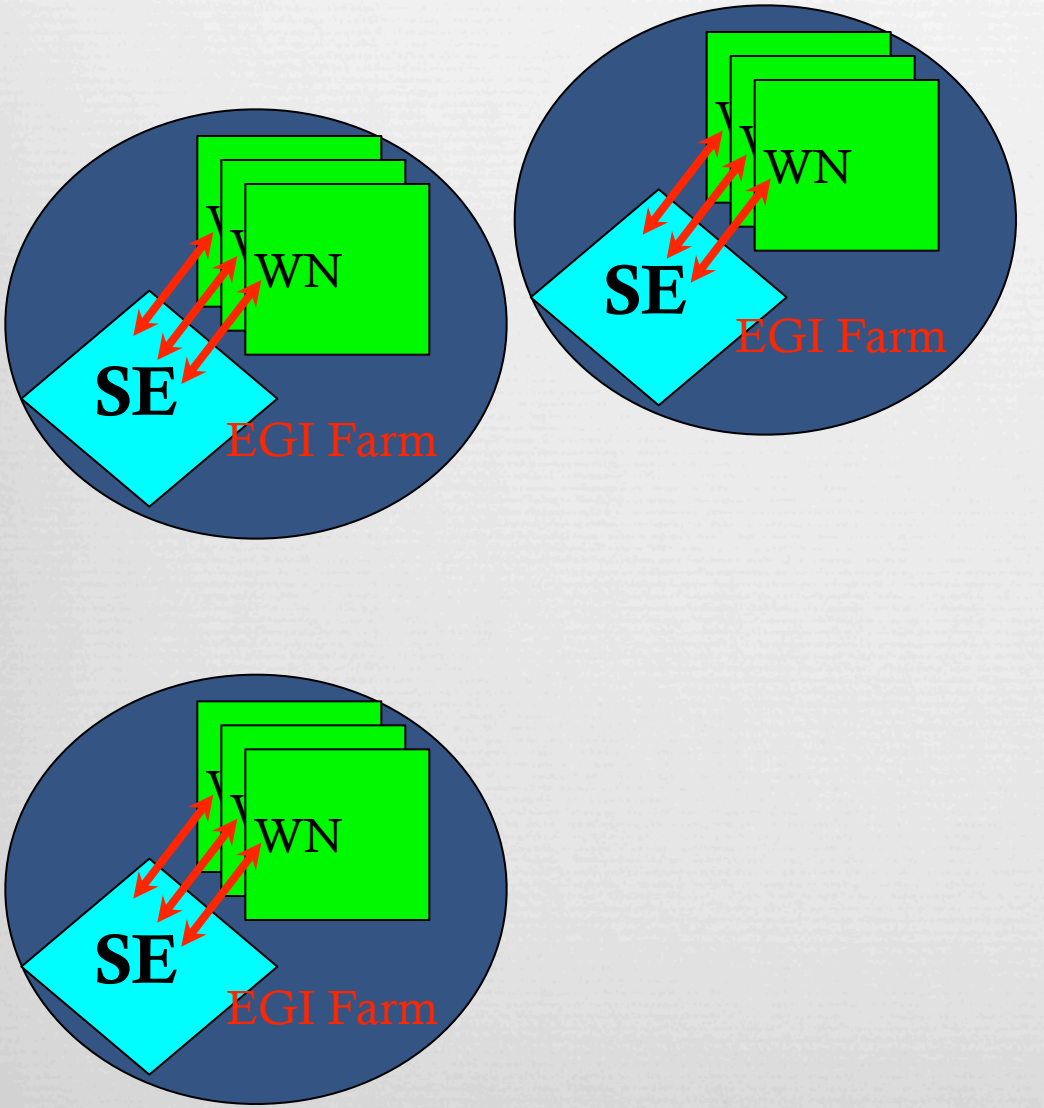
Data-Management and Bioinformatics applications



Data-Management and Bioinformatics applications



Data-Management and Bioinformatics applications



Storage Element

JST

Data-Management and Bioinformatics applications

- Most of the bioinformatic challenge produces hundred of thousand (or million) of very small files
  - writing a file using standard grid catalogues has a not negligible overhead
  - the catalogue is usually single point of failure (usually no High Availability solution is in place)
  - we can not trust a single Storage Element on the grid:
    - it could fail or get overloaded
- We are using an automatic procedure deal about SE failures and overload
- Information about the output of each task is stored in the TaskListDB (within the JST framework)
- Each job over the grid produces hundreds (or thousands) of files which are packed together before storing them on SEs

