Use cases and challenges of distributed workflows at JCVI

Andrey Tovchigrechko JCVI

J. Craig Venter™ INSTITUTE

JCVI sequencing and computing

<u>Illumina (HiSeq2000), 454 (Titanium) and Sanger</u>

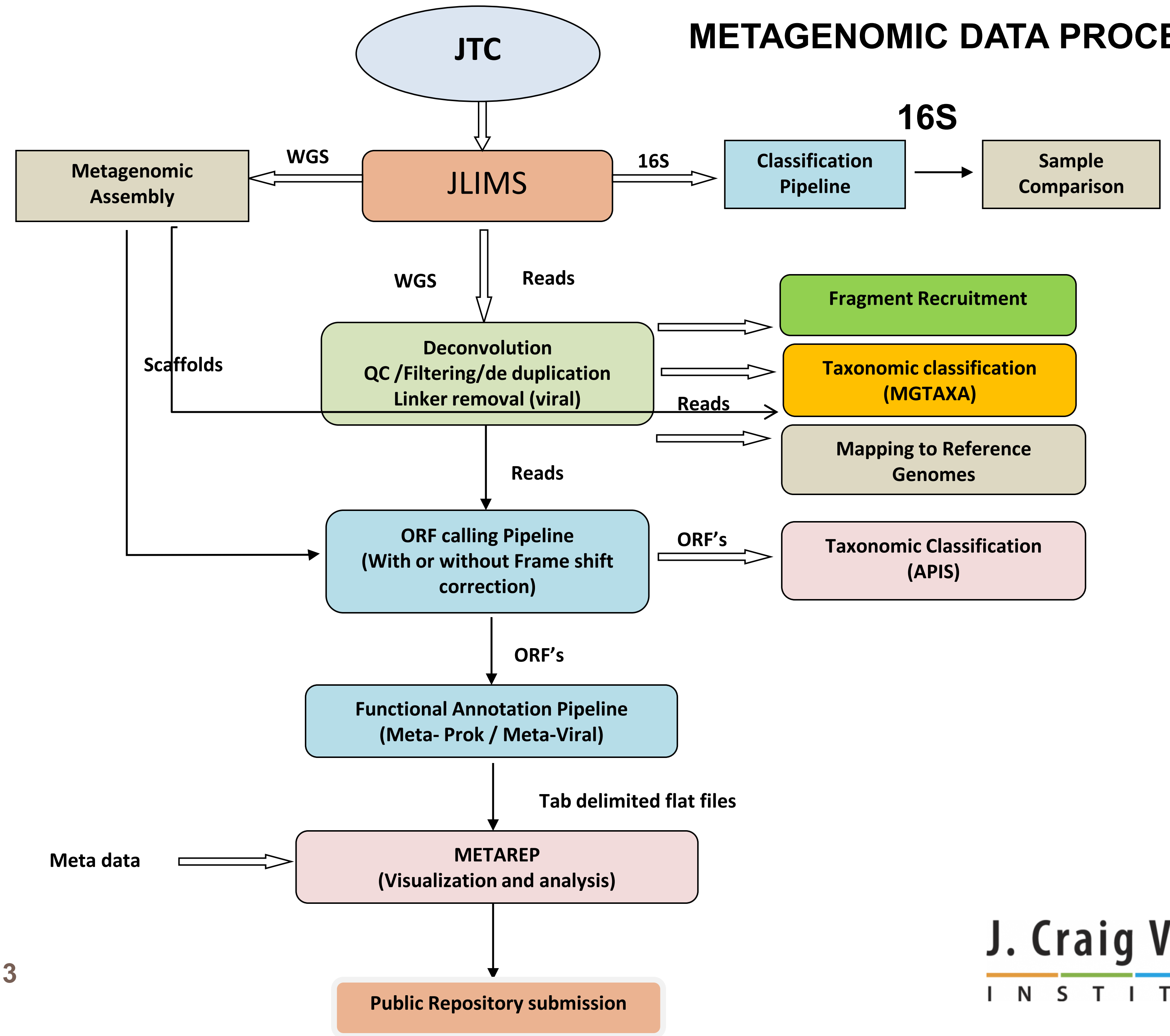
Generation per month: 6.6 billion reads   668 Gbp
Plus external data

454 is still viable for metagenomics and viral assembly. In one recent test on marine dataset, it gave better assemblies and more read-based BLASTP protein hits than 100x more Illumina data from the same sample, with corresponding savings in CPU cost

