- **DONVITO Giacinto (INFN)**
- GAIDO, Luciano (INFN)
- CESINI, Daniele (INFN)
- MOSSUCCA, Lorenzo (Istituto Superiore Mario Boella)
- TERZO, Olivier (Istituto Superiore Mario Boella)
- RUIU, Pietro (Istituto Superiore Mario Boella)
- ACQUAVIVA, Andrea (Politecnico di Torino)

TOPHAT ON THE GRID: AN AUTOMATIC WORKFLOW FOR SEQUENCE ALIGNMENT EXPLOITING EGI/IGI GRID INFRASTRUCTURE

EGI TF, 17-21 September 2012, Prague

OUTLINE

- Scientific problem description
- Description of the workflow
- Description of the grid distributed execution
- Problem & Solutions
- Conclusion and future works

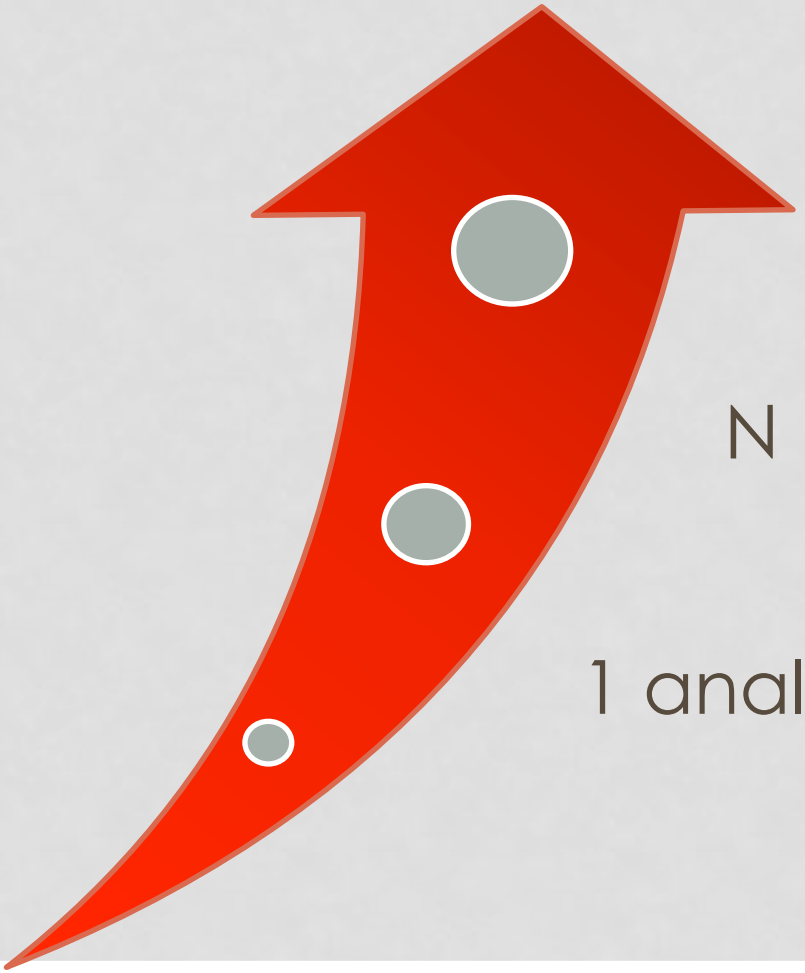
APPLICATION CONTEXT - NGS

- Is opening the way to new and more accurate biological analysis.
- Is extremely helpful to the detection of various forms of disease.
- Enables the sequencing of entire genomes more efficiently and economically and with greater depth than ever before.

