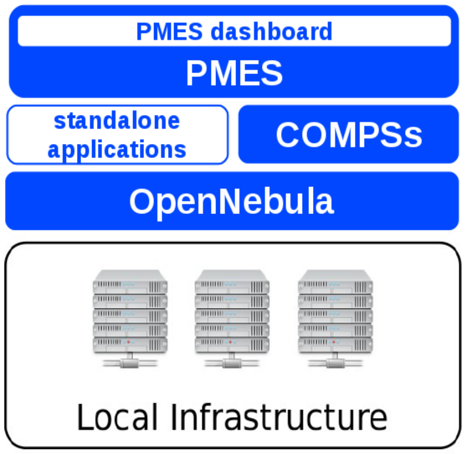
Guide to run TransPLANT applications

1. Introduction

TransPLANT cloud computing environment is made accessible over the internet through the PMES dashboard. PMES[1] (Programming Model Execution Service) is used as the entry point to the TransPLANT local infrastructure and operates OpenNebula[2], the middle-ware responsible of the virtual machines management where TransPLANT applications are executed.

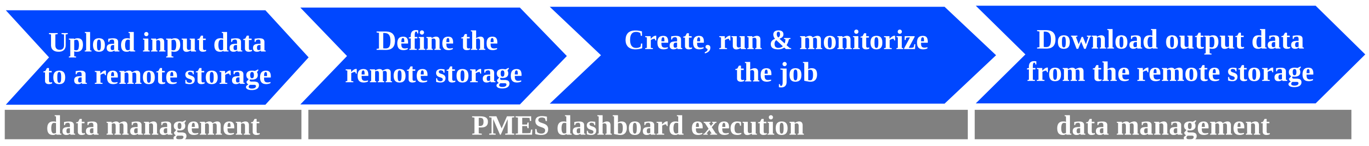


Fiaure 1 TransPLANT aeneral architecture

PMES dashboard handily enables the user to create, submit and monitorize job requests invoking either stand-alone or COMPSs[] applications. Such applications are merely the piece of code to be executed in the cloud infrastructure, and they can correspond to bare executable and web service wrappers, or to more complex pipelines. In any case, the distributed computational power is exploited, being the parallelism built-in in straightforward applications, or being featured or enhanced through COMPS superscalar, a programming model that pushes the task items to the available nodes in an orchestrated fashion.

In a more advanced mode, PMES dashboard also permits to create and configure a new application, yet a virtual machine image containing the application code needs to be manually uploaded to OpenNebula. TransPLANT users do not require to create new applications, since they already have an available collection of pilot applications - described below. More information regarding the introduction of new applications can be found in the *PMES dashboard manual*[].

Hence, the essential steps to be followed by a user are resumed here:


Upload input data to a remote storage
Define the remote storage
Create, run & monitorize the job
Download output data from the remote storage
data management
PMES dashboard execution
data management

The present document describes the two essential parts a user deals with when running an application using PMES: the data management and the application execution itself. Besides, the list of currently available TransPLANT applications is also detailed.

2. Data management: Input/Output storage

The cloud infrastructure needs to gain access to the user input data, just like the user will require access to the output data. In order to make such input/output operations possible, a remote storage must be defined, specifically an FTP site. Its URL and credentials are petitioned by the PMES dashboard and the data is transparently transferred.

