The Computing Revolution in Biosciences

BU-ISCIII

Unidades Comunes Científico Técnicas – SGSAFI-ISCIII

2428 Octubre 2022, 4ª Edición
Programa Formación Continua, ISCIII

Index

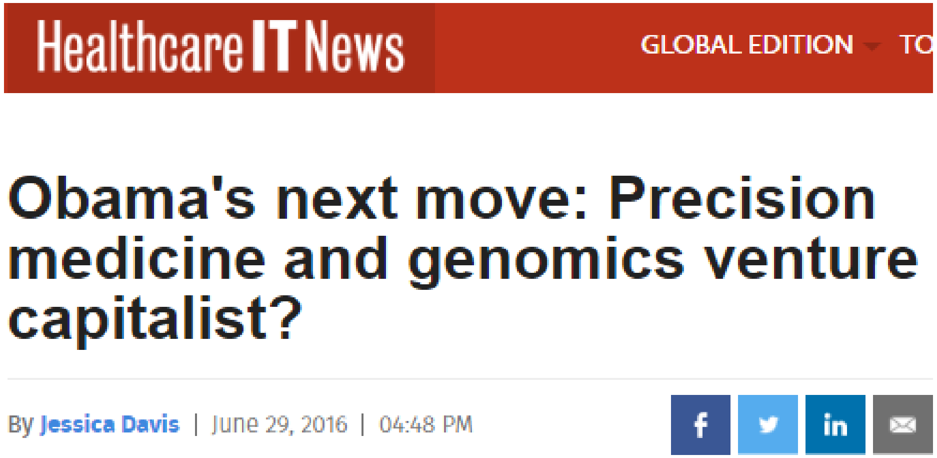
The Computing Revolution in Biosciences:

- The Century of Biology
- Computing in Biosciences
- The Omics Era
- Change of Paradigm
- HPC infrastructure
- Workflows
- The need of standardisation
- Nextflow
- Containers
- Singularity

The Century of Biology

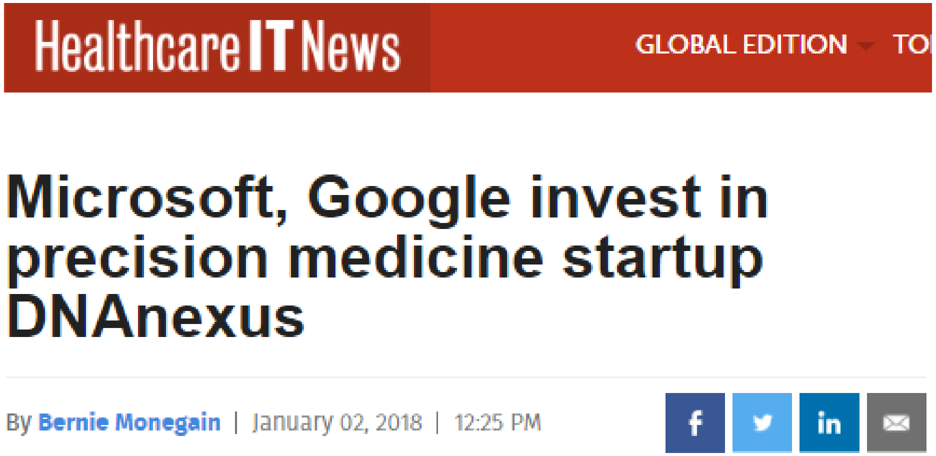
"If the 20th century was the century of physics, the 21st century will be the century of biology. While combustion, electricity and nuclear power defined scientific advance in the last century, the new biology of genome research—which will provide the complete genetic blueprint of a species, including the human species—will define the next."

VENTER, C., & COHEN, D. (2004). The Century of Biology. New Perspectives Quarterly, 21(4), 73–77. doi:10.1111/j.1540-5842.2004.00701.x



Obama's next move: Precision medicine and genomics venture capitalist?

By Jessica Davis | June 29, 2016 | 04:48 PM



Microsoft, Google invest in precision medicine startup DNAnexus

By Bernie Monegain | January 02, 2018 | 12:25 PM

Computing in Biosciences I

Research used to focussed in a small number of samples and researchers analysed them with the whatever means they had and/or felt more comfortable with:

- Windows based PC using programs with visual interface
- Macs and Linux based workstations
- Remote web servers
- Web-based platforms (i.e. Galaxy) and remote HPC
- HPC local environments

Computing in Biosciences II

- Windows based PC using programs with visual interface

Pros	Cons
Data remains private	No backups or data management schemes
Software easy to install	Software version not easy to control, binaries are black boxes
Graphic interface	No control over hidden parameters
	Analysis are irreproducible

Computing in Biosciences III

- Macs and Linux based workstations

Pros	Cons
Data remains private	No backups or data management schemes
Control over software installed versions, open source programs	Software may not be easy to install, library and dependencies problems
All parameters are available for the command	Command line interface
	Analysis are irreproducible

Computing in Biosciences IV

- Remote web servers

Pros	Cons
No need to storage intermediate files	Your data is in someone else's computer No backups or data management schemes
No need to install software	Software version not easy to control, black boxes
Graphic interface	No control over hidden parameters
	Quotas Analysis are irreproducible