	2009	2010	Target
Time	years	weeks	days
Cost [$]	100K	10K	< 1K

DATA EXPLOSION

- Needs of new computational environment



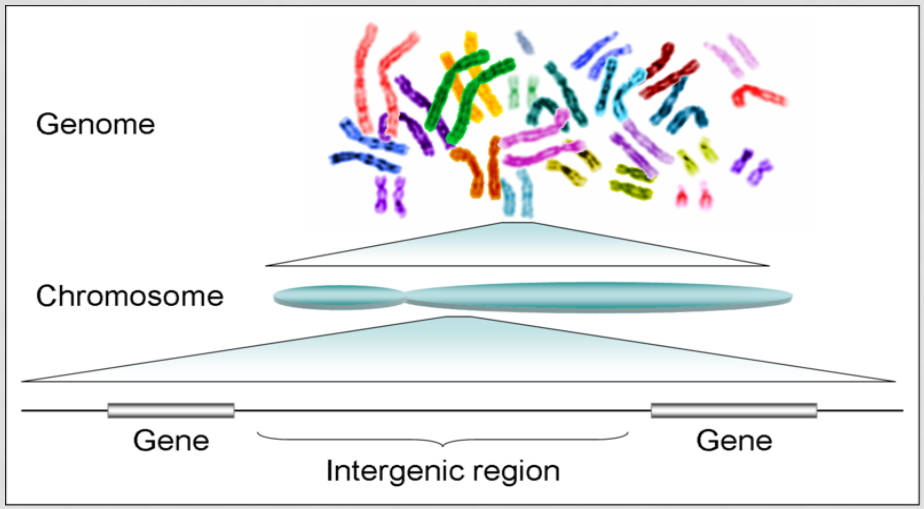
NGS data bottleneck

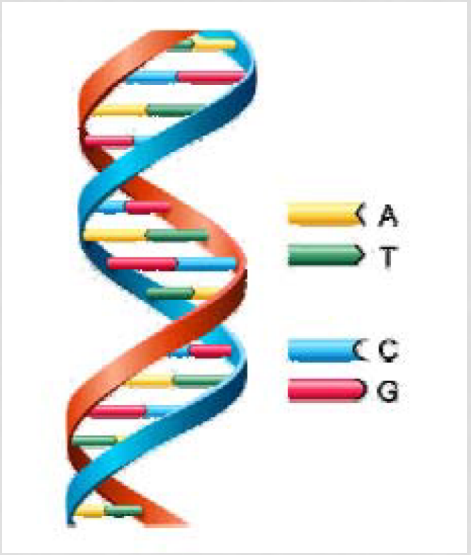
N users * 80M of reads

1 analysis = 80M of reads

GENETICS DATA

- NGS technologies chop the DNA/RNA molecules into small fragments called reads.
- (In our case)NGS data coming from the analysis of Chronic Myeloid Leukemia.
- The reads must be aligned to a reference genome (HG19).

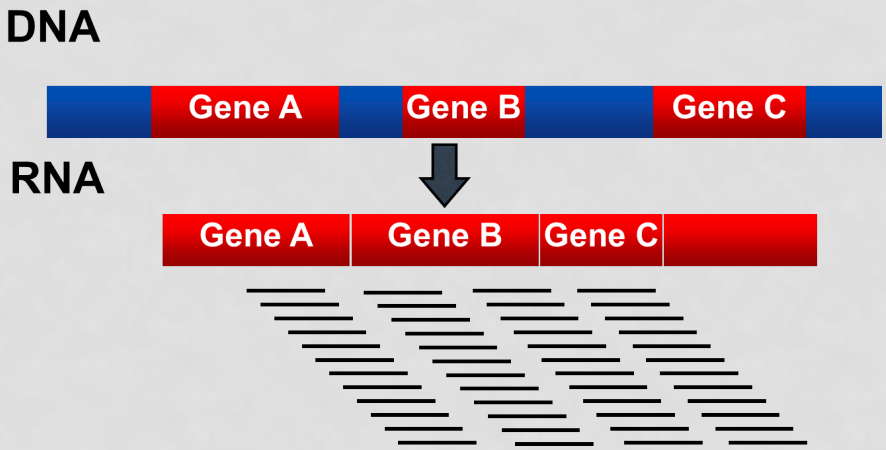




ALIGNMENT PHASE

Most common operation in these kind of analysis

- Reads are mapped onto a reference genome
- Alignment tool: Bowtie
- Two steps in the alignment process:
  - Candidate lookup: reduces the search space of the local alignment from the entire genome to a short list of possible alignment locations
  - Local alignment: generally solved using the Smith-Waterman