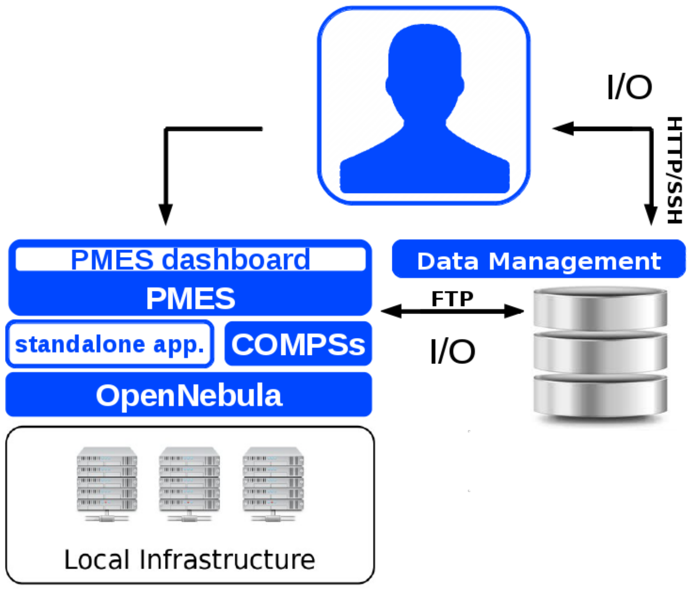


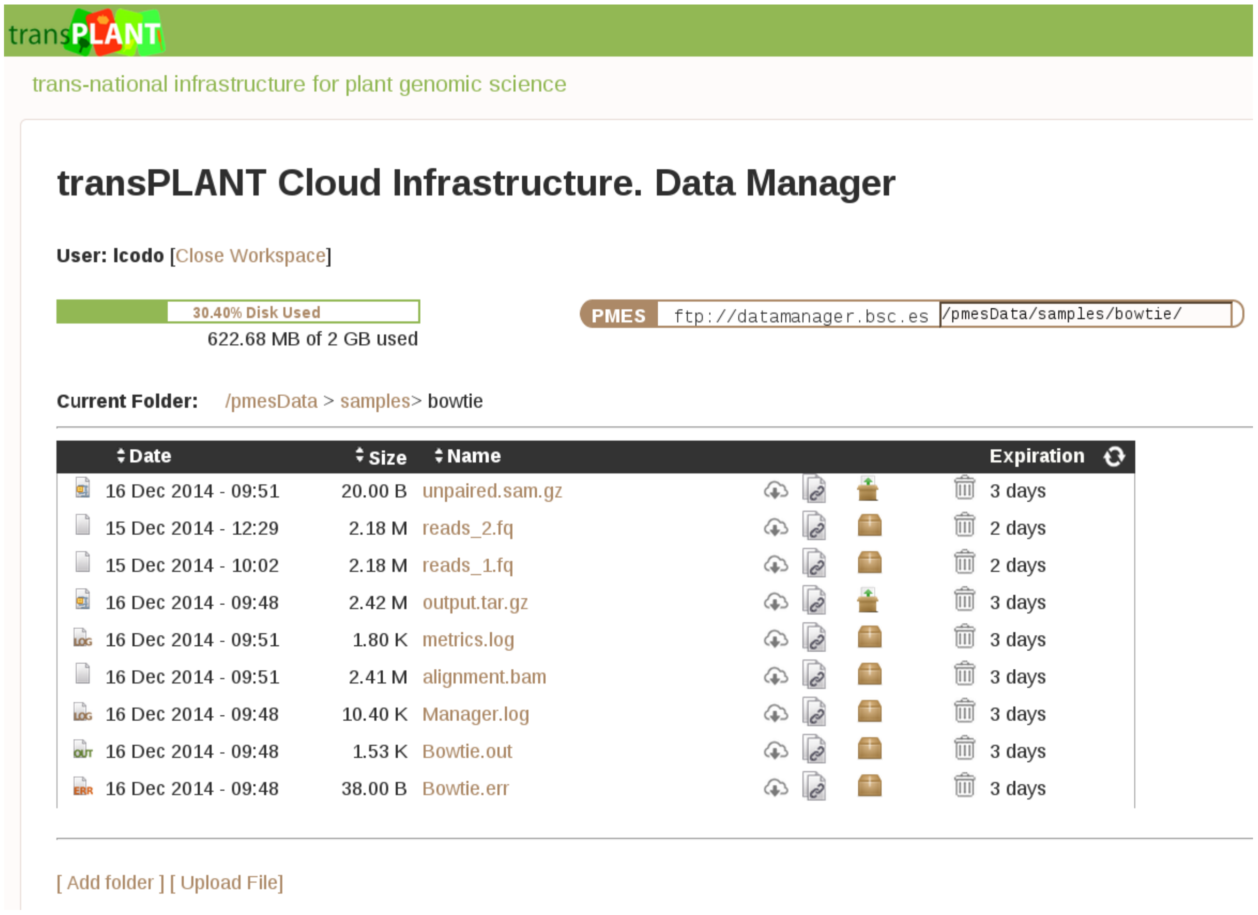
Figure 2: General transPLANT infrastructure description

Although any FTP server could be defined in PMES dashboard as the remote I/O storage, our cloud infrastructure operates within strict security constraints, hence, access is granted only to local FTP servers.

Two different servers are habilitated for transPLANT users, being one accessible through the Data Manager web portal, and the other through the cloud frontend itself. Account users are obtained by contacting transplantdb@bsc.es.

The Data Manager portal (login at http://transplantdb.bsc.es) is the usual strategy to transfer data to and from the transPLANT cloud (figure 3). Through this web application, users can upload PMES input data as well as download PMES results via HTTPS. This data is directly stored in the FTP site *ftp://datamanager.bsc.es*, to which PMES freely access.

Additionally, the cloud frontend would allow users who need to transfer large amounts of data, to use the conventional SSH protocol. Here, the FTP server in use would be *ftp://login-inb.bsc.es*, also with PMES access.



transPLANT

trans-national infrastructure for plant genomic science

transPLANT Cloud Infrastructure. Data Manager

User: lcodo [Close Workspace]

30.40% Disk Used

622.68 MB of 2 GB used

PMES ftp://datamanager.bsc.es /pmesData/samples/bowtie/

Current Folder: /pmesData > samples> bowtie

Date	Size	Name	Expiration
16 Dec 2014 - 09:51	20.00 B	unpaired.sam.gz	3 days
15 Dec 2014 - 12:29	2.18 M	reads_2.fq	2 days
15 Dec 2014 - 10:02	2.18 M	reads_1.fq	2 days
16 Dec 2014 - 09:48	2.42 M	output.tar.gz	3 days
16 Dec 2014 - 09:51	1.80 K	metrics.log	3 days
16 Dec 2014 - 09:51	2.41 M	alignment.bam	3 days
16 Dec 2014 - 09:48	10.40 K	Manager.log	3 days
16 Dec 2014 - 09:48	1.53 K	Bowtie.out	3 days
16 Dec 2014 - 09:48	38.00 B	Bowtie.err	3 days

[Add folder] [Upload File]

Figure 3: Data manager portal snapshot (login at http://transplantdb.bsc.es). The application lists the content of the local FTP site ftp://datamanager.bsc.es, and allows a set of file operations: uploads from local, directory creation, file visualization, compression options, etc.
This particular snapshot shows the files owned by the user 'lcodo' at the path /pmesData/samples/bowtie/. In fact, the listed files are the inputs required to run the Bowtie applicacion, and the set of outputs and log files returned by PMES once the application is finished.

Apart from these FTP servers, a special device, part of the local infrastructure, is also made ready for read-only data which might be frequently used and could reach a considerable size, for instance, resources like databases or libraries. For such cases, BSC has made available a data storage system called DATA2. It is accessible from the cloud and required by some TransPLANT applications.

3. Application execution (use case: MAKER)

The present section numerates the steps to follow in order to run a TransPLANT application via the PMES dashboard. To get a comprehensive and detailed description of the totality of the dashboard features, consult the *PMES dashboard manual* document.

3.1 Configuring PMES

The first thing to do is to log in https://transplantdb.bsc.es/pmes and get familiar with the new interface. The dashboard is conceived as a portal to a remote and a distributed computational infrastructure like the TransPLANT cloud. Hence, the central panel is occupied by the current state of the user's jobs. Jobs can be monitorized, debugged and edited, but first of all, the user should create and submit them.

In order to create a job request, PMES needs to be configured so that the framework will know from/to where the I/O data should to be transferred, as well as which are the specific environment variables to be set (if any). These arrangements are stored in the application, thus they only are required to be configured once.

