Soluzione di Software as a Service (SaaS) per Applicazioni Scientifiche. Come sfruttare risorse di calcolo e storage distribuito in Grid tramite WorkFlow in modo semplice e trasparente

Giacinto Donvito
INFN-Bari
ReCaS
giacinto.donvito@ba.infn.it

e-infrastructure

CAPACITIES

Outlook

- Overview of the Use Cases
- Overview of the SaaS framework
  - Front-end and Back-end overview
  - Job Submission features
  - Data Management features
- Test and results
- Conclusions and Work in progress

Biodiversity Virtual e-Laboratory

BioVeL is an international network of experts

•Connects two scientific communities: IT and biodiversity.

•Offers an international network of IT expert scientists in BioVeL's data processing services.

•Shares expertise in workflow studies among BioVeL's users.

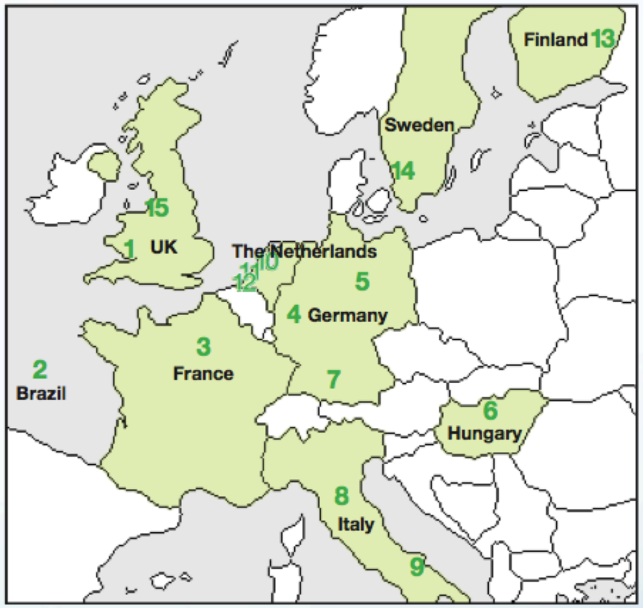
•Fosters an international community of researchers and partners on biodiversity issues.

•BioVeL is an e-laboratory that supports research on biodiversity using large amounts of data from cross-disciplinary sources.

Biodiversity Virtual e-Laboratory

BioVeL is a consortium of 15 partners from 9 countries

1. Cardiff University, UK – Coordinator
2. Centro de Referência em Informação Ambiental, Brazil
3. Foundation for Research on Biodiversity, France
4. Fraunhofer-Gesellschaft, Institute IAIS, Germany
5. Free University of Berlin – Botanical Gardens and Botanical Museum, Germany
6. Hungarian Academy of Sciences Institute of Ecology and Botany, Hungary
7. Max Planck Society, MPI for Marine Microbiology, Germany
8. National Institute of Nuclear Physics, Italy
9. National Research Council: Institute for Biomedical Technologies and Institute of Biomembrane and Bioenergetics, Italy
10. Netherlands Centre for Biodiversity (NCB Naturalis), The Netherlands
11. Stichting European Grid Initiative, The Netherlands
12. University of Amsterdam, Institute of Biodiversity and Ecosystem Dynamics, The Netherlands

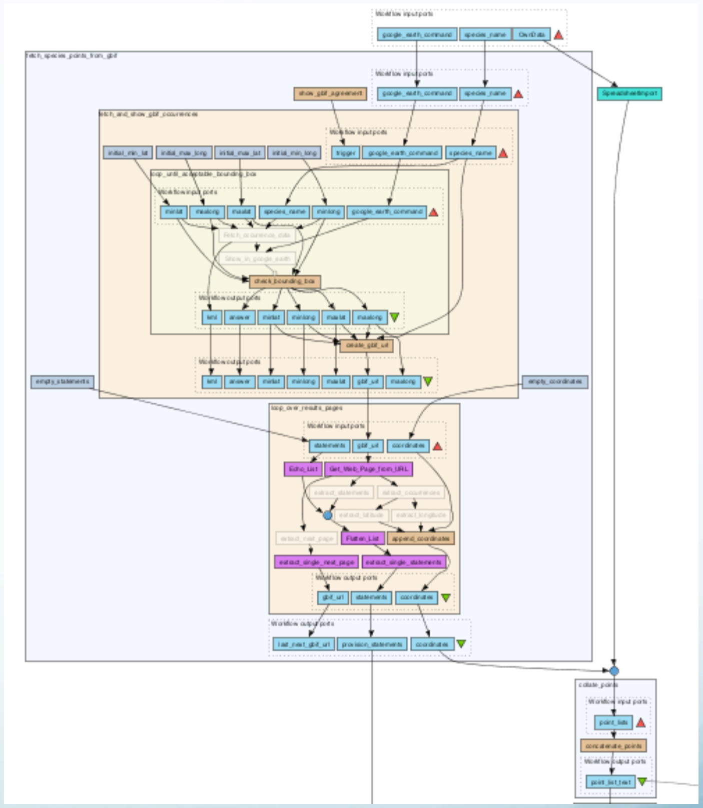


13. University of Eastern Finland, Finland
14. University of Gothenburg, Sweden
15. University of Manchester, UK

Biodiversity Virtual e-Laboratory

BioVeL is a powerful data processing tool

- Import data from one's own research and/or from existing libraries.
- "Workflows" (series of data analysis steps) allow to process vast amounts of data.
- Build your own workflow: select and apply successive "services" (data processing techniques.)
- Access a library of workflows and re-use existing workflows.
- Cut down research time and overhead expenses.
- Contribute to LifeWatch and GEO BON.

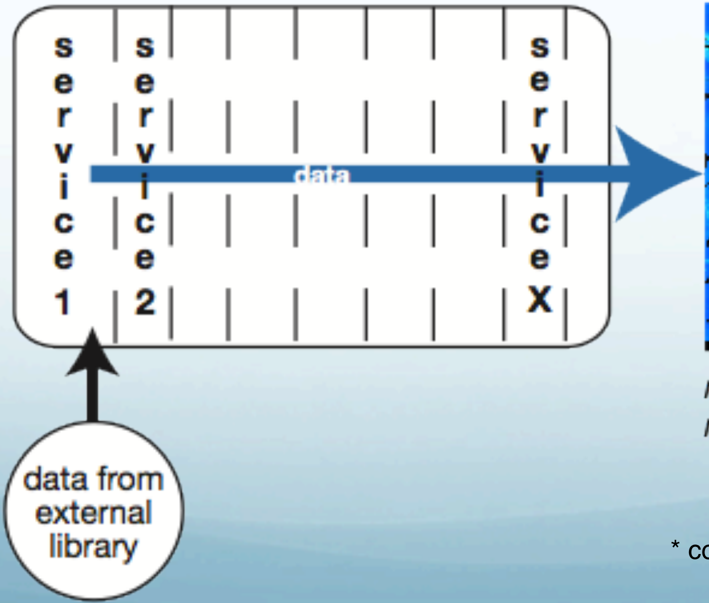


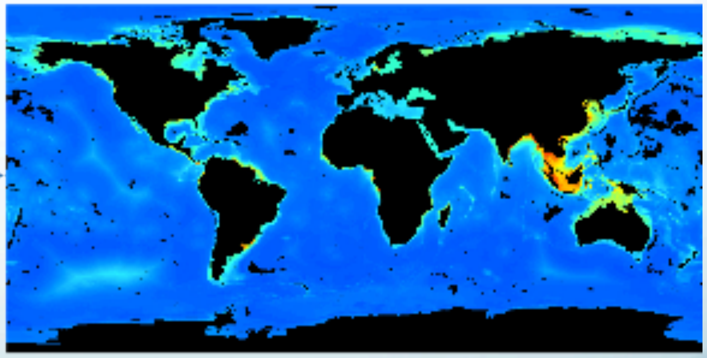
Part of a workflow to study the ecological niche of the horseshoe crab

Biodiversity Virtual e-Laboratory

Showcase study 1: create a workflow*

Study on the ecological niche of the south east Asian horseshoe crab, an endangered species:
•Import south east Asian data from external library
•Apply succession of "services" = workflow
•Result: ecological niche map





results: map showing the potential ecological niche of the south-east Asian horseshoe crab

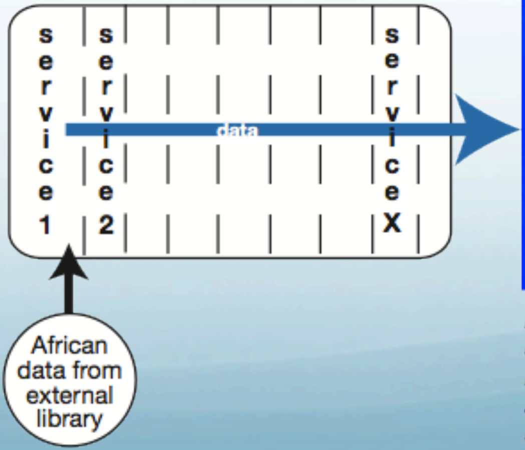
* courtesy Matthias Obst, University of Gothenburg, Sweden

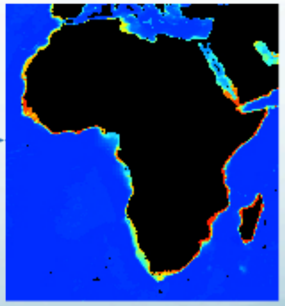
Showcase study 2: re-use a workflow

Study on the ecological niche of the American horseshoe crab
- Import American data
- Re-use south east Asia crab study workflow
- Result: ecological niche map for American horseshoe crab

Compare the ecological niches of the south east Asian and American crabs.

Potential study of the ecological niche of an African animal
- Import African data
- Re-use horseshoe crab study workflow
- Result: ecological niche map for African animal

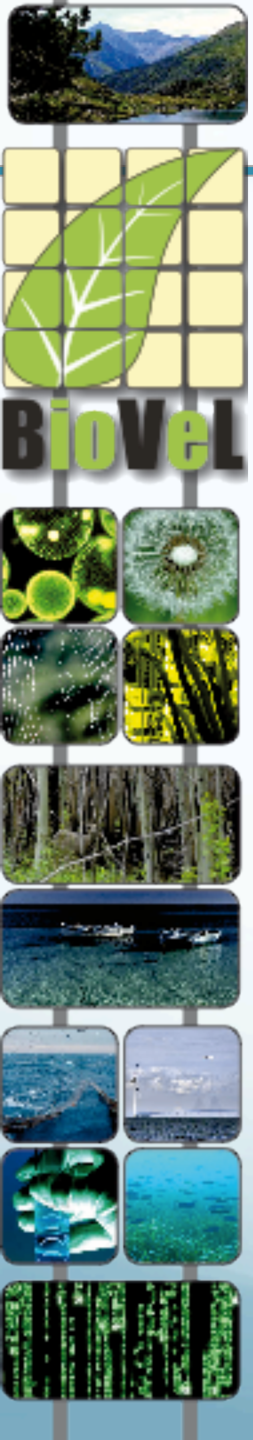




results: map showing the potential ecological niche of an African species re-using the same workflow --or an altered version of it-- as for the south east Asia horseshoe crab.

Medical Use case and LONI Pipeline

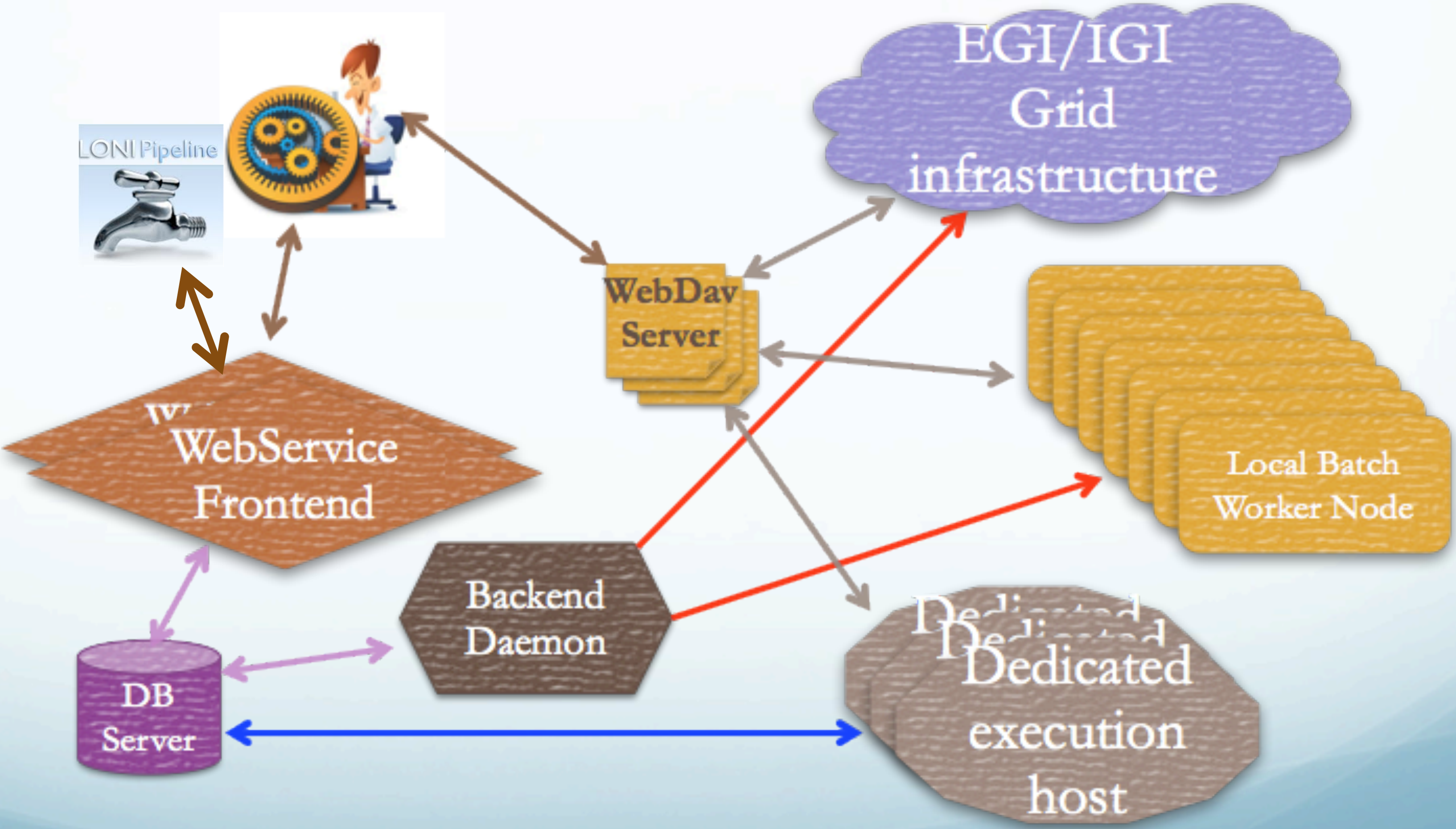
- Analysis of neuro-images to diagnose the *Alzheimer* disease
- Several different libraries used:
  - Matlab, ITK, etc
- LONI Pipeline used to orchestrate the complex analysis workflow
- The analysis chain is quite long in terms of number of different programs to be executed
  - usually more then 10 algorithms are applied to a single image
- The whole analysis chain on a single image takes about 200 CPU/hours
  - Usually a research group need to process thousands of images
- The LONI Pipeline is able to exploit WSDL services



General overview of the framework

- **FrontEnd:**
  - REST-FUL and Soap Web service
    - Apache TomCat
    - DBMS: MySql 5
    - Framework Jersey
    - Framework Java EE 6.0
    - SDK Asynchronous operations
  - It is able to deal with bunch operations (Submit&Check Status)
- Username&Password based security
- **BACKEND** written in JAVA (Multithread)
  - Reads DB, submits and executes jobs
  - At the moment we support:
    - PBS, EGI/IGI grid infrastructure, dedicated servers

Web service Logical Design



General overview of the framework

- Each call to the web service is intended to ask for an execution of a well specified application:
  - Only supported applications (and well known to the service provider) could be executed
  - Supporting a new application is usually few days of works from the service provider point of view
  - Most of the application only requires one or few input files
    - The user can request a run, by choosing the name of the application and the name (and location) of the input files
    - You can also use a external file available through http, ftp, etc.
  - When needed the user could change also parameters used in the command line

- The output of the runs at the end will be available (also to other services) via http link

Describing the application

- Each application is described by:
  - A bash script that prepare the environment and run the real application
    - Hidden to the final user
  - A set parameters
    - Input location and file name
    - Arguments for the executable
- Returns:
  - Status
  - Output URL

Features Supported

- Requesting execution of application for:
  - Huge challenges on distributed computing infrastructure (EGI)
    - >1000 jobs && >1 month of CPU
    - Response time: few days

READY-TO-GO RESOURCES

  - Hundreds of parallel executions on a local farm (INFN-Bari--ReCaS)
    - Few hundreds-thousand of jobs
    - Response time: from few minutes to few hours

READY-TO-GO RESOURCES

  - Single fast execution per real time analysis
    - ~10 concurrent execution
    - Response time: ~ 5-10 seconds

READY-TO-GO RESOURCES

- Each of the application/service is already configured to run on a specific infrastructure

Describing the Job Submission Tool

Describing the Job Submission Tool

- Job Submission Tool

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - The TaskListDB is then used to control the assignment of tasks to the jobs and to monitor the jobs execution

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - 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/workflow

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - 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/workflow
  - Job: it is the process executed on the grid worker nodes that takes care of a specific task execution

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - 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/workflow
  - 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)

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - 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/workflow
  - 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 status of TaskListDB: it submits new jobs as soon as new task appears

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - 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/workflow
  - 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 status of TaskListDB: it submits new jobs as soon as new task appears
  - The same job is submitted every time

Describing the Job Submission Tool

- Job Submission Tool
  - Each requested application run is inserted into a RDBMS (the TaskListDB).
  - 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/workflow
  - 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 status of TaskListDB: it submits new jobs as soon as new task appears
  - The same job is submitted every time
  - The differences is only related to the task they have to complete that is assigned only when it got executed

Job Submission Tool Features

Job Submission Tool 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

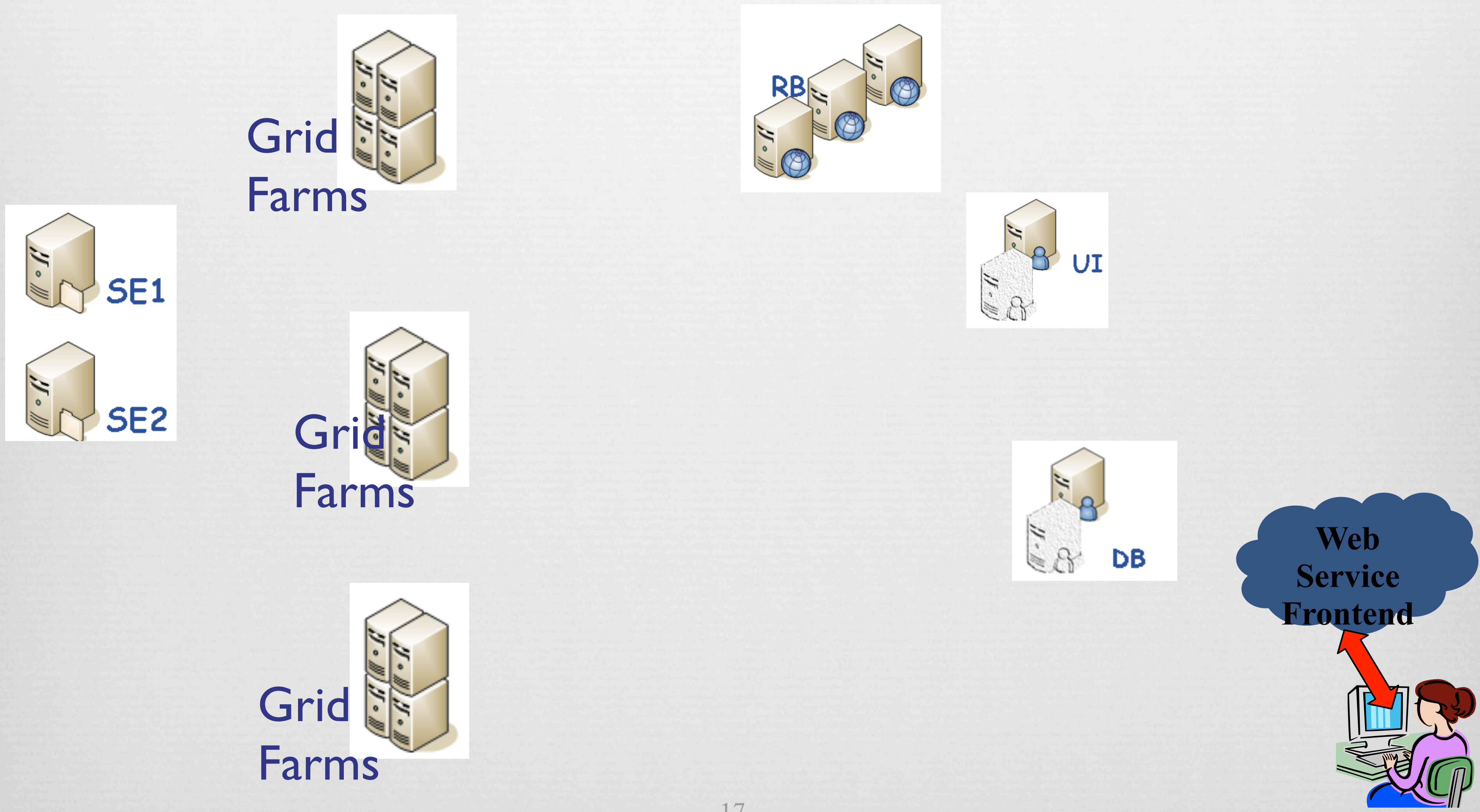
Job Submission Tool Features

- 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.
- 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

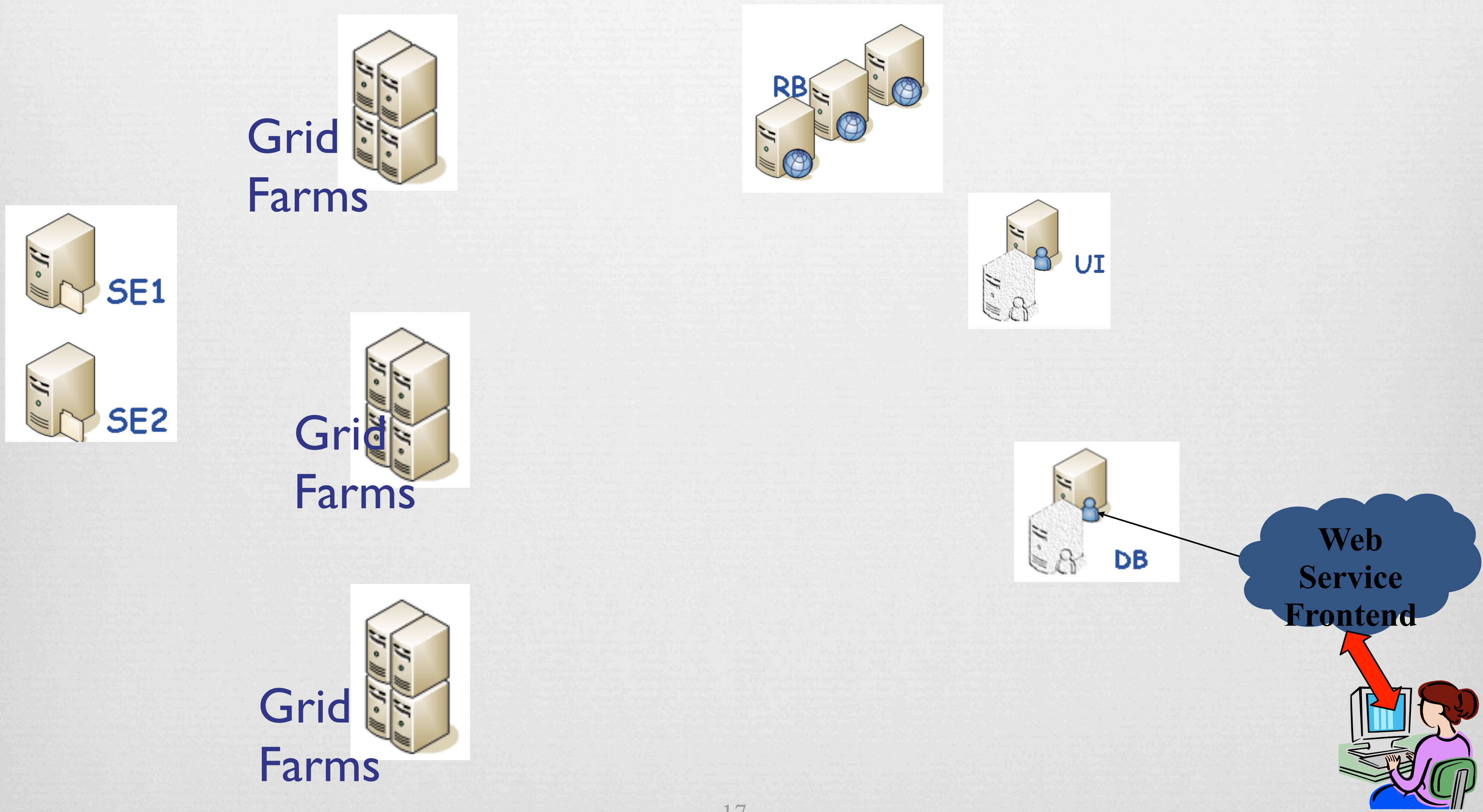
Job Submission Tool Wrapper

- Requests from the TaskListDB a tasks to be executed
- Retrieves the application executable (it has to be available with one protocol among: https, http, gridftp, 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