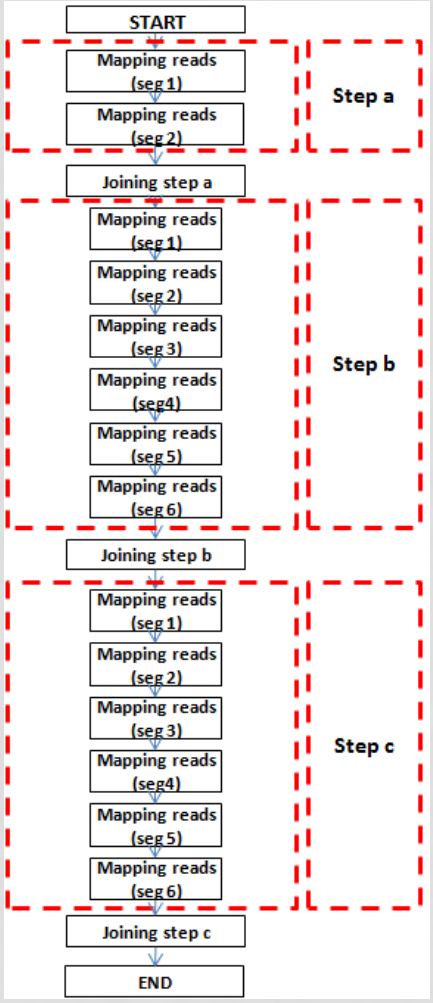


TOPHAT & BOWTIE

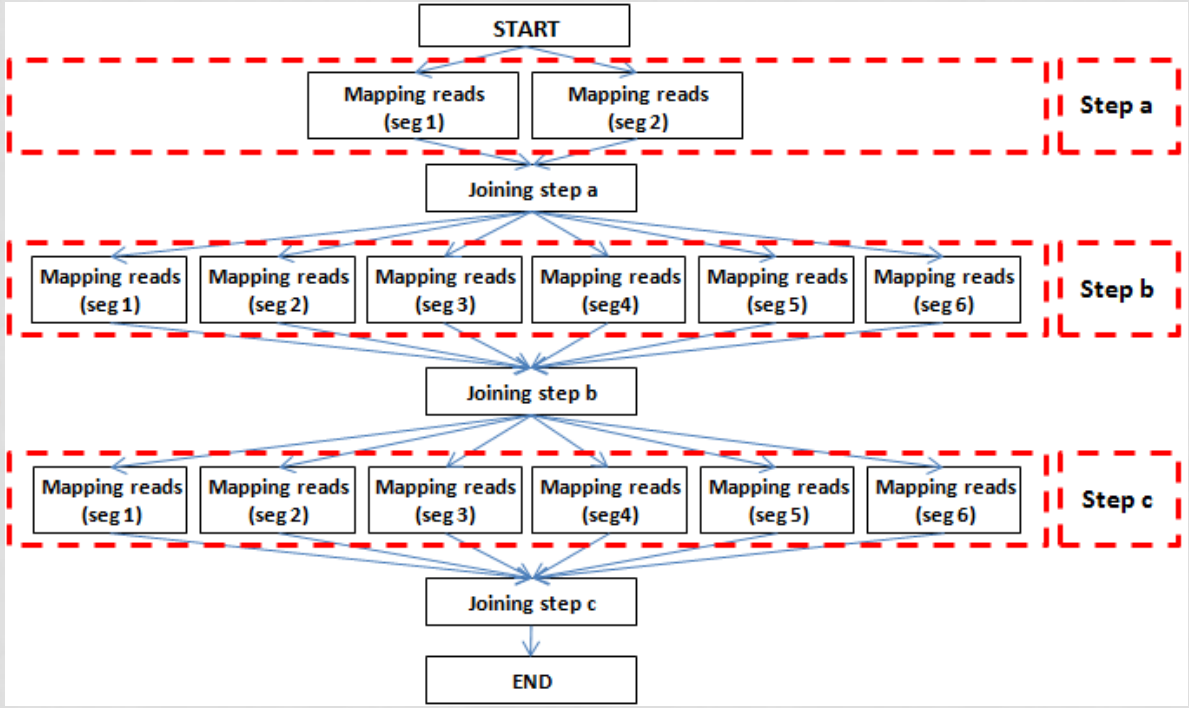
- TopHat is a fast splice junction mapper for RNA Seq reads.
- It aligns RNA-Seq reads to mammalian-sized genomes using the ultra high-throughput short read aligner Bowtie, and then analyzes the mapping results to identify splice junctions between exons.
- TopHat receives as input reads produced by the Illumina Genome Analyzer
- The short reads alignment is surely the most common operation in RNA-Seq data analysis. The purpose of the alignment is to map each short read fragment onto a genome reference.