Computing in Biosciences V

- Web-based platforms (i.e. Galaxy) and remote HPC

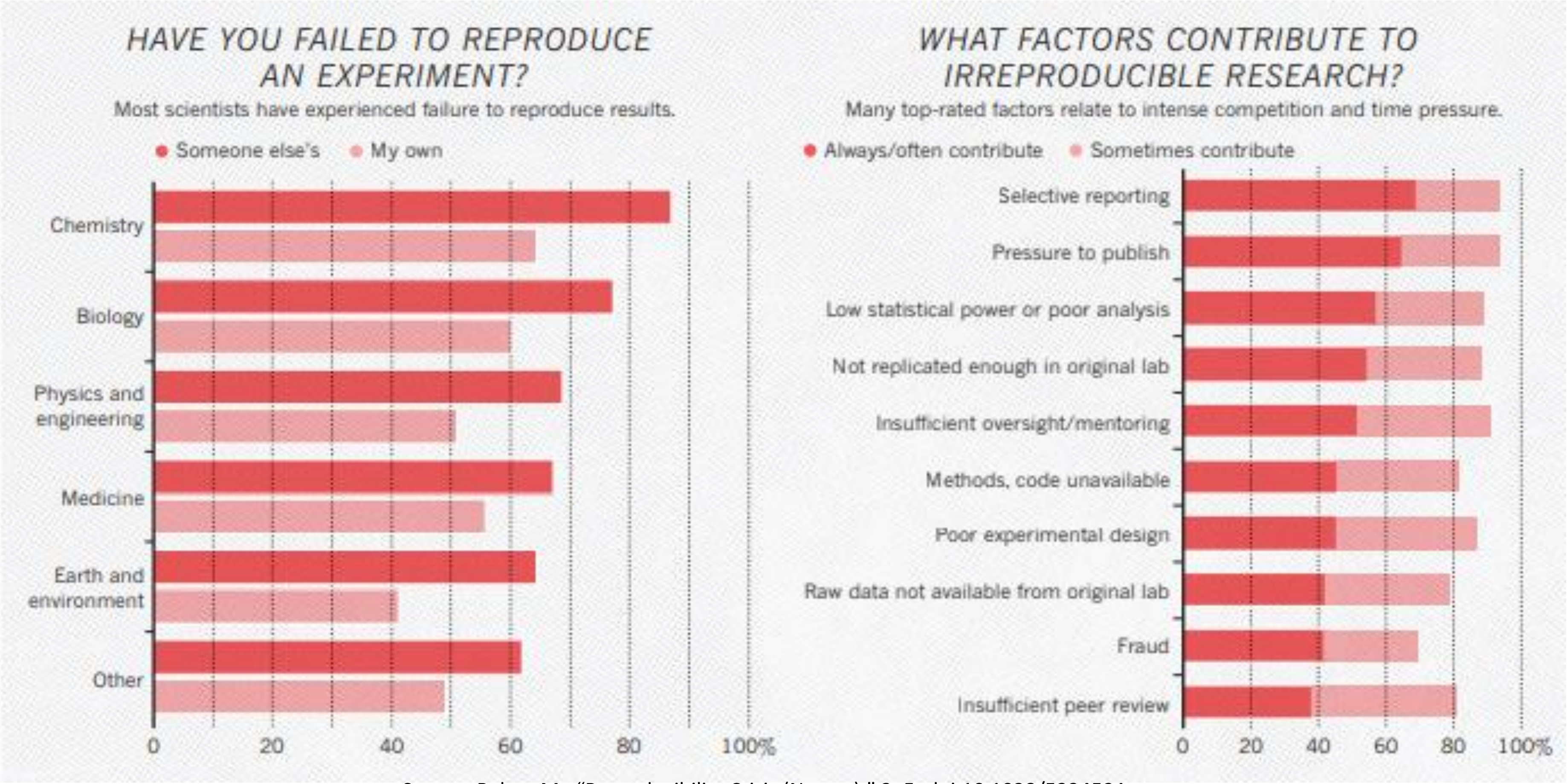
Pros	Cons
No need to storage intermediate files	Your data is in someone else's computer No backups or data management schemes
No need to install software Partial control over installed software	No control over installed software, versions and future availability
Graphic interface	No control over hidden parameters
Analysis are partially reproducible	Quotas