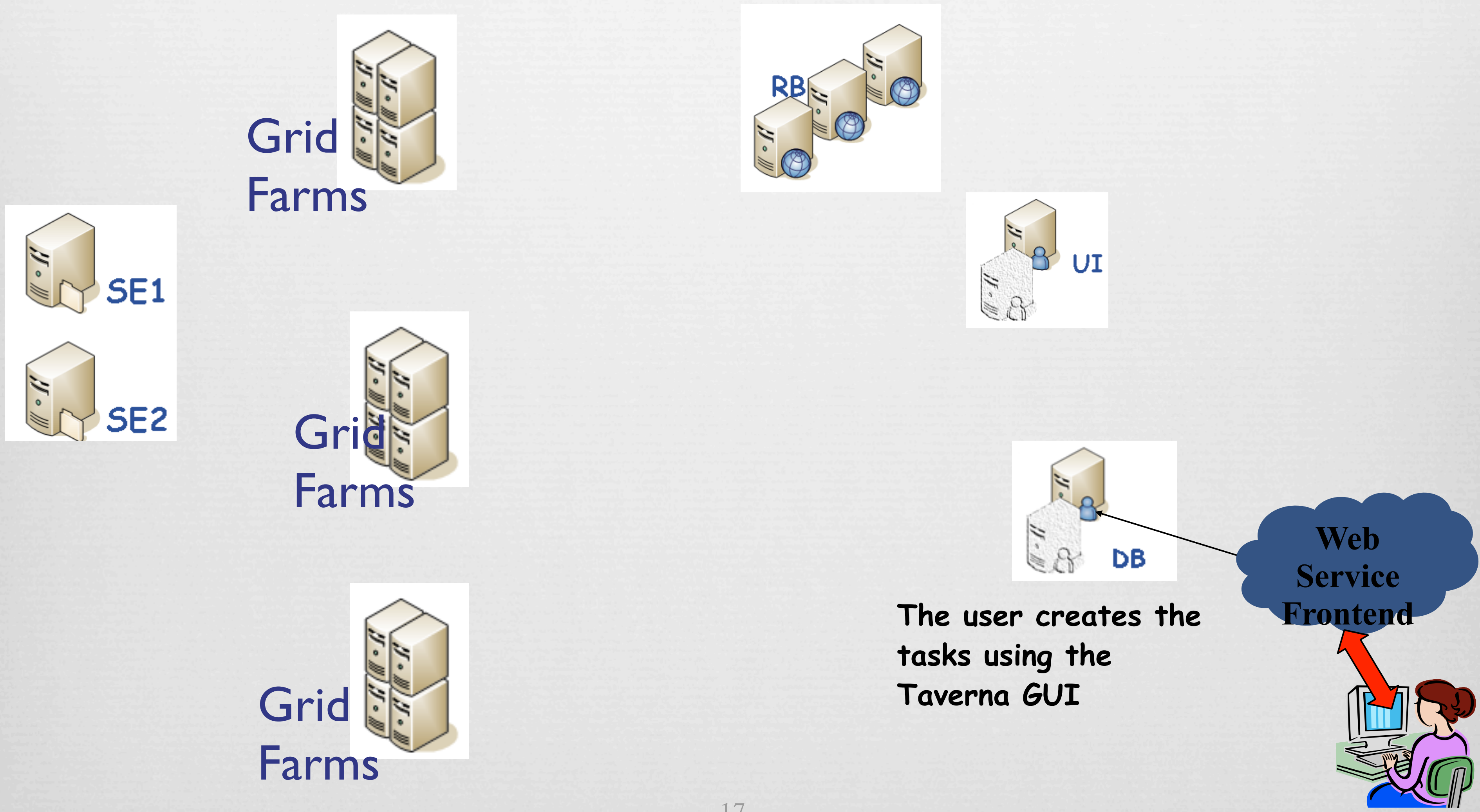
JST Detailed schema



JST Detailed schema

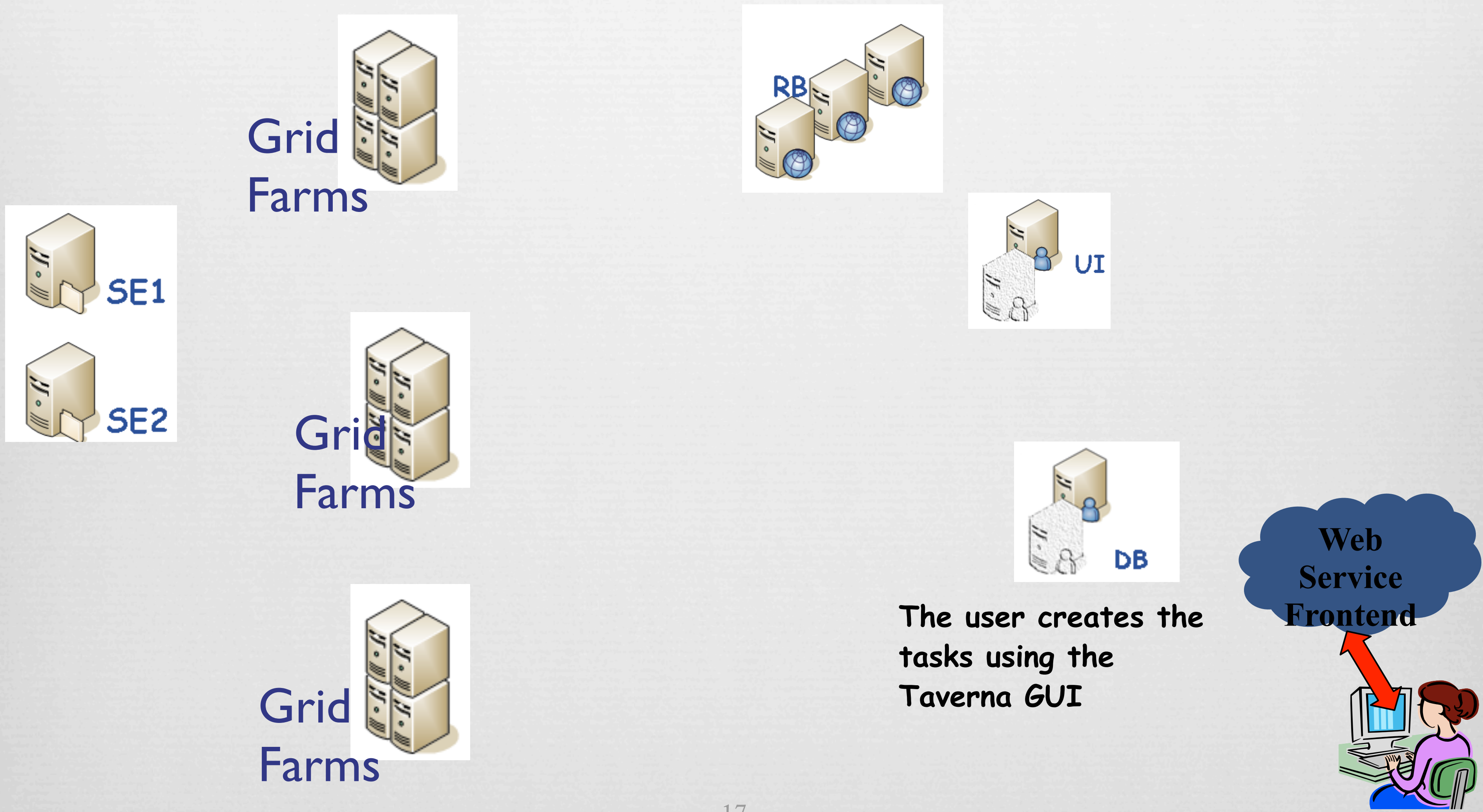


JST Detailed schema



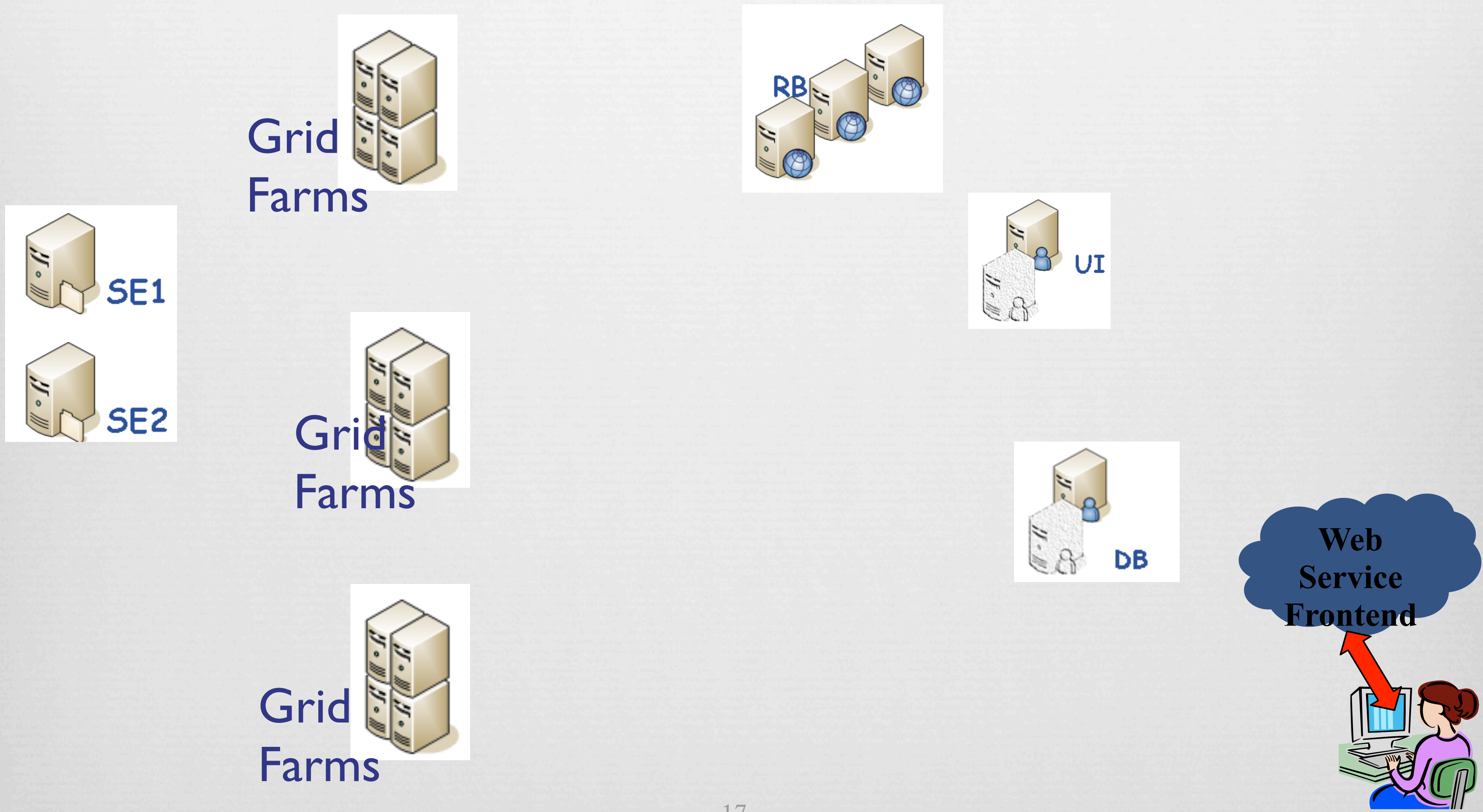
The user creates the tasks using the Taverna GUI

JST Detailed schema

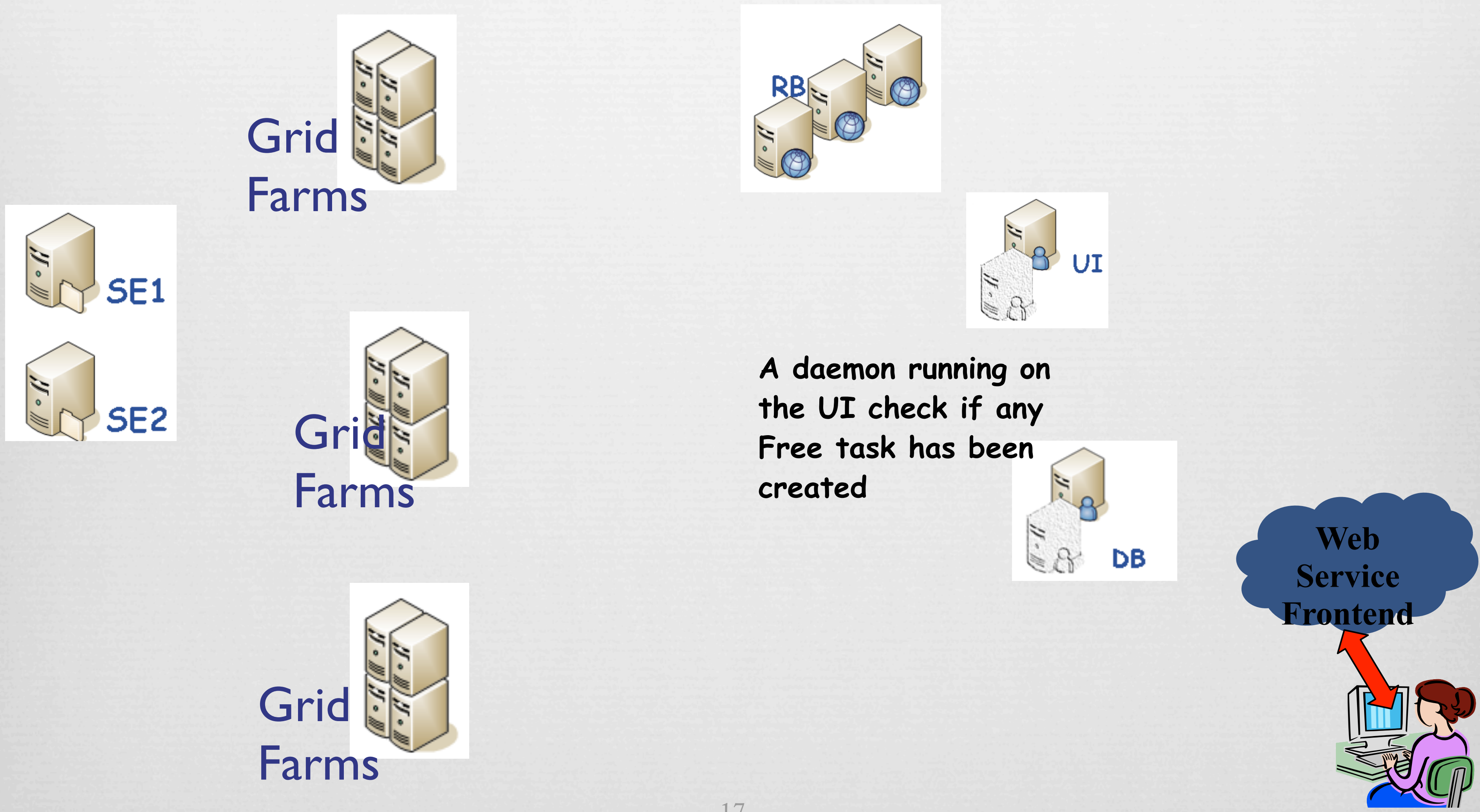


The user creates the tasks using the Taverna GUI

JST Detailed schema

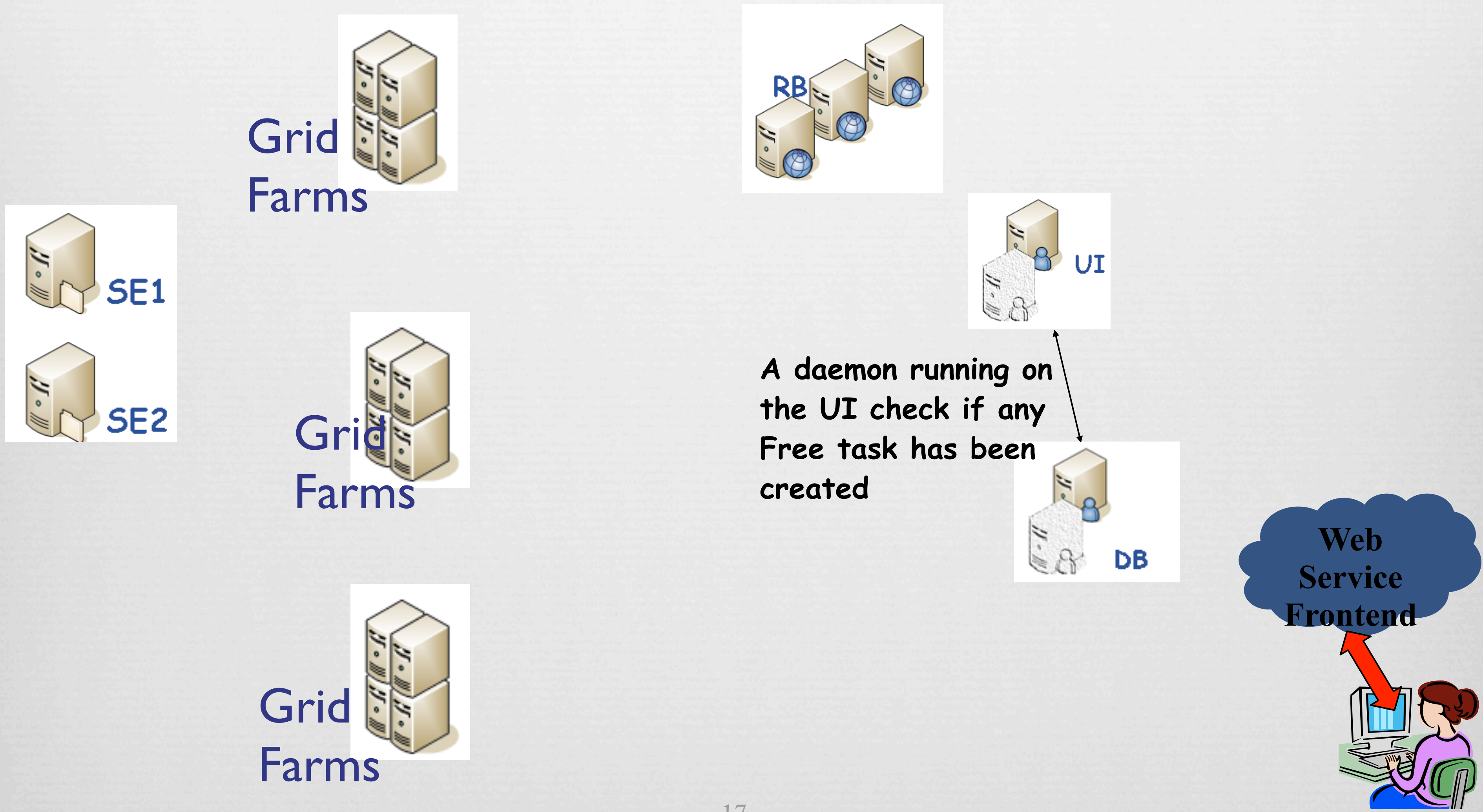


JST Detailed schema



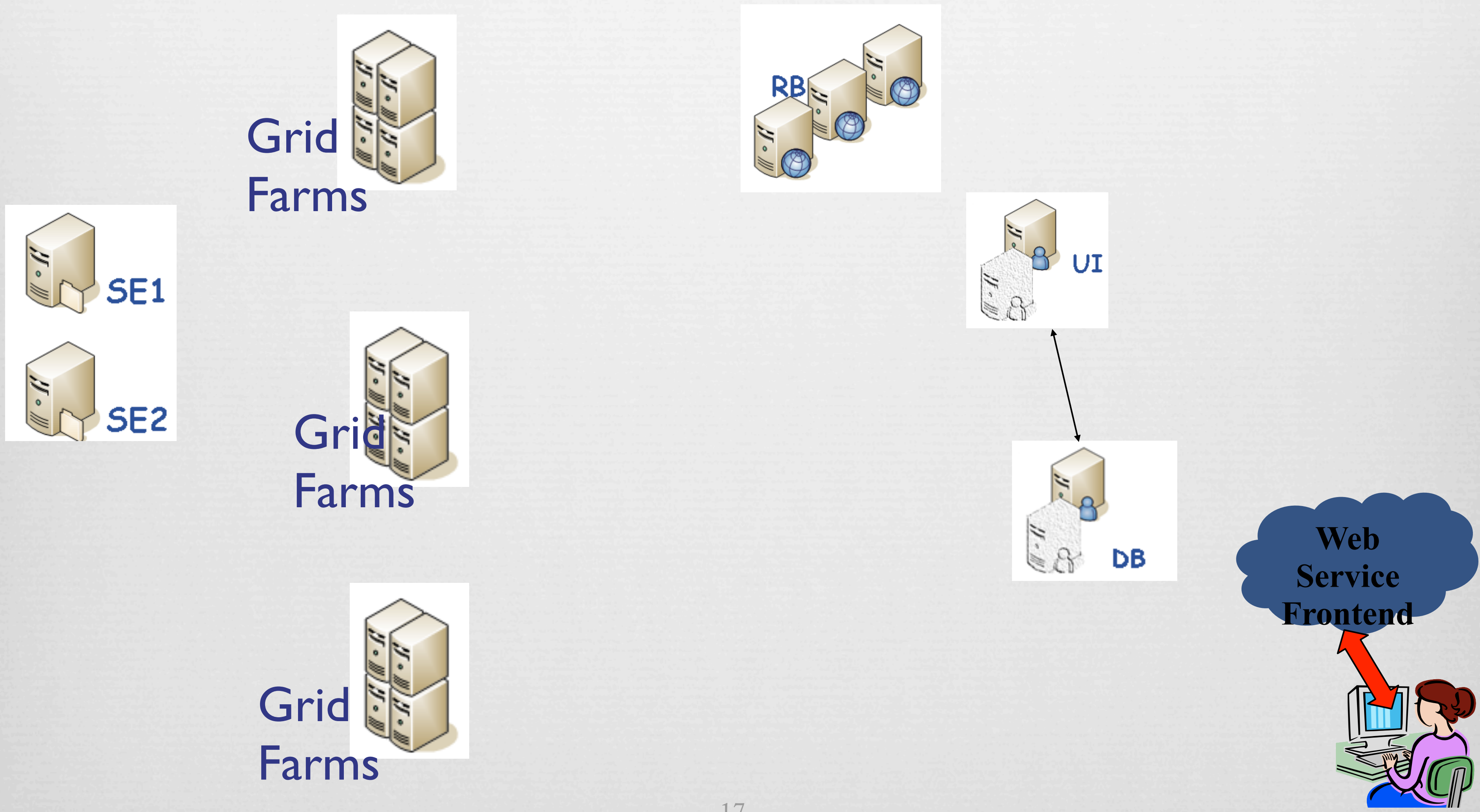
A daemon running on the UI check if any Free task has been created