TOPHAT OPTIMIZATION

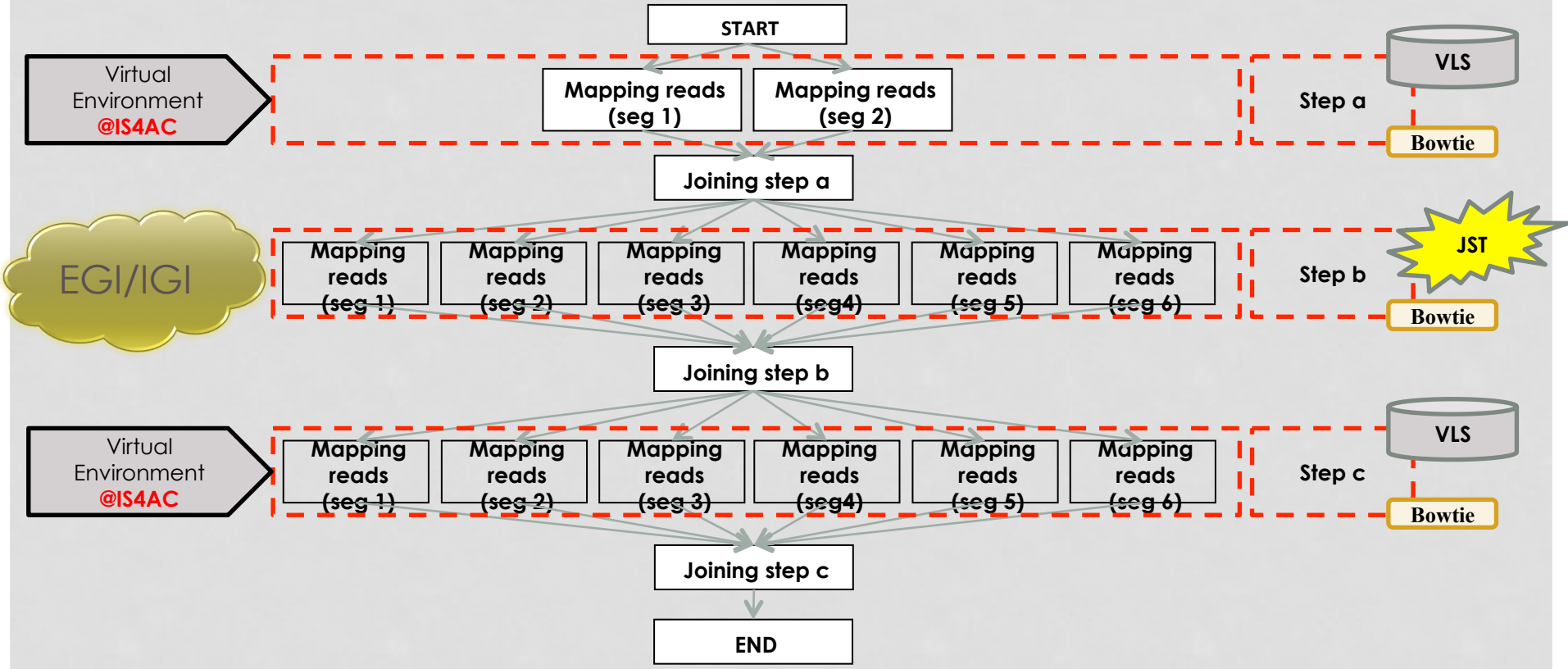
Original version



Distributed version



EGI - TOPHAT WORKFLOW



The size of the problem:

- Each run of the "step b" requires about 2GB (compressed) of reference genome and and about 500MB of input files
- It is easy to have thousands of runs to be executed over the grid infrastructures

HOW TO SUBMIT TO THE GRID ?

- The problem with this workflow is how to deal with grid job submission
- The main problems are:
  - The used scheduler is not “grid aware”
  - The job failure is very difficult to be handled in such a scenario
  - The status of the jobs is “done” only when the output file is available on the common storage server
  - It could be time consuming to check the status of each submitted job

THE SOLUTION: USING A "GATEWAY"

- We already have a framework for submitting jobs over grid infrastructure called Job Submission Tool (JST)
- This is a pilot system based on the interaction with a Task Queue databases.
- The workflow itself is already managed using a database
  - We can use the same tools and logic already in production at the Boella LAB(IS4AC).
- The JST is able to deal with job monitoring and re-submission of the failed jobs
  - It is also able to deal with the status of the stage-out
- JST is based on a RDBMS so it is easy to interact with the VLS