<u>JCVI's current computing capability:</u>

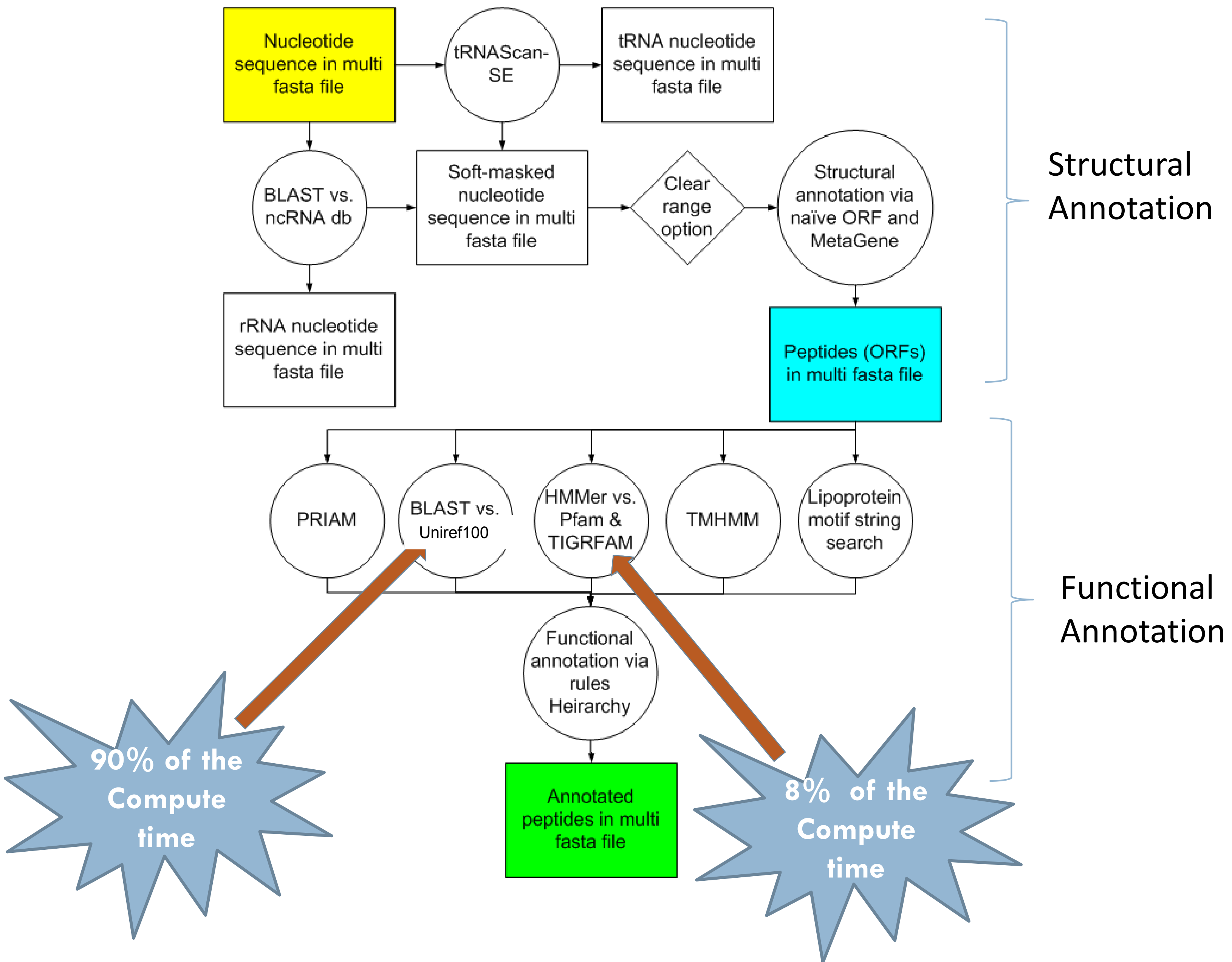
1700 cores, several petabytes of storage
1 million 454 reads can be annotated in
24 CPU hours (1 day) with 180 nodes (720 cores) = 17280 CPU*hrs.

METAGENOMIC DATA PROCESSING


JTC
JLIMS
WGS
16S
Metagenomic Assembly
16S
Classification Pipeline
Sample Comparison
WGS
Reads
Scaffolds
Deconvolution QC /Filtering/de duplication Linker removal (viral)
Fragment Recruitment
Taxonomic classification (MGTAXA)
Reads
Mapping to Reference Genomes
Reads
ORF calling Pipeline (With or without Frame shift correction)
ORF's
Taxonomic Classification (APIS)
ORF's
Functional Annotation Pipeline (Meta- Prok / Meta-Viral)
Tab delimited flat files
Meta data
METAREP (Visualization and analysis)
Public Repository submission

J. Craig Venter™ INSTITUTE

Metagenomic Annotation Pipeline Overview



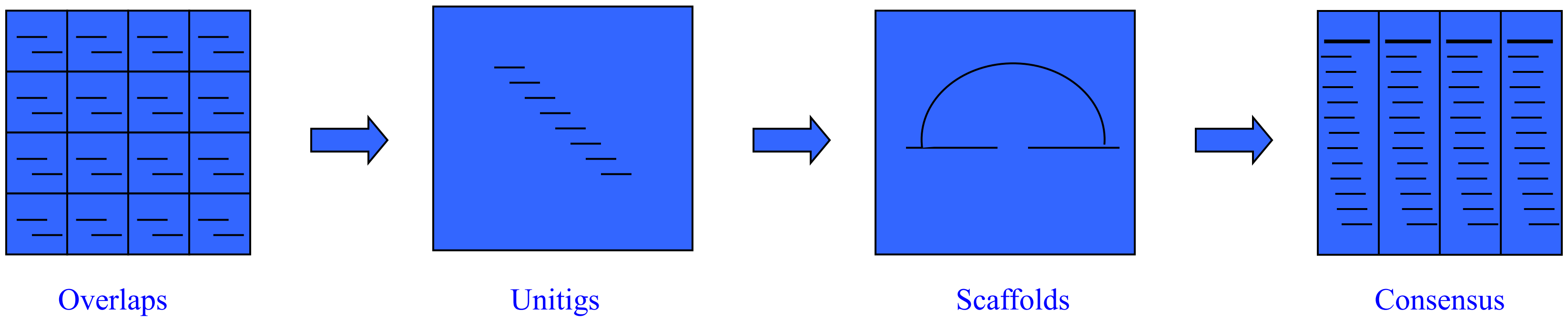
Nucleotide sequence in multi fasta file
tRNAScan-SE
tRNA nucleotide sequence in multi fasta file
BLAST vs. ncRNA db
Soft-masked nucleotide sequence in multi fasta file
Clear range option
Structural annotation via naïve ORF and MetaGene
rRNA nucleotide sequence in multi fasta file
Peptides (ORFs) in multi fasta file
Structural Annotation
PRIAM
BLAST vs. Uniref100
HMMer vs. Pfam & TIGRFAM
TMHMM
Lipoprotein motif string search
Functional Annotation
Functional annotation via rules Heirarchy
Annotated peptides in multi fasta file
90% of the Compute time
8% of the Compute time

Main workflow support

- VICS – Venter Institute Compute Services
- Java server application
- A workflow is described by a mix of XML (sequence of workflow steps as local jobs, single SGE jobs, SGE array jobs) and Java classes (generation of each step inputs and command lines)
- Talks to the SGE cluster via DRMAA
- Web form interface is generated (some extra Java code is needed to describe input fields)
- SOAP bindings are generated automatically
- Workflow becomes part of the code base and needs to be deployed into a specific server
- No restart capability – if one step fails because of node failure, workflow has to be re-run from scratch
- Plans are to use Kepler workflows inside VICS in place of VICS native implementation

Celera Assembler Pipeline

Stage



Structure

Workflow is self-resubmitting SGE jobs with dynamic modification of existing job dependencies, Perl script

Matrix partition, parallel computation.	Sequence-free overlap graph in core RAM.	Scaffold graph in core RAM with mate links.	Progressive consensus, partition by scaffold.