Storage

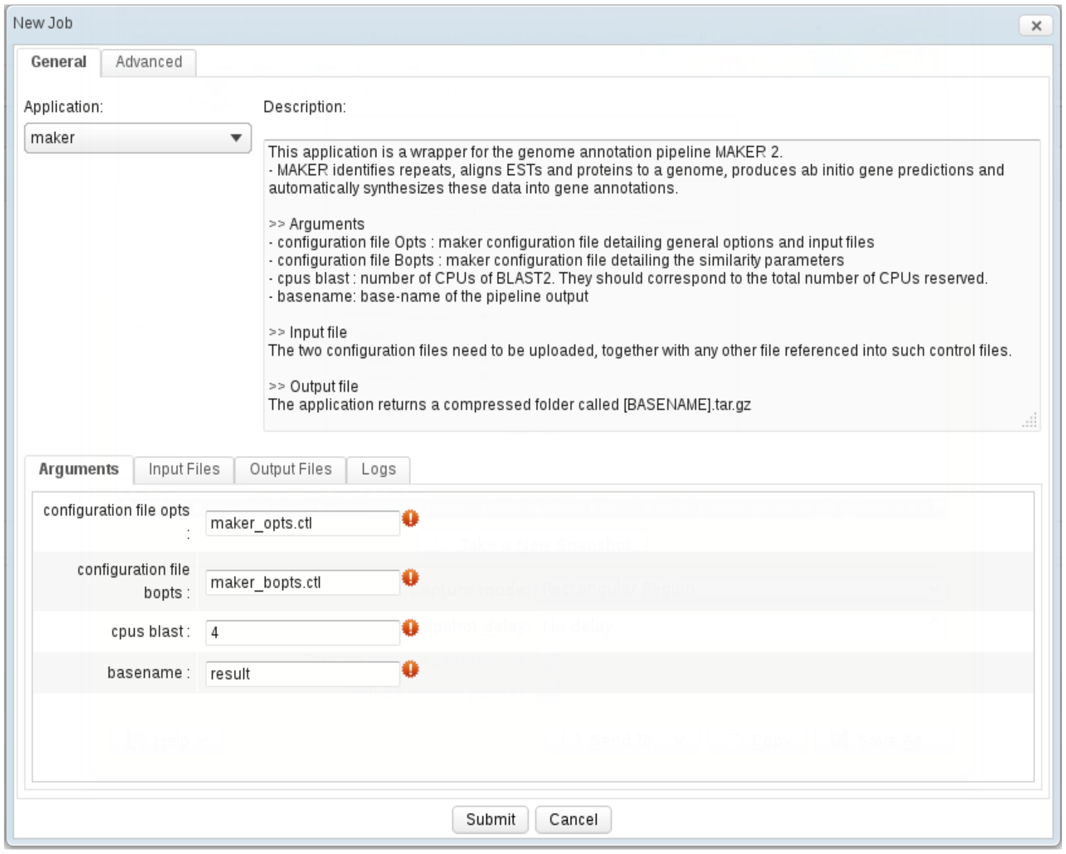
The storage where input and output data is located should be defined in *edit→storage*, on the top-left corner. Here the user will have the list of FTP sites to which the virtual machines will have access in order to upload/download data[1]. Type in the URL where you will upload your input data in the form ftp://myFTPsite/myFolder/ (notice the final slash). Later on, when attempting to connect to the storage, the application will petition your FTP site credentials.

3.2 Creating a job request

Here we are going to detail how to run MAKER application, a stand-alone job. In order to create a new job, we go to *Jobs→new→single Job*. If the application was prepared to be parallelized using COMPS , we would select *Jobs→new→COMPSs job*. The appearing window allows the user to choose the application we want to run, configure all its parameters and define from where the inputs should be retrieved and where the final output should be transferred to.

By choosing “maker” in the *Application* selector, its how-to description appears on the top-right corner frame (*fig. 3*). The text offers a short description of what the application does and details each of the arguments and the expected outputs.

1 When working on our cloud, remember that only two FTP servers are made accessible (ftp://datamanager.bsc.es *ftp://login-inb.bsc.es*) as refered in 'Data management: Input/Output storage'.



This application is a wrapper for the genome annotation pipeline MAKER 2.
- MAKER identifies repeats, aligns ESTs and proteins to a genome, produces ab initio gene predictions and automatically synthesizes these data into gene annotations.

>> Arguments
- configuration file Opts : maker configuration file detailing general options and input files
- configuration file Bopts : maker configuration file detailing the similarity parameters
- cpus blast : number of CPUs of BLAST2. They should correspond to the total number of CPUs reserved.
- basename: base-name of the pipeline output

>> Input file
The two configuration files need to be uploaded, together with any other file referenced into such control files.

>> Output file
The application returns a compressed folder called [BASENAME].tar.gz

configuration file opts :	maker_opts.ctl
configuration file bopts :	maker_bopts.ctl
cpus blast :	4
basename :	result

Figure 4: New job window

On the second half of the windows, four tabs indicate the four sequential pieces of information to fill in before submitting a job:

<u>*Arguments*</u>:

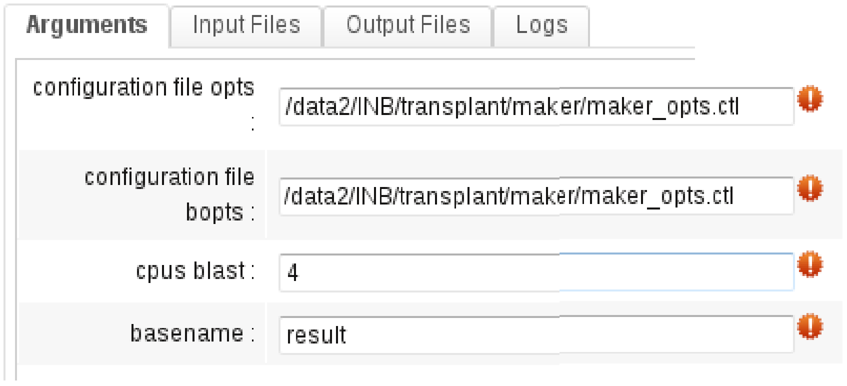
By completing the application arguments, the user is in fact building up the command line that will be later executed in the virtual machine. If an option already has a value, it corresponds to its default value. MAKER application takes only four argument which for this tutorial we will set as shown in figure 3.

Two of the arguments are files. In fact, they are configuration files containing, among other parameters, the path to other files required by the MAKER executable. This is precisely the data the user need to upload to an FTP site to make it accessible to the virtual machines. It is done in the *Input files* tab.

<u>*Input files:*</u>

In this tab, the user indicates which is the data that PMES should transfer to the cloud infrastructure. The three columns correspond to: (1) the user FTP site where the data is located, (2) the user FTP file or directory to be securely transferred and (3) the destination file or directory of the virtual machine where the data is to be transferred.