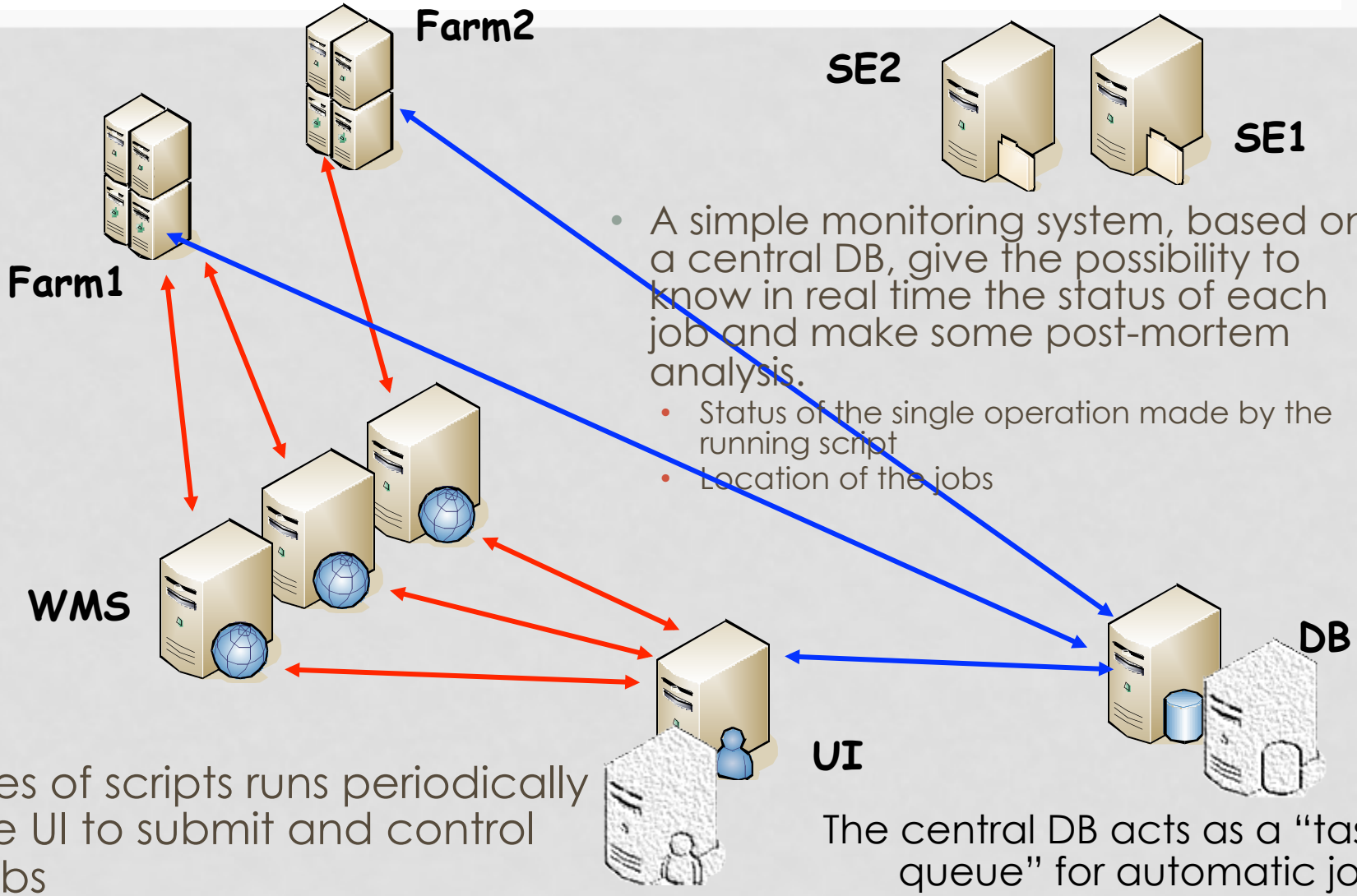
GENERIC "JST": PROCEDURES

- Each of the segment is an independent tasks
- In order to submit and control automatically (without human control) all the tasks, we use a procedure made by few elements:
  - DB Server -- Repository of all tasks
    - It is really useful to monitor the running jobs
    - It always have knowledge of done, running, undone and failed tasks
    - Can easily deal about prioritization, dependency, different kind of application or different users, etc.
    - Takes care of association between: task, executable, input and output files
    - In a separate table/DB there are also monitoring information like:
      - The exit status of each "error prone" action
      - CE/HOST of the execution
      - JOB_ID, TASK
      - …

GENERIC "JST": PROCEDURES (2)

- Job Wrapper -- manage all the general actions and make starts the real application on the WN
  - It takes care of "choosing" the task to be executed (querying the DB server)
  - It takes care of retrieving input files; storing output files (handling error conditions), updating the DB server accordingly
  - It launches the execution of the task and handles the exit_status updating the DB server accordingly
- Job Submitter -- Submit job to different WMS
  - It submits jobs to the grid infrastructure at different rates, taking care of the queued jobs
  - It uses different WMS in order to avoid single point of failure.
- Job Controller -- Checks for task status and can choose action to be executed
  - It can retrieve automatically the output
  - Advice the user (in this case the DB in the VLS) for the end of a task