Requirement (one HMP sample)

SGE grid (~100 core ½ day)	Big RAM box (~100GB ½ day)	Big RAM box (~100GB ½ day)	SGE grid (~100 core ½ day)

Ways to scale BLAST at JCVI

- Example used is BLASTP search of proteins annotated on Velvet assembly of one HMP sample against PANDA DB (124,000 peptides of average length 176). E-value cutoff 10E-5.
- **"Matrix" of serial NCBI BLAST jobs** <Query chunk> X <DB partition> on HTC cluster, followed by a single aggregator job
  - Runs under VICS workflow manager
  - 660 CPU*hrs
- **TimeLogic Tera-BLAST**
  - FPGA card, closed source, implements something similar to older NCBI one-hit extension algorithm with default word size 4 (for BLASTP)
  - Runs under VICS
  - 1.3 CPU*hrs (where "CPU" is one TimeLogic card)
  - 500x speed up over NCBI BLAST on one CPU core
  - Lost about 10,000 hits out of 700,000
- **mpiBLAST from VA-Tech on TACC Ranger**
  - NCBI C Tookit code patched and integrated into MPI application. Worker nodes send hits from seeds to master node that tells them which ones to extend, thus providing superlinear speed-up. Many other optimizations (MPI-IO, virtual DB files etc).
  - We were not able to run it stably on TACC Ranger in 2010 despite going through the matrix of different (compiler)(MPI)(runtime options) and support lists. It had some fixes after that but we did not try again.
- Lessons learned: TimeLogic seems to be a solution (even considering its price), but it will never give the same results as NCBI BLAST. It is simply somewhat different algorithm.

Storage

- Current solutions at JCVI - NFS (NetApp) and NFS-compatible clustering FS (Isilon)
- Most flexible – provides tradition random access file operations on a shared filesystem
- Expensive to scale with increasing number of compute nodes
- Solutions like Hadoop Distributed Filesystem are economical to scale, but at the expense of recasting all applications into streaming data access patterns
- JCVI has a 16 node Hadoop cluster that is currently used to a test-bed for Cloud Viral Annotation project
- If external computing resources are used, the transfer to and from its filesystems becomes a bottleneck. 25G BLAST DB took 50 min to push from JCVI to TACC with gridFTP (we are connected to Internet 2). Transfer can easily take longer than processing.
- Local solutions like TimeLogic do not have this network transfer issue (you only pushing to a dedicated machine on your local network)

Bioinformatics on large cyber infrastructure

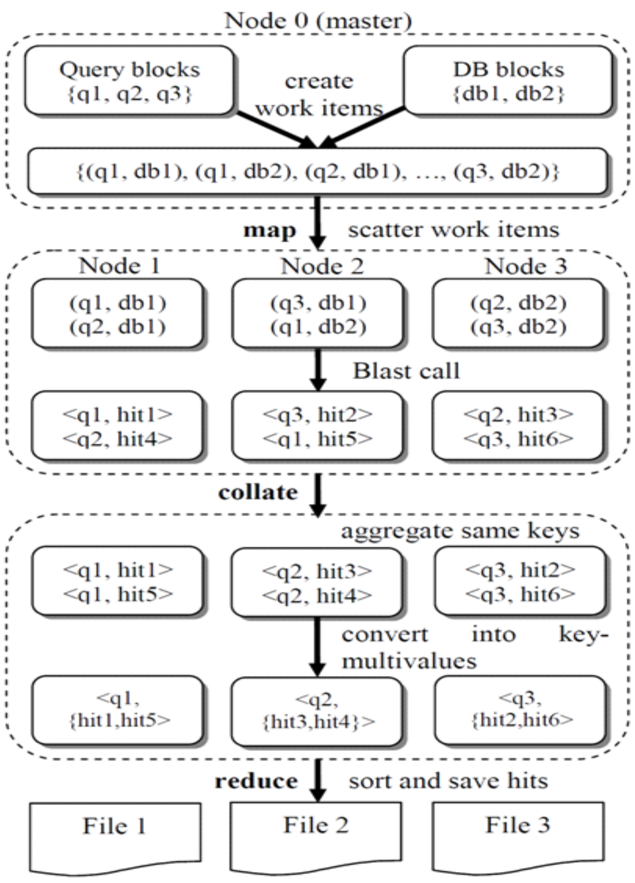
- Two NSF projects (metagenomic taxonomic assignment and proteogenomics) had been requested by the funding agency to use TeraGrid/XSEDE
- Most TeraGrid clusters schedule efficiently only large MPI jobs (100s of CPU cores)
- Workflows of many serial jobs are not supported by local resource managers
- Two ways to use such supercomputers for serial workflows:
  - Fake serial High Throughput Computing
  - Re-write workflows into MPI parallel code, at least for critical parts