JST Detailed schema

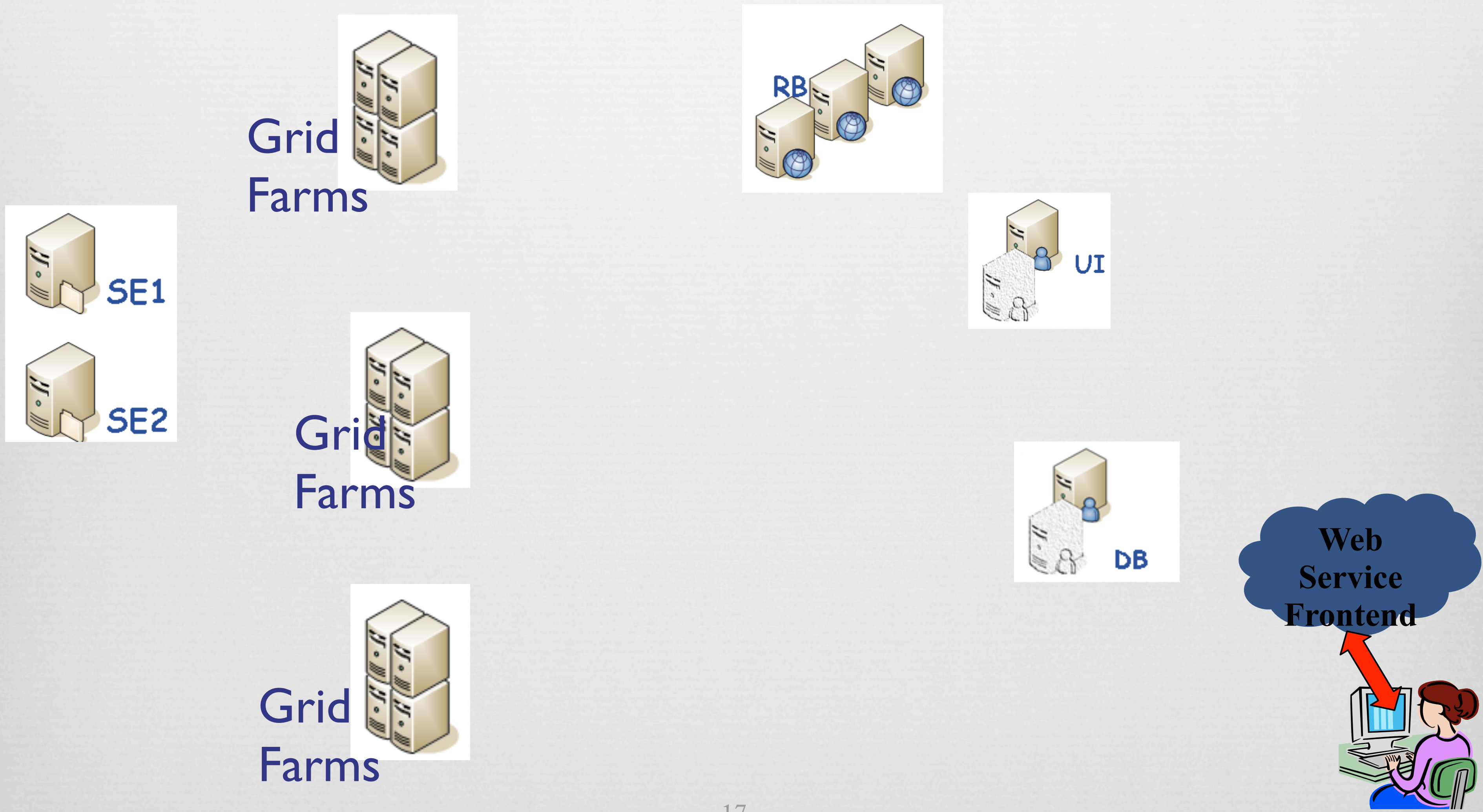


A daemon running on the UI check if any Free task has been created

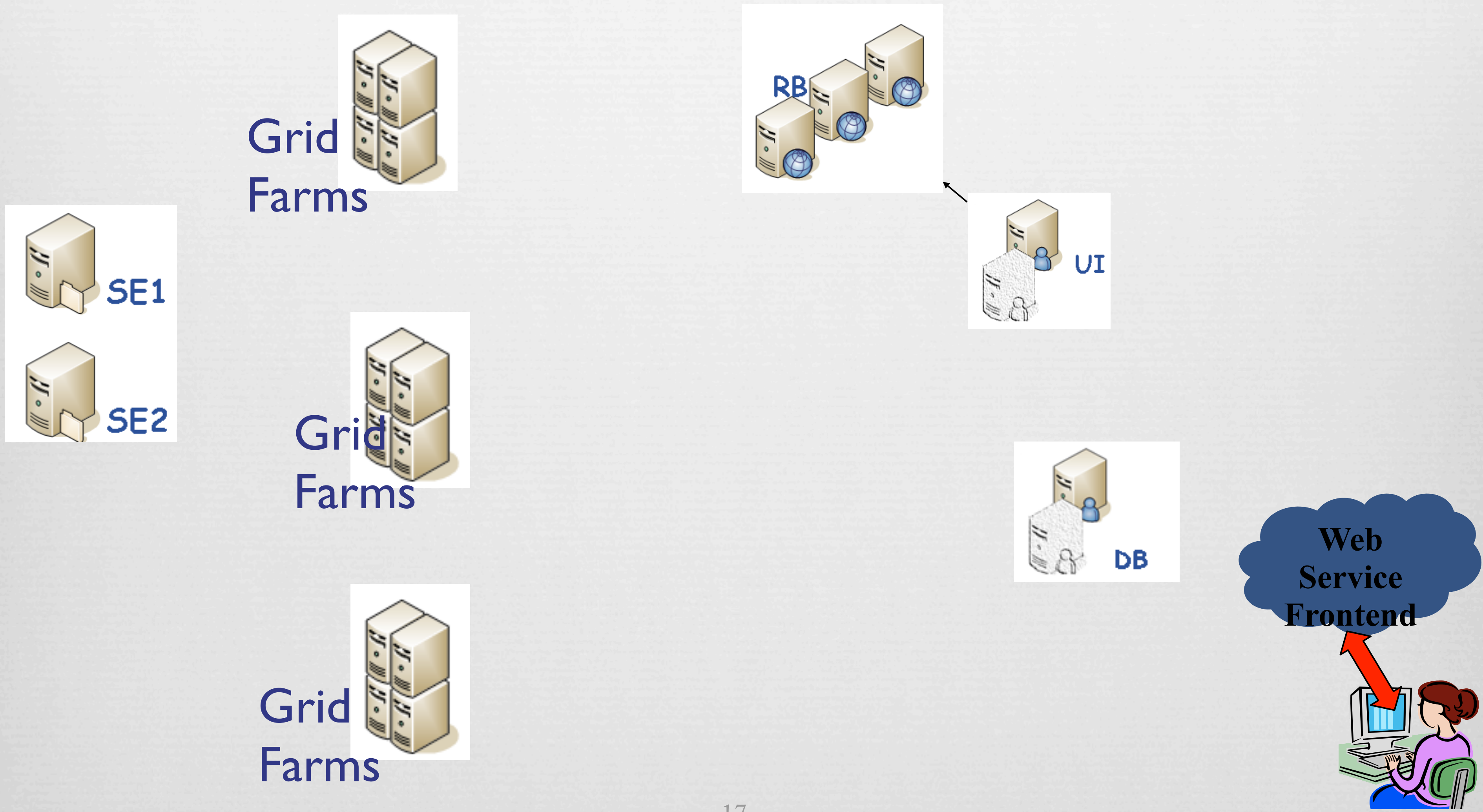
JST Detailed schema



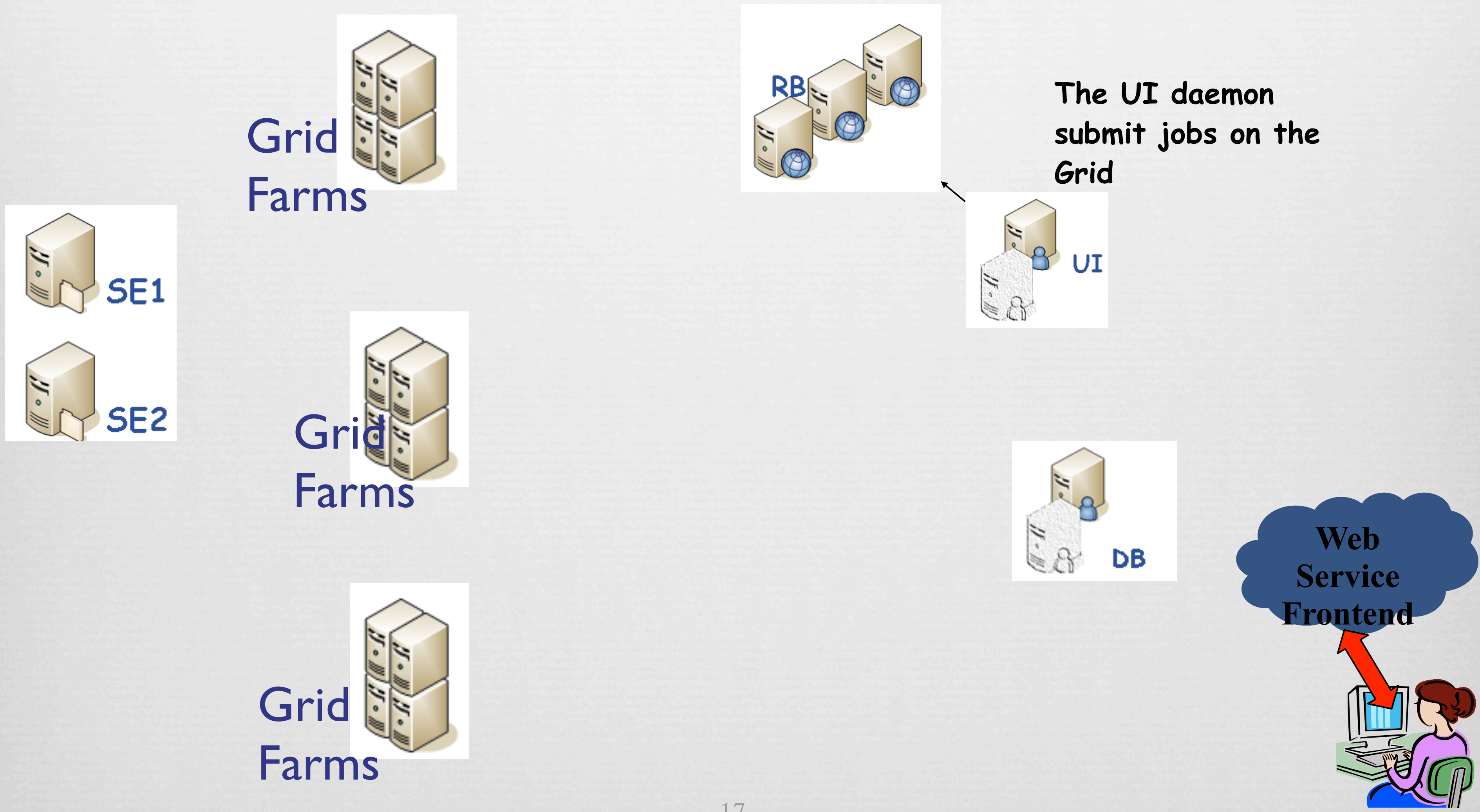
JST Detailed schema



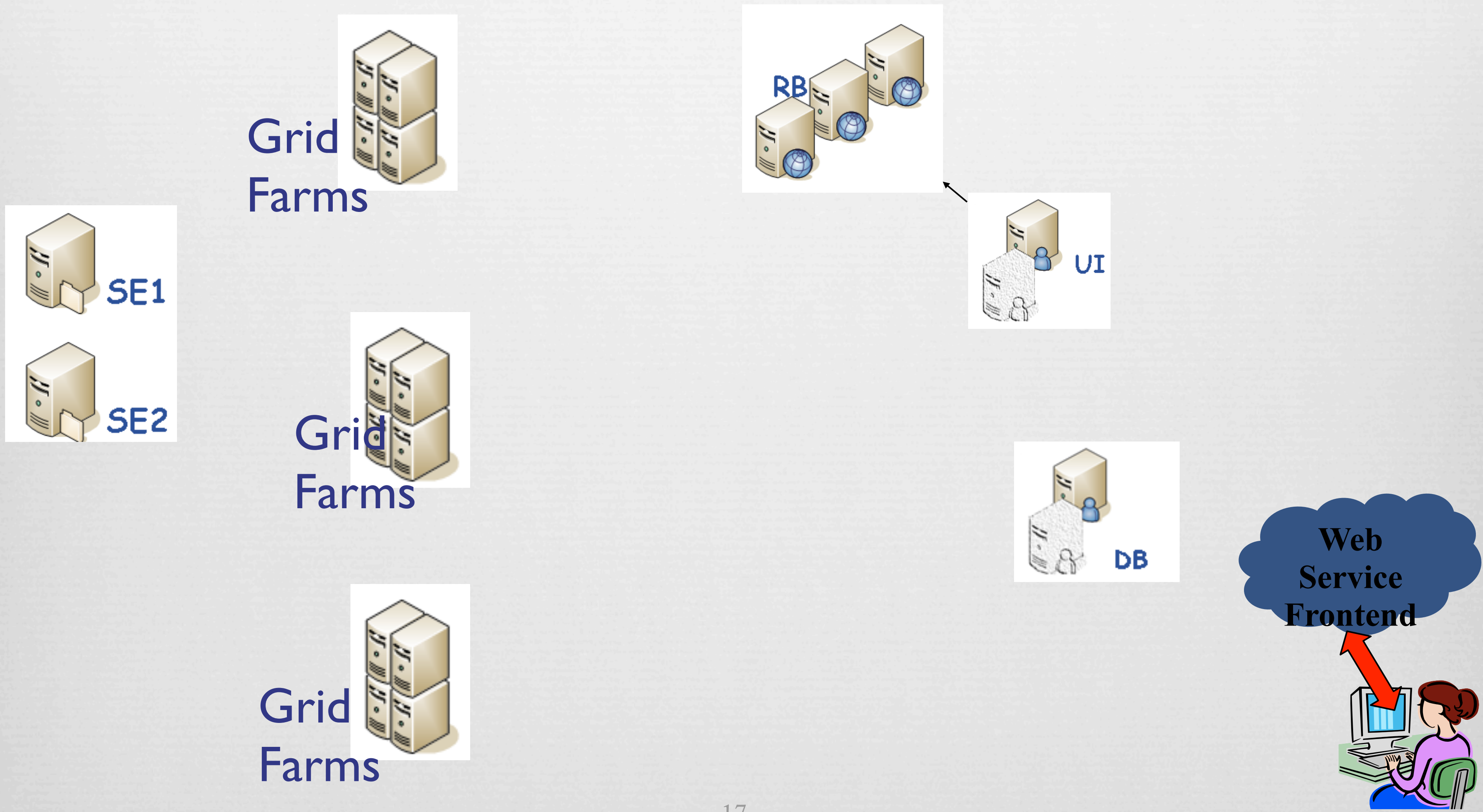
JST Detailed schema



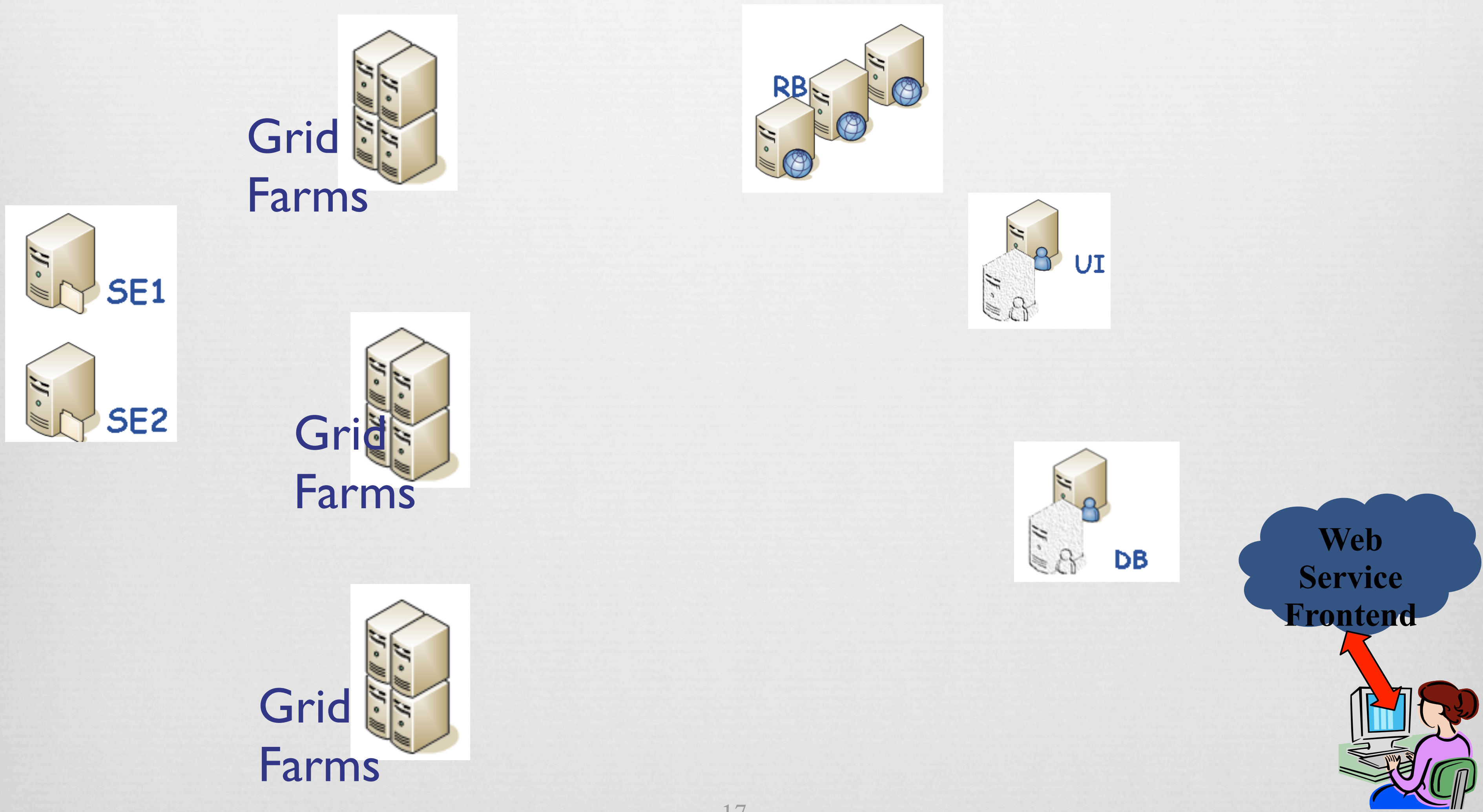
JST Detailed schema



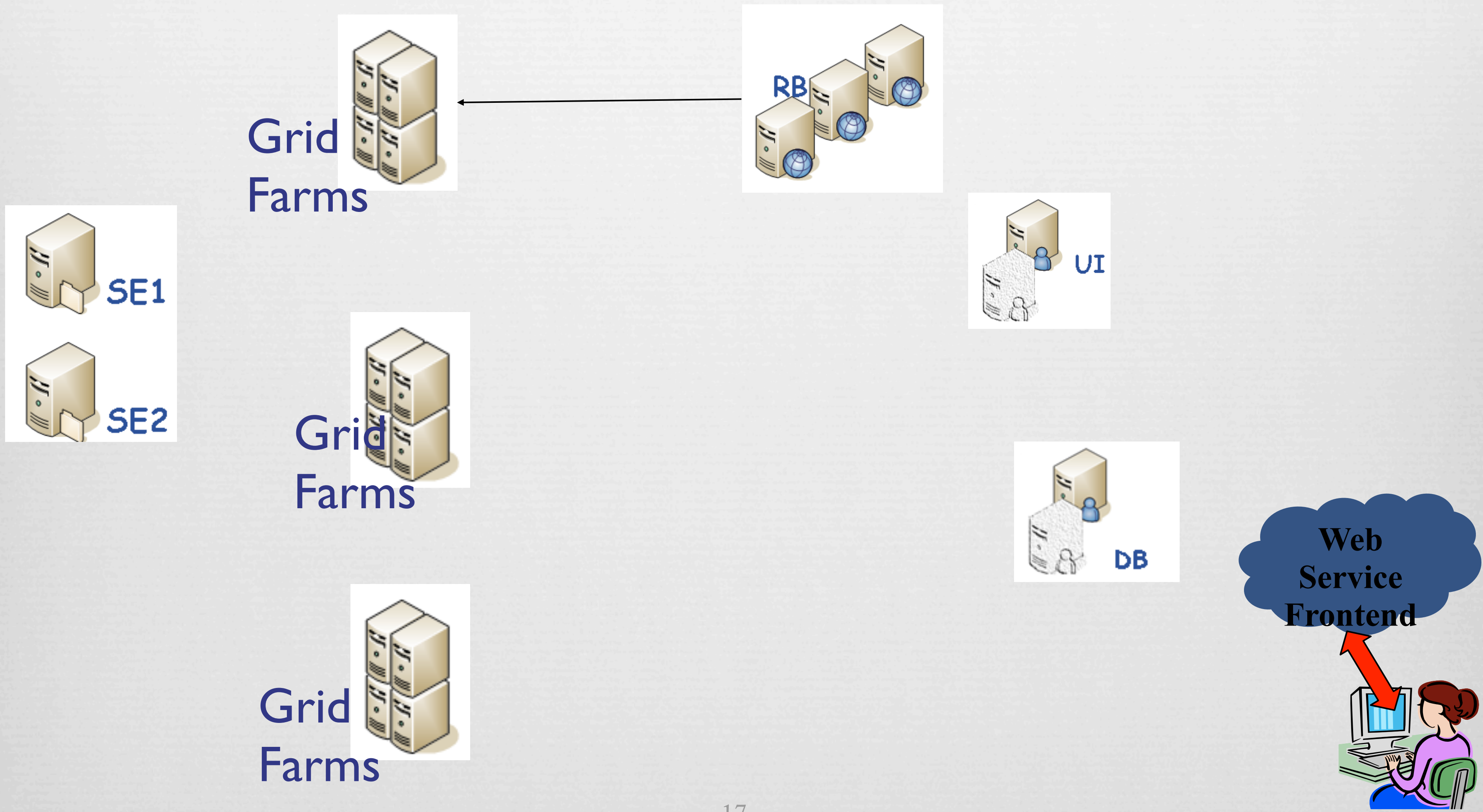
JST Detailed schema



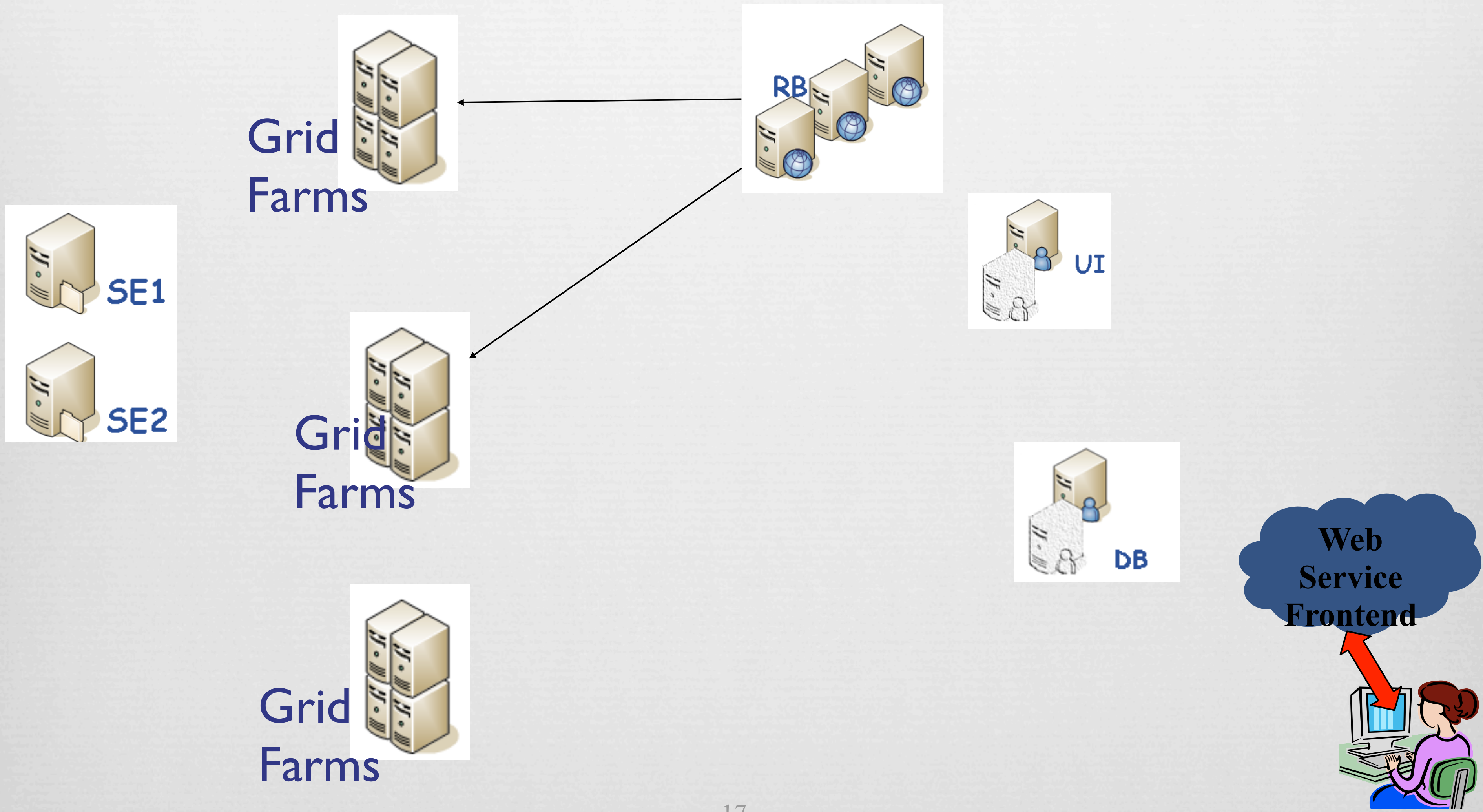
JST Detailed schema



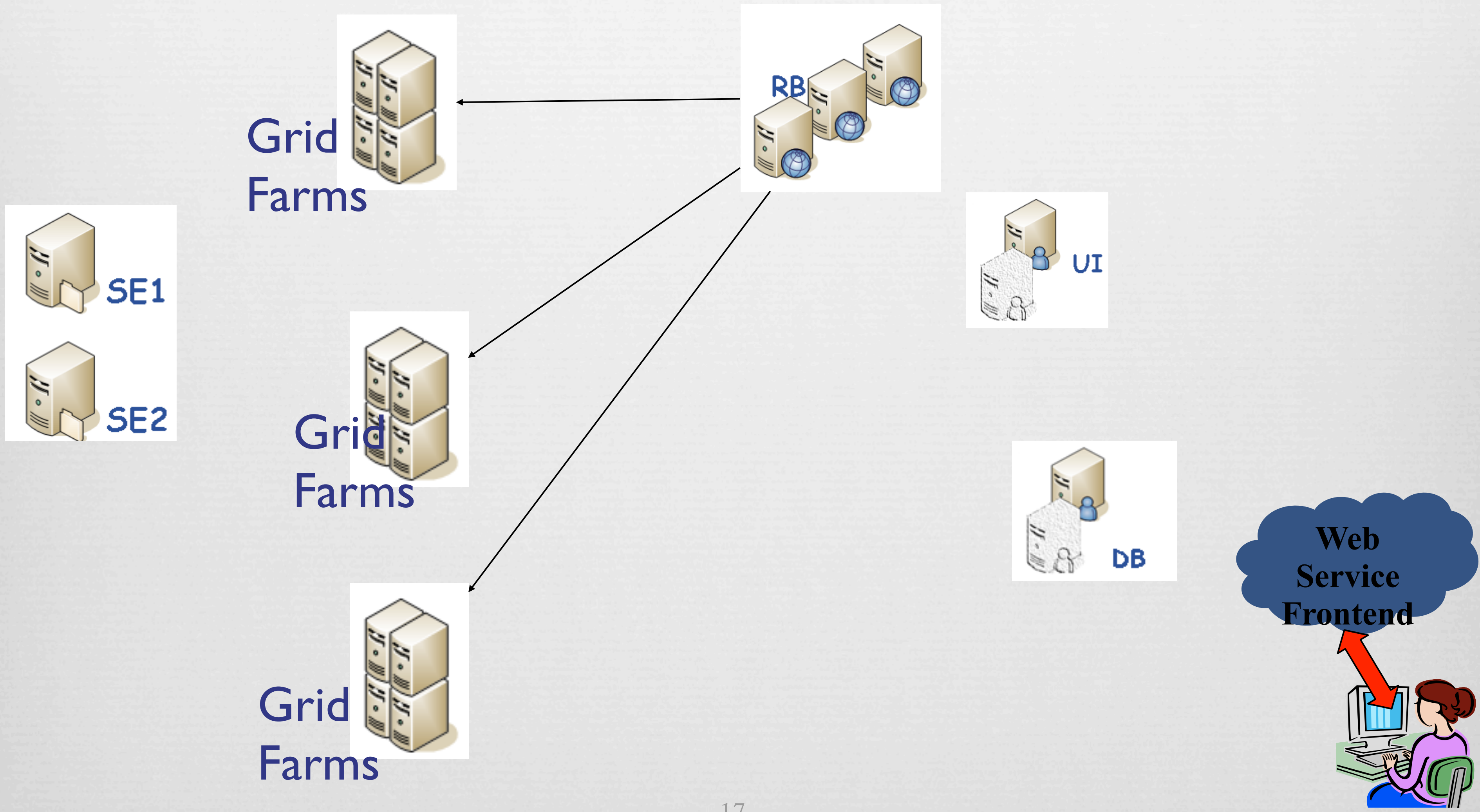
JST Detailed schema



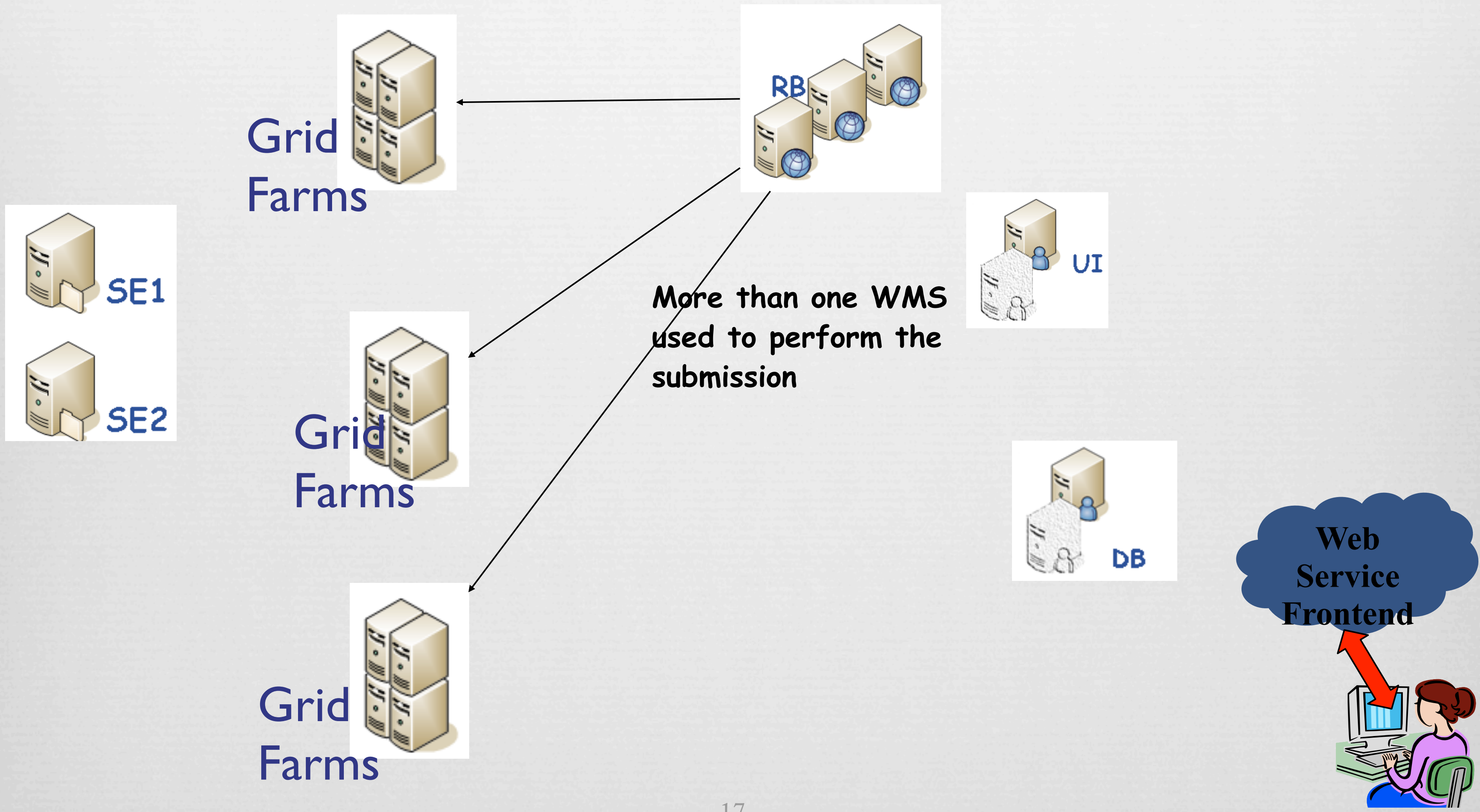
JST Detailed schema



JST Detailed schema

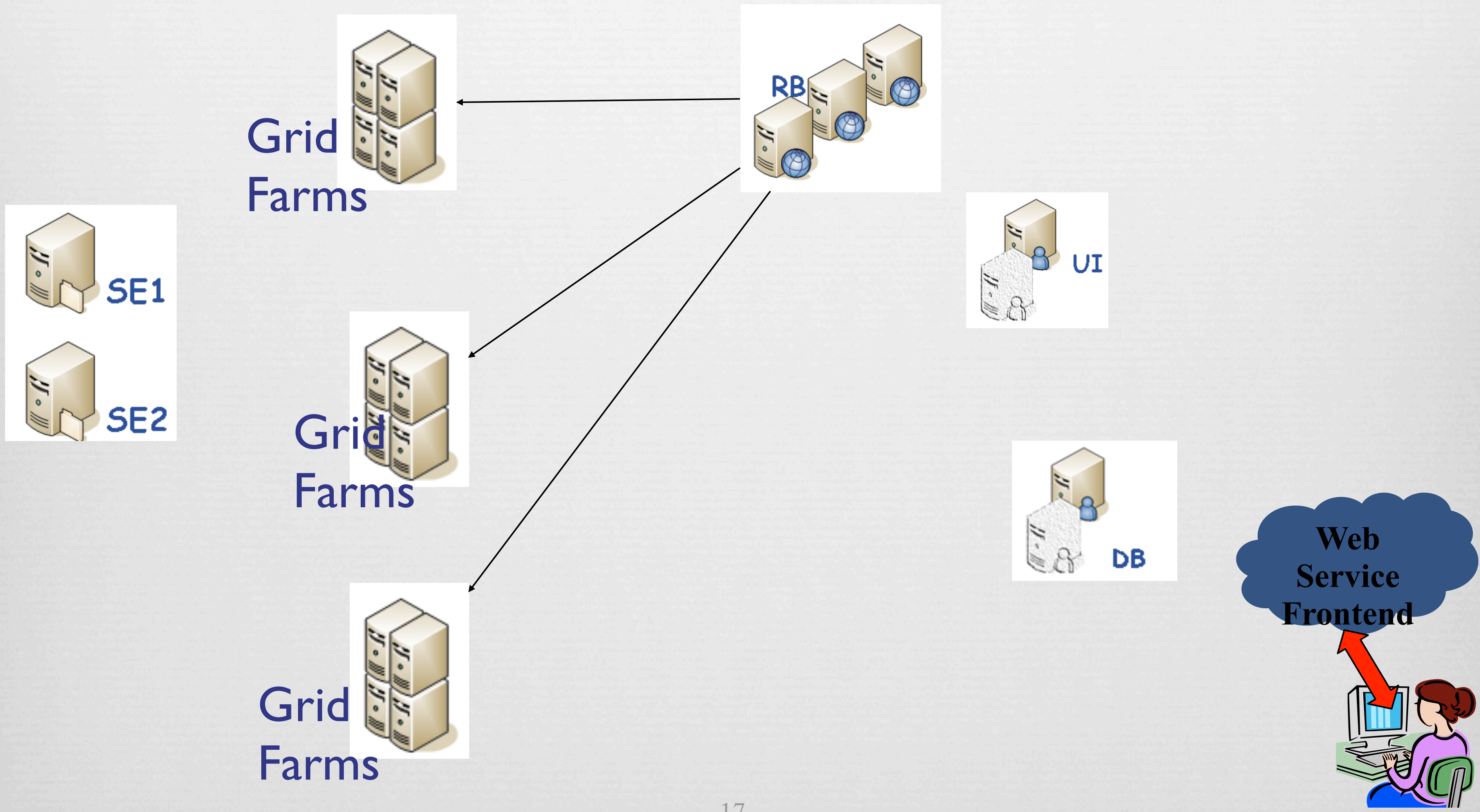


JST Detailed schema

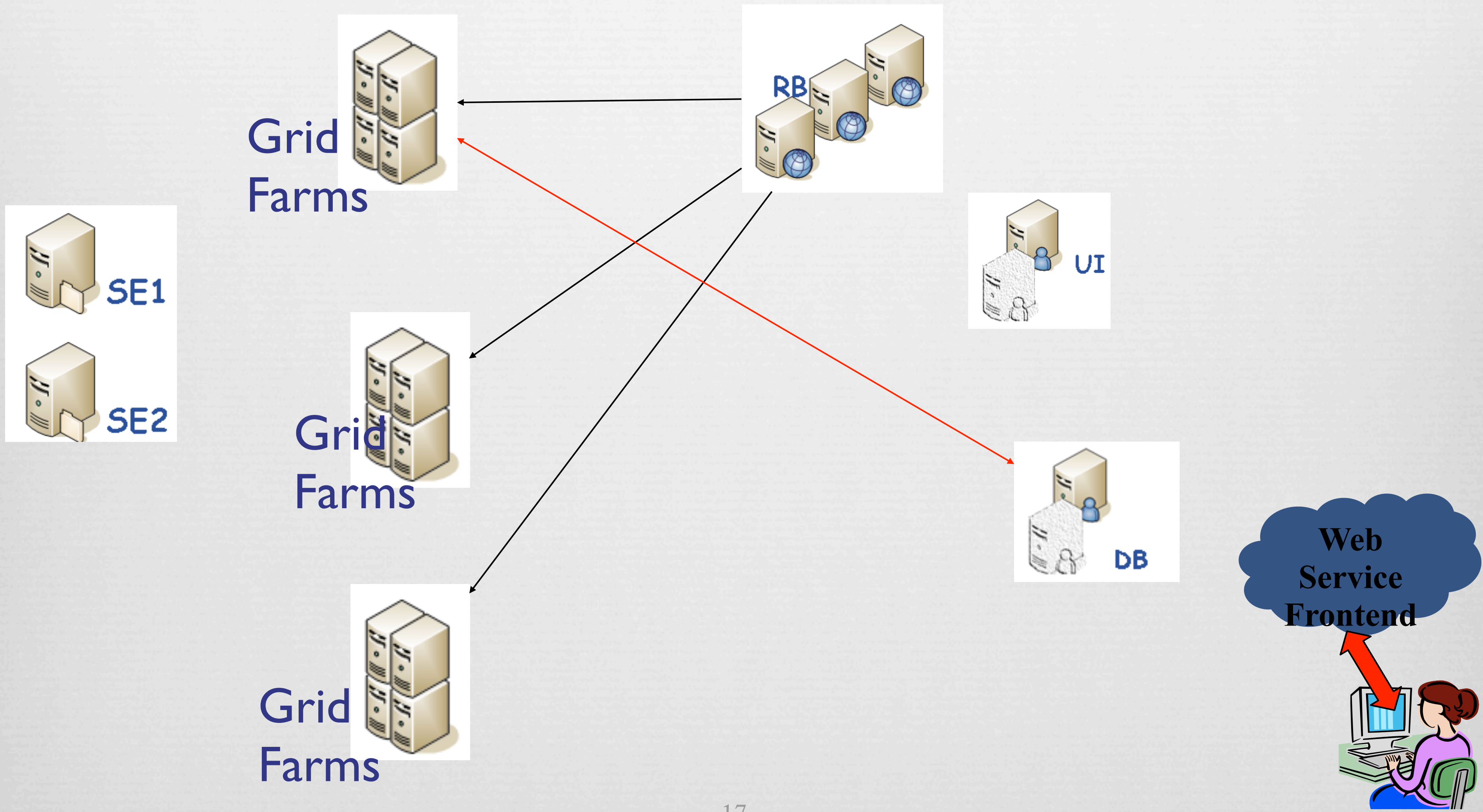


More than one WMS used to perform the submission