Computing in Biosciences VI

- HPC local environments

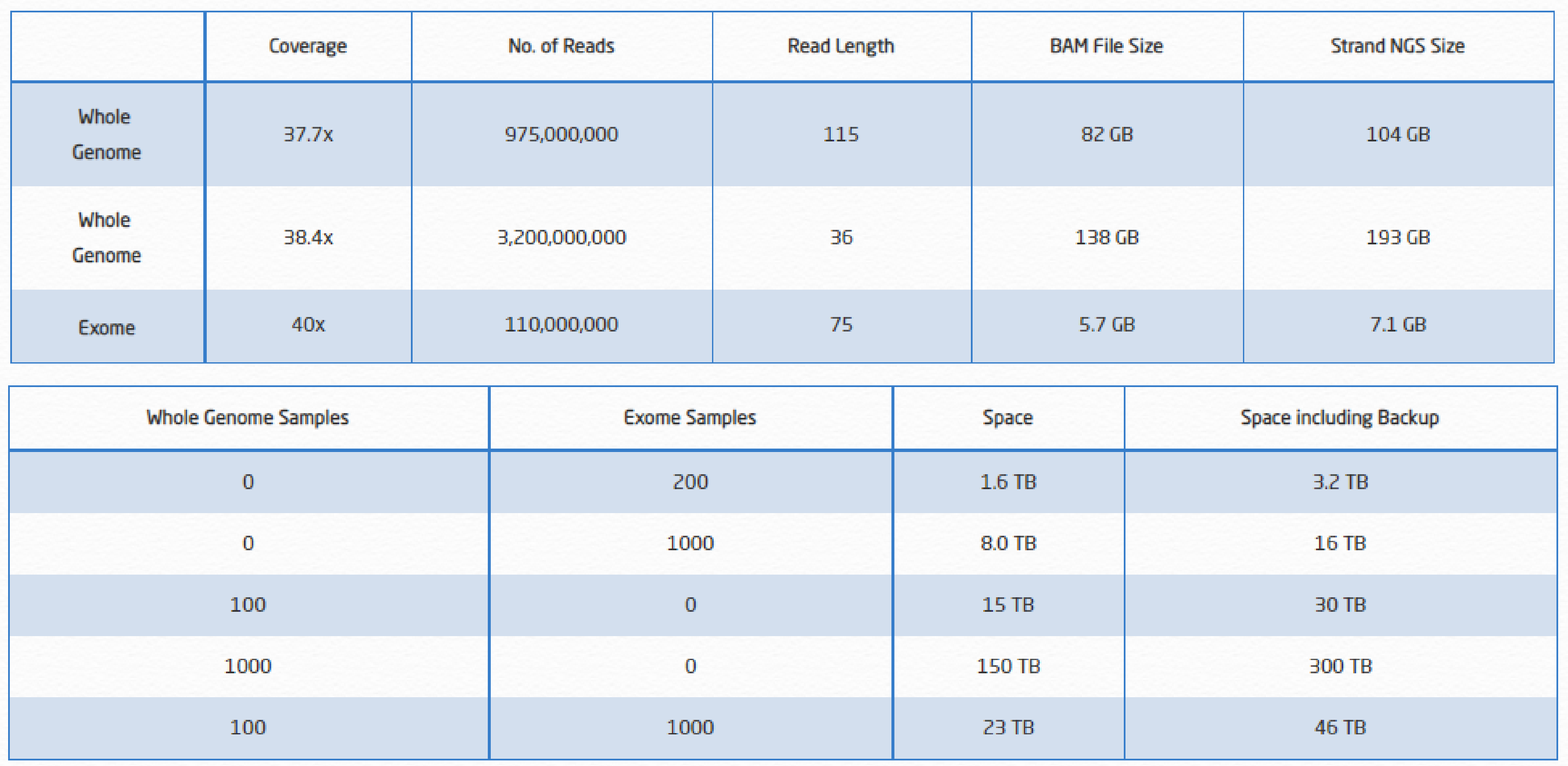
Pros	Cons
Data remains private Backups and data management schemes	Quotas
No need to install software Partial control over installed software	No control over installed software, versions and future availability
All parameters are available for the command	Command line interface
Possibility of suggesting new software installations	Analysis may be irreproducible

Is there a reproducibility crisis?



Source: Baker, M. "Reproducibility Crisis (Nature)," 3–5. doi:10.1038/533452A.