A pair of sample MAKER configuration files and the rest of the required files to test MAKER can be found in two different locations. The first is the DATA2 storage in the following directory: /data2/INB/transplant/maker. The second location is in the documentation section of the BSC transplant web (http://transplantdb.bsc.es/documents/samples/maker/). If the user choose to use the DATA2 sample inputs, there is no need to upload any input file to any FTP site, as PMES has access to DATA2 storage (read section Data management: Input/Output storage). Hence, no further work has to be done in this "*Input files*" tab. Nevertheless, the user should come back to the "*Arguments*" tab, and modify it accordingly:

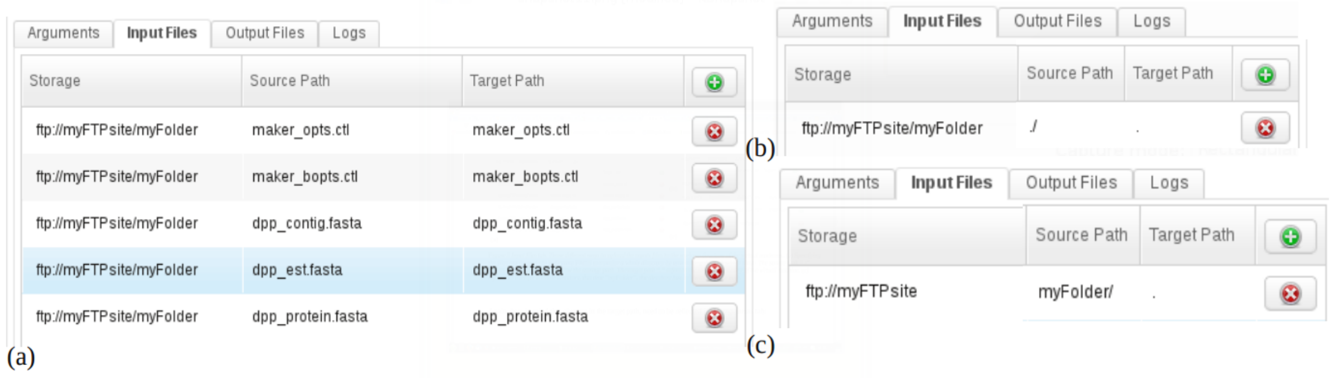


configuration file opts :	/data2/INB/transplant/maker/maker_opts.ctl
configuration file bopts :	/data2/INB/transplant/maker/maker_opts.ctl
cpus blast :	4
basename :	result

Figure 5: Argument tabs. Inputs files do not need to be uploaded if they are already accessible to PMES, for instance, through the DATA2 storage. Then, the arguments that are files, need to include the total path to indicate PMES where exactly the input files are located.

On the contrary, the user can download them or use their own data. In any case, the user should upload them to the FTP site specified when configuring PMES (ftp://myFTPsite/myFolder/).

There are different ways to fill in *input files* form, but in any case, the destination file must correspond to the file name as specified in the arguments section. Figure 6 shows several combinations that eventually upload the same files and match up with the arguments as previous reported:



(a)

Storage	Source Path	Target Path
ftp://myFTPsite/myFolder	maker_opts.ctl	maker_opts.ctl
ftp://myFTPsite/myFolder	maker_bopts.ctl	maker_bopts.ctl
ftp://myFTPsite/myFolder	dpp_contig.fasta	dpp_contig.fasta
ftp://myFTPsite/myFolder	dpp_est.fasta	dpp_est.fasta
ftp://myFTPsite/myFolder	dpp_protein.fasta	dpp_protein.fasta

(b)

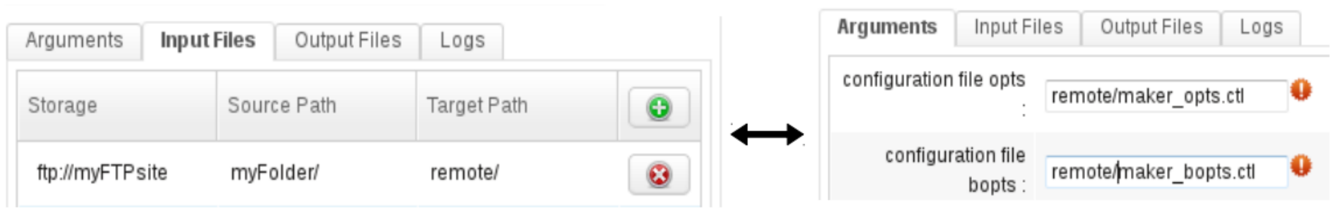
Storage	Source Path	Target Path
ftp://myFTPsite/myFolder	./	.

(c)

Storage	Source Path	Target Path
ftp://myFTPsite	myFolder/	.

Figure 6: Input files tab. Three different ways to upload the inputs files to the working directory of the virtual machines. (a) Specifying the files one by one. (b) Indicating PMES to transfer a whole directory by terminating the path with a slash. The source path "./" referes to the directory just below the storage path. The target path "." refers to the working directory of the virtual machine. (c) Again, we transfer a whole directory, this time "myFolder", the directory just below the storage path.

Again, bear in mind that changes in the target path, need to be reflected in the arguments tab:



Storage	Source Path	Target Path
ftp://myFTPsite	myFolder/	remote/

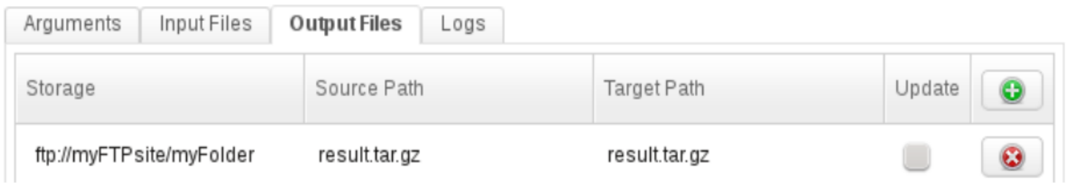
configuration file opts :	remote/maker_opts.ctl
configuration file bopts :	remote/maker_bopts.ctl

Figure 7: The remote target path needs to correspond with the file name specified in the arguments tab.