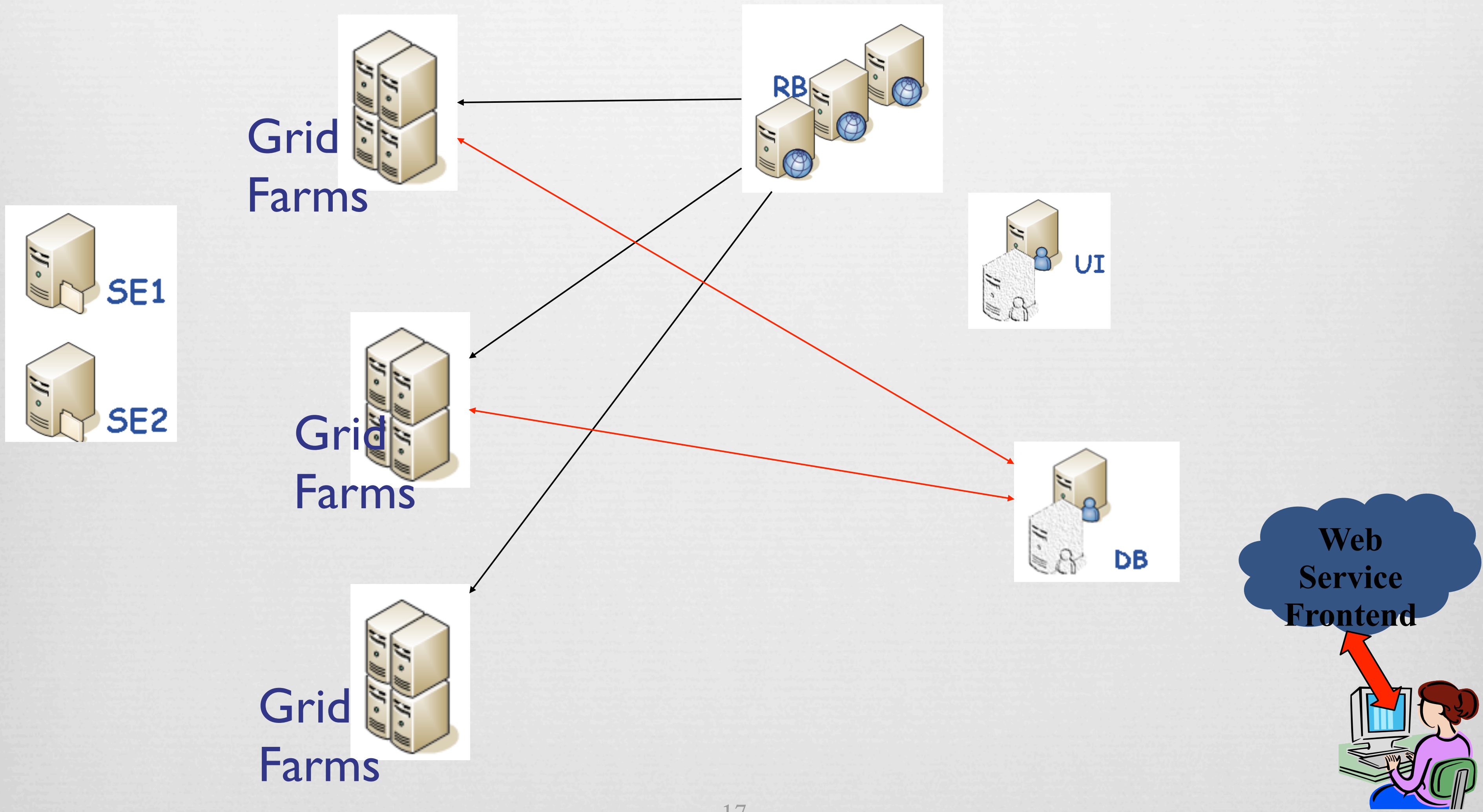
JST Detailed schema



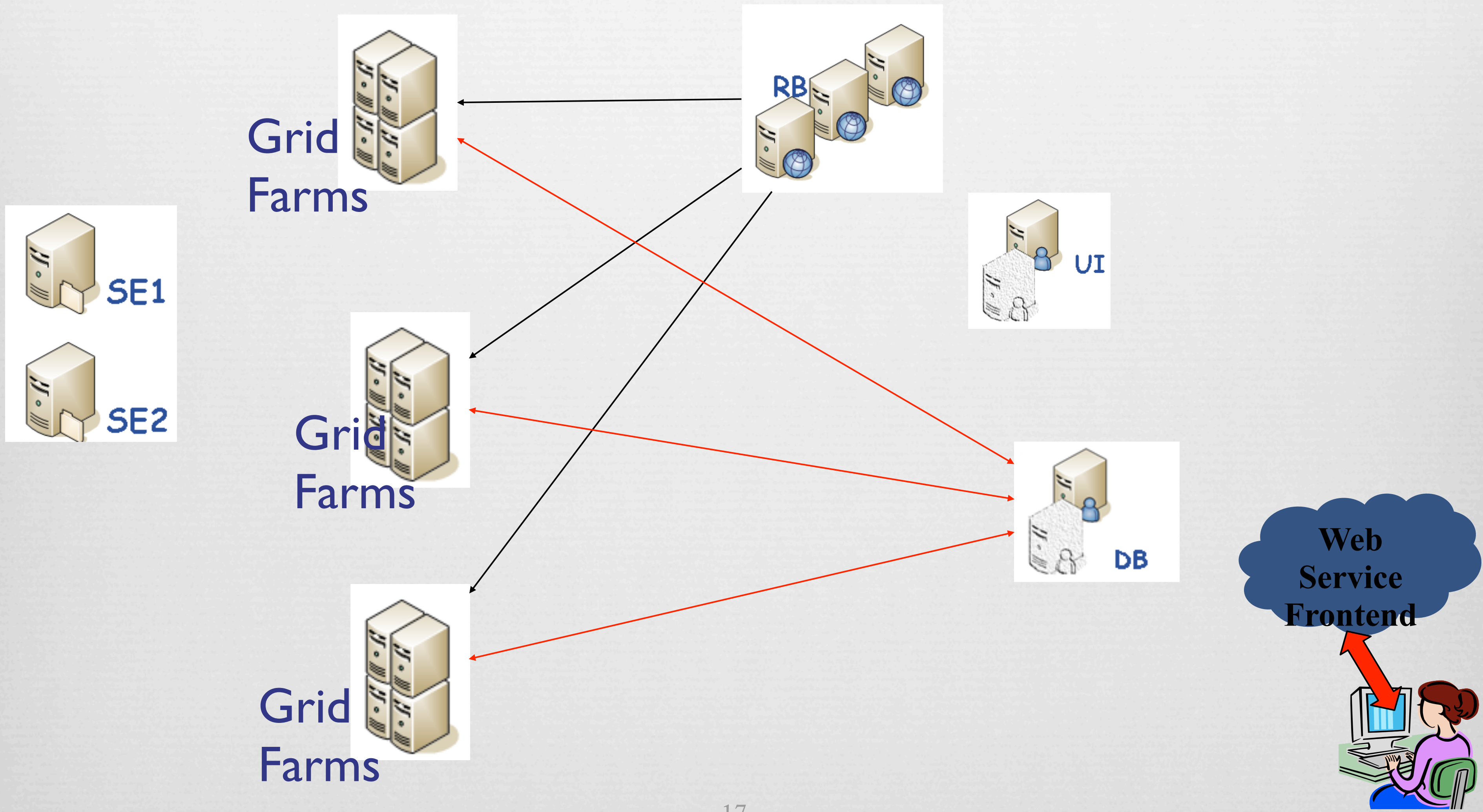
JST Detailed schema



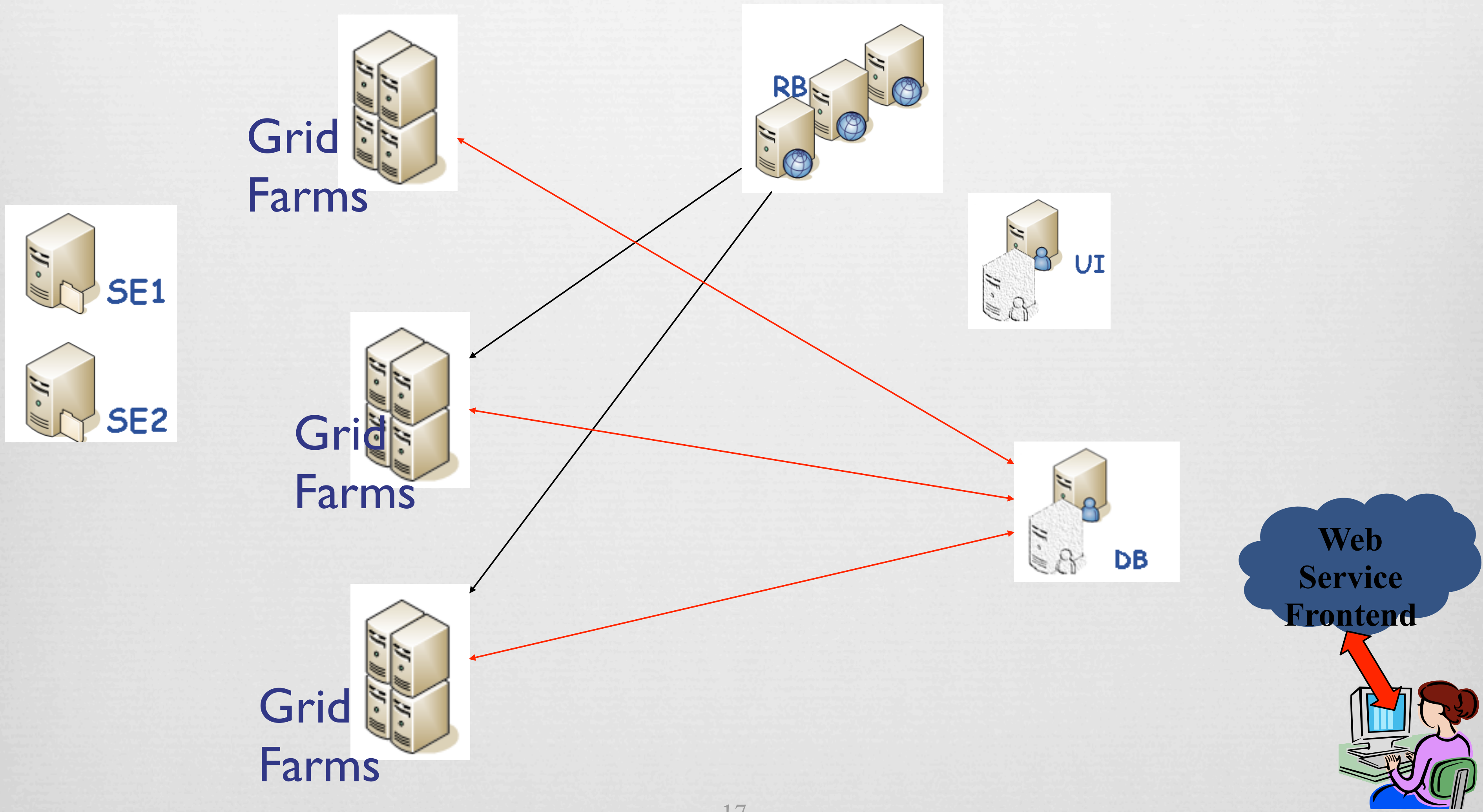
JST Detailed schema



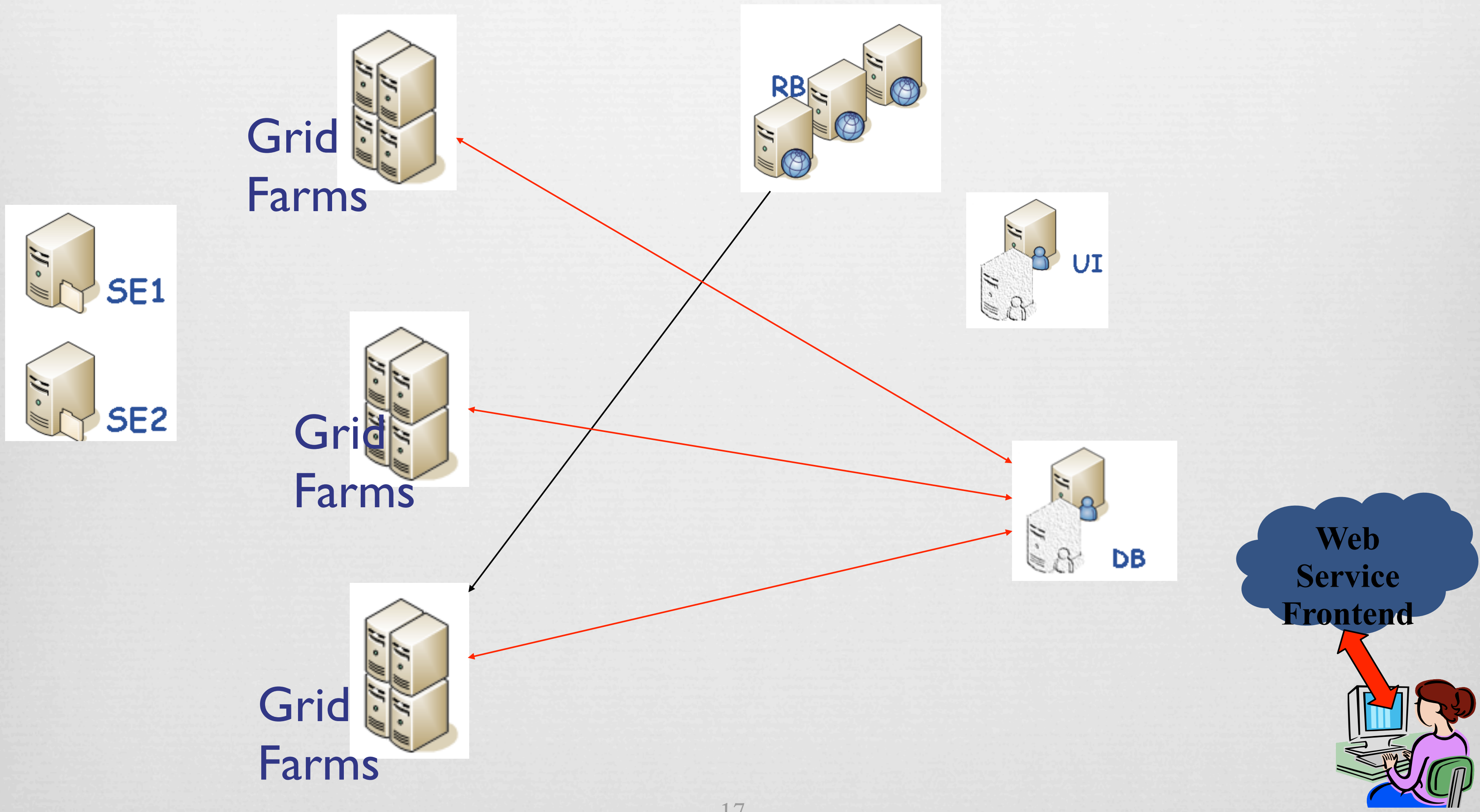
JST Detailed schema



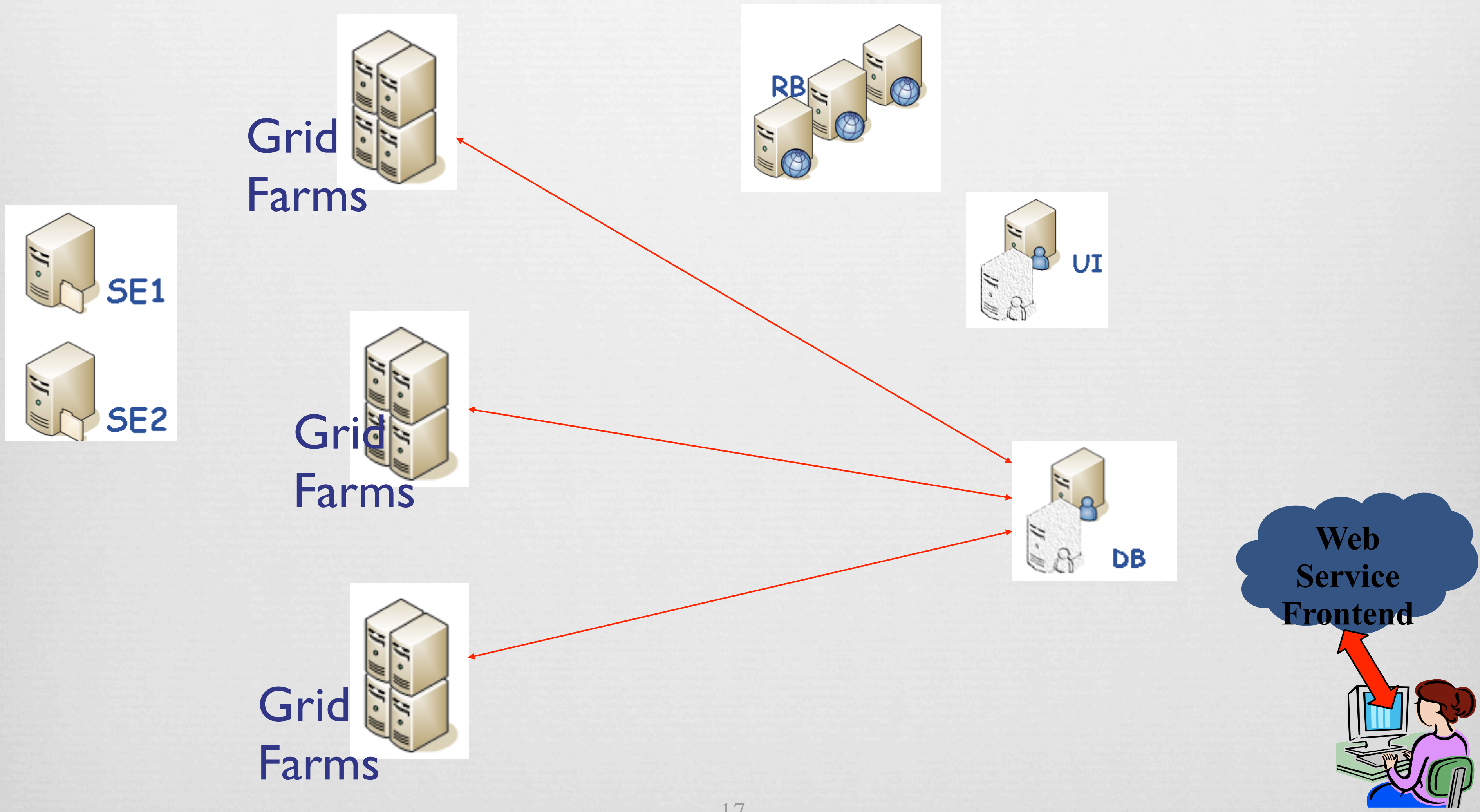
JST Detailed schema



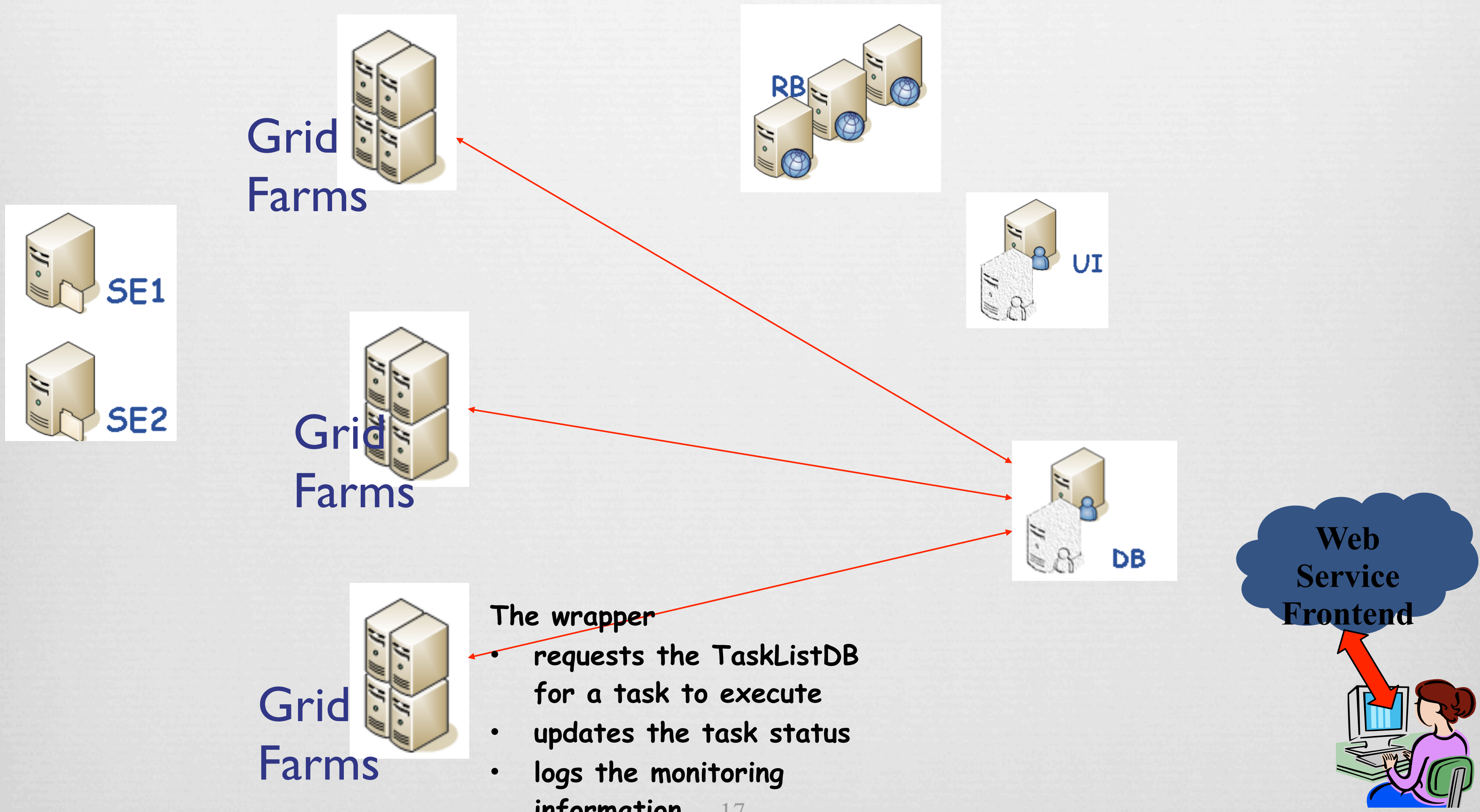
JST Detailed schema



JST Detailed schema



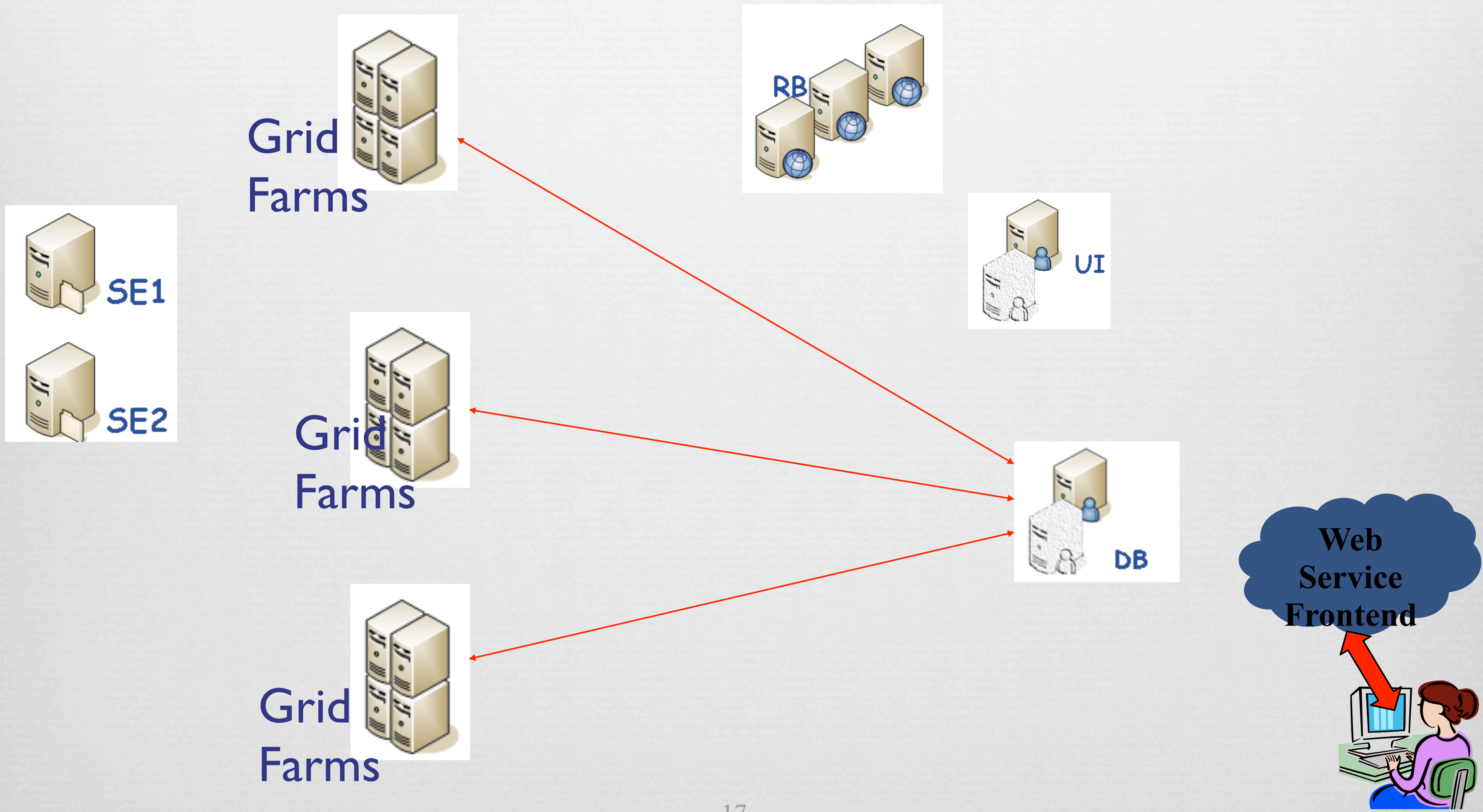
JST Detailed schema



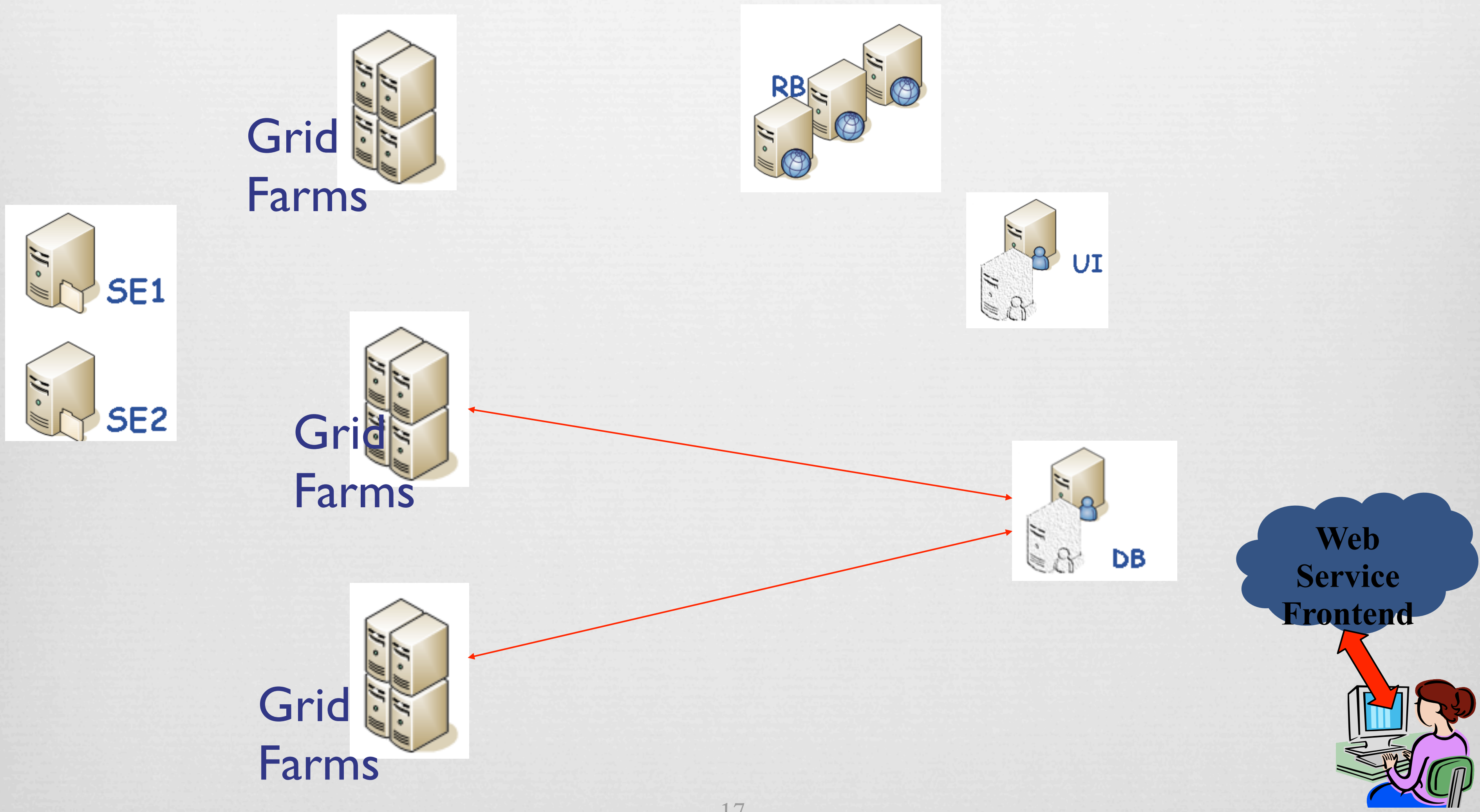
The wrapper

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

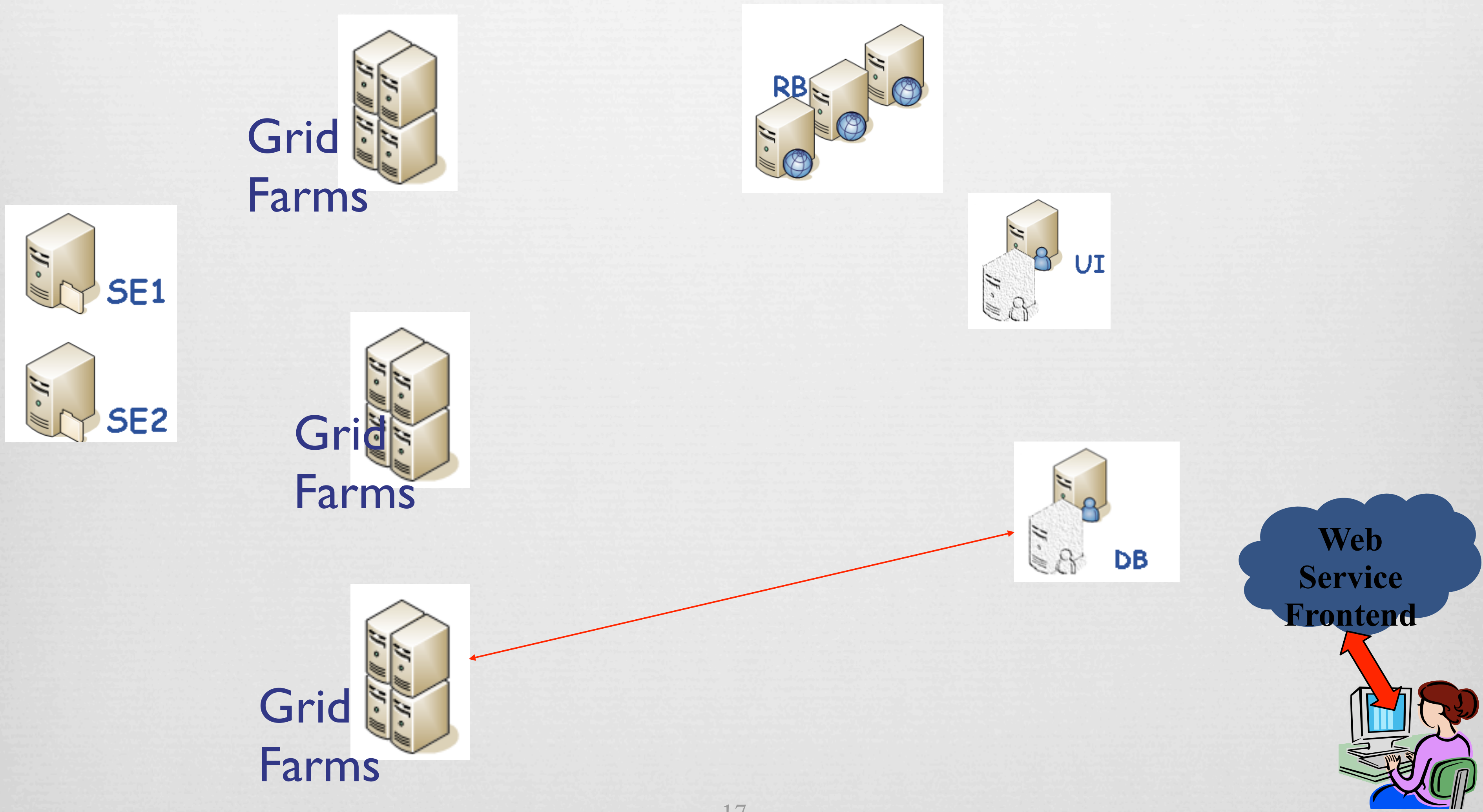
JST Detailed schema



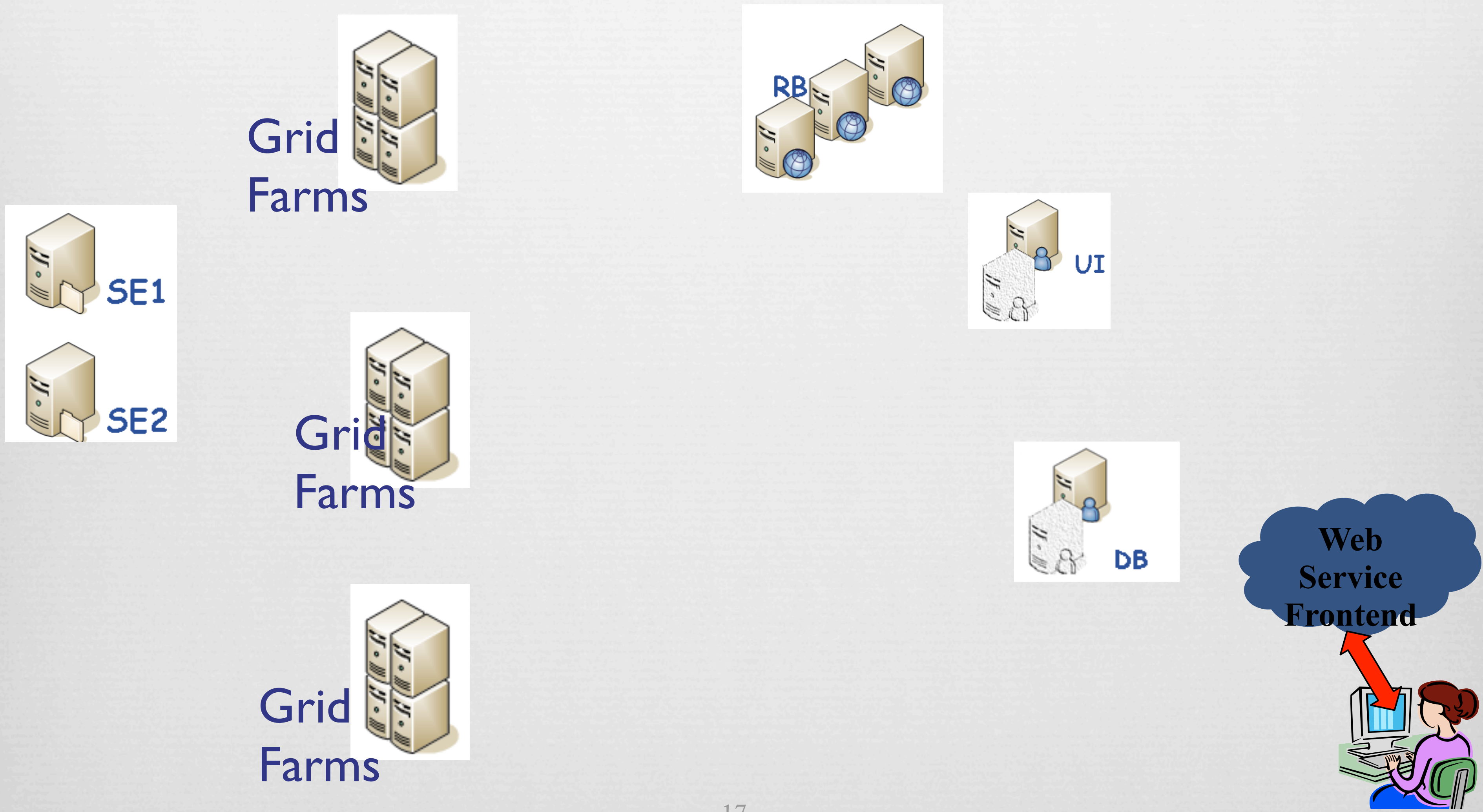
JST Detailed schema



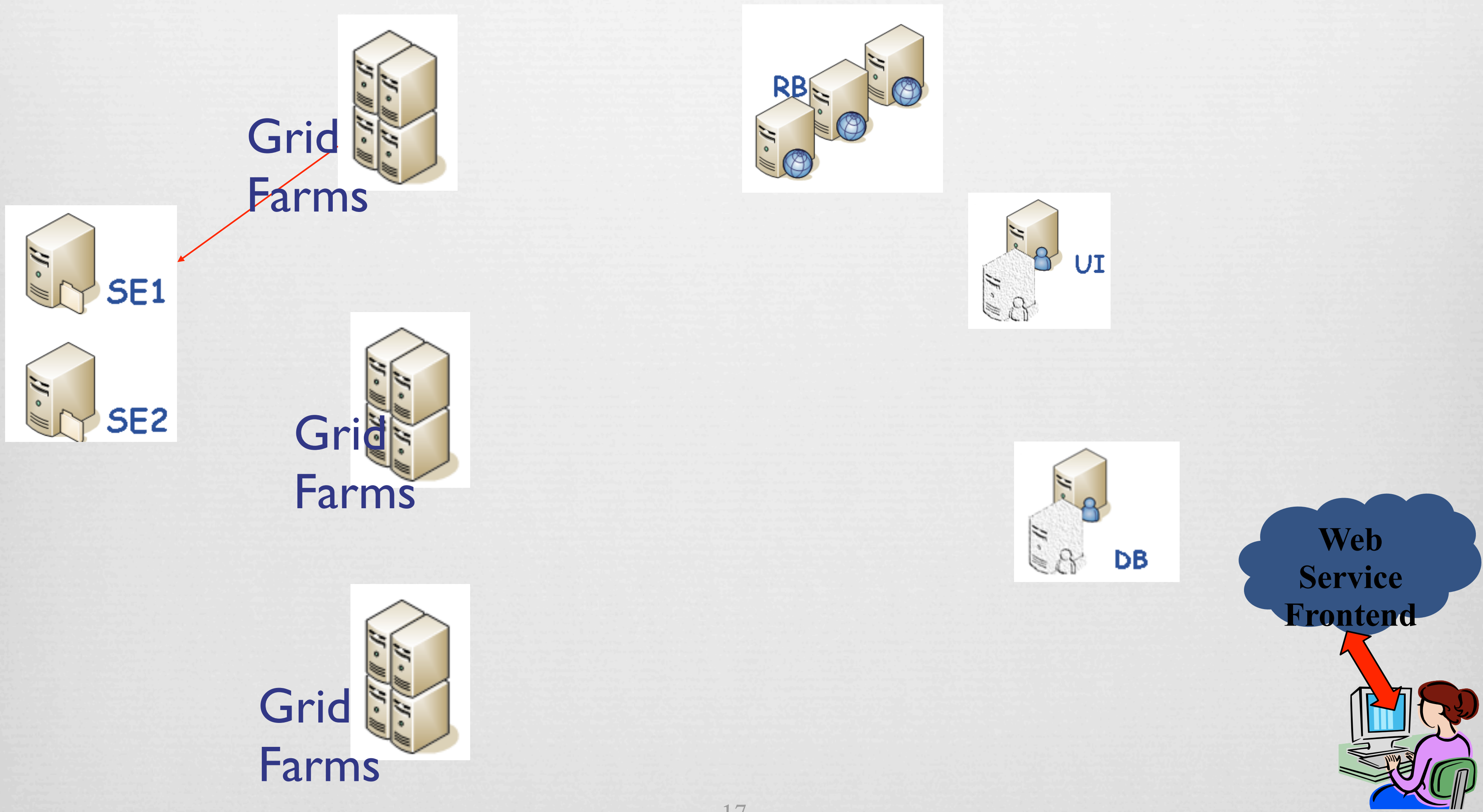
JST Detailed schema



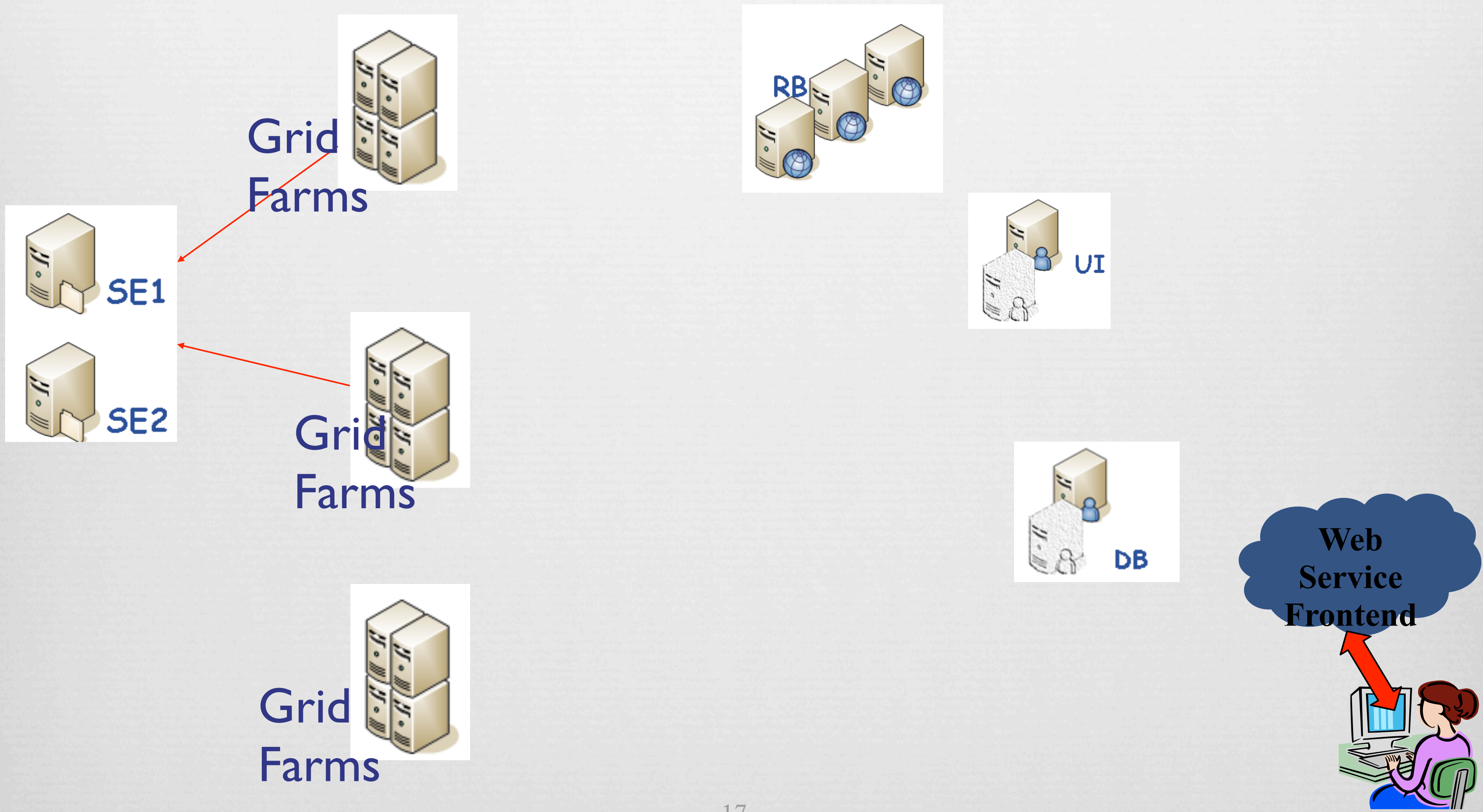
JST Detailed schema