JST SCHEMA

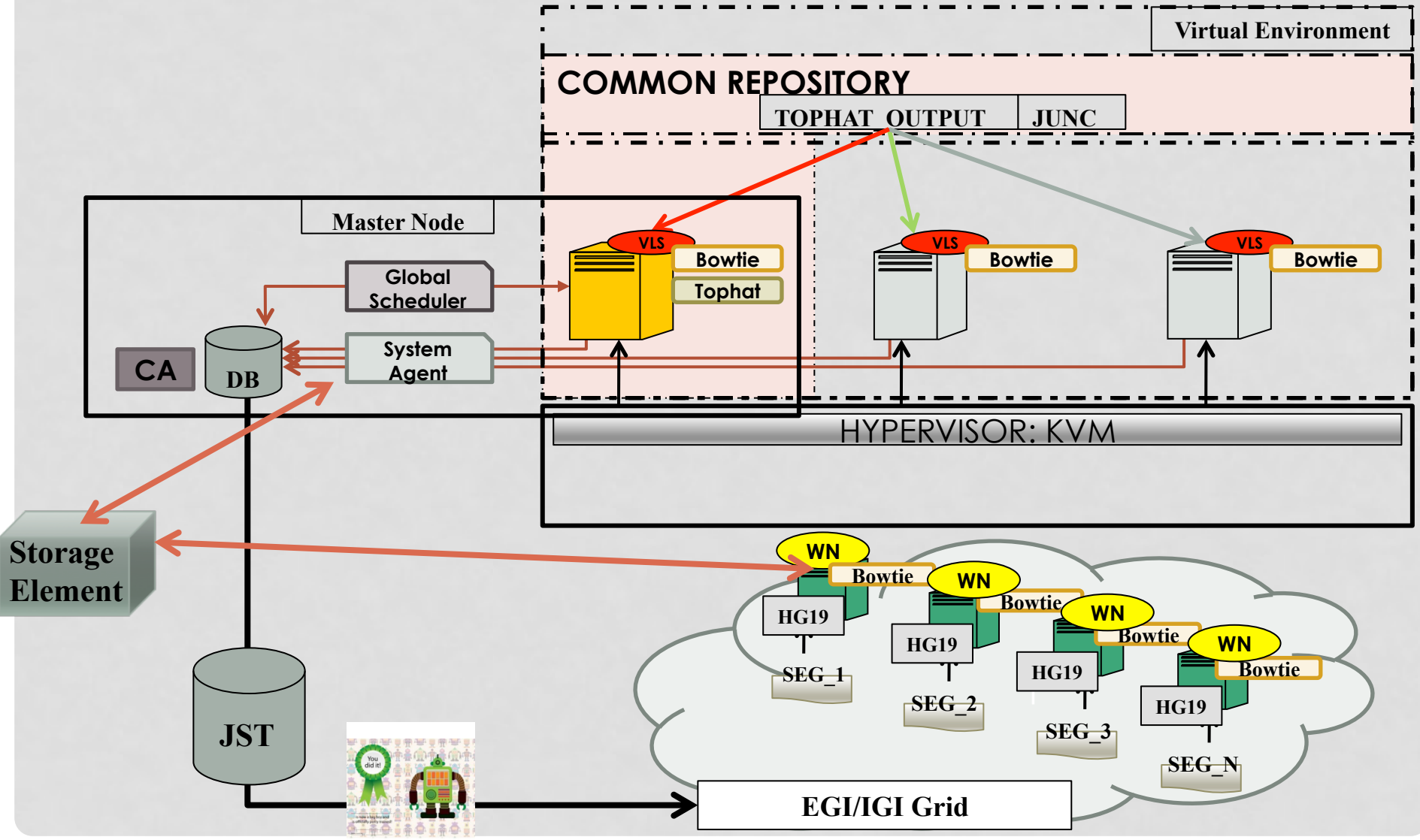


- A simple monitoring system, based on a central DB, give the possibility to know in real time the status of each job and make some post-mortem analysis.
  - Status of the single operation made by the running script
  - Location of the jobs