<u>*Output files:*</u>

In this tab we indicate the file or directory to upload the results back to the user FTP site once the application finished. Following the *input files* tab philosophy, the three columns correspond to: (1) the user FTP site where the output is to be transferred, (2) the remote location of the output in the virtual machine, and (3) the user FTP site destination file or directory. The name of outputs generated by the application and required to complete the *source path*, are usually specified in the text describing the application.

In our sample case, and according to the basename argument filled in the first tab, the form looks like follows:

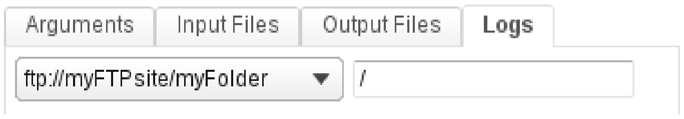


Storage	Source Path	Target Path	Update
ftp://myFTPsite/myFolder	result.tar.gz	result.tar.gz	☐

Figure 8: Output files tab.

<u>*Logs:*</u>

PMES dashboard returns a series of log files and in this tab, the user is asked where upload them. They are periodically updated as the job execution goes. In our case, we copy them in the same FTP folder than the output files:



ftp://myFTPsite/myFolder	/

Figure 9:Logs tab.

Reached this point, we are ready to simply send the job to the cloud infrastructure by pressing the "submit" button.

However, we also can modify some execution parameters through the *Advanced* tab, on the top-left corner of the "new job" window. The user can decide whether to reduce or increase the number of CPU's of the job, for instance. Changes in this section may need adjustments of some arguments and the user should be aware of this. In our case, the CPU's job needs to match with the argument *cpus* *blast*, because our MAKER application is not using MPI, thus BLAST2 is the program within the pipeline consuming more resources.

3.3 Monitorizing and debugging the job

Once the job is submitted, it appears in the central pannel, together with its state (PENDING/FINISHED/FAILED). Selecting a job, the user obtains more information. The icon displays the states of the different virtual machines that compound that job. The icon allows to edit the job and launch it again, if needed. In the lower panel, the three types of job logs are presented: Manager.log (the OpenNebula logfile), Stdout.log (the application standard output) and Stderr.log (the application standard error). All of them are periodically uploaded to the FTP site, as defined while setting the job. When the job successfully finishes, the output files are uploaded to the FTP site, as the user defined in the *output files tab*.

4. Application repository

GENE DETECTION		
Application Type	COMPSs	
This application is a pipeline that detects genes in a genome using a reference protein from a closely related species using Genewise. It makes the process more efficient by restricting the Genewise executions to the most relevant regions with the help of Blast and Blast2Gene. - GENEWISE: tool that compares a protein sequence to a genomic DNA sequence, allowing for introns and frameshifting errors. (PMID: 10779496) - BLAST: The Basic Local alignment tool that finds regions of local similarity between sequences. - BLAST2GENE: program that allows a detailed analysis of genomic regions containing completely or partially duplicated genes.		
Arguments	Genome file	A genome multifasta file
	Proteins file	Collection of related protein sequences (fasta).
	Results directory	Destination of results
Input Files	Both, the genome and the collection of proteins	
Output Files	Name of the results directory	
Sample Files		
Special requirements		

BLAST		
Application Type	COMPSs	
This application is a wrapper of the widely used local alignment tool BLAST. The application allows the user to compose the complete command line. - BLAST: program that compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches.		
Arguments	blast binary	Path to the Blast binary. *Default: /binary/blastall*
	query sequence	Input sequence filename in FASTA o multifasta.
	database	Path to the database. *Default: /data2/INB/ddbb/NCBI/bin/blast/[DB_type]*. Available DB_types are: - all_contig: Homo sapiens build 37.3 genome - env_nt: environmental samples - est_human: GenBank Human EST entries - est_mouse: GenBank Mouse EST entries

BLAST		
		- est: GenBank+EMBL+DDBJ sequences from EST Divisions - est_others: GenBank non-mouse and non-human EST entries - htgs: Unfinished High Throughput Genomic Sequences sds ssds - human_genomic: NCBI genome chromosomes - human - nt: Nucleotide collection (nt) - other_genomic: NCBI genome chromosomes - other - patnt: Nucleotide sequences from the Patent division - pdbnt: PDB nucleotide database - refseq_genomic: NCBI Genomic Reference Sequences - refseq_rna: NCBI Transcript Reference Sequences - env_nr: Proteins from WGS metagenomic projects - nr: non-redundant GenBank CDS translations +PDB +SwissProt +PIR +PRF - pataa: Protein sequences from the Patent division - pdbaa: PDB protein database - refseq_protein: NCBI Protein Reference Sequences - swissprot: Non-redundant UniProtKB/SwissProt
	fragment number	Number of fragment of the input sequence to use
	command line args	Comand line arguments for blast. Example: “-n blastp”. Options: http://www.ncbi.nlm.nih.gov/staff/tao/URLAPI/blastall/
	temporary directory	Path to the machine temporary diretory. *Default: /tmp*
	output file	Filename containing the results of the application
	debug	Activaves the debug mode of the application (optional).
Input Files	The query sequence file.	
Output Files	The file named as defined in the argument “Output file”	
Sample Files		
Special requirements	The application requires access to the DATA2 data storage, where all databases are stored.	

MAKER		
Application Type	stand alone	
This application is a wrapper for the genome annotation pipeline MAKER 2. - MAKER: identifies repeats, aligns ESTs and proteins to a genome, produces *ab initio* gene predictions and automatically synthesizes these data into gene annotations.		
Arguments	configuration file Opts	Maker configuration file detailing general options and input files

MAKER		
	configuration file Bopts	Maker configuration file detailing the similarity parameters
	cpus blast	Number of CPUs of BLAST2. They should correspond to the total number of CPUs reserved.
	basename	Base-name of the pipeline output
Input Files	The two configuration files need to be uploaded, together with any other file referenced into such control files.	
Output Files	The application returns a compressed folder called [BASENAME].tar.gz	
Sample Files	Configuration files: http://transplantdb.bsc.es/documents/samples/maker/maker_opts.ctl http://transplantdb.bsc.es/documents/samples/maker/maker_bopts.ctl Files refered into the configuration file Opts: http://transplantdb.bsc.es/documents/samples/maker/dpp_contig.fasta http://transplantdb.bsc.es/documents/samples/maker/dpp_est_fasta	
Special requirements		