Accommodating supercomputer to existing bioinformatics pipelines

- Two-level scheduling ("glide-ins")
- Condor "glide-ins", glideinWMS, SWIFT, MyCluster and Makeflow with MPI WorkQueue workers.
- E.g. SWIFT takes care of everything – data movement, resubmission of failed jobs, optimizing glide-in allocation, quick scheduling of very small jobs.
- WorkQueue implementation is more restricted, but the big advantage is that the same Makeflow file can be executed on a serial SGE or Condor cluster
- Glide-in tools require certain minimum level of host OS support (Posix "fork" – not available on earlier Blue Gene OSes)
- SWIFT develops its own ecosystem that includes a Linux kernel on BlueGene ("fork"; shared RAM caching drives local to the groups of compute nodes; tuning to BG network topology)
- Data exchange and synchronization is through files, unless a specific pattern is implemented internally by the workfllow engine like AllPairs or MapReduce in CCTools

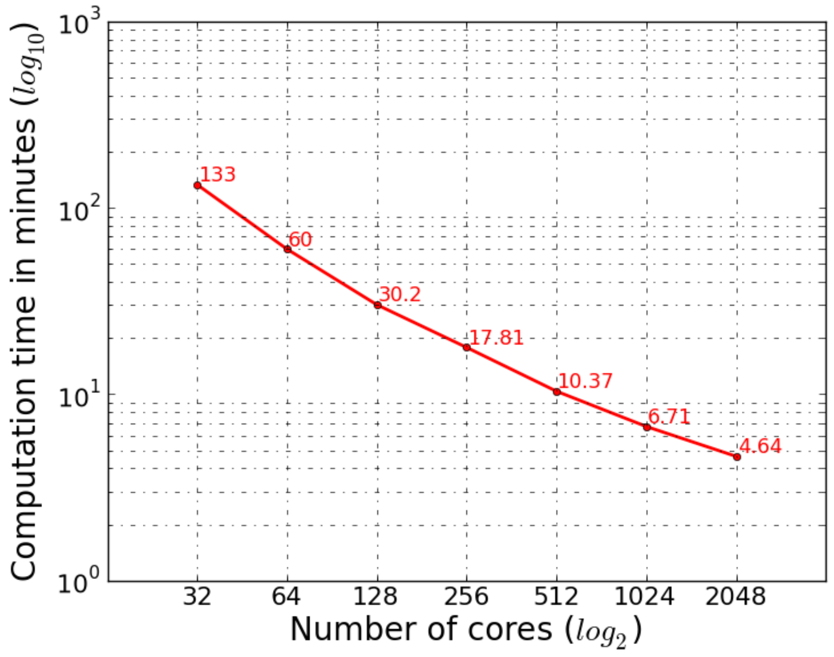
MR-MPI BLAST+ implementation

- A regular MPI program – runs on any supercomputer with a shared file system
- Makes high-level API calls to unmodified NCBI C++ Toolkit – results are fully compatible with the upstream NCBI code; easy to keep up to date
- Implemented with MapReduce MPI (MR-MPI) library from Sandia Lab that helps organizing computations and data flow
- The library scheduler was modified to solve the problem of maintaining context between map() calls – a common problem with the classical MapReduce algorithm
- Performs sorting of final output in parallel. This is an advantage compared to a typical HTC matrix-split implementation with a single combiner job for sorting, if the number of hits is large

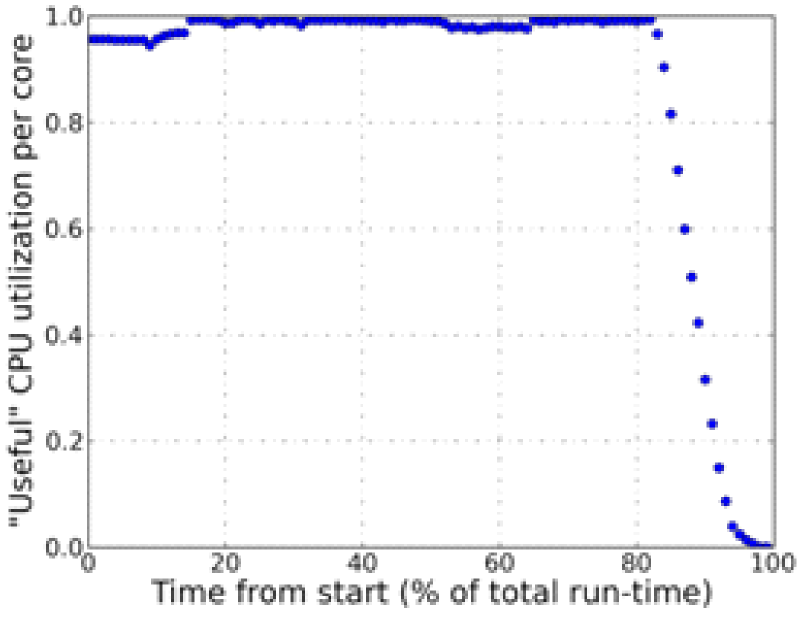


Control flow of the MR-MPI BLAST

MR-MPI BLASTN and BLASTP scaling

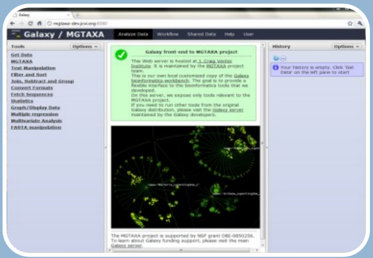


Scaling chart for MR-MPI BLASTN showing process wall clock time at different total core counts in MPI job. The total number of query sequences is 40,000. The sequences are split into 40 blocks of 400 kbp. Each block, when combined with one DB partition, forms a sequential work unit for the MapReduce algorithm. The data point labels represent time in minutes.
DB is NCBI WGS+nt+hg (109 1G formatted partitions).



"Useful" CPU utilization per core during the course of the computation for the MR-MPI BLASTP run with 1024 cores. CPU user time used at any given moment within a BLAST call was divided by the corresponding wall clock time, summed over all concurrent calls, and divided by a total number of cores allocated to the MPI program.