Optimized Scheduling

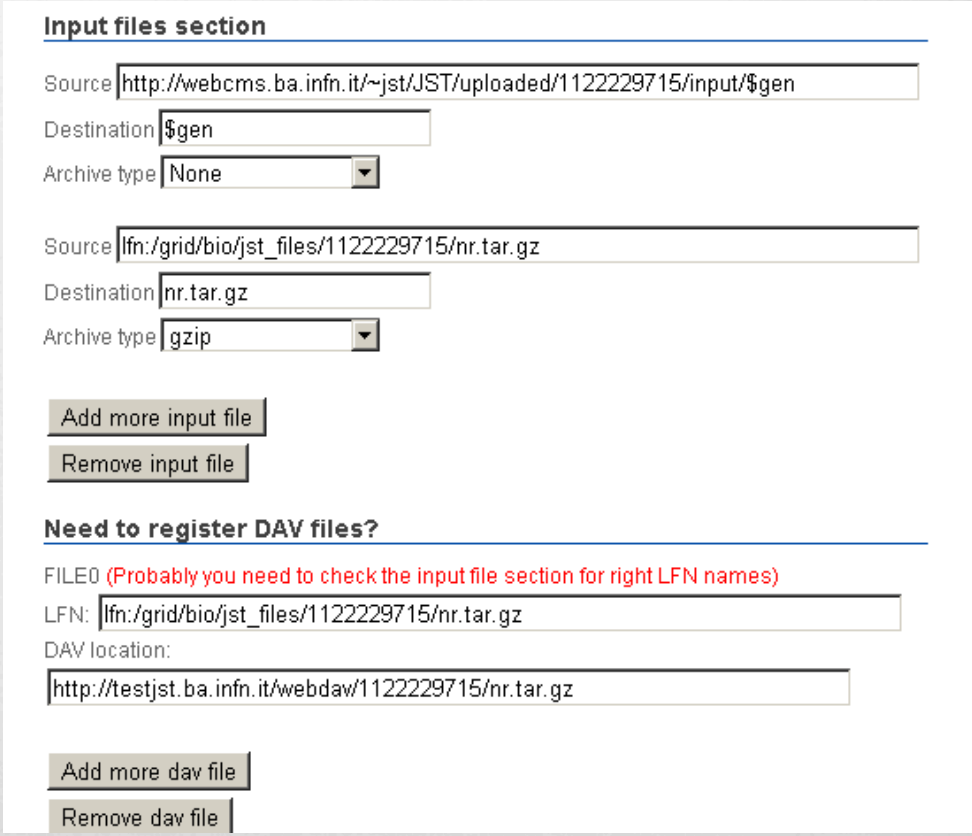
- The LIBI bioinformatic group in Italy do not have dedicated resources but we try to exploit all grid sites also if they have few free CPU available
  - we submit jobs to all the sites on the EGI infrastructure
  - we trust on the match making done by gLite WMS with the default ranking
  - we need to be sure not to overload a single site and leave unused resources on other farms
- For a better control on the number of jobs queued on a given farm, a new table within JST DB was set up to keep track of the status of each submitted grid job
  - We do a bulk query to WMS used looking for "scheduled" jobs and farms in which to much "scheduled" job are waiting to be executed are excluded from the usable sites

Input files: problems

- Quite often the size of the input files is O(GB) so it means is quite difficult to upload it using the standard web interface
- Typical Bioinformatics users do not know how to register input files into grid storage elements and catalogues
- We need to provide an easy interface to manage large files and then transfer it to the grid in a transparent way
- This transfers service should:
  - Have at least one client in every platform (Windows/MacOS/Linux)
  - Provide authentication at least with username/password
  - Provide high performance on high-latency networks