AbySS		
Application Type	stand-alone	
This application runs ABySS. The parallel version is implemented using MPI and it is capable of assembling large genomes. - ABYSS: *de novo*, parallel, paired-end sequence assembler, designed for short reads. The single-processor version is useful for assembling genomes up to 100 Mbases in size.		
Arguments	kmer size	k-mer size to test. *Format: k=[value]*.
	number of threads	Number of threads used in the execution. *Format: j=[value]*
	name	Name to identify the execution (i.e., specie) . *Format=[value]*
	pair-end libraries	A name to identify the paired-end library. *Format: lib='[value1], [value2]'. Default: lib='pe1'*
	mate-pair libraries	A name to identify the mate-pair libraries. *Format: mp='[value1], [value2]'. Default: mp='mp1'*
	pe library files	The name of the files with the paired-end reads. *Format: lib_name='[value1] [value2]'. Default: pe1='file1 file2'*
	mp library files	The name of the files with the mate-pair reads. *Format: lib_name='[value1] [value2]'. Default: mp1='file1 file2'*
Input Files	The fastq files required to run the application are those corresponding to the arguments 'pe library files' and 'mp_library files'.	

AbySS	
Output Files	The application returns a folder called [NAME].tar.gz
Sample Files	As an example, you can run the assembly of staphylococcus_aureus with the data from the GAGE project [4]. The libraries are in: http://gage.cbcb.umd.edu/data/index.html
Special requirements	

BWA		
Application Type	stand alone	
This application is a sequential pipeline that uses BWA to align paired-end reads against a reference genome and converts the resulting alignment into a BAM file using SAM Tools. - BWA (Burrows-Wheeler Alignment): software package for mapping low-divergent sequences against a large reference genome. - SAM Tools: provides various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format.		
Arguments	fastq1	paired-end reads file 1 in fastq format.
	fastq2	paired-end reads file 2 in fastq format.
	Reference Genome	indexed reference genome (Ensembl release 20). Options: arabidopsis_lyrata, arabidopsis_thaliana, brachypodium_distachyon, brassica_rapa, chlamydomonas_reinhardtii, cyanidioschyzon_merolae, glycine_max, hordeum_vulgare, medicago_truncatula, musa_acuminata, oryza_brachyantha, oryza_glaberrima, oryza_indica, oryza_indica, oryza_sativa, physcomitrella_patens, populus_trichocarpa, selaginella_moellendorffii, setaria_italica, solanum_lycopersicum, solanum_tuberosum, sorghum_bicolor, triticum_aestivum, triticum_urartu, vitis_vinifera, zea_mays
	Output basename	Base-name of the pipeline output
Input Files	The files required to run the application correspond to the arguments fastq1 and fastq2.	
Output Files	The pipeline generates an output file called [BASENAME].bam	
Sample Files	Fastq1: http://transplantdb.bsc.es/documents/samples/bwa/1.fastq.gz Fastq2: http://transplantdb.bsc.es/documents/samples/bwa/2.fastq.gz Reference Genome: arabidopsis_thaliana Output base-name: results	
Special	The application requires access to the DATA2 data storage, where Ensembl database is	

BWA	
requirements	stored.

EXONERATE		
Application Type	stand alone	
Exonerate is a generic tool for pairwise sequence comparison. It allows you to align sequences using many alignment models, either exhaustive dynamic programming, or a variety of heuristics.		
Arguments	Query	query sequence/s required. These must be in a FASTA format file. Single or multiple query sequences may be supplied in one or more files.
	Target	target sequence/s required. Also, must be in a FASTA format file. As the query sequences, single or multiple target sequences and files may be supplied. Are also available though the shared storage, the Plant Esembl genomes (release 20) and the GRCh37 human genome. In order to use them, specify one of the following options: arabidopsis_lyrata, arabidopsis_thaliana, brachypodium_distachyon, brassica_rapa, chlamydomonas_reinhardtii, cyanidioschyzon_merolae, glycine_max, hordeum_vulgare, medicago_truncatula, musa_acuminata, oryza_brachyantha, oryza_glaberrima, oryza_indica, oryza_sativa, physcomitrella_patens, populus_trichocarpa, selaginella_moellendorffii, setaria_italica, solanum_lycopersicum, solanum_tuberosum, sorghum_bicolor, triticum_aestivum, triticum_urartu, vitis_vinifera, zea_mays, Homo_sapiens.chromosome.NUM where NUM = chr. Num, X, Y or MT]
	Options	original optional Exonerate parameters. Consult them in: http://www.ebi.ac.uk/~guy/exonerate/exonerate.man.html Default: --model ungapped --bestn 0 --score 100 --exhaustive FALSE --showtargetgff yes
	Chunks Query	equivalent to querychunktotal. Number of chunks into which the query will by split in order to run on different nodes (*).
	Chunks Target	equivalent to targetchunktotal. Number of chunks into which the

EXONERATE		
		target will by split in order to run on different nodes (*).
	Output basename	basename of the output
Input Files	The Query FASTA file/s need to be uploaded. And also the Target FASTA file/s, unless the target correspond to a sequence included in the Plant Esembl genomes (release 20) and or the GRCh37 human release. In such cases, only the “Target“ parameter need to be specified.	
Output Files	The application generates a GZIP file containing the concatenation of all Exonerate outfiles. File: BASENAME.gz	
Sample Files	Query: http://transplantdb.bsc.es/documents/samples/exonerate/TAIR_partial.fa Target: arabidopsis_lyrata Chunks Query: 1 Chunks Target: 4 Adanved tab → Cores: 4 (*) (*) Consider that the total number of cores reserved in the cloud should correspond to: Chunks-Query multiplied by Chunks-Target. If, for example, you wish to split the target database into 3 parts and the query into 2, 6 exonerate jobs would run, so 6 cores need to be reserved. The granularity of the chunk goes down to a single sequence.	
Special requirements	The application requires access to the DATA2 data storage, is no Target file is uploaded and instead, Ensembl or GRCh37 databases are specified.	

TOPHAT

Application Type	stand alone

This application is a wrapper for the TopHat program. Additionally, it runs bowtie2-build to build the genome bowtie2 indexes.