MGTAXA – composition-based taxonomic classification in metagenomes: http://andreyto.github.com/mgtaxa/



Galaxy front-end to MGTAXA project

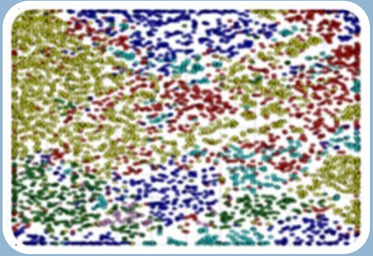
This Web server is hosted at J. Craig Venter Institute. It is maintained by the MGTAXA project team.
This is our own local customized copy of the Galaxy bioinformatics workbench. The goal is to provide a flexible interface to the bioinformatics tools that we developed.
On this server, we expose only tools relevant to the MGTAXA project.
If you need to run other tools from the original Galaxy distribution, please visit the Galaxy server maintained by the Galaxy developers.

Your history is empty. Click 'Get Data' on the left pane to start

The MGTAXA project is supported by NSF grant DBI-0850256. To learn about Galaxy funding support, please visit the main Galaxy server.

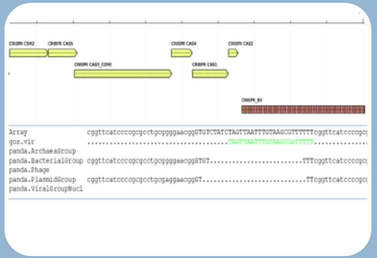
Predict taxonomy for bacterial metagenomic sequences

- Glimmer ICM based classifier (similar to Phymm approach). Parallelized on HTC cluster, Galaxy Web interface. Sequences above 300 bp.
- BLAST+ and Batch SOM implementations for HPC clusters with MPI-MapReduce framework. Calls to pristine NCBI BLAST+ API – full compatibility. Scales to 2000 cores on TeraGrid (TACC Ranger).



Predict hosts for bacteriophages in metagenomes

- Explores compositional similarity between phage and host
- ICM and SOM for prediction and visualization
- Assigns phage scaffolds (5Kbp) to bacterial sequences
- Uses infrastructure and interfaces of the bacterial classifier

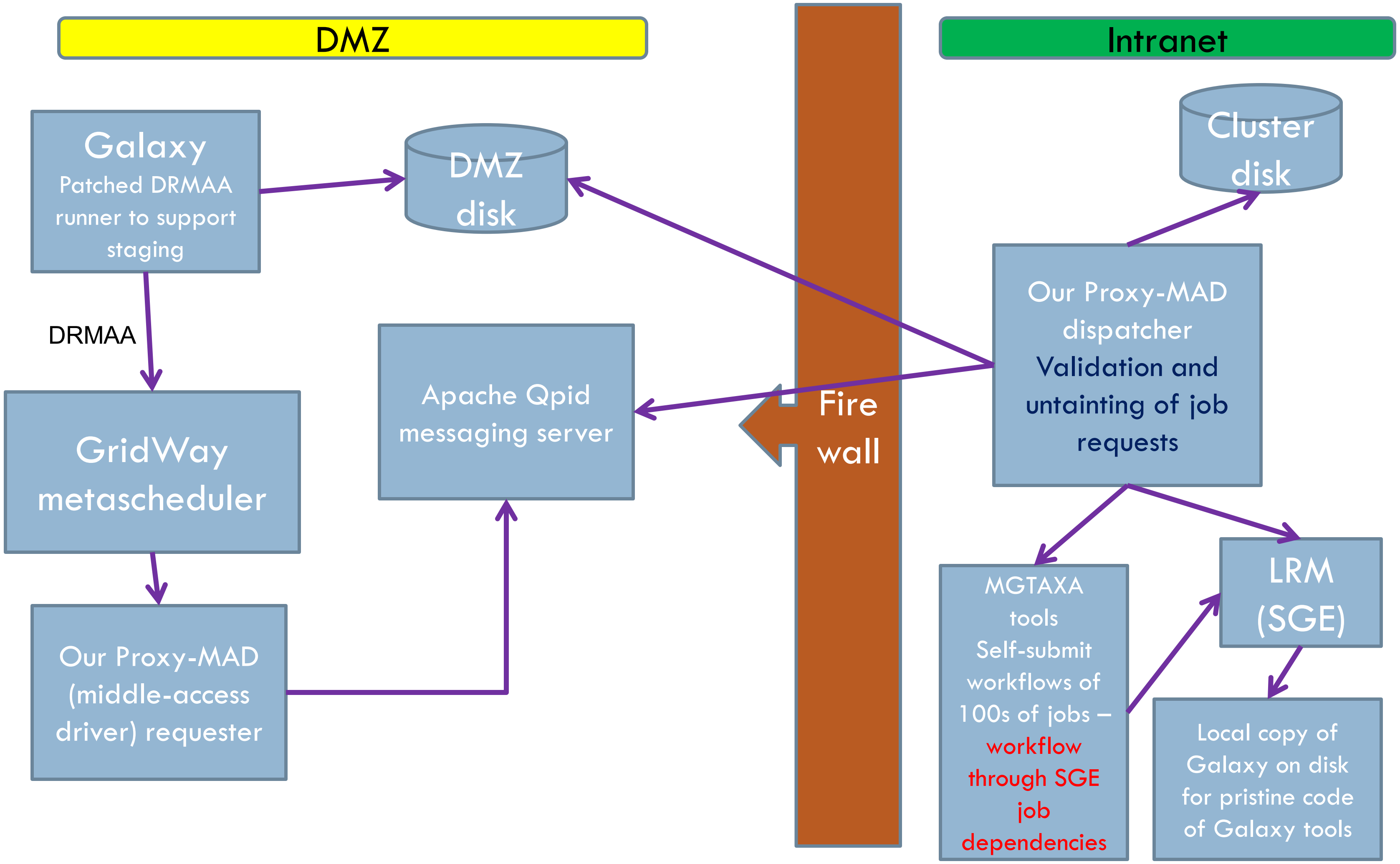


```
Array               cggttcatccccgcgcctgcggggaacggGTGTCTATCTAGTTAATTTGTAAGCGTTTTTTcggttcatccccgc
gos.vir             ...................................TAGTTAATTTGTAAGCGaTTTTT.............
panda.ArchaeaGroup
panda.BacterialGroup cggttcatccccgcgcctgcggggaacggGTGT.......................TTTcggttcatccccgc
panda.Phage
panda.PlasmidGroup  cggttcatccccgcgcctgcgaggaacggGT.........................TTcggttcatccccgc
panda.ViralGroupNucl
```