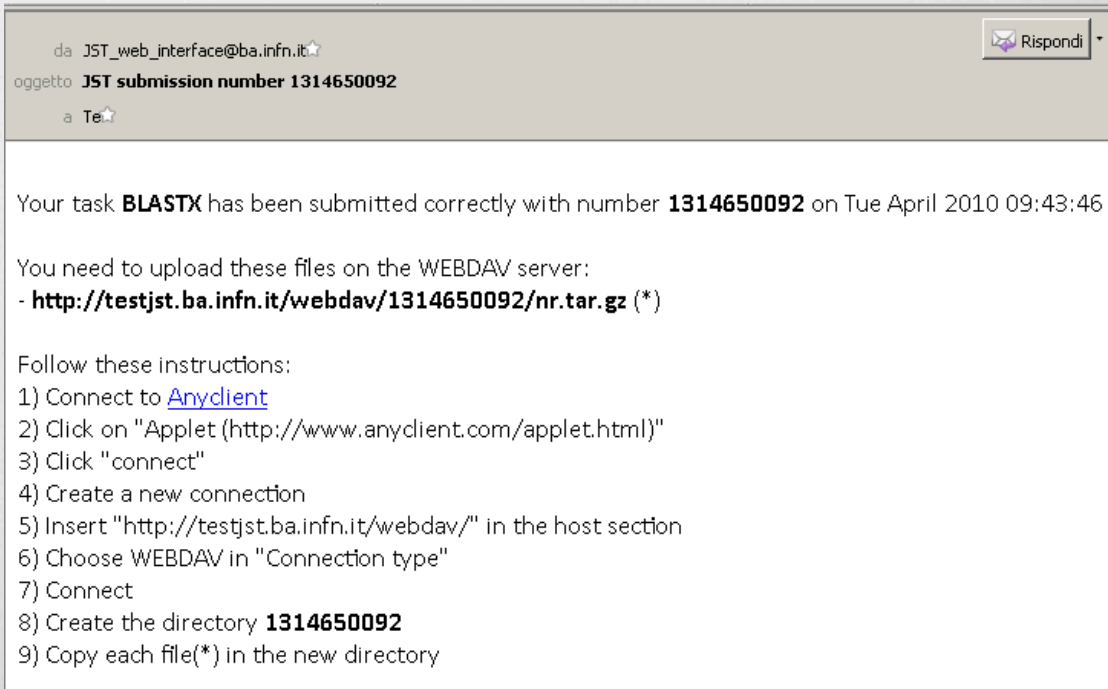
WebDAV service

- **We found that WebDAV fulfil all the requirements**
- The web submission page allows users to define which files have to be uploaded on the webdav server.
- Both the lfn and the webdav destination url are provided
  - Lfn: logical file name that will be used to register the file on the grid storage elements
  - The webdav link is useful to know where to upload the input file



WebDAV service II

- As soon as the procedure is finished on the Web GUI the user will receive an e-mail with the instructions to upload the files on the WebDAV server
- The jobs submission will start as soon as all the needed files are uploaded to the WebDAV server
- For security and privacy reasons every access to the WebDAV server is protected with the same security mechanism of the JST server



da JST_web_interface@ba.infn.it

oggetto **JST submission number 1314650092**

a Te

Rispondi

Your task **BLASTX** has been submitted correctly with number **1314650092** on Tue April 2010 09:43:46

You need to upload these files on the WEBDAV server:
- **http://testjst.ba.infn.it/webdav/1314650092/nr.tar.gz** (*)

Follow these instructions:
1) Connect to Anyclient
2) Click on "Applet (http://www.anyclient.com/applet.html)"
3) Click "connect"
4) Create a new connection
5) Insert "http://testjst.ba.infn.it/webdav/" in the host section
6) Choose WEBDAV in "Connection type"
7) Connect
8) Create the directory **1314650092**
9) Copy each file(*) in the new directory

Results

- Applications executed:
  - GAF: Gene Analogous Finder
    - 6000 jobs submitted on Grid
    - More than 200 WNs used
  - ASPic: Alternative Splicing Prediction
    - One complete genome analyzed (Mouse) in 3 days
  - PAML: maximum likelihood analysis using approximate derivatives
    - 6690 cases evaluated in 36 hours
  - FT-COMAR: Protein Tertiary Structure Prediction
    - 21.582.680 runs executed in 5 days
  - MrBayes: Bayesian inference of phylogeny
    - over 7200 different runs; ~5 CPU/years
    - ~20 days of run on EGI infrastructure
  - Emboss "vrnalfold": applications for molecular sequence
    - over 0.5 M of sequences analyzed
  - MAFFT: multiple sequence alignment
    - more than 5000 sequences aligned
  - Solexa Illumina: Sequence clustering
    - 2.5M sequences clustered
  - BayeSSC: modified version of SerialSIMCOAL
    - 1M of runs

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, EMBOSS, BayeSSC, MAFFT and many others
- Running a large challenge on the grid requires only small or no human effort as most of the steps are automatic and fault tolerant
- We used this tool in order to run several challenges with the mentioned applications
- Roughly we have executed jobs for more than 800 CPU/years
- Not only bioinformatics applications could be executed on the Grid using JST
  - If your application could be split in single independent tasks it perfectly fits JST features

Conclusions

- It is used successfully also to submit jobs using local batch system
  - the modular structure is the key feature to give the possibility to support different computational infrastructure
- The use of the Robots Certificate make the use of the grid as much transparent as possible to a non expert user
- We tested WebDAV interface up to several GB of input files and it always proved to be reliable and fast enough
- Our automatic files distribution procedure let us to run concurrently on several farms over the grid infrastructure without overloading SEs and without a pre-data-placement strategy

References

- People involved:
  - Giacinto DONVITO
    - giacinto.donvito@ba.infn.it
  - Guido CUSCELA
    - guido.cuscela@ba.infn.it

- Web site:
  - http://webcms.ba.infn.it/cms-software/cms-grid/index.php/Main/JobSubmissionTool
  - http://webcms.ba.infn.it/~jst/JST/