The Omics Era I

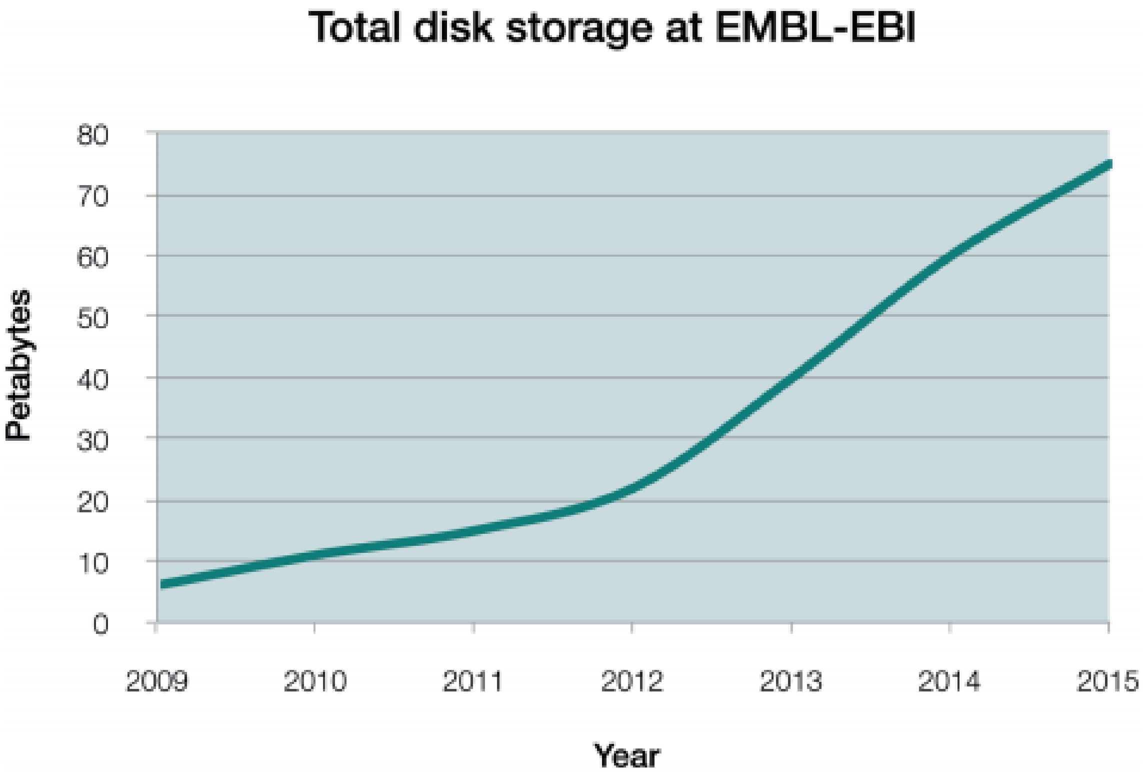


	Coverage	No. of Reads	Read Length	BAM File Size	Strand NGS Size
Whole Genome	37.7x	975,000,000	115	82 GB	104 GB
Whole Genome	38.4x	3,200,000,000	36	138 GB	193 GB
Exome	40x	110,000,000	75	5.7 GB	7.1 GB

Whole Genome Samples	Exome Samples	Space	Space including Backup
0	200	1.6 TB	3.2 TB
0	1000	8.0 TB	16 TB
100	0	15 TB	30 TB
1000	0	150 TB	300 TB
100	1000	23 TB	46 TB

Source: https://www.strand-ngs.com/support/ngs-data-storage-requirements

The Omics Era II



Installed (2008–2015) storage at EMBL-EBI. These figures include all installed storage, counting multiple backups for all data resources as well as unused storage to handle submissions in the immediate future

Source: Cook, Charles E et al. “The European Bioinformatics Institute in 2016: Data growth and integration” *Nucleic acids research* vol. 44,D1 (2015): D20-6.

Change of Paradigm I

1 sample	multiple samples
Research only: NGS was still a new thing, no applications 10 years ago	**Many applications:** research, clinical, industrial, forensic, military, …
Reproducibility is not needed: Why would anyone reanalyse this?	**Reproducibility, scalability , portability and standardisation are required**
Storage is not an issue: files of 1 sample fits everywhere in my HDD, maybe I will copy it in a CD-ROM	**Storage is challenging:** storage, indexation and backup required, privacy and legal standards
Computing is simple: no need to worry about resources or optimisation	**Computing requires optimisation and lots of resources**

Change of Paradigm II

- Nowadays scientific computing paradigm

Pros	Cons
Data remains private Backups and data management schemes	High storage space Dedicated file systems Databases to index files
Control over software installed versions, open source programs	Many versions of the same software coexists
All parameters are available for the command	You have to understand all software variations
Analysis are reproducible and public	You have to publish and document your work