[bowtie2-build] → TopHat

- TOPHAT: is a program that aligns RNA-Seq reads to a genome in order to identify exon-exon splice junctions. It is built on the ultrafast short read mapping program Bowtie.

The wrapper, as the original software, behaves differently according to the given arguments. For instance:

- Align reads: <read> <read2> <index> <cpus> <output> <options [...]>
- Build transcriptome from GTF: <index> <cpus> <output> <options [--GTF input.gtf --transcriptome -index newDir/prefix ...] >
- Resume: <output> <options [--resume tophat_out] >

Arguments	read	A comma-separated list of files containing reads in FASTQ or FASTA format. For paired-end reads, this should be the *_1 files.
	read2	A comma-separated list of files containing reads in FASTA or FASTA format. Only used for paired end reads. It contains the *_2

TOPHAT		
		set of files, which must appear in the same order as the *_1 files.
	index	Genome to be searched. The parameter accepts two types of values: -1 : Bowtie2 indexes basename. The program will look index*bt2 and index*rev.bt2 files, which require to be uploaded (Input tab). -2: comma-separated list of files containing reads in FASTA format. They will be indexed using Bowtie2-build program.
	cpus	Number of threads to align reads. They should correspond to the number of cores reserved in the 'advanced' tab. Notice that Bowtie2-build do not parallelise.
	output	Basename of the directory in which TopHat will write all of its output.
	topHat options	Native options of TopHat program. Notice that some options are input files (i.e. -j file.juncs), therefore, they require to be uploaded to the cloud though the 'input' tab. Check options at: http://ccb.jhu.edu/software/tophat/manual.shtml
Input Files	When running Tophat to align RNA-Seq reads, they need to be uploaded to the virtual machine in FASTQ or FASTA format. The target name or target path (3th column), should correspond to the argument 'read' and 'read2'. When using pre-built indexes in 'index', *.1.bt2 and *.rev.1.bt2 files need to be uploaded. When new indexes are to be build, the original genomic FASTA files should be transfered. Additionally, when user supplies their own insertions, deletions, or list of known transcripts, the corresponding .GTF, .BED, .JUNCS, etc., files need to be correclty specified within 'TopHat options', as well as uploaded through the 'input' tab.	
Output Files	The application returns a [OUTPUT].tar.gz, a compressed version of the standard Tophat ouput directory. When a new transcriptome index is created (--GTF & --transcriptome-index within 'TopHat options'), it is included in the [OUTPUT].tar.gz, so it can be reused in other TopHat runs.	
Sample Files	read: http://transplantdb.bsc.es/documents/samples/tophat/reads_1.fq read2: http://transplantdb.bsc.es/documents/samples/tophat/reads_2.fq index: http://transplantdb.bsc.es/documents/samples/tophat/indexes/ cpus: 8 topHat options: -r 20	
Special requirements		

AUGUSTUS		
Application Type	stand alone	
AUGUSTUS is a program that predicts genes in eukaryotic genomic sequences. It can be used as an ab initio program, but the program may also incorporate hints on the gene structure coming from extrinsic sources such as EST, MS/MS, protein alignments and synthenic genomic alignments.		
Arguments	query sequence	The query file contains the DNA input sequence and must be in uncompressed (multiple) fasta format
	specie	Choose one of the followings, for which Augustus has been trained: human, fly, arabidopsis, brugia, aedes, tribolium, schistosoma, tetrahymena, galdieria, maize, toxoplasma, caenorhabditis, , aspergillus_fumigatus, aspergillus_nidulans, aspergillus_oryzae, aspergillus_terreus, botrytis_cinerea, candida_albicans, candida_guilliermondii, candida_tropicalis, chaetomium_globosum, coccidioides_immitis, coprinus, coprinus_cinereus, cryptococcus_neoformans_gattii, cryptococcus_neoformans_neoformans_B, cryptococcus_neoformans_neoformans_JEC21, debaryomyces_hansenii, encephalitozoon_cuniculi_GB, eremothecium_gossypii, fusarium_graminearum, histoplasma_capsulatum, kluyveromyces_lactis, laccaria_bicolor, lamprey, leishmania_tarentolae, lodderomyces_elongisporus, magnaporthe_grisea, neurospora_crassa, phanerochaete_chrysosporium, pichia_stipitis, rhizopus_oryzae, saccharomyces_cerevisiae_S288C, saccharomyces_cerevisiae_rm11-1a_1, schizosaccharomyces_pombe, trichinella, ustilago_maydis, yarrowia_lipolytica, nasonia, tomato, chlamydomonas, amphimedon, pneumocystis
	optional parameters	Original optional Augustus parameters. Default: --strand=both --genemodel=partial --maxDNAPieceSize=200000 Consult http://augustus.gobics.de/binaries/README.TXT to modify the default parameters or add others.
	output	Base-name of the GFF that will be generated
Input Files	Augustus only requires the FASTA file corresponding to the 'query sequence' parameter.	
Output Files	The pipeline generates an output file called [OUTPUT].gff	
Sample Files	query sequence: http://transplantdb.bsc.es/documents/samples/augustus/sequence.fa specie: arabidopsis	
Special requirements		

BOWTIE		
Application Type	stand alone	
This application is essentially a wrapper for the fast aligner Bowtie2. Additionally, the wrapper allows to build the indexes from input reference genomes, if the pre-build indexes of ENSEMBL Plants full version genomes were not suitable. The application includes the following software: [bowtie2-build] → bowtie2 → samtools - BOWTIE2: It is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. It is particularly good at aligning reads of about 50 up to 100s or 1,000s of characters to relatively long (e.g. mammalian) genomes. - BOWTIE2-build: The program indexes the genome with an FM Index (based on the Burrows-Wheeler Transform or BWT) to keep its memory footprint small. This step is only performed when the user supplies a genome sequence to be indexed. - SAM Tools: provide various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format.		
ents	read	unpaired reads to be aligned OR paired-end reads containing mate 1s. FASTQ is the default format. A list of comma-separated read files is also accepted.
	read2	paired-end reads containing mate 2s when mate 1s is specified in argument read. FASTQ is the default format. A list of comma-separated read files is also accepted.:/BWA
	reference	reference genome against whom reads are aligned. The parameter accept two possible type of data: 1. genome sequence : comma-separated list of FASTA files containing the reference sequences to be aligned to. From them, FM indexes will be created running bowtie2-build program. 2. genome index: base-name of the index of a pre-build reference genome. The base-name is the name of any of the index files up to but not including the final *.1.bt2, *.rev.1.bt2, etc. User can upload their own index files (2.a), or use those available in the shared disk. Following are the basenames of the pre-build genomes (2.b): arabidopsis_lyrata, oryza_sativa arabidopsis_thaliana, physcomitrella_patens brachypodium_distachyon, populus_trichocarpa brassica_rapa, selaginella_moellendorffii chlamydomonas_reinhardtii, setaria_italica cyanidioschyzon_merolae, solanum_lycopersicum glycine_max, solanum_tuberosum