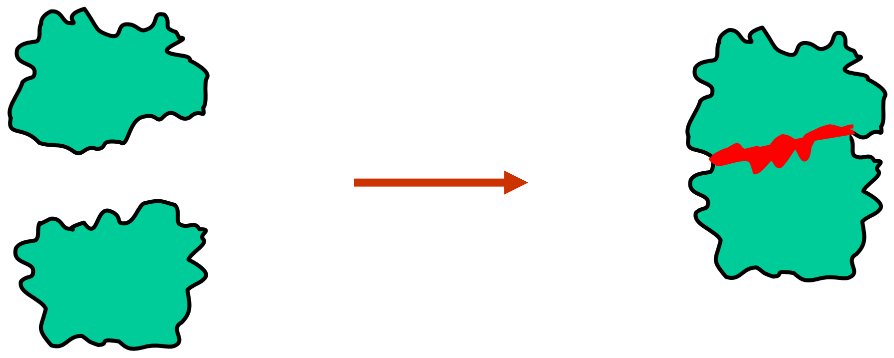
CRISPR pipeline

- Scans genomes & metagenomes for CRISPR arrays and genes
- Connects with viral metagenomes through spacer matches – alternative way to establish the bacteriophage-host relationship
- Parallelized on HTC cluster

Server - architecture

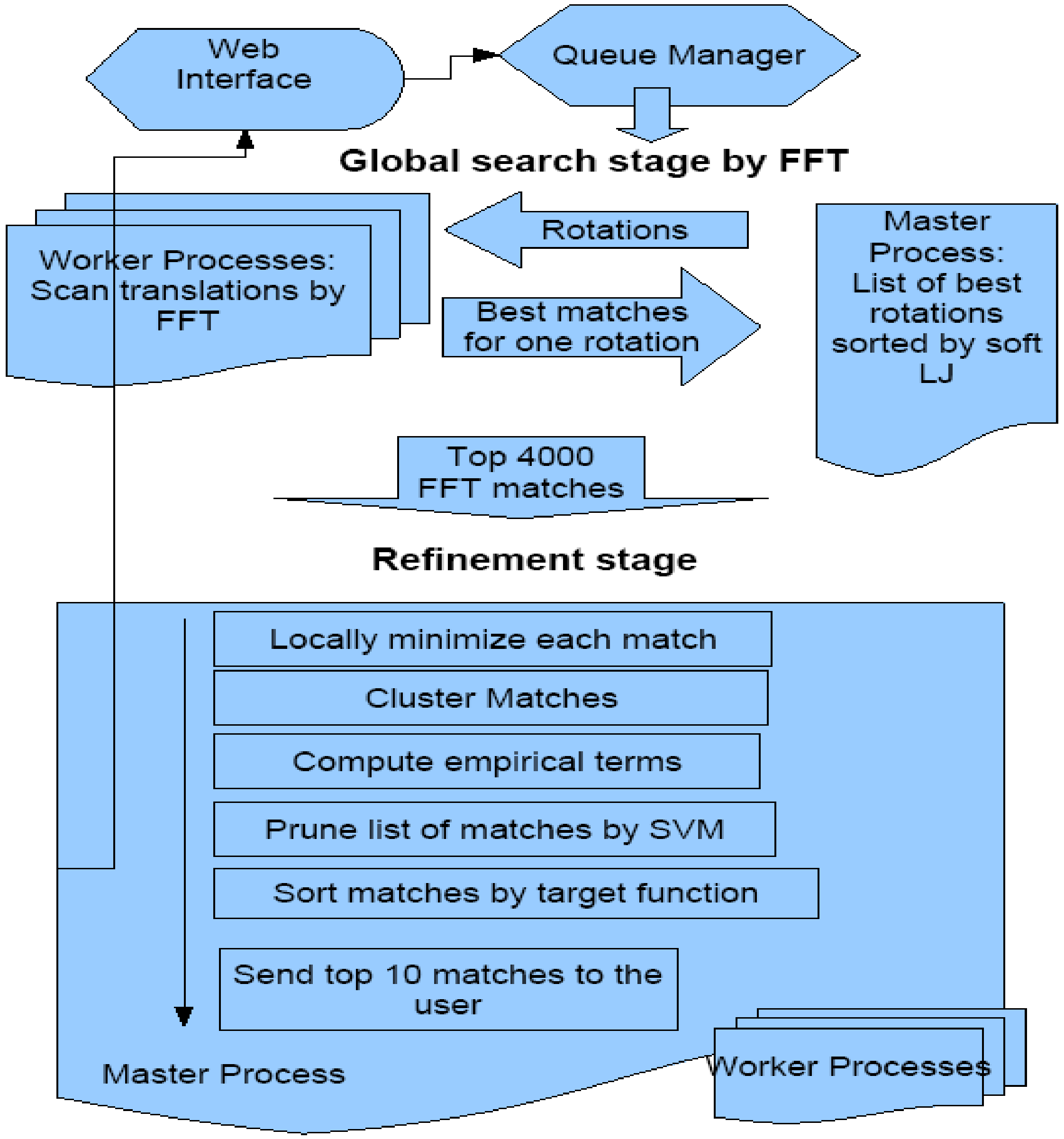

DMZ
Intranet
Galaxy
Patched DRMAA runner to support staging
DMZ disk
Cluster disk
DRMAA
Our Proxy-MAD dispatcher
Validation and untainting of job requests
Apache Qpid messaging server
Fire wall
GridWay metascheduler
Our Proxy-MAD (middle-access driver) requester
MGTAXA tools
Self-submit workflows of 100s of jobs – workflow through SGE job dependencies
LRM (SGE)
Local copy of Galaxy on disk for pristine code of Galaxy tools