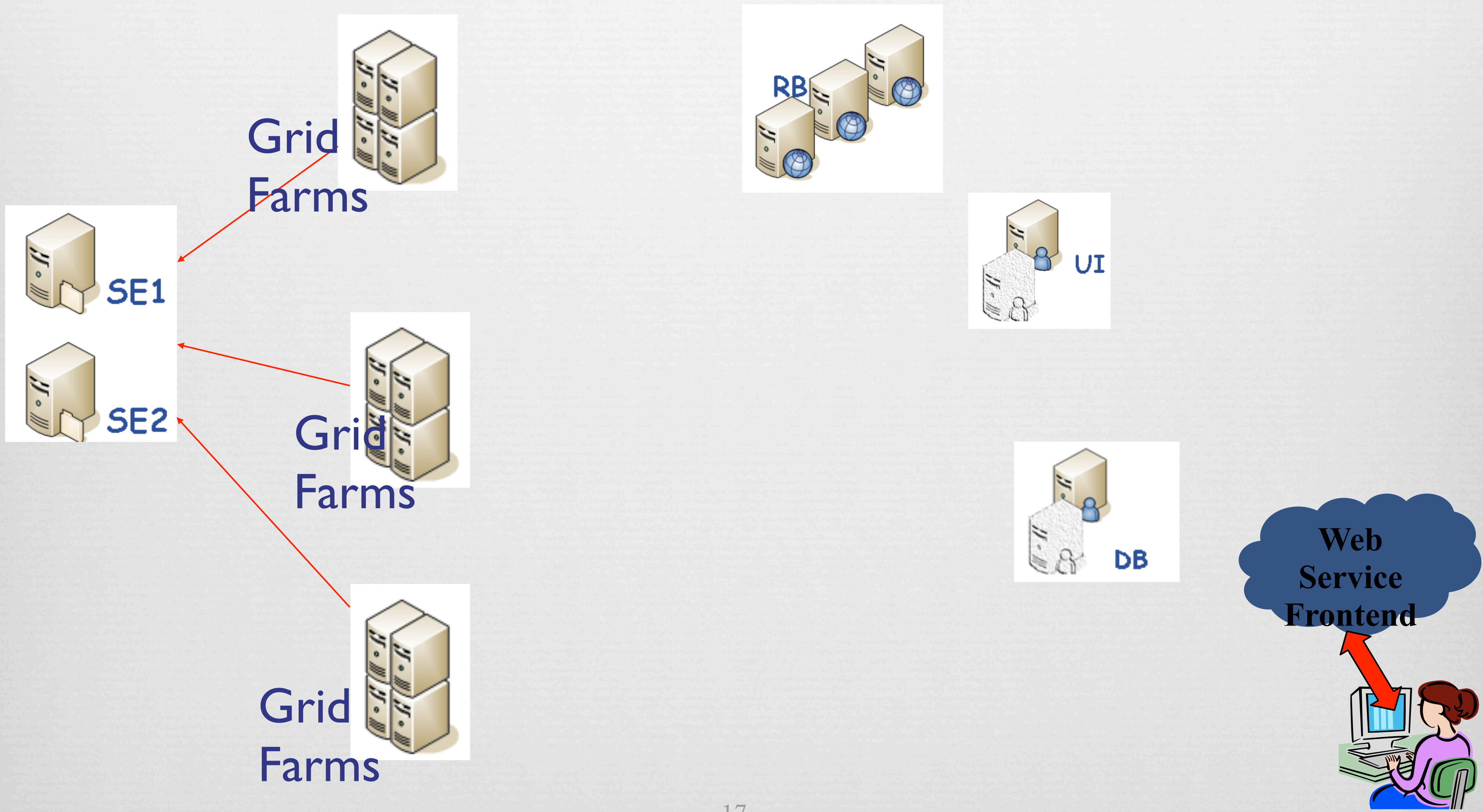
JST Detailed schema



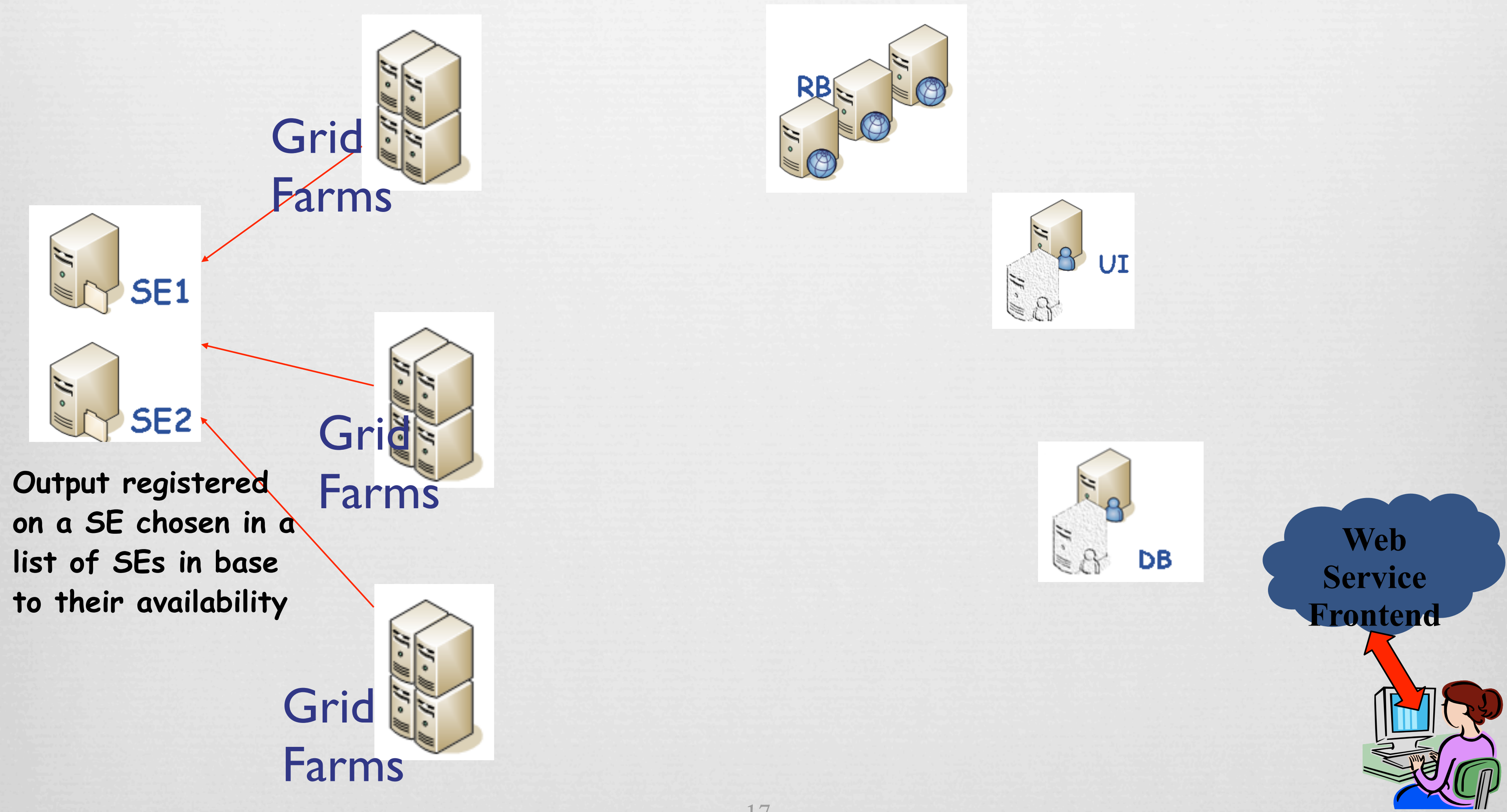
JST Detailed schema



JST Detailed schema

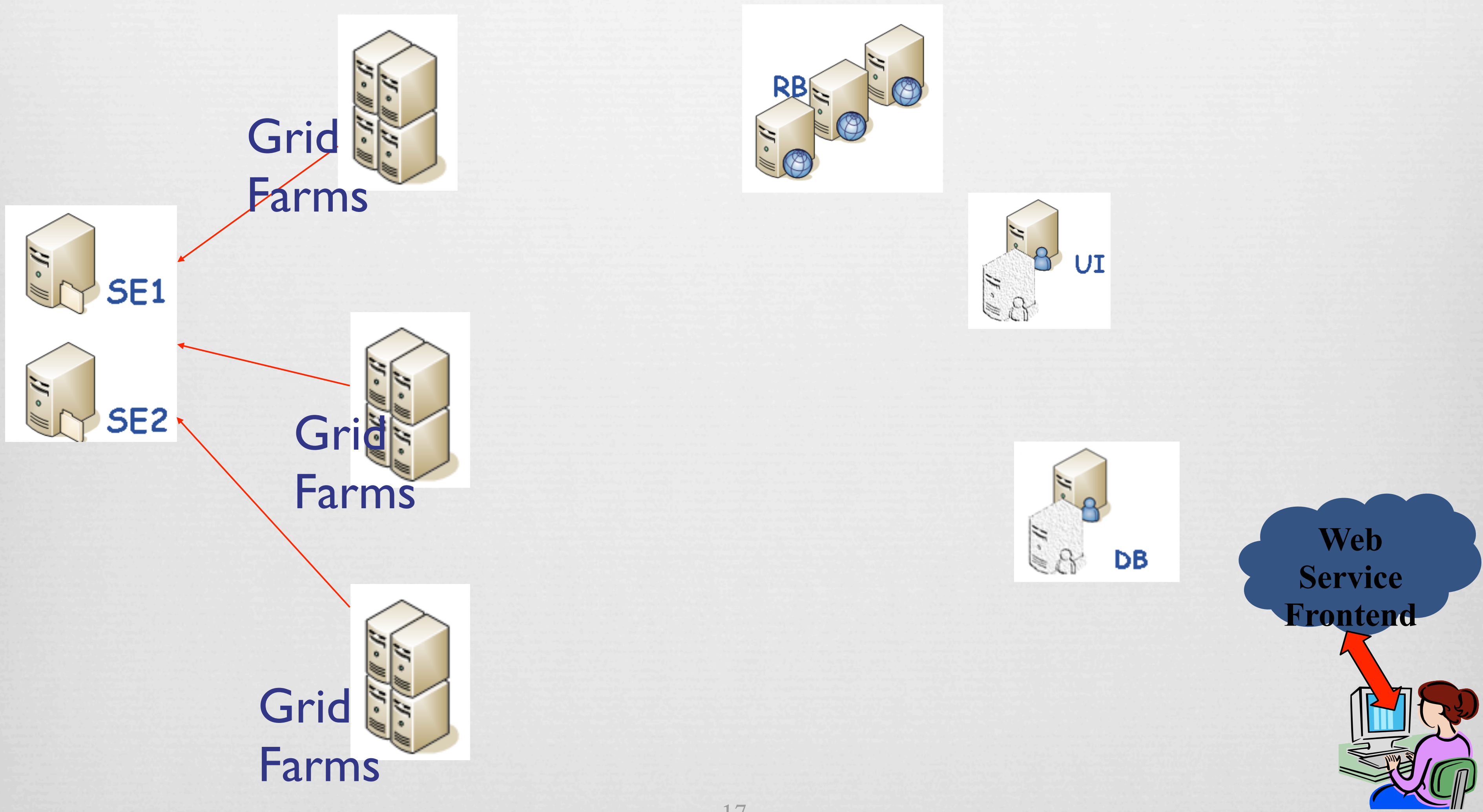


JST Detailed schema

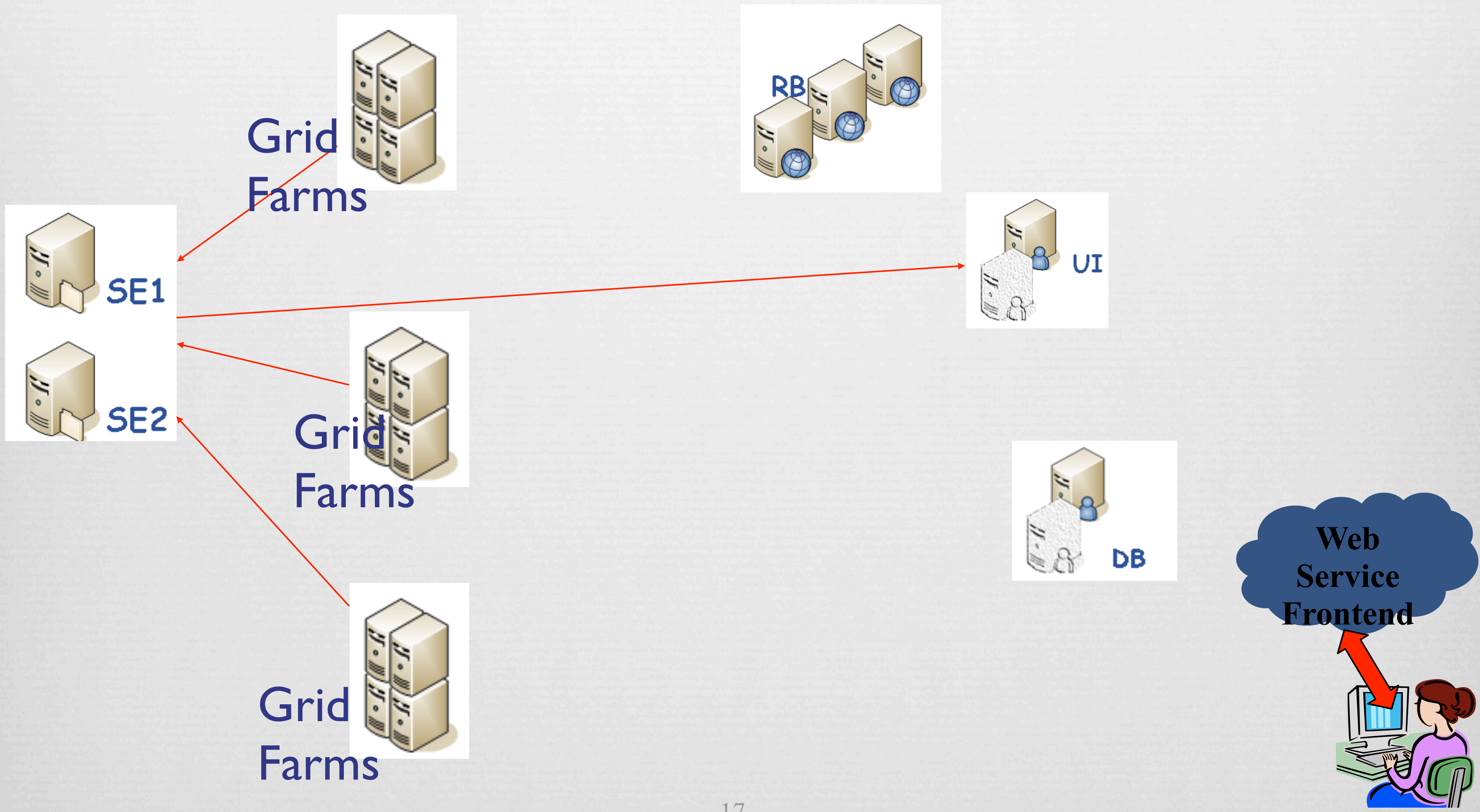


Output registered on a SE chosen in a list of SEs in base to their availability

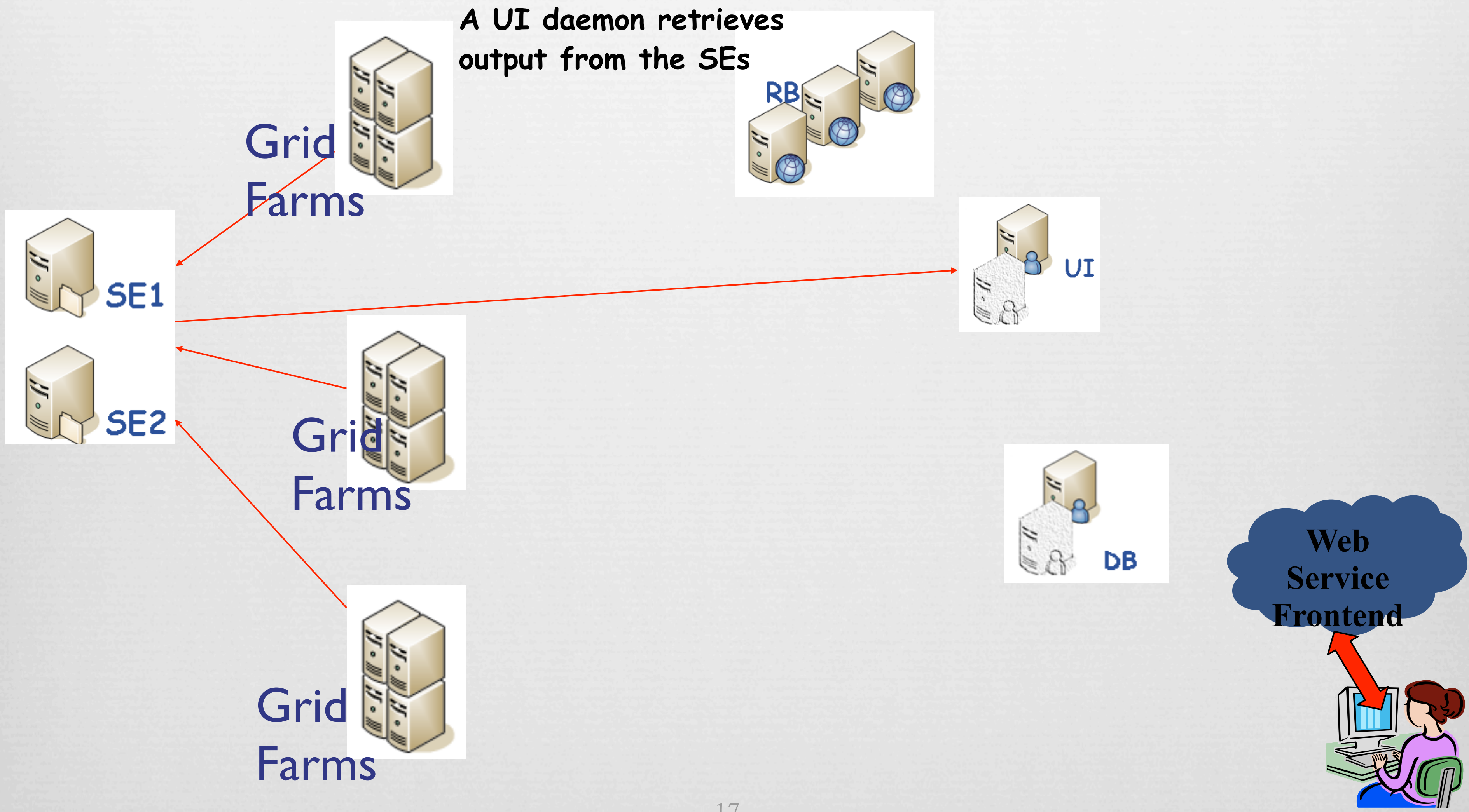
JST Detailed schema



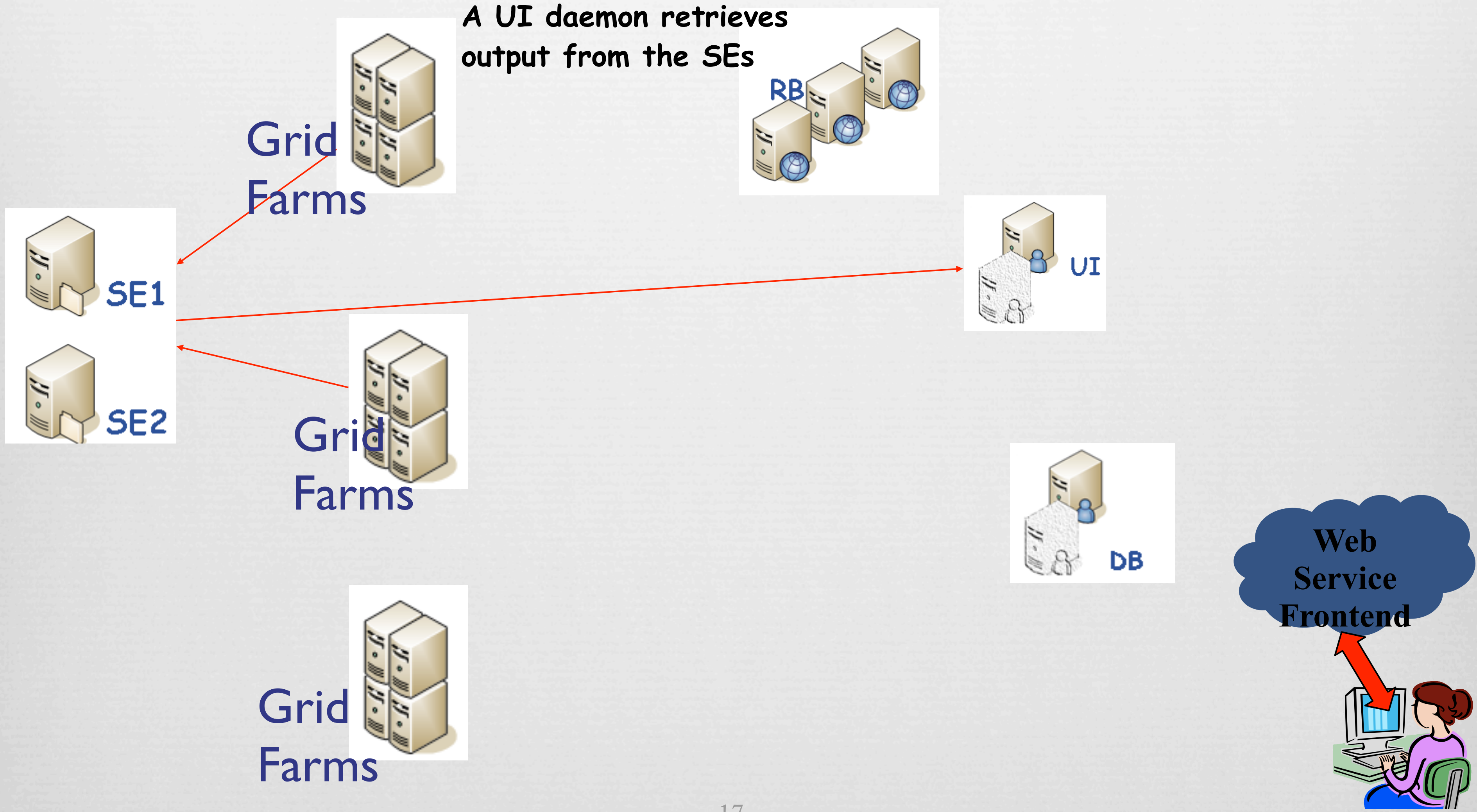
JST Detailed schema



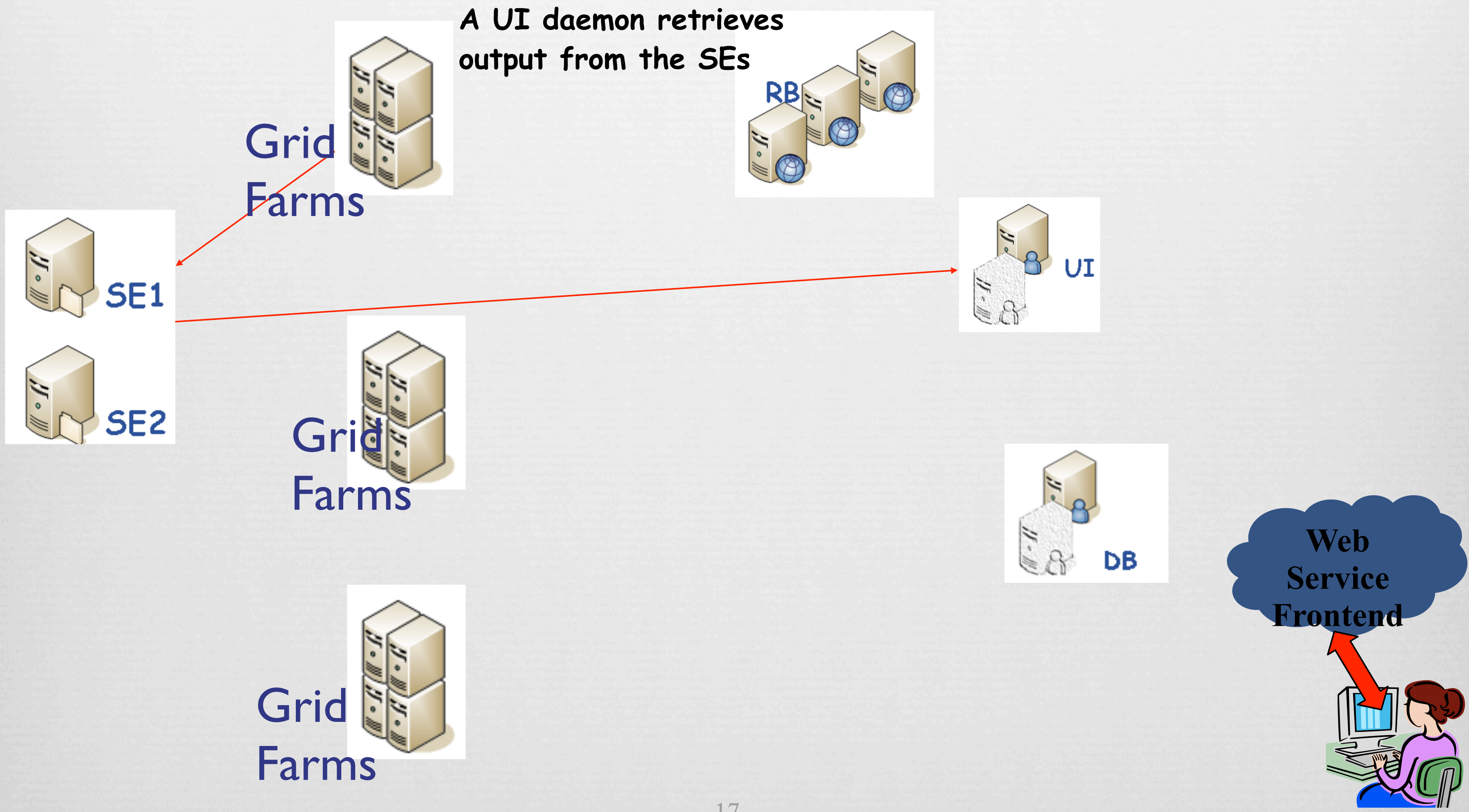
JST Detailed schema



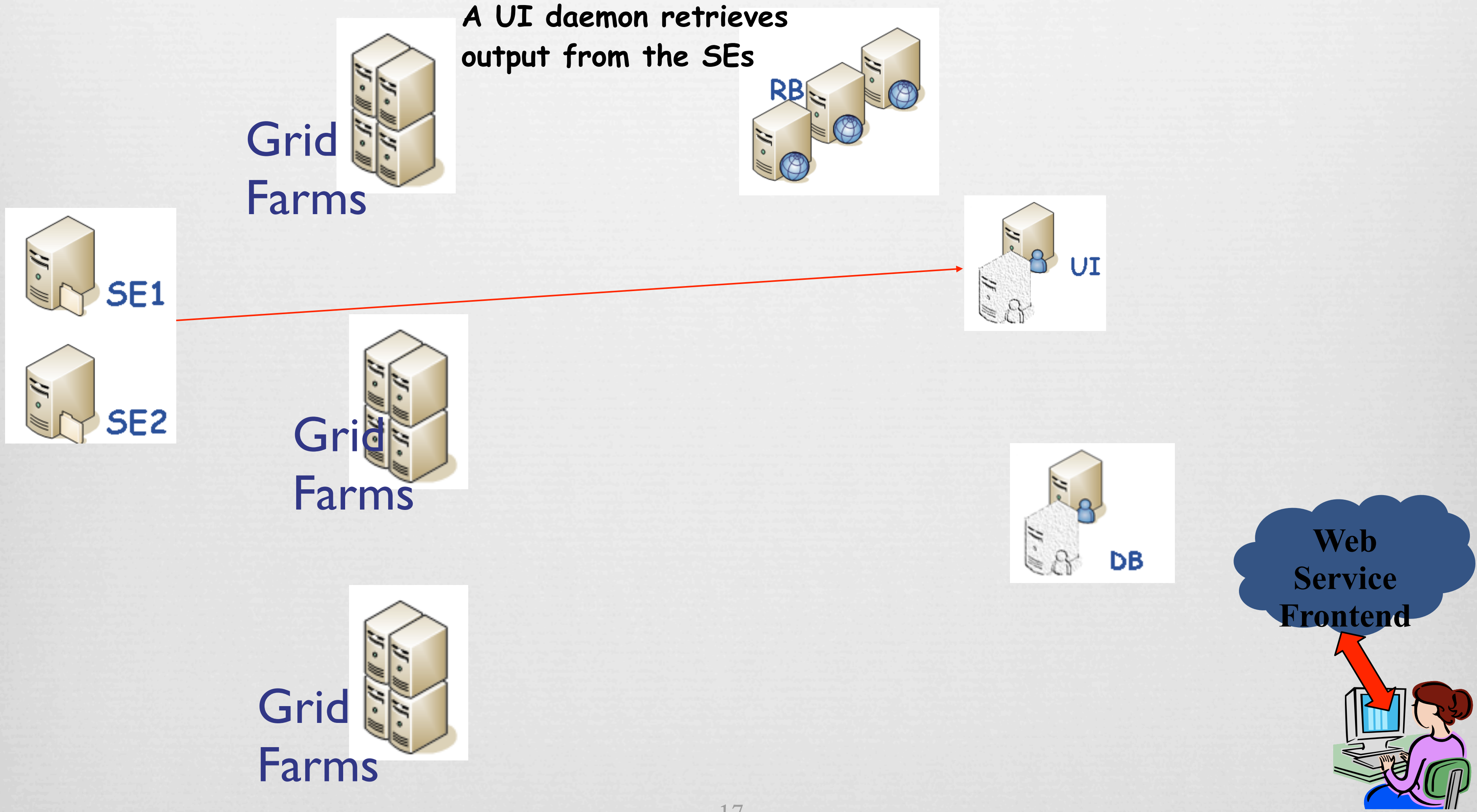
JST Detailed schema



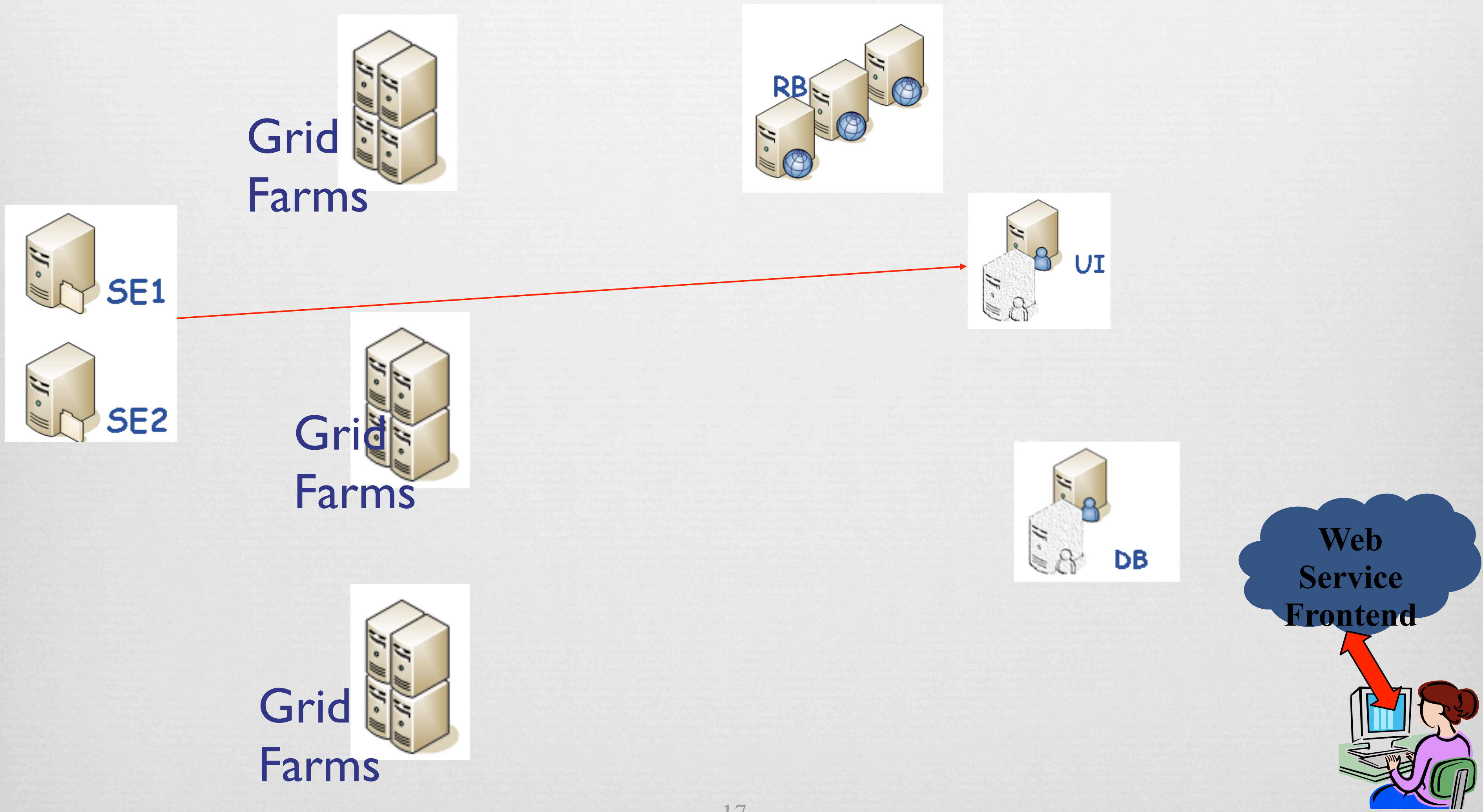
JST Detailed schema



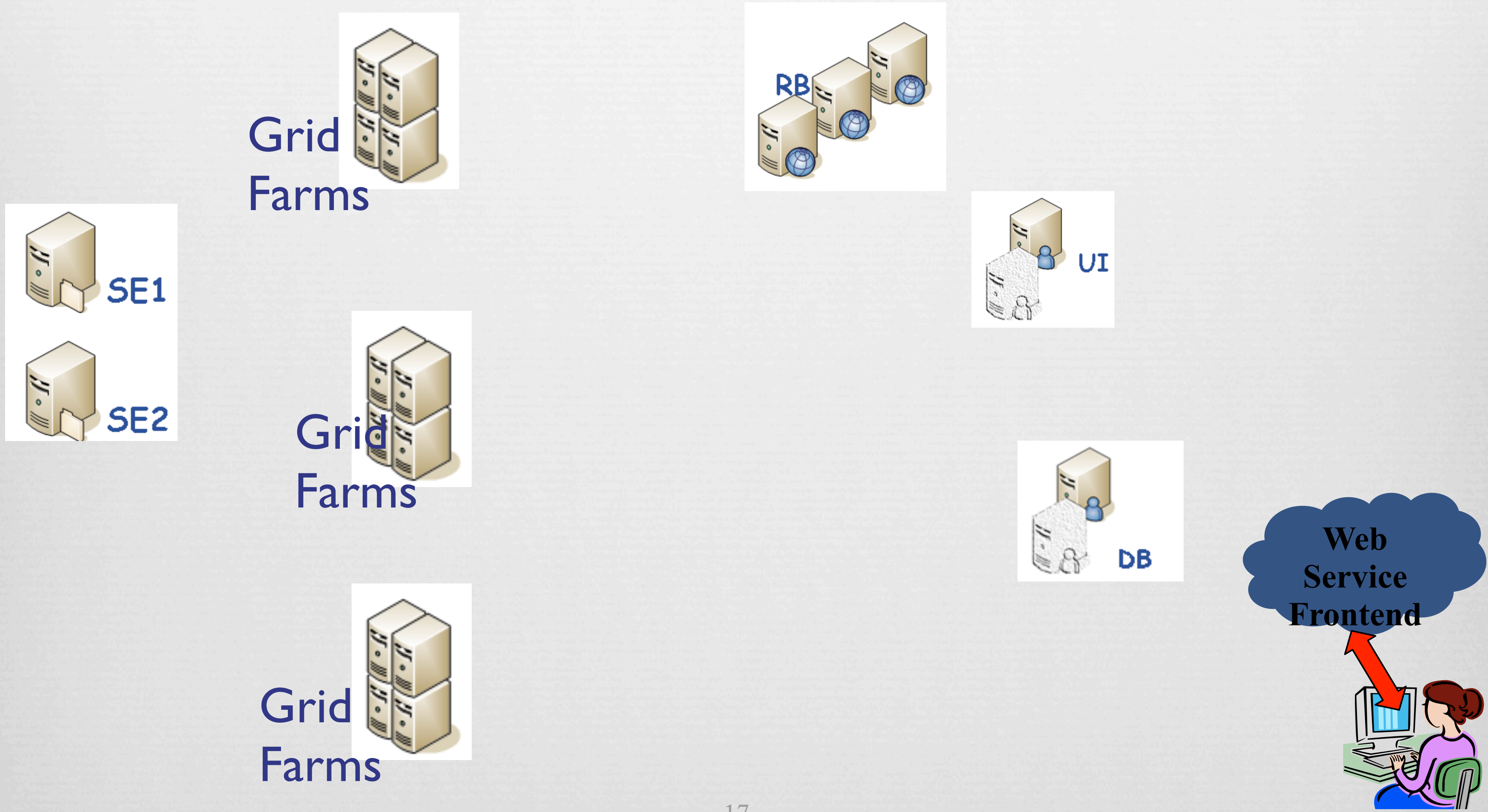
JST Detailed schema



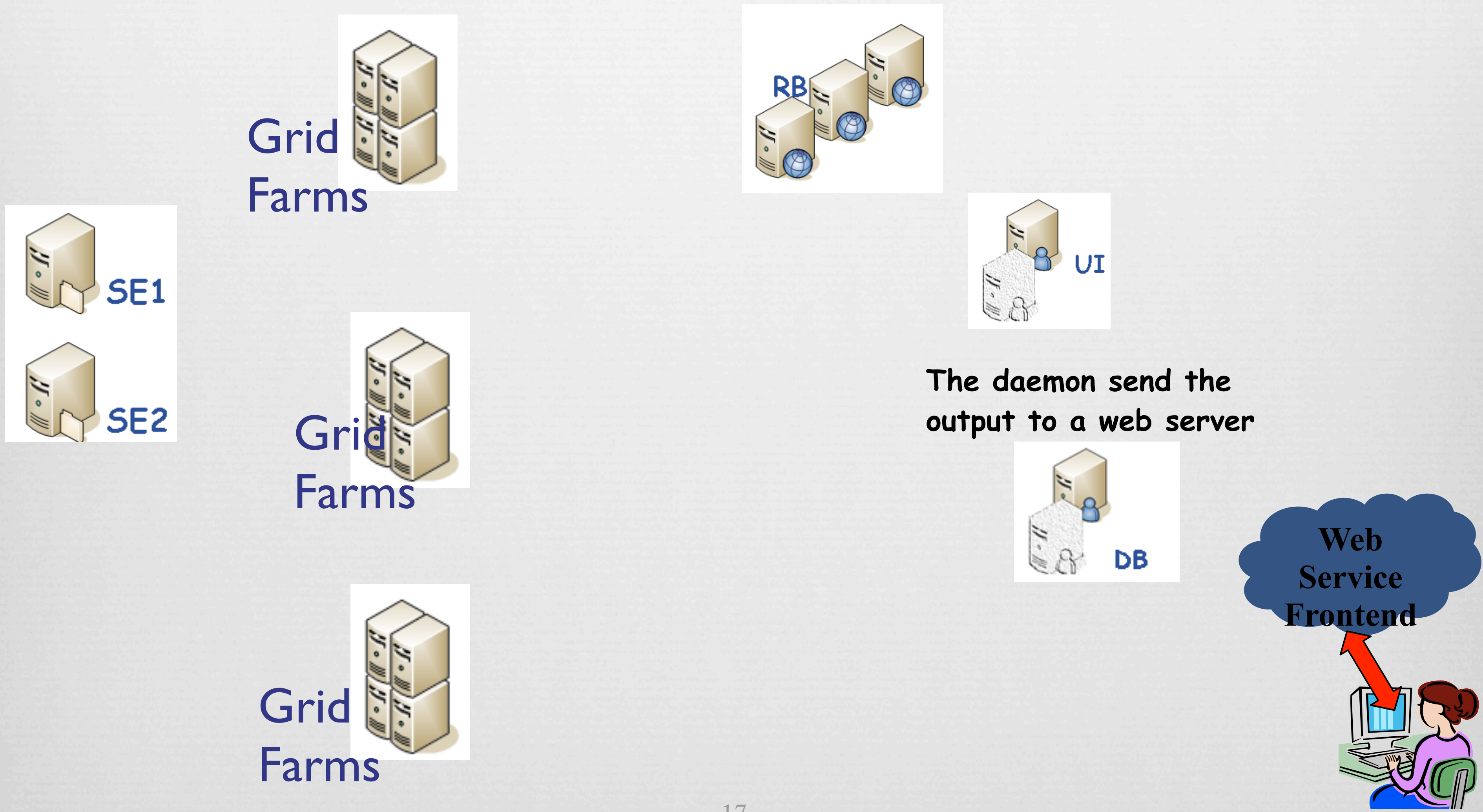
JST Detailed schema



JST Detailed schema

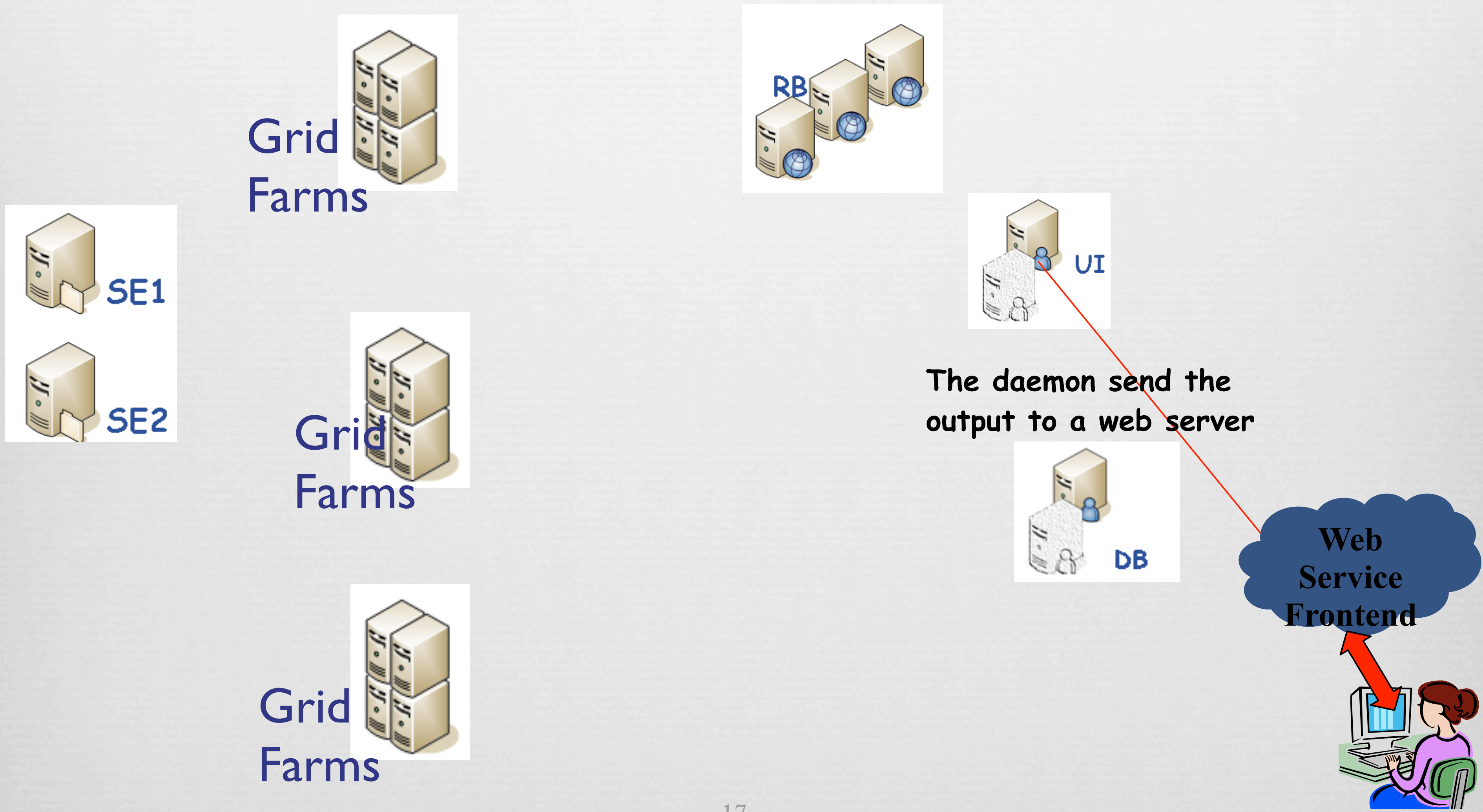