BOWTIE		
		hordeum_vulgare medicago_truncatula musa_acuminata oryza_brachyantha oryza_glaberrima oryza_indica sorghum_bicolor triticum_aestivum triticum_urartu vitis_vinifera zea_mays
	only index	yes|no. Return only the indexes build from 'reference' sequence/s. As no reads will be aligned, the following arguments will be ignored:'read', 'read2', 'cpus', 'bowtie2-parameters'. *Default: no*
	cpus	number of thhreads created by bowtie2. They should correspond to the number of cores reserved in the 'advanced' tab. Notice that bowtie2-build do not parallelise.
	output	base-name of the final packed and compressed output
	bowtie2-build parameters	native options of bowtie2-build program. Consult them in: http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#the-bowtie2-build-indexer Default: -f --offrate 5
	bowtie2 parameters	native options of bowtie. Consult them in http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml#command-line Default: --end-to-end --sensitive --un-gz unpaired.sam.gz --met-file metrics.log --time
Input Files	Read files in FASTQ or Illumina's QSEQ format (bowtie2 parameters = --qseq) need to be uploaded to the virtual machine. The target name or target path (3th column) should correspond to the argument 'read' and 'read_2'. Additionally, when user supplies their own indexes, all *.1.bt2 and *.rev.1.bt2 files need to be uploaded, and their path and base-names set in the argument 'reference'. However, if 'reference' argument refers to genomic sequences, the files to upload are the FASTA reference genome sequences.	
Output Files	The application returns a [OUTPUT].tar.gz file. It will contain a variable number of files, created according to the provided options: • indexes files (reference*.bt2 and reference.rev.*.bt2) • SAM and BAM read's alignments (alignment.sam, alignment.bam) • file containing unpaired reads that fail to align (named according to --un options) • file containing unpaired reads that align at least once (named according to --al options)	

BOWTIE	
	• file containing paired-end reads that fail to align concordantly (named according to --un-conc options) • file containing paired-end reads that align concordantly at least once to file (named according to --al-conc options) • bowtie2 metrics file (named according to --metrics-file)
Sample Files	read: http://transplantdb.bsc.es/documents/samples/bowtie/reads_1.fq read2: http://transplantdb.bsc.es/documents/samples/bowtie/reads_2.fq reference: arabidopsis_lyrata only_ index: no
Special requirements	The application requires access to the DATA2 data storage, if Ensembl database is used as reference genome.

TEDENOVO		
Application Type	stand alone	
The application is is a wrapper of the TEdenovo 2.2 pipeline included in the REPET package (https://urgi.versailles.inra.fr/) - TEdenovo: pipeline dedicated to detect, annotate and analyse repeats in genomic sequences, in particular, transposable elements (TEs). The pipeline starts by comparing the genome with itself using BLASTER, clusters the resulting matches with GROUPER, RECON and PILER, and for each cluster, it builds a multiple alignment from which a consensus sequence is derived. At the end, we obtain a library of classified, non-redundant consensus sequences.		
Arguments	project	project name as defined within TEdenovo configuration file (i.e.TEdenovo.cfg).
	Configuration File	standard TEdenovo configuration file. Is remarcable that the paths defined in the configuration file, should not be anymore absolute, but relative to the project working directory (defined in the configuration file tag 'project_dir'). In turn, it will be normally defined as the current directory (“project_dir: .”), or as the directory just created when uploading input files (i.e. “project_dir: uploaded_dir/”).
	Step	TEdenovo step/s to be excuted. *Format: a single step (i.e. 1), or a set of them (i.e. 1-5)*
	Dectection Mean	mean of TE detection (Blaster / LTRharvest / both). *Default: Blaster*
	Cluster HSPs	program used to cluster HSPs (Grouper / Recon / Piler) or a combination of them (GrpRec / GrpPil / RecPil / GrpRecPil). *Default: Grouper*
	Multi-alignment	program used to build the multiple sequence aligment (Map / MUSCLE / TCOFFE). *Default: Map*

TEDENOVO		
	Cluster TEs	program used to cluster consensus to find consensus families (Blastclust / MCL). *Default: Blastclust*
Input Files	The pipeline requires theTEdenovo configuration file, as well as the input contigs FASTA(*). Also, those files specified within configuration file (*i.e* TE_nucl_bank, HMMER_profiles, ReatScoupt_bank, etc.) are required if the default versions do not fit user needs: • repbase17.01_aaSeq_cleaned.fsa • repbase17.01_ntSeq_cleaned.fsa • ProfilesBankForREPET_Pfam26.0_GypsyDB.hmm • hmmbank_test (*) Contig filename should be in the format: [PROJECT_DIR.fa], like the original software.	
Output Files	The application returns a compressed directory under the name [PROJECT].tar.gz. It includes all the outputs generated in each step, and a dump file of the TE database created.	
Sample Files	project: Dmel configuration: http://transplantdb.bsc.es/documents/samples/TEdenovo/TEdenovo.cfg contig file: http://transplantdb.bsc.es/documents/samples/TEdenovo/Dmel.fa step: 1-8	
Special requirements	The application requires access to the DATA2 if the default bank file need to be used.	

5. References

[1] Programming Model Enactment Service – http://venus-c.sourceforge.net/
[2] OpenNebula – http://www.opennebula.org/
[3] COMP Superscalar – http://sourceforge.net/projects/compss/
[4] PMES dashboard manual:
http://transplantdb.bsc.es/documents/man/PMES_Dashboard_Manual.pdf