Protein-protein docking



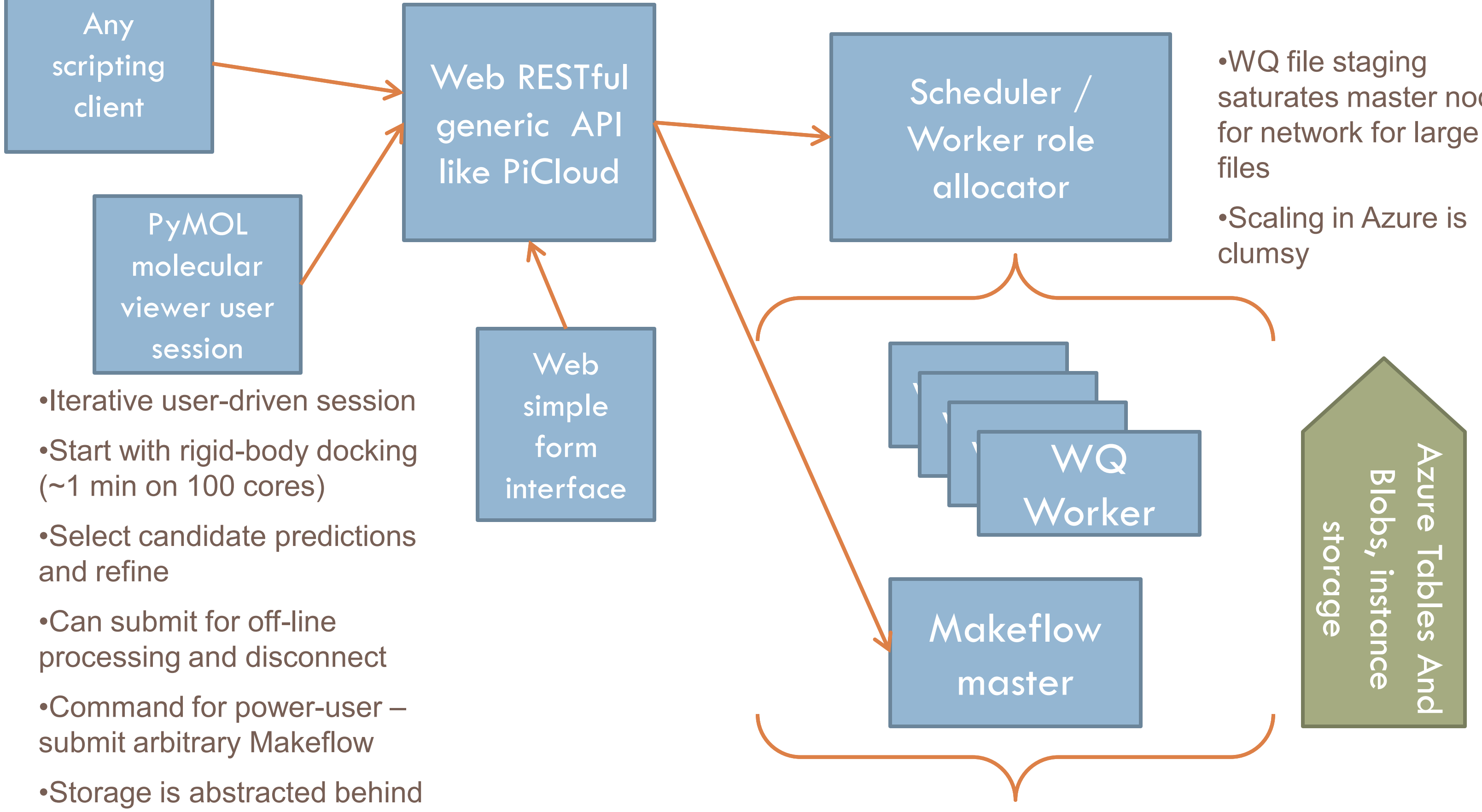
- A way to model protein interactions
- Given a 3D structures of two unbound proteins, predict a 3D structure of their complex (transient or obligate)
- GRAMM-X docking Web server created while at KU, has been running since 2006
- Current NSF-Microsoft grant to implement enhanced version in Azure cloud

GRAMM-X Data Flow and Process Organization


Web Interface
Queue Manager
Global search stage by FFT
Worker Processes: Scan translations by FFT
Rotations
Best matches for one rotation
Master Process: List of best rotations sorted by soft LJ
Top 4000 FFT matches
Refinement stage
Locally minimize each match
Cluster Matches
Compute empirical terms
Prune list of matches by SVM
Sort matches by target function
Send top 10 matches to the user
Master Process
Worker Processes

Client-cloud docking application (Azure)

Generic self-spinning, self-scaling framework for running any Makeflows in Azure


Any scripting client
PyMOL molecular viewer user session
Web RESTful generic API like PiCloud
Web simple form interface
Scheduler / Worker role allocator
WQ Worker
Makeflow master
Azure Tables And Blobs, instance storage

•WQ file staging saturates master node for network for large files

•Scaling in Azure is clumsy

•Iterative user-driven session

•Start with rigid-body docking (~1 min on 100 cores)

•Select candidate predictions and refine

•Can submit for off-line processing and disconnect

•Command for power-user – submit arbitrary Makeflow

•Storage is abstracted behind REST API – can be node local or Azure Blob internally