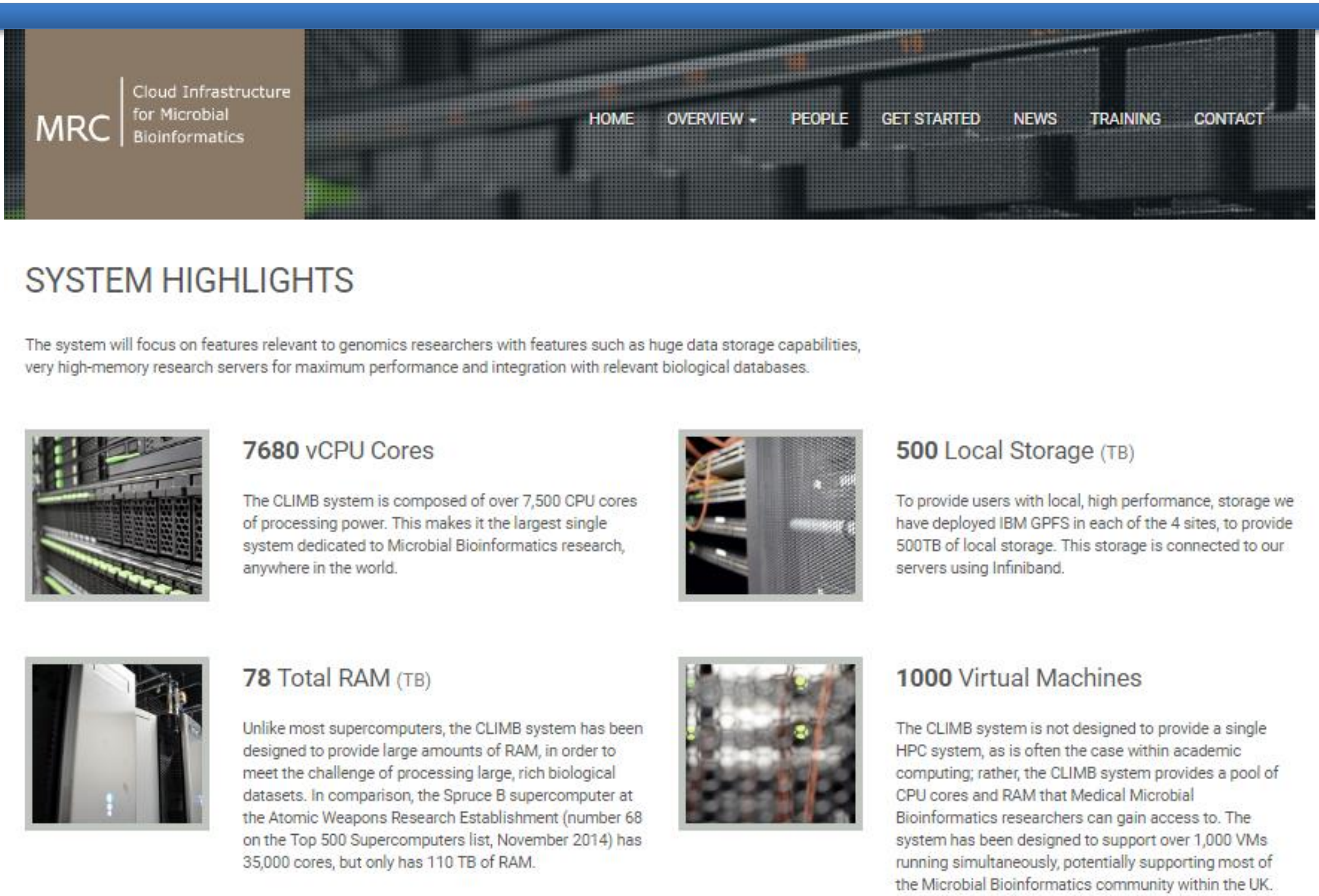
HPC infrastructure I

Machine	OS	Software	CPU	RAM	Storage
Workstation (x5)	Centos 6.9	/opt(*)	4 cores	32 Gb	4 TB
Bioinfo01 (1 node)		/opt(*)	16 cores	120 Gb	500 Gb
HPC (16 nodes)		/opt(*)	320 cores	8 TB	500 Gb

2 shared data storage disk boxes: 70TB + 250 TB

VMs, ISCIII's Windows personal terminals, personal laptops mobile platforms, cloud computing platforms, cloud storage, remote services, …

HPC infrastructure II



SYSTEM HIGHLIGHTS

The system will focus on features relevant to genomics researchers with features such as huge data storage capabilities, very high-memory research servers for maximum performance and integration with relevant biological databases.

7680 vCPU Cores

The CLIMB system is composed of over 7,500 CPU cores of processing power. This makes it the largest single system dedicated to Microbial Bioinformatics research, anywhere in the world.

500 Local Storage (TB)

To provide users with local, high performance, storage we have deployed IBM GPFS in each of the 4 sites, to provide 500TB of local storage. This storage is connected to our servers using Infiniband.

78 Total RAM (TB)

Unlike most supercomputers, the CLIMB system has been designed to provide large amounts of RAM, in order to meet the challenge of processing large, rich biological datasets. In comparison, the Spruce B supercomputer at the Atomic Weapons Research Establishment (number 68 on the Top 500 Supercomputers list, November 2014) has 35,000 cores, but only has 110 TB of RAM.

1000 Virtual Machines

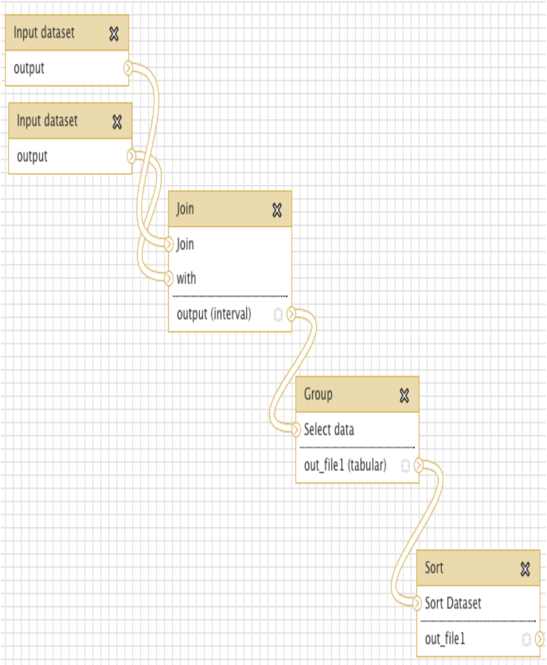
The CLIMB system is not designed to provide a single HPC system, as is often the case within academic computing; rather, the CLIMB system provides a pool of CPU cores and RAM that Medical Microbial Bioinformatics researchers can gain access to. The system has been designed to support over 1,000 VMs running simultaneously, potentially supporting most of the Microbial Bioinformatics community within the UK.

Workflows I

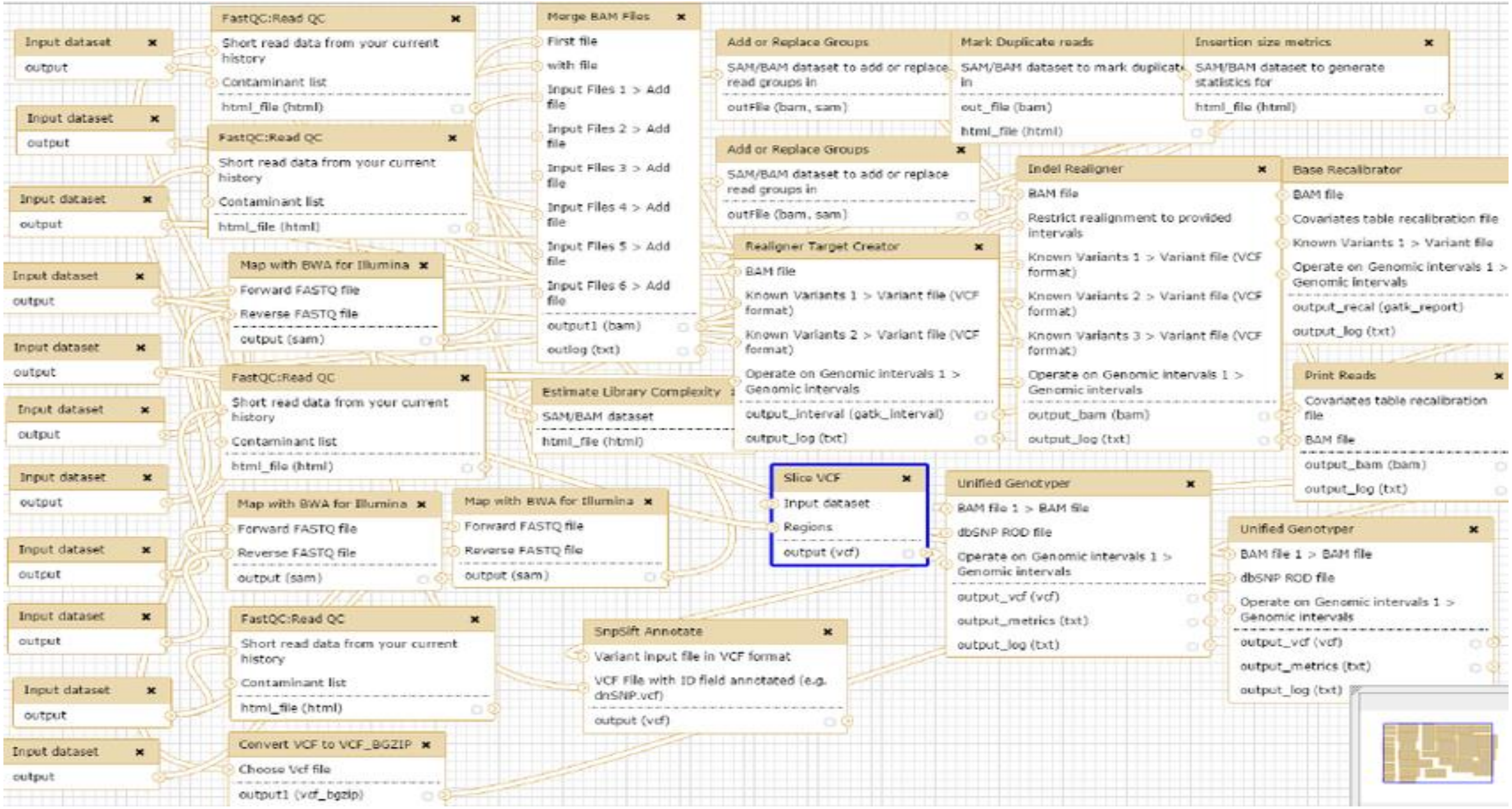
- Bioinformatic analyses invariably involve shepherding files through a series of transformations, called a **pipeline** or a **workflow.**

- These transformations are done by executable **command line software** written for Unix-compatible operating systems.

- They need to be **reproducible, easy to maintain, portable and scalable.**



Workflows II


Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

Input dataset
output

FastQC:Read QC
Short read data from your current history
Contaminant list
html_file (html)

FastQC:Read QC
Short read data from your current history
Contaminant list
html_file (html)

Map with BWA for Illumina
Forward FASTQ file
Reverse FASTQ file
output (sam)

FastQC:Read QC
Short read data from your current history
Contaminant list
html_file (html)

Map with BWA for Illumina
Forward FASTQ file
Reverse FASTQ file
output (sam)

FastQC:Read QC
Short read data from your current history
Contaminant list
html_file (html)

Convert VCF to VCF_BGZIP
Choose Vcf file
output1 (vcf_bgzip)

Merge BAM Files
First file
with file
Input Files 1 > Add file
Input Files 2 > Add file
Input Files 3 > Add file
Input Files 4 > Add file
Input Files 5 > Add file
Input Files 6 > Add file
output1 (bam)
outlog (txt)

Estimate Library Complexity
SAM/BAM dataset
html_file (html)

Map with BWA for Illumina
Forward FASTQ file
Reverse FASTQ file
output (sam)

SnpSift Annotate
Variant input file in VCF format
VCF File with ID field annotated (e.g. dnSNP.vcf)
output (vcf)

Add or Replace Groups
SAM/BAM dataset to add or replace read groups in
outFile (bam, sam)

Add or Replace Groups
SAM/BAM dataset to add or replace read groups in
outFile (bam, sam)

Realigner Target Creator
BAM file
Known Variants 1 > Variant file (VCF format)
Known Variants 2 > Variant file (VCF format)
Operate on Genomic intervals 1 > Genomic intervals
output_interval (gatk_interval)
output_log (txt)

Slice VCF
Input dataset
Regions
output (vcf)

Mark Duplicate reads
SAM/BAM dataset to mark duplicates in
out_file (bam)
html_file (html)

Indel Realigner
BAM file
Restrict realignment to provided intervals
Known Variants 1 > Variant file (VCF format)
Known Variants 2 > Variant file (VCF format)
Known Variants 3 > Variant file (VCF format)
Operate on Genomic intervals 1 > Genomic intervals
output_bam (bam)
output_log (txt)

Unified Genotyper
BAM file 1 > BAM file
dbSNP ROD file
Operate on Genomic intervals 1 > Genomic intervals
output_vcf (vcf)
output_metrics (txt)
output_log (txt)

Insertion size metrics
SAM/BAM dataset to generate statistics for
html_file (html)

Base Recalibrator
BAM file
Covariates table recalibration file
Known Variants 1 > Variant file
Operate on Genomic intervals 1 > Genomic intervals
output_recal (gatk_report)
output_log (txt)

Print Reads
Covariates table recalibration file
BAM file
output_bam (bam)
output_log (txt)

Unified Genotyper
BAM file 1 > BAM file
dbSNP ROD file
Operate on Genomic intervals 1 > Genomic intervals
output_vcf (vcf)
output_metrics (txt)
output_log (txt)

The need of standardisation I

Sequencing techniques are starting to be used in clinical diagnosis, and therefore workflows have to assure:

- **Reproducibility**

Results always have to be reproducible

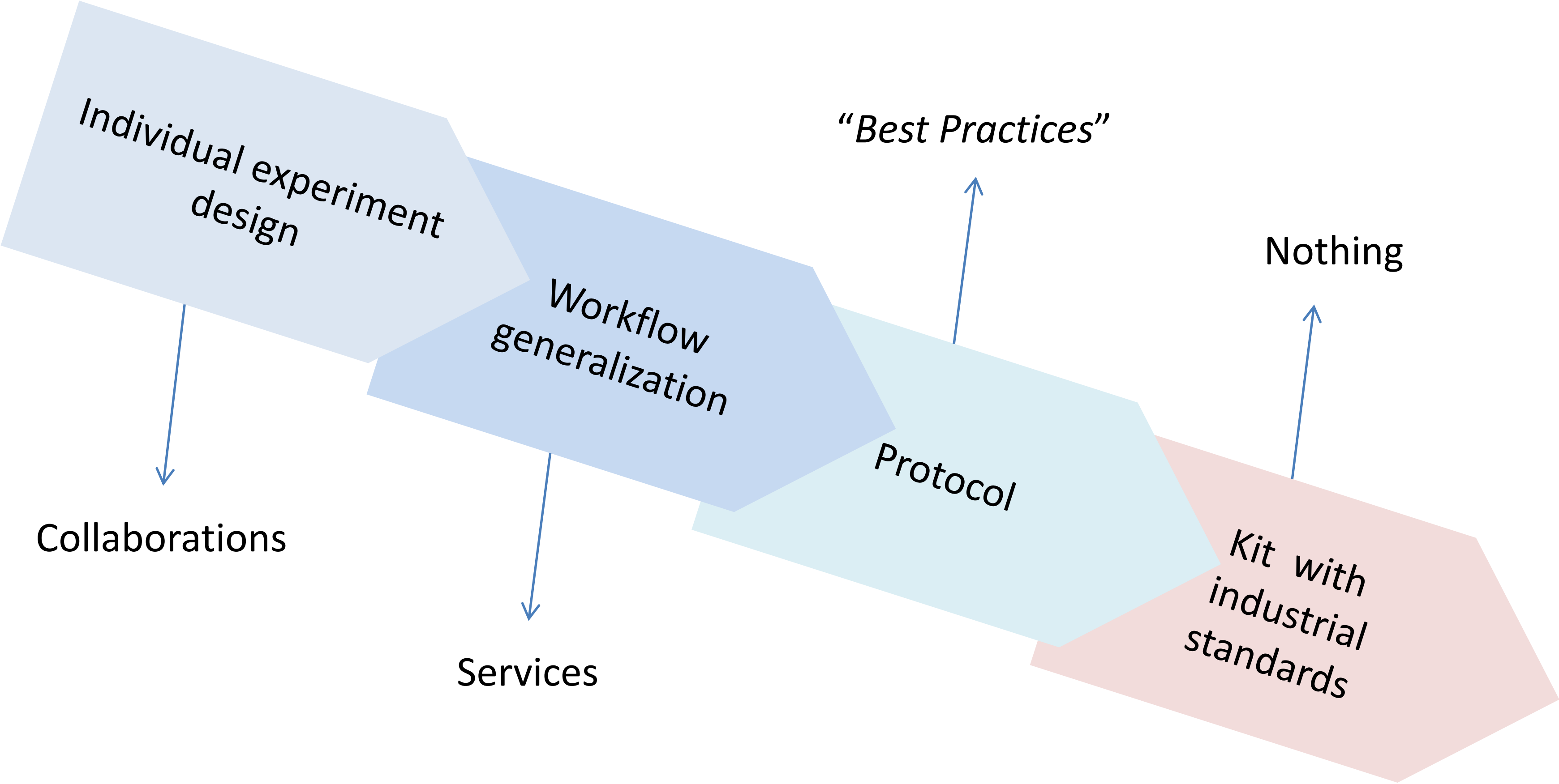
- **Portability**

The analysis workflow must be executable in different platforms

- **Scalability**

The analysis workflow must be able to work with different numbers of samples

The need of standardisation II



Nextflow I

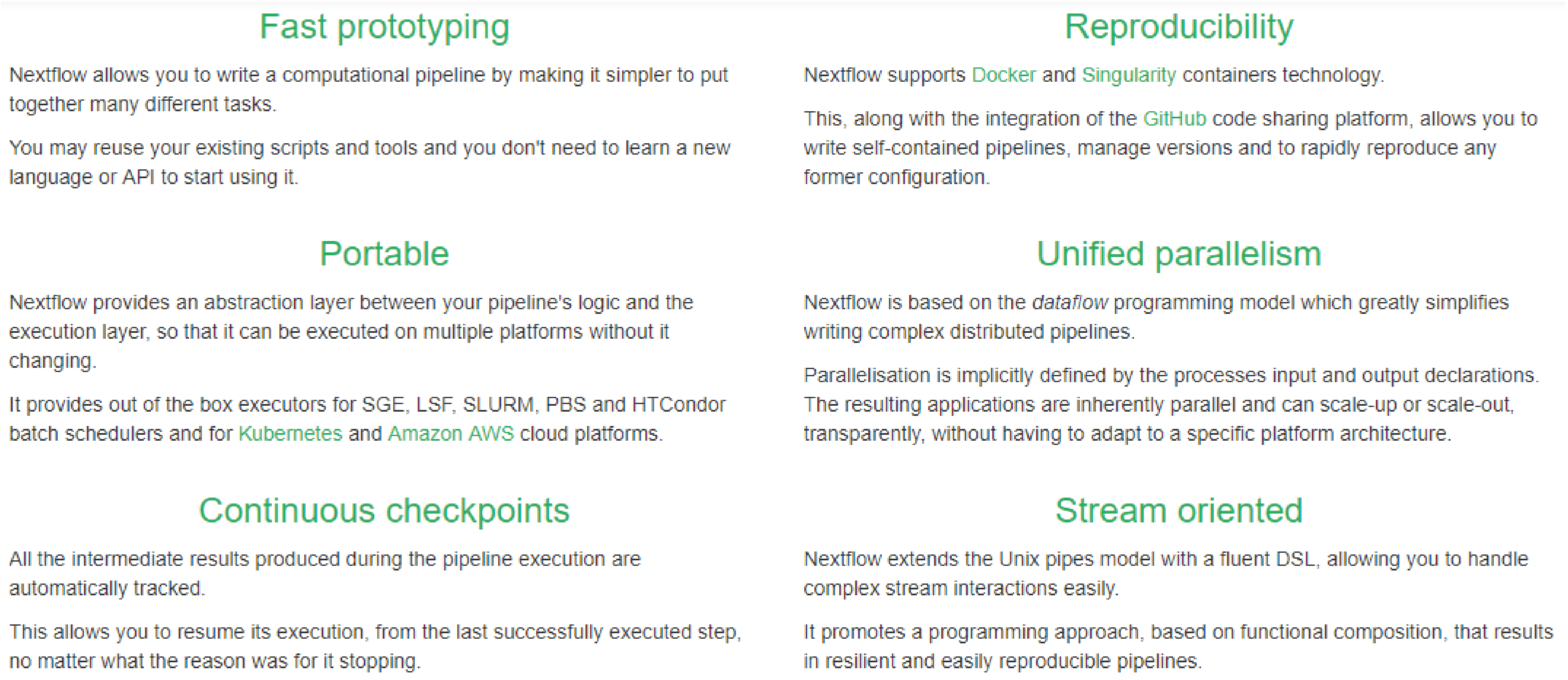
- **Nextflow** is a DSL for parallel and scalable computational pipelines.

- It enables **scalable and reproducible scientific workflows** using software **containers**.

- It **allows the adaptation of pipelines** written in the most common scripting languages.

- Its fluent **DSL** simplifies the implementation and the deployment of complex parallel and reactive workflows on clouds and clusters.

Nextflow II



Fast prototyping

Nextflow allows you to write a computational pipeline by making it simpler to put together many different tasks.

You may reuse your existing scripts and tools and you don't need to learn a new language or API to start using it.

Portable

Nextflow provides an abstraction layer between your pipeline's logic and the execution layer, so that it can be executed on multiple platforms without it changing.

It provides out of the box executors for SGE, LSF, SLURM, PBS and HTCondor batch schedulers and for Kubernetes and Amazon AWS cloud platforms.

Continuous checkpoints

All the intermediate results produced during the pipeline execution are automatically tracked.

This allows you to resume its execution, from the last successfully executed step, no matter what the reason was for it stopping.

Reproducibility

Nextflow supports Docker and Singularity containers technology.

This, along with the integration of the GitHub code sharing platform, allows you to write self-contained pipelines, manage versions and to rapidly reproduce any former configuration.

Unified parallelism

Nextflow is based on the *dataflow* programming model which greatly simplifies writing complex distributed pipelines.