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

DISTRIBUTED ARCHITECTURE



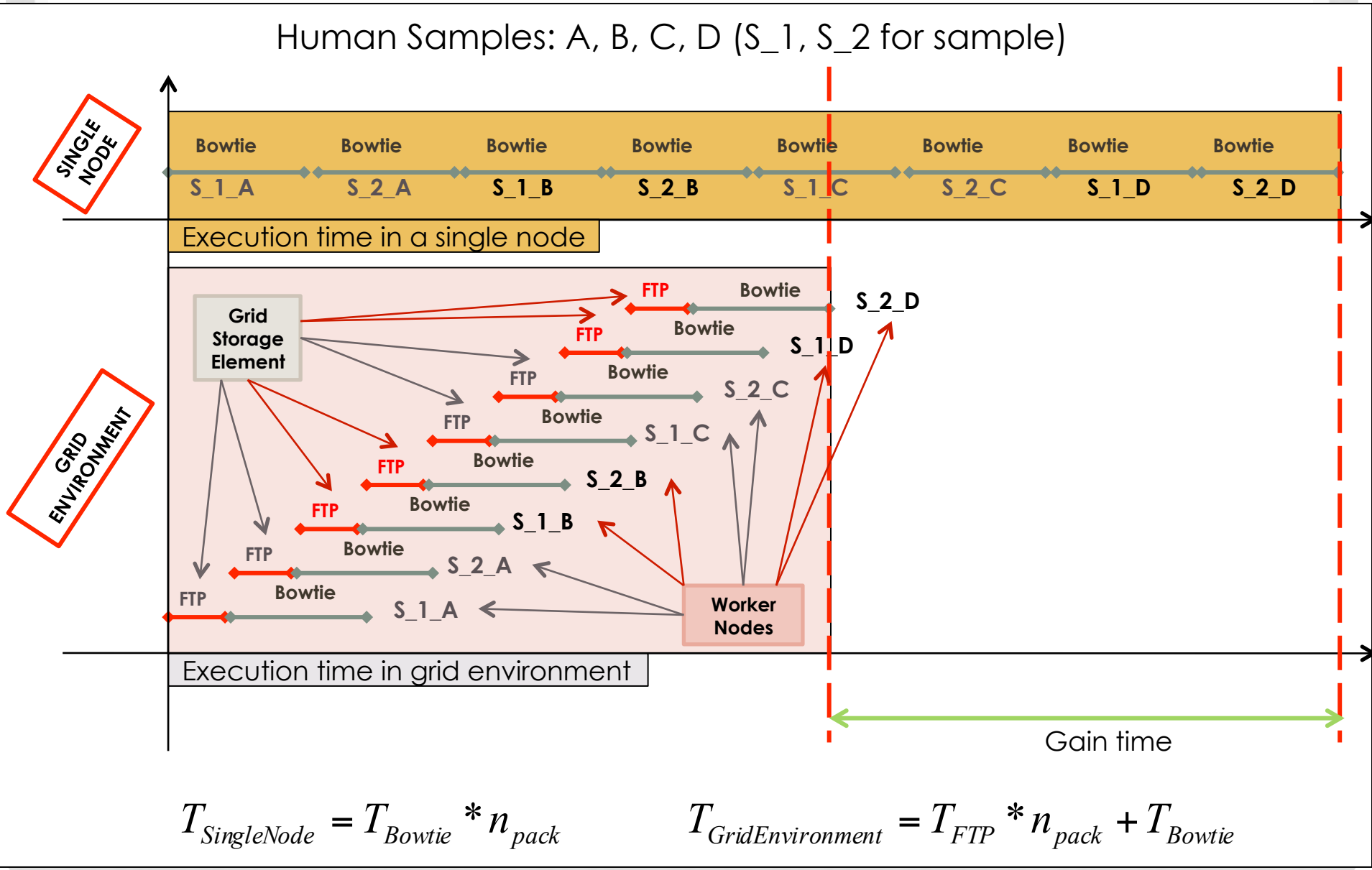
LEGENDA

- JST: Grid Scheduler (Grid Environment)
- VLS: Virtual Local Scheduler (Virtual Environment)
- System Agent (monitoring agent)
- HG19: Last Human Genome
- CR: Common Repository - NFS
- DataBase
- Hypervisor – KVM

DESCRIPTION OF THE DISTRIBUTED WORKFLOW

- The "step a" is executed as usual within the virtual environment on the Boella LAB, using the common repository
- After the completion of this step, TopHat goes on with the creation of the input file of the "step b" on the grid storage element…
- … and create the task for the "step b" both in the VLS and in the JST DBs
- This triggers the job submission over the grid infrastructure …
- … JST monitors and resubmit the jobs in case of failure
- If the output is correctly stored from the job on the Grid Storage Element the JST Job Controller updates VLS DB
- A daemon in the Boella LAB transfers the output of the grid job from the grid storage element to the "common repository"
- The rest of the workflow could run as usual