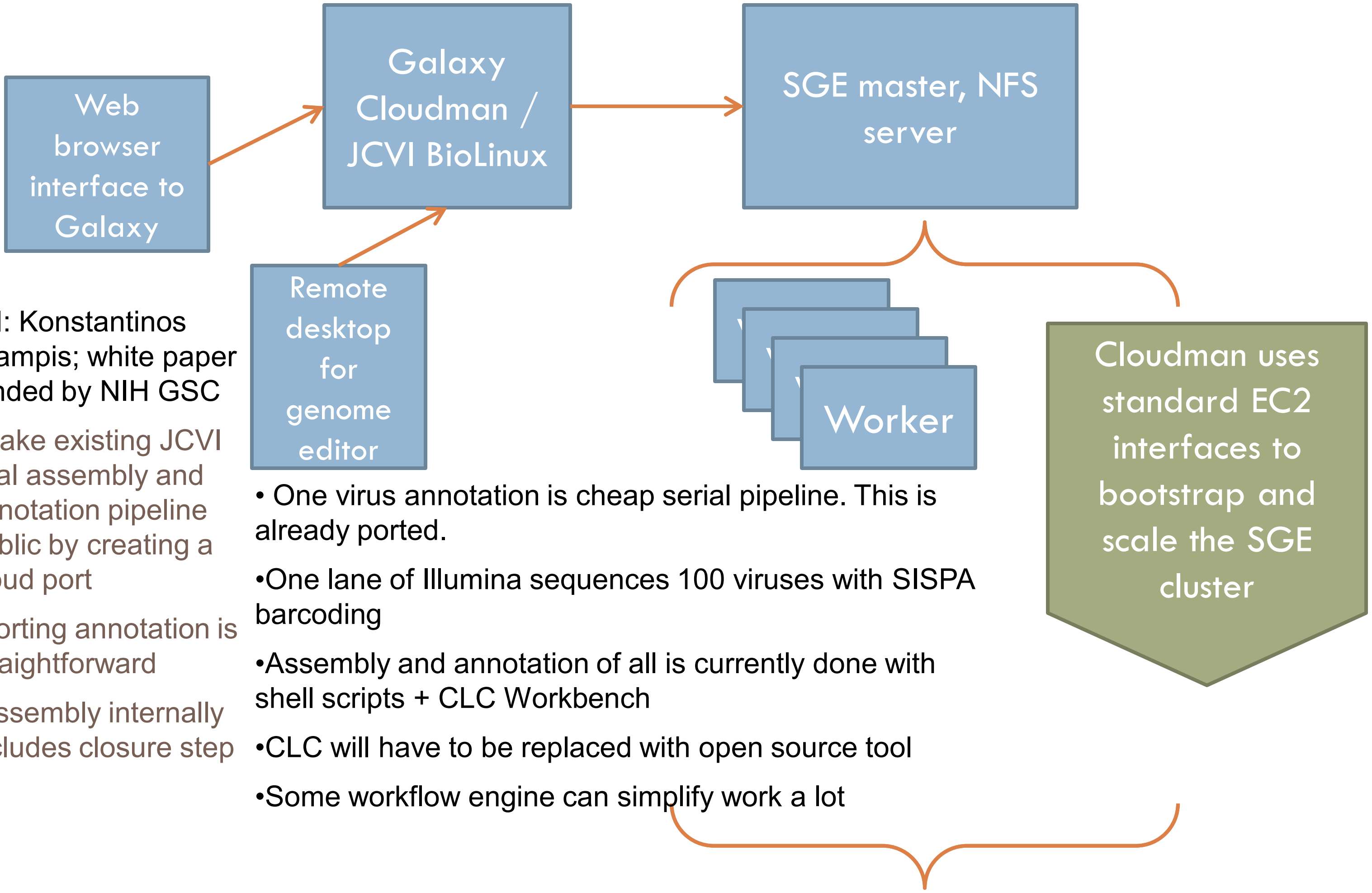
•Framework is designed for portability to other clouds or clusters

•Only a small bootstrap .NET module – the rest is Java

Porting Proteogenomics Pipeline to XSEDE

- NSF project to use proteomics data for improving genome annotation (PI Sam Payne, now at PNNL)
- Was originally implemented as VICS pipeline with a mix of single and array SGE jobs
- 10 GB of input files, 200 jobs, total CPU time 300 hrs
- Using Makeflow, it was made portable across XSEDE SGE MPI sites and serial SGE clusters like JCVI’s
- Still assumes shared file system on the cluster – too much work to get rid of deep directory structure in each workflow instance

Viral annotation pipeline in EC2 and Ecalyptus


Web browser interface to Galaxy
Galaxy Cloudman / JCVI BioLinux
SGE master, NFS server
Remote desktop for genome editor
Worker
Cloudman uses standard EC2 interfaces to bootstrap and scale the SGE cluster

•PI: Konstantinos Krampis; white paper funded by NIH GSC

•Make existing JCVI viral assembly and annotation pipeline public by creating a cloud port

•Porting annotation is straightforward

•Assembly internally includes closure step

• One virus annotation is cheap serial pipeline. This is already ported.

•One lane of Illumina sequences 100 viruses with SISPA barcoding

•Assembly and annotation of all is currently done with shell scripts + CLC Workbench

•CLC will have to be replaced with open source tool

•Some workflow engine can simplify work a lot

Acknowledgments

- Mathangi Thiagarajan (Metagenomics pipelines)
- Jason Miller (Celera Assembly)
- Jeff Hoover (TimeLogic BLAST)
- Seung-Jin Sul (MR-MPI BLAST)
- Shibu Yooseph (Metagenomics pipelines)
- Dana Busam (Sequencing stats)
- Hyunsoo Kim (Azure)
- Support from NSF, NIH, DOE, Microsoft, Battelle