JST Detailed schema



The daemon send the output to a web server

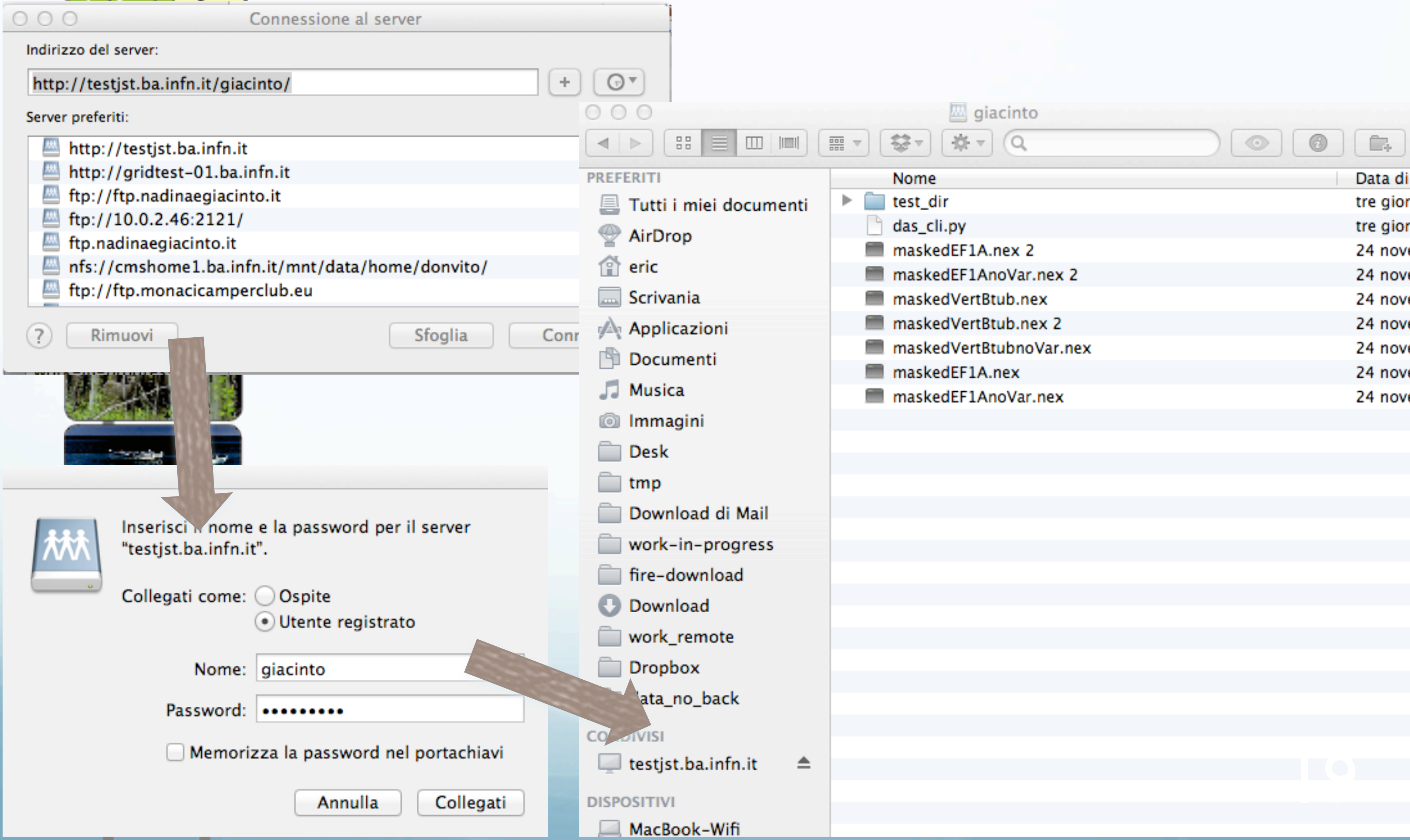
JST Detailed schema



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 service 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
    - Reduce the file transfers between services and users desktop to the minimum (temporary files should be already available to

Screenshots:
WebDav DataManagement Service



Connessione al server

Indirizzo del server: http://testjst.ba.infn.it/giacinto/

Server preferiti:
- http://testjst.ba.infn.it
- http://gridtest-01.ba.infn.it
- ftp://ftp.nadinaegiacinto.it
- ftp://10.0.2.46:2121/
- ftp.nadinaegiacinto.it
- nfs://cmshome1.ba.infn.it/mnt/data/home/donvito/
- ftp://ftp.monacicamperclub.eu

Inserisci i nome e la password per il server "testjst.ba.infn.it".

Collegati come: ( ) Ospite [✓ Utente registrato]

Nome: giacinto

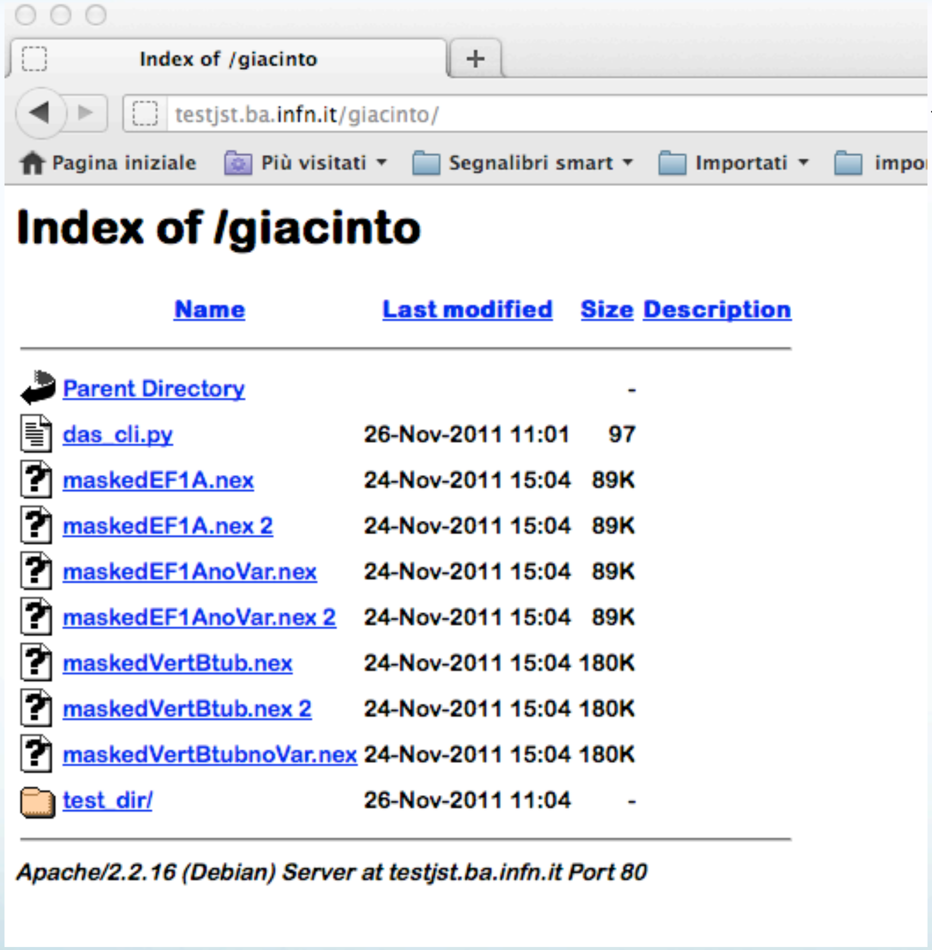
Password: ••••••••••

☐ Memorizza la password nel portachiavi

giacinto

Nome	Data di
test_dir	tre gior
das_cli.py	tre gior
maskedEF1A.nex 2	24 nove
maskedEF1AnoVar.nex 2	24 nove
maskedVertBtub.nex	24 nove
maskedVertBtub.nex 2	24 nove
maskedVertBtubnoVar.nex	24 nove
maskedEF1A.nex	24 nove
maskedEF1AnoVar.nex	24 nove

Screenshots: WebDav DataManagement Service



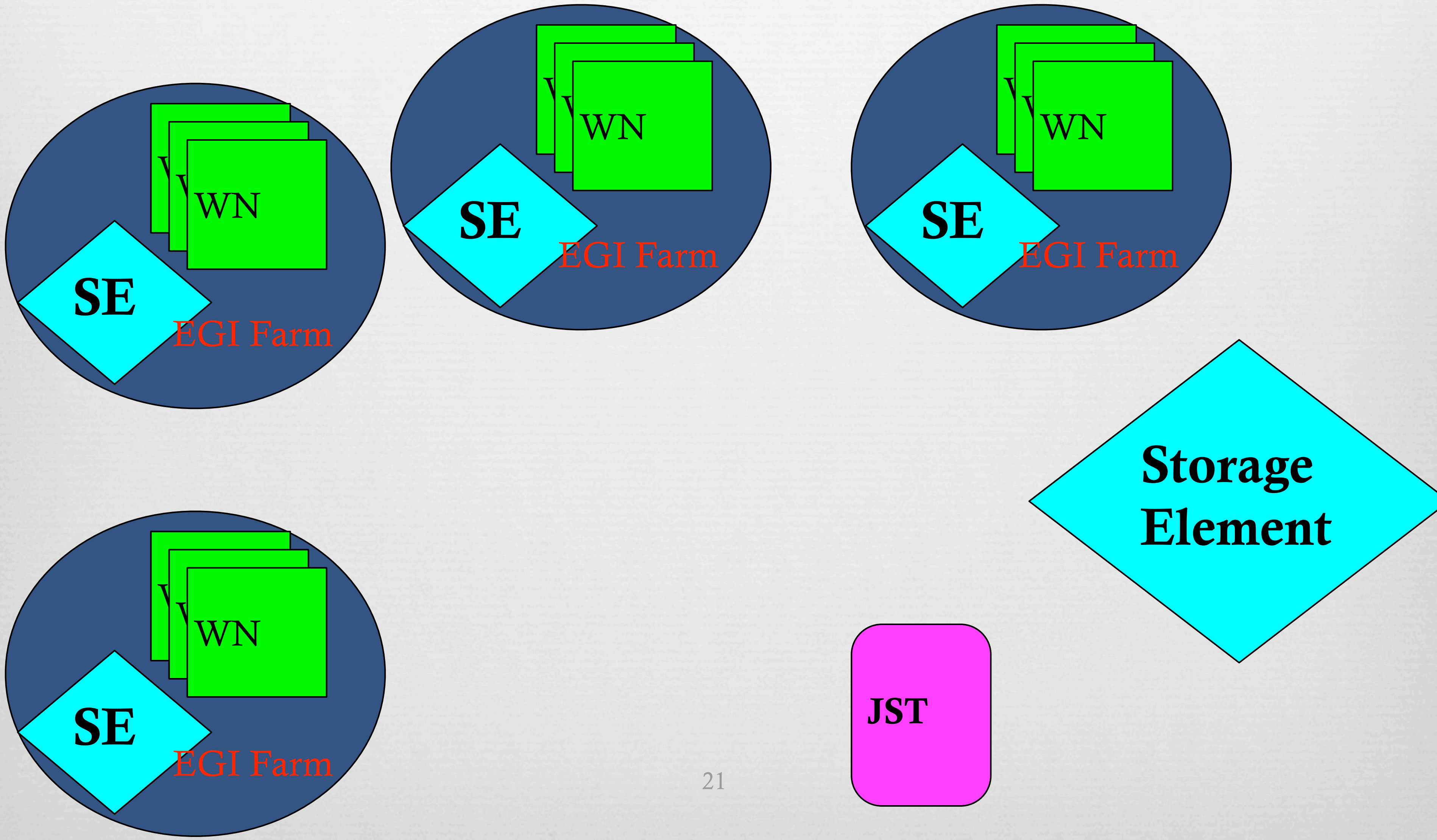
Index of /giacinto

Name	Last modified	Size	Description
Parent Directory		-	
das_cli.py	26-Nov-2011 11:01	97	
maskedEF1A.nex	24-Nov-2011 15:04	89K	
maskedEF1A.nex 2	24-Nov-2011 15:04	89K	
maskedEF1AnoVar.nex	24-Nov-2011 15:04	89K	
maskedEF1AnoVar.nex 2	24-Nov-2011 15:04	89K	
maskedVertBtub.nex	24-Nov-2011 15:04	180K	
maskedVertBtub.nex 2	24-Nov-2011 15:04	180K	
maskedVertBtubnoVar.nex	24-Nov-2011 15:04	180K	
test_dir/	26-Nov-2011 11:04	-	

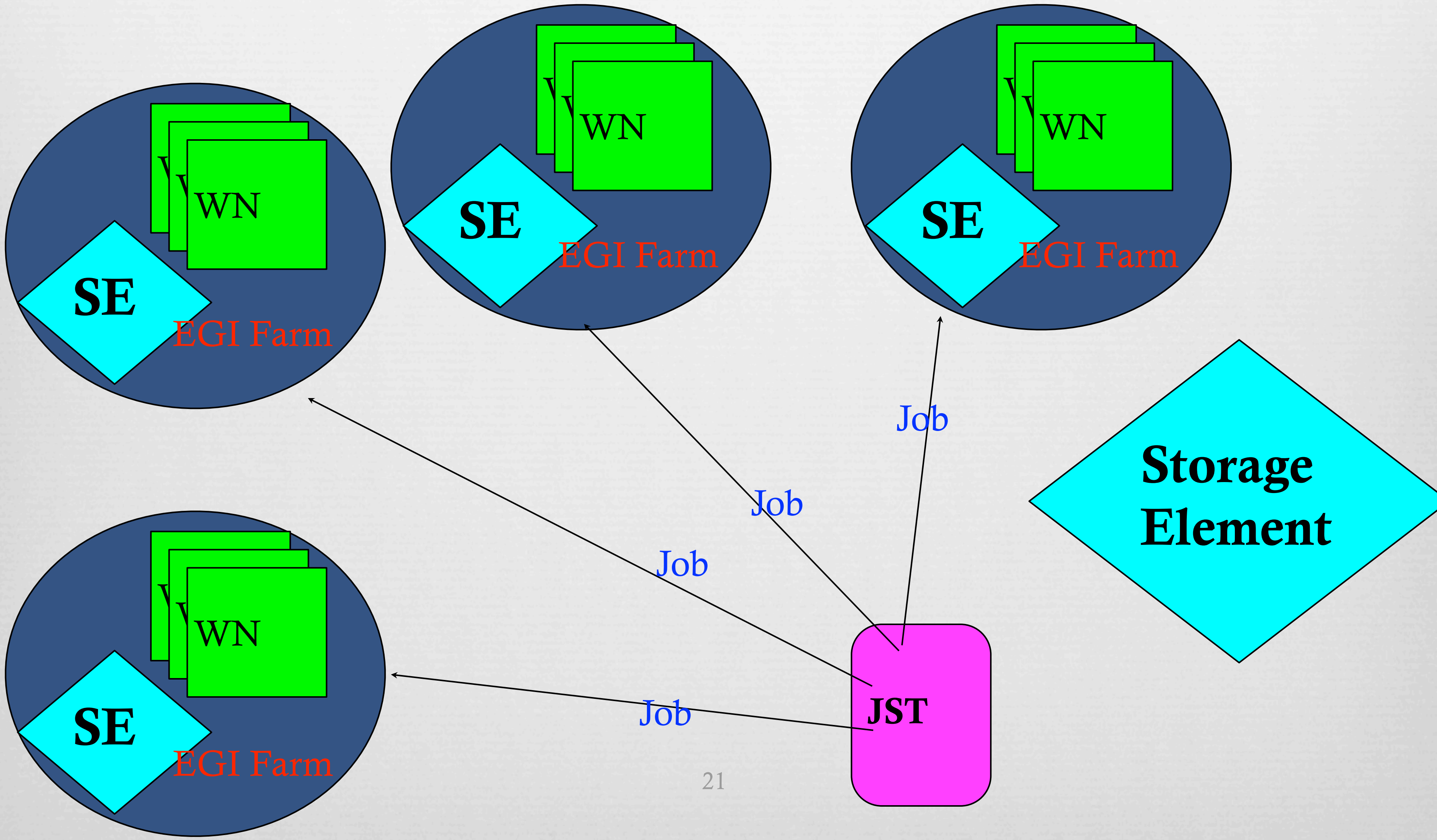
Apache/2.2.16 (Debian) Server at testjst.ba.infn.it Port 80

- You can access those files using web browser:
  - You can easily share your data with others colleagues
  - Or use the input/output within other (web) services

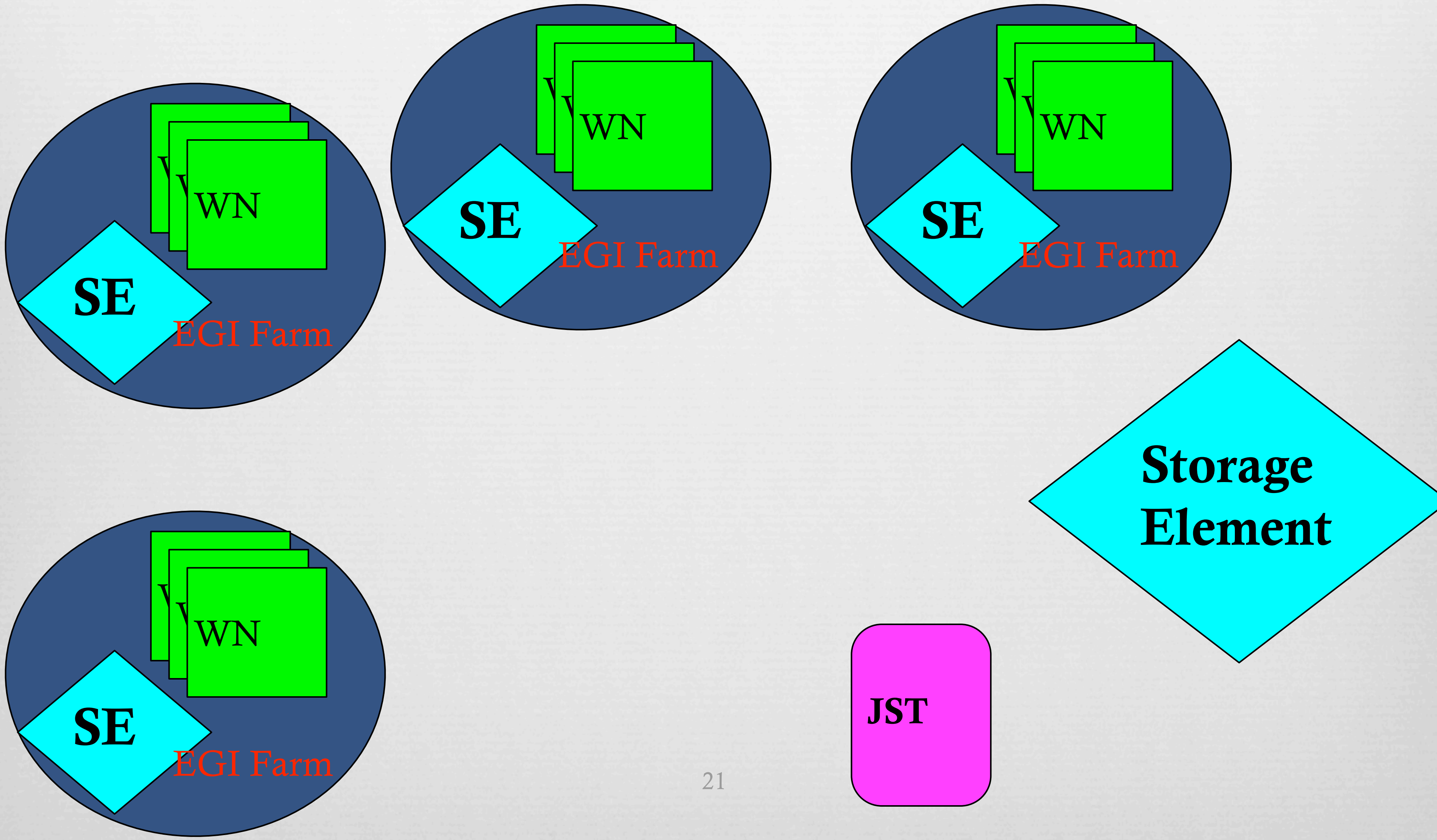
Data-Management and Bioinformatics applications



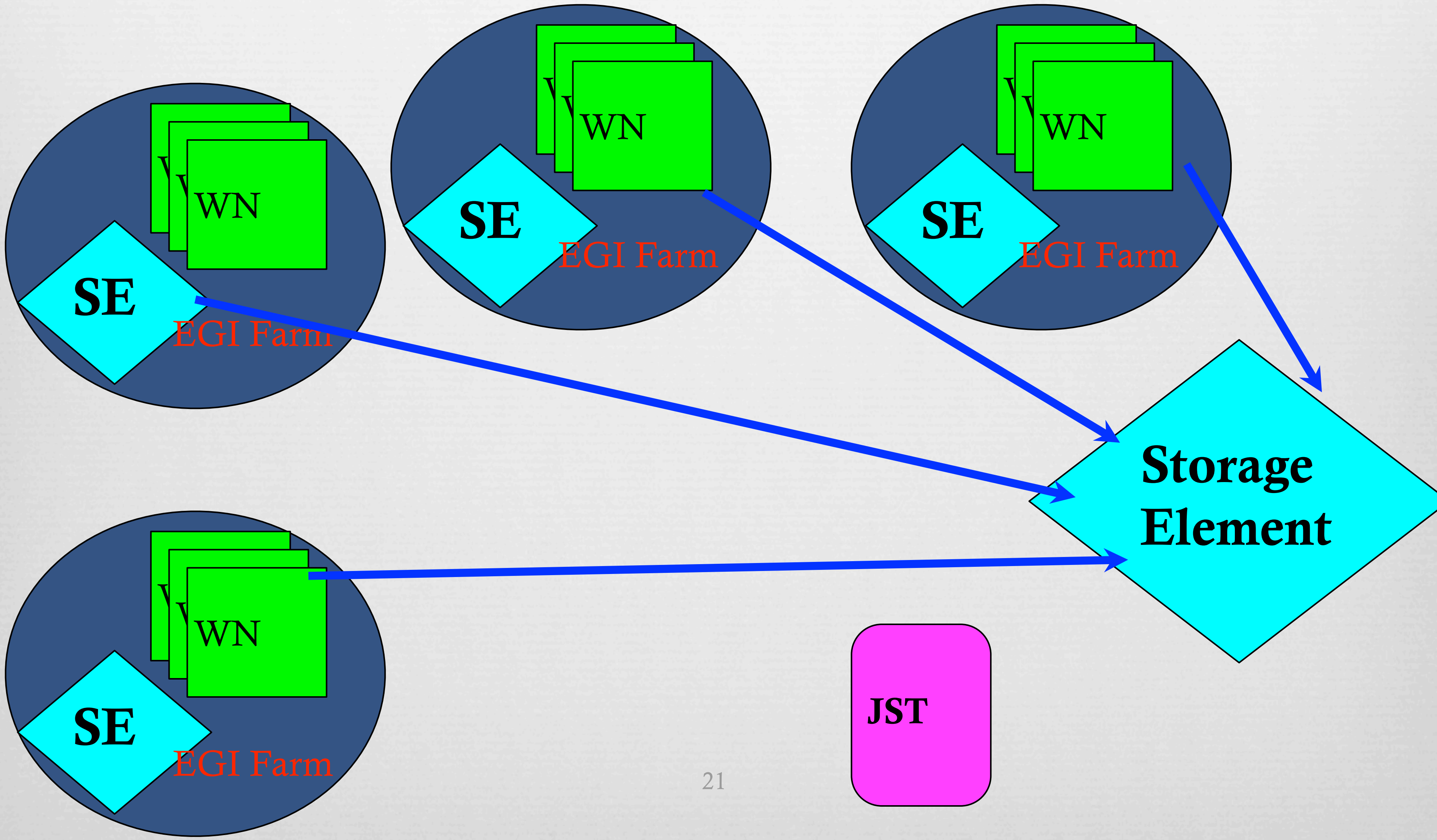
Data-Management and Bioinformatics applications



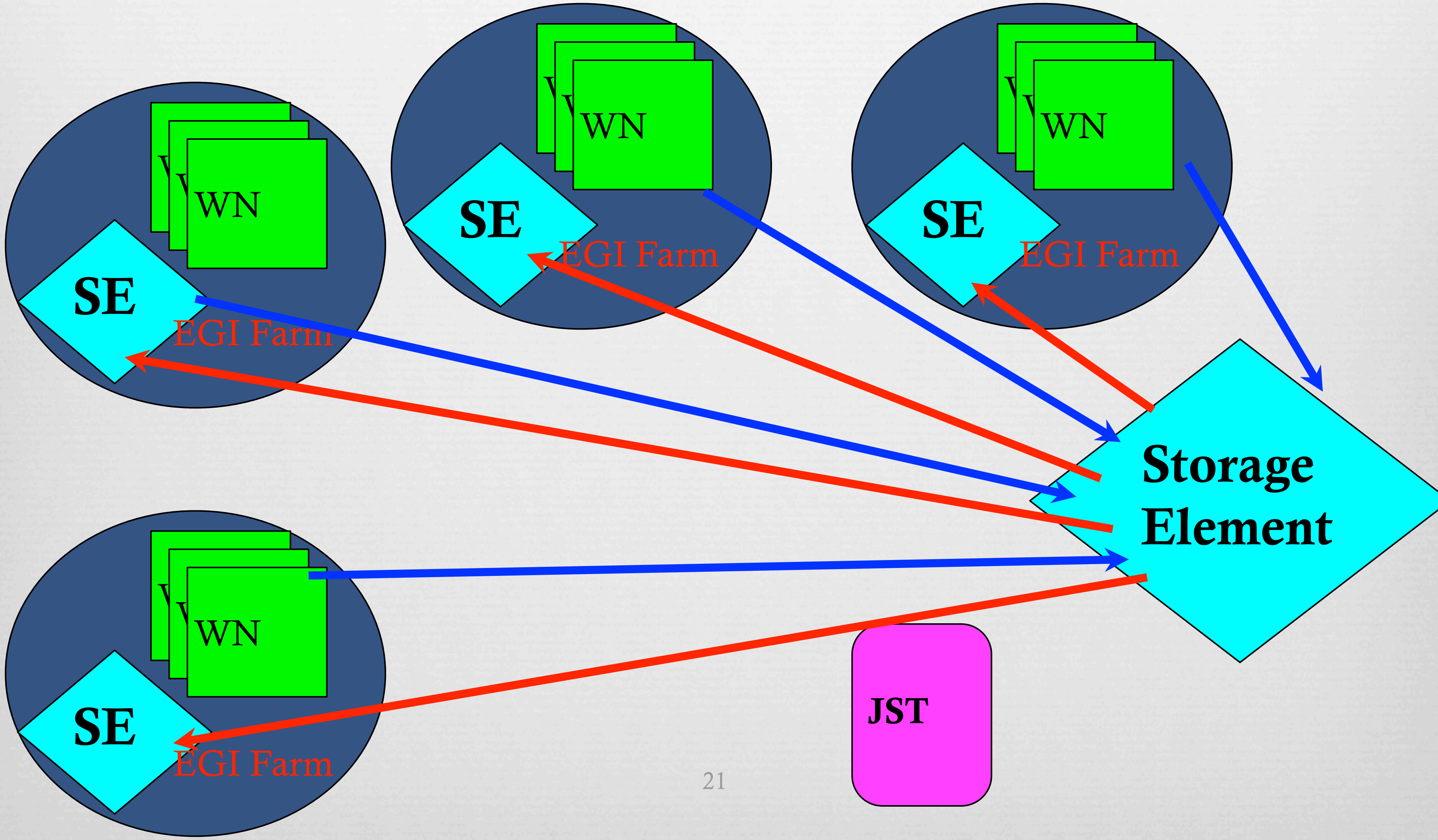
Data-Management and Bioinformatics applications



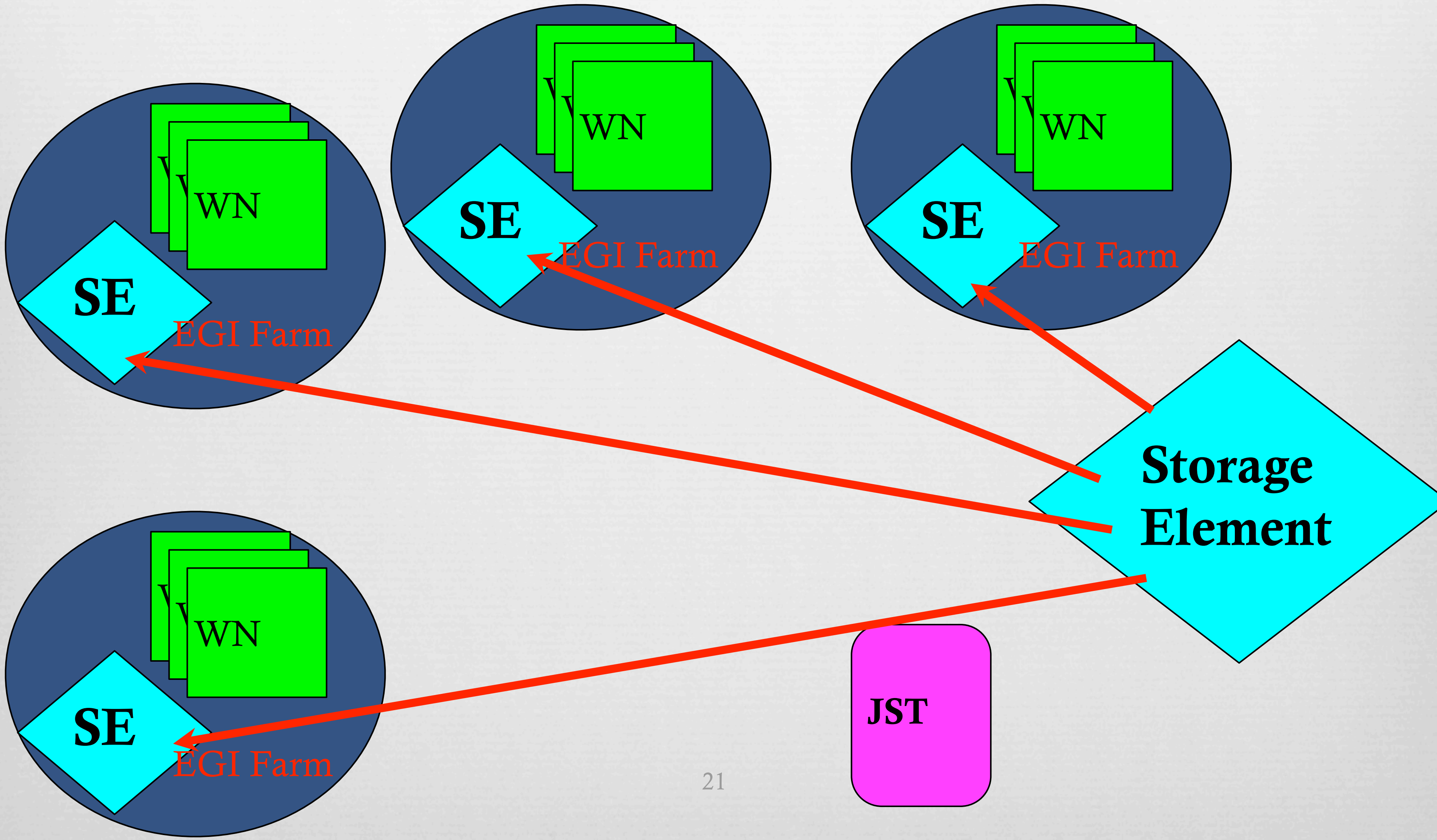
Data-Management and Bioinformatics applications



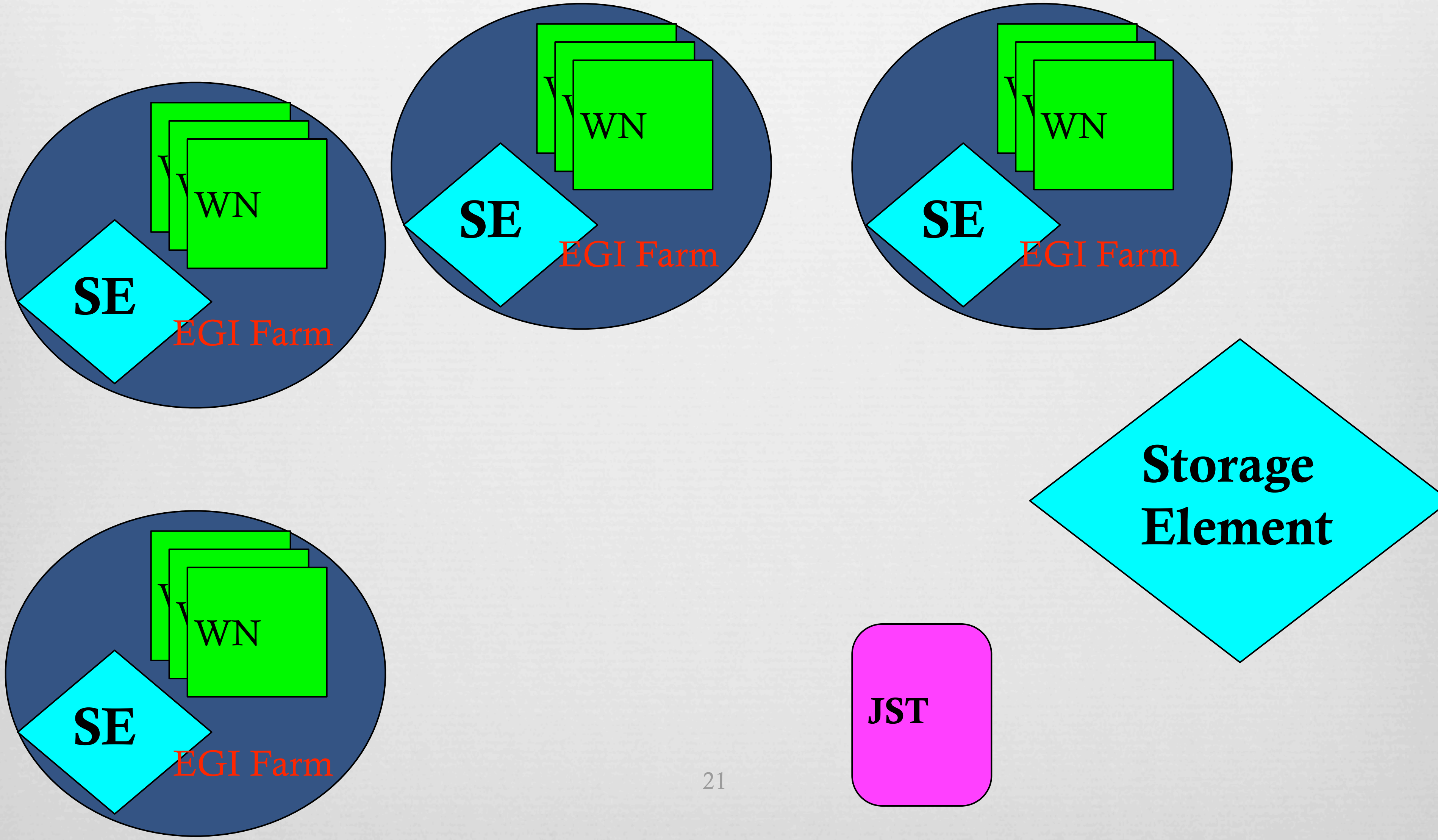
Data-Management and Bioinformatics applications



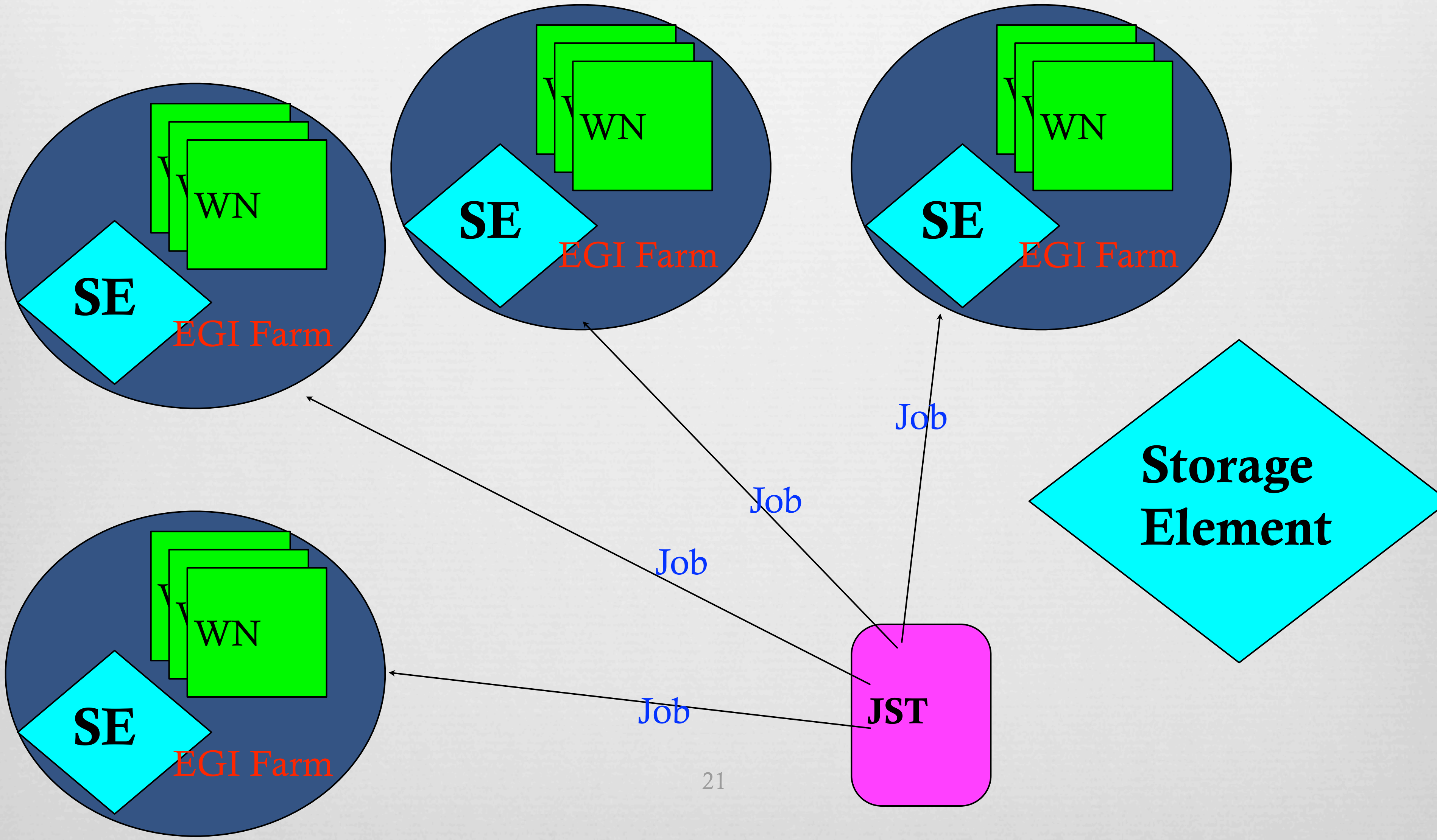
Data-Management and Bioinformatics applications



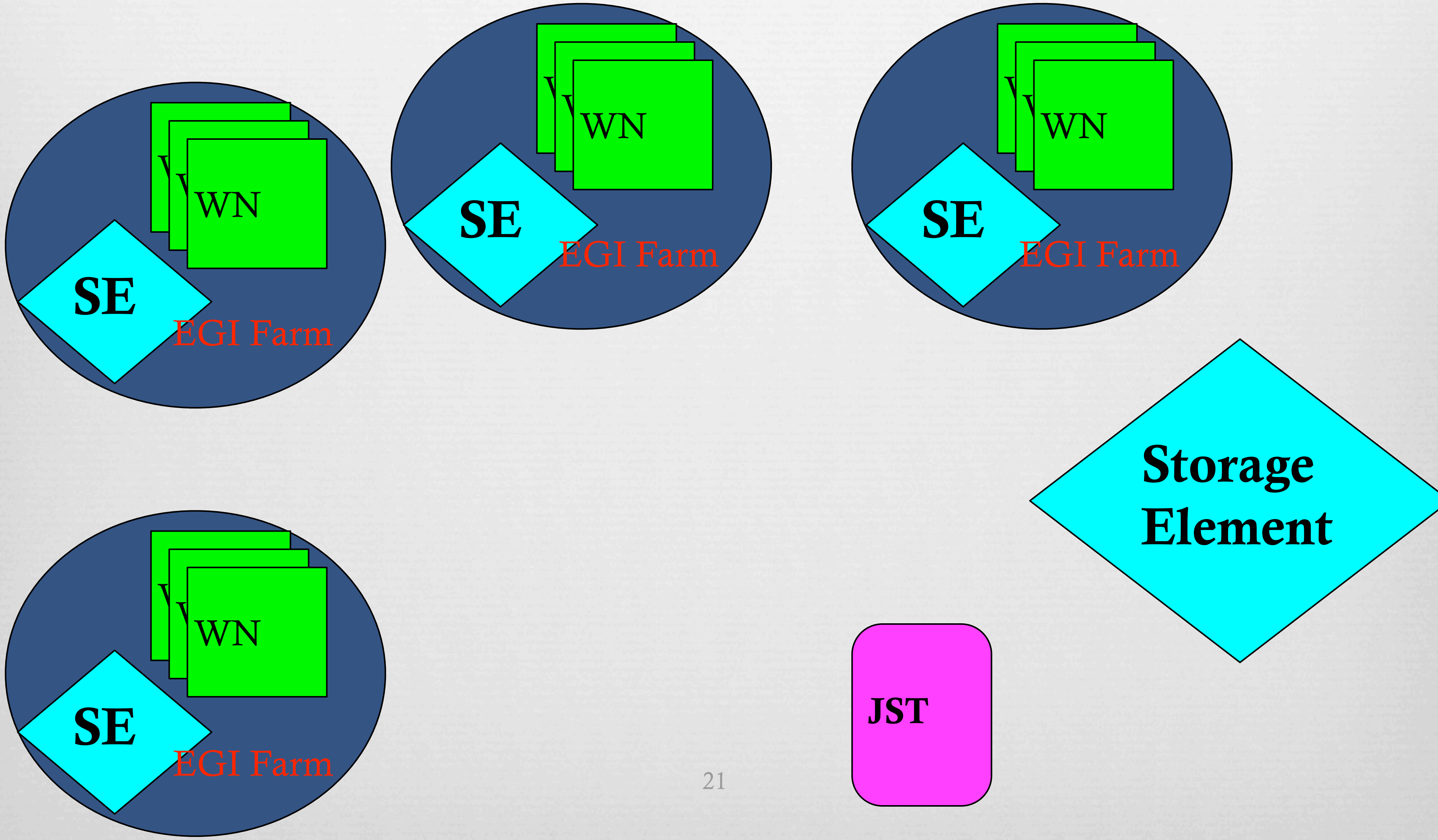
Data-Management and Bioinformatics applications



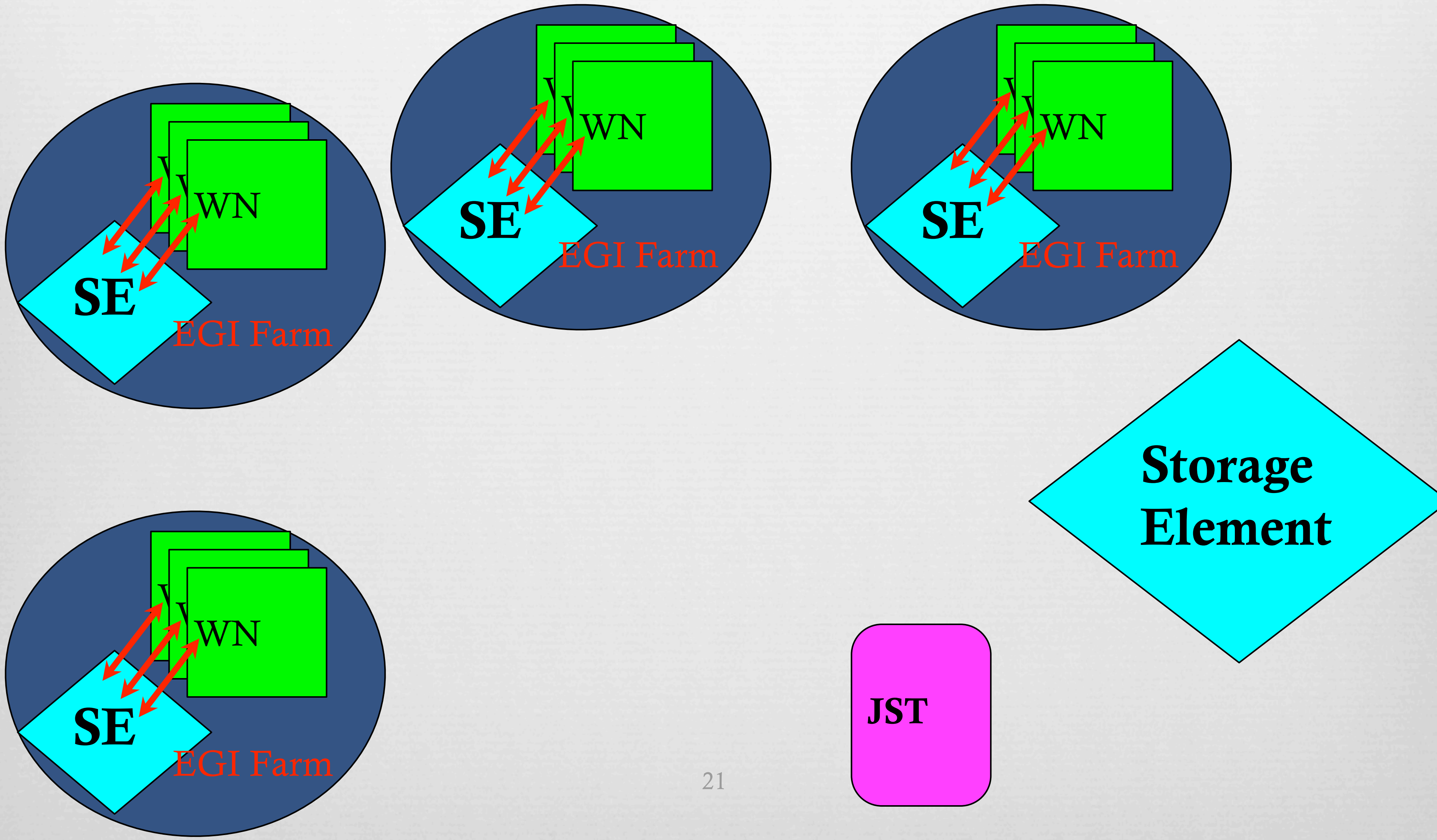
Data-Management and Bioinformatics applications



Data-Management and Bioinformatics applications

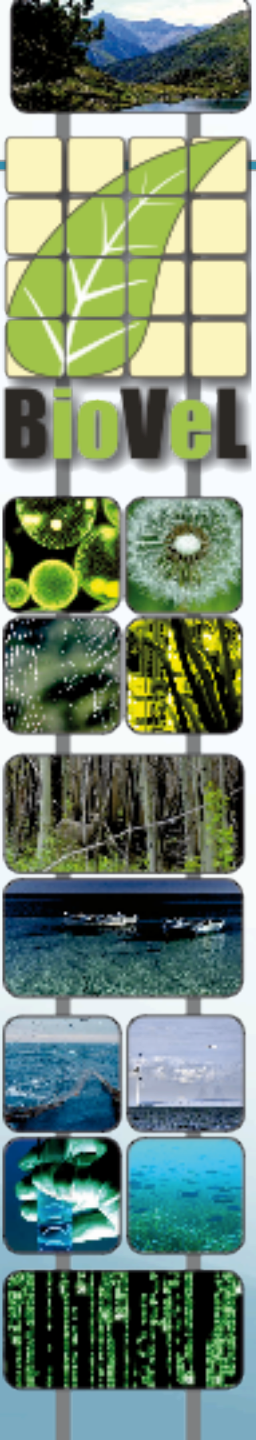


Data-Management and Bioinformatics applications



Features Supported

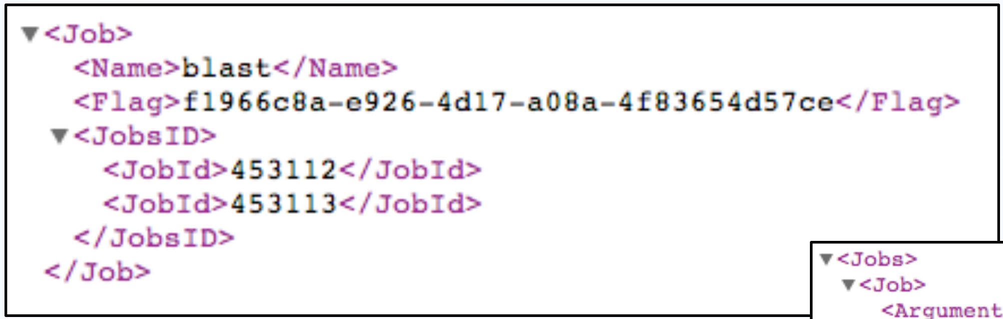
- Transferring files and directory:
  - user friendly
  - good performance
  - Well established protocol (WebDav)
- Submitting and monitoring the status of one or several runs with simple Soap/ REST APIs
  - Multiple submission (up to thousands tasks) with a single web services call
  - Monitoring the whole run with a single WS call
  - High reliability of the job execution with re-submission in case of failures
    - also stage-in/out operation is checked to deal with job failures



Rest Web service example

Insert Jobs:

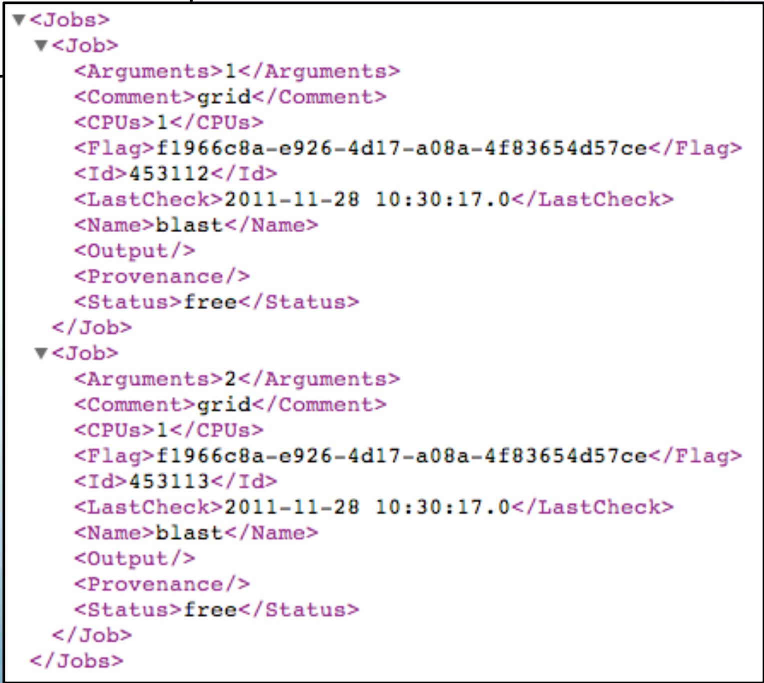
http://localhost:8080/RestService/services/QueryJob/InsertJobs?NAME={blast}&arguments={http://webtest.ba.infn.it/vicario/FinalFusariumDB_2.nex ArgOne; http://webtest.ba.infn.it/vicario/FinalFusariumDB_1.nex ArgTwo;}



```
▼<Job>
   <Name>blast</Name>
   <Flag>f1966c8a-e926-4d17-a08a-4f83654d57ce</Flag>
  ▼<JobsID>
     <JobId>453112</JobId>
     <JobId>453113</JobId>
   </JobsID>
 </Job>
```

Select Jobs:

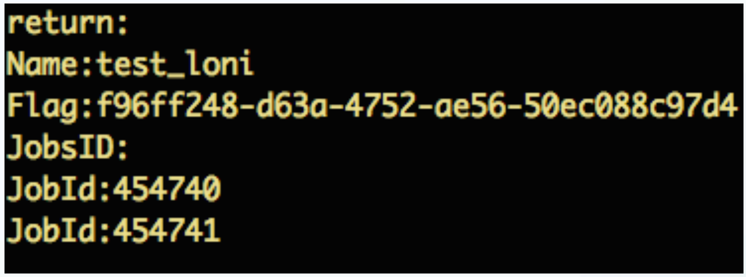
http://localhost:8080/RestService/services/QueryJob/SelectJobs?FLAG={20b3cbf8-6805-47b4-ad7c-7b40bc706741}



```
▼<Jobs>
  ▼<Job>
     <Arguments>1</Arguments>
     <Comment>grid</Comment>
     <CPUs>1</CPUs>
     <Flag>f1966c8a-e926-4d17-a08a-4f83654d57ce</Flag>
     <Id>453112</Id>
     <LastCheck>2011-11-28 10:30:17.0</LastCheck>
     <Name>blast</Name>
     <Output/>
     <Provenance/>
     <Status>free</Status>
   </Job>
  ▼<Job>
     <Arguments>2</Arguments>
     <Comment>grid</Comment>
     <CPUs>1</CPUs>
     <Flag>f1966c8a-e926-4d17-a08a-4f83654d57ce</Flag>
     <Id>453113</Id>
     <LastCheck>2011-11-28 10:30:17.0</LastCheck>
     <Name>blast</Name>
     <Output/>
     <Provenance/>
     <Status>free</Status>
   </Job>
 </Jobs>
```

Soap Web service example

- wsdlpull 'http://localhost:8080/INFN.Grid.SoapFrontEnd/SoapServiceMethodsPort?wsdl' **InsertJobs** admin admin test_loni 'MatLabRUN1 input_test 12; MatLabRUN2 input_test2 24' pasq.notra@ba.infn.it



```
return:
Name:test_loni
Flag:f96ff248-d63a-4752-ae56-50ec088c97d4
JobsID:
JobId:454740
JobId:454741
```

- wsdlpull 'http://localhost:8080/INFN.Grid.SoapFrontEnd/SoapServiceMethodsPort?wsdl' **SelectJobs** admin admin 20b3cbf8-6805-47b4-ad7c-7b40bc706741

Upload the user’s inputs

Run MrBayes: a MPI application that could run for several hours

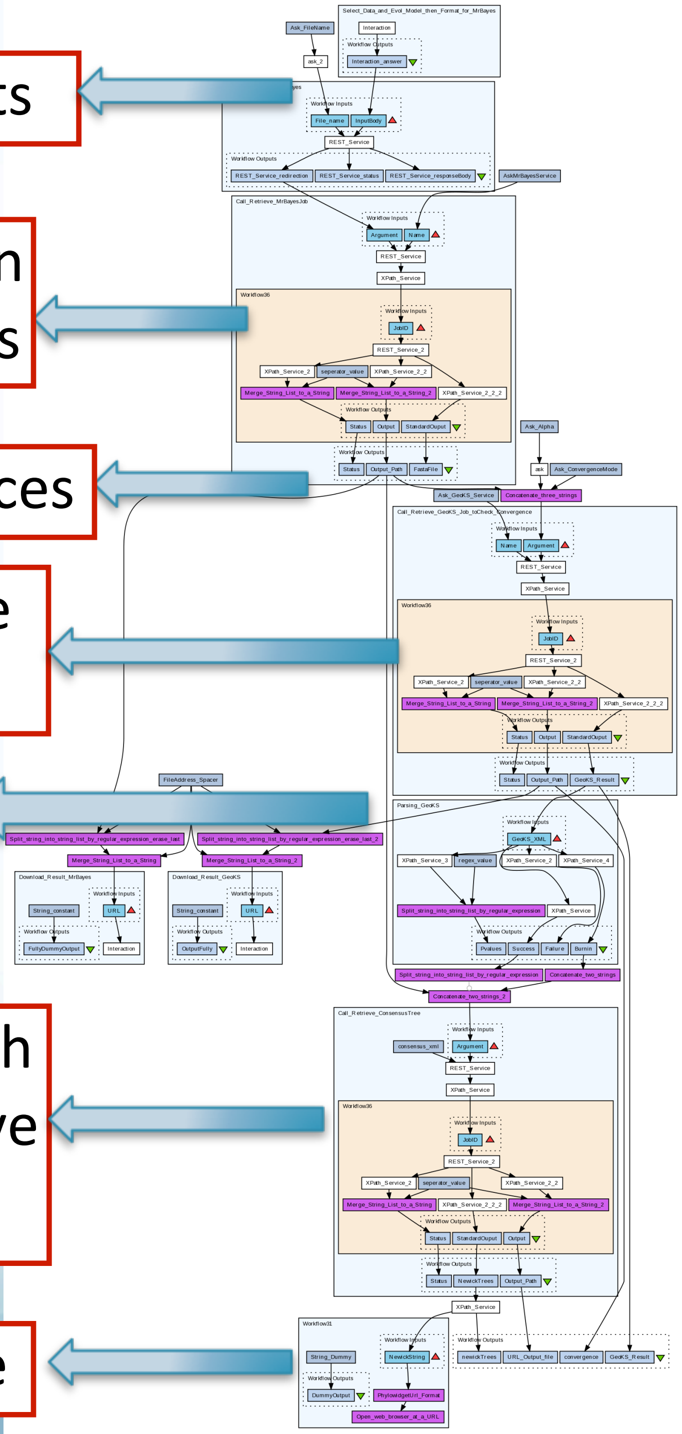
Pass the output to the next services

Check the convergence of the model

Retrieving the output and parsing the XML

calculate the consensus tree of th e posterior distribution of MrBaye s output

Graphical view of the tree



Description of the resources

- Grid distributed computing infrastructures (EGI Production Grid)
  - ~ 200'000 Cores, Hundreds of distributed sites
  - ~ 20PB
  - There sites that offers a good support to biomed VO
  - ***Very useful for huge number of independent tasks***
- Local batch INFN-Bari (ReCaS) farm
  - ~4000 Cores
  - 1.6PB of storage
  - Supporting several VOs
  - 10% of the share are dedicated to opportunistic VOs (like bioinformatics)
  - ***From tens to hundreds of concurrent execution, very good support for MPI application***
- Dedicated servers
  - Big servers: 24Cores, 48GB of RAM (will be higher in the future)
  - Specialized servers: 2 Tesla C2070 GPUs
  - ***Thread based parallel applications, GPU enabled applications, short and high priority tasks***

Test & Results

- Stress test already passed:
  - 100’000 insert in a loop… no memory leak or similar problems
  - Up to 100 concurrent clients without problems
  - 1000 tasks insert in a single REST call
  - ~1M of tasks managed from DB+backend
- A lot of experience in porting Bioinformatics application over EGI distributed computing infrastructure:
  - Hmmer, MrBayes, Blast, PAML, MUSCLE, EMBOSS, Biopython, AmpliconNoise, ABCtool, Bowtie, BayeSSC, GeoKS, hyphy, raxmlHPC, phylocom, consensus_xml, Matlab, etc…
- 25 different services already provided to users communities

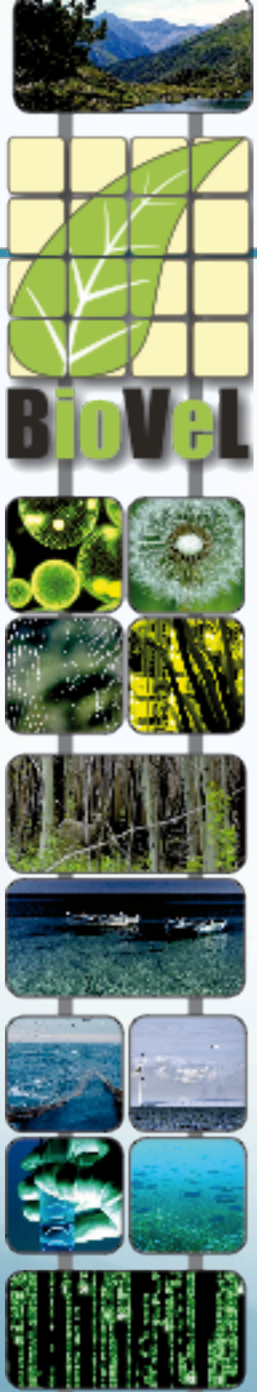
Conclusions & To-do

- We have a high scalable and solid service that could be used to supports execution of applications over different computing infrastructure
- We have also a high performance data transfer and sharing service
- We publish both services and WorkFlows on BioCatalogue and MyExperiments as soon as they are available
- It will be quite easy to add new application in the near future
- We will soon add OpenID authentication, and
  - as soon as it will be required it would be easy to add GSI or Shibboleth security on the front-end

People involved in the development

- Giacinto Donvito (INFN-ReCaS)
- Pasquale Notarangelo (INFN)
- Saverio Vicario (CNR)
- Bachir Balech (CNR)

Biodiversity Virtual e-Laboratory



BioVeL is funded by the European Commission 7th Framework Programme (FP7).
It is part of its e-Infrastructures activity.

Under FP7, the e-Infrastructures activity is part of the Research Infrastructures programme, funded under the FP7 'Capacities' Specific Programme. It focuses on the further development and evolution of the high-capacity and high-performance communication network (GÉANT), distributed computing infrastructures (grids and clouds), supercomputer infrastructures, simulation software, scientific data infrastructures, e-Science services as well as on the adoption of e-Infrastructures by user communities.

BioVeL is free and available via internet.

www.biovel.eu, contact Alex Hardisty: HardistyAR@cardiff.ac.uk