GRID PERFORMANCE

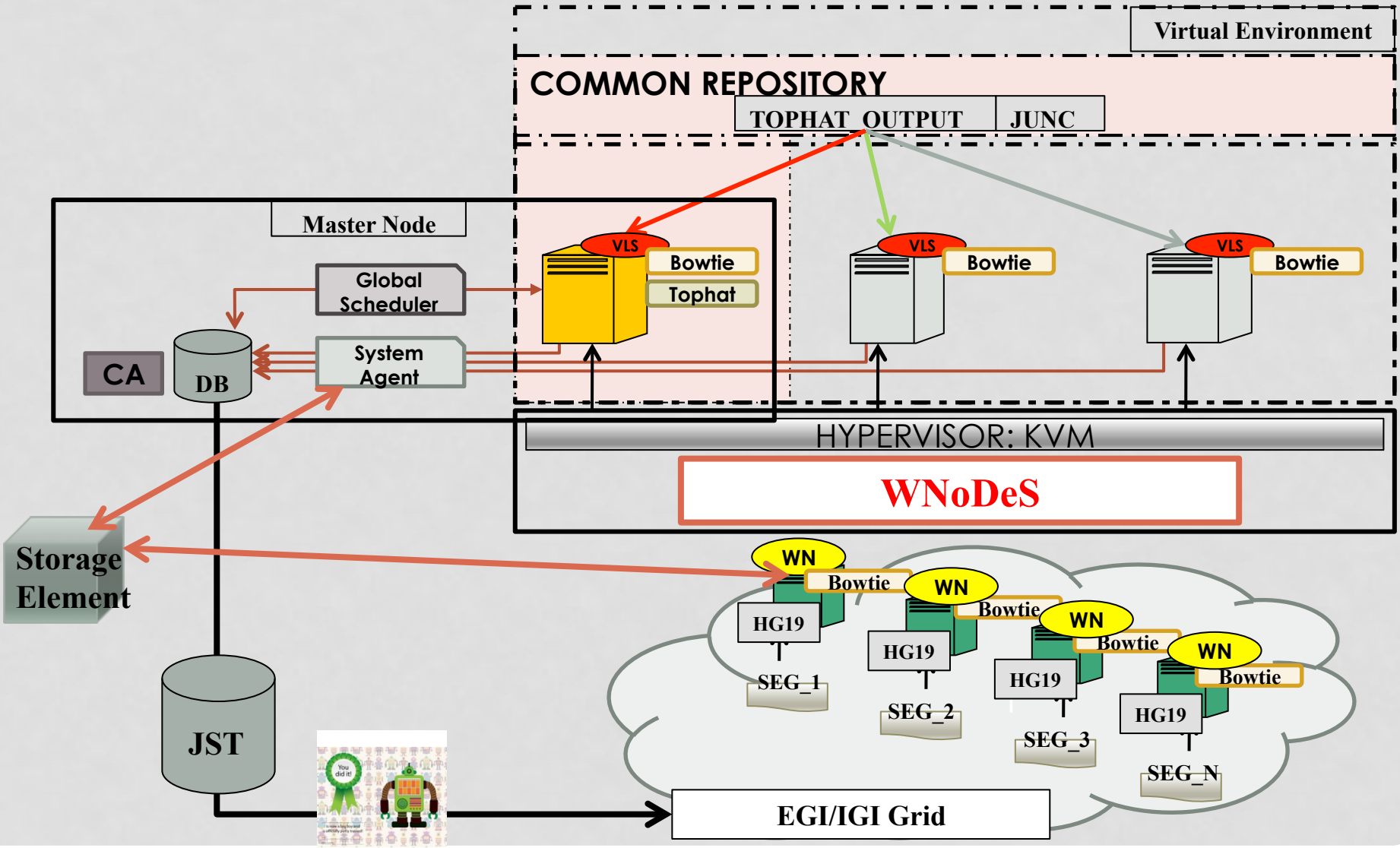


$$T_{SingleNode} = T_{Bowtie} * n_{pack} \qquad T_{GridEnvironment} = T_{FTP} * n_{pack} + T_{Bowtie}$$

OPEN ISSUES AND SOLUTIONS

- The main problem in this set-up is the data transfers between grid storage element and the Laboratory where the VLS is running
- To run the others steps of the workflow it is needed to have way more bandwidth to transfer data among nodes
- We compressed all the input and output files
  - Together with the reference genome
- In order to port also the other step will be needed to move the VLS from the laboratory network to a more fast network
- The best solution will be to deploy the VLS on a IaaS service on a farm with fast network
  - WNoDeS

DISTRIBUTED ARCHITECTURE – NEXT STEP



CONCLUSIONS AND FUTURE WORKS

- The porting of this kind of workflows could be of great interest for bioinformatics researchers as it will give the possibility to reduce the overall processing time
- Executing huge amount of runs over grid infrastructure could be far more easy by means of a gateways like JST
  - That takes cares of submitting, monitoring and resubmitting the jobs in case of failures
- The size of the data poses new challenges for running on the grid infrastructure
- Exploiting cloud solution like WNoDeS could help in dealing with this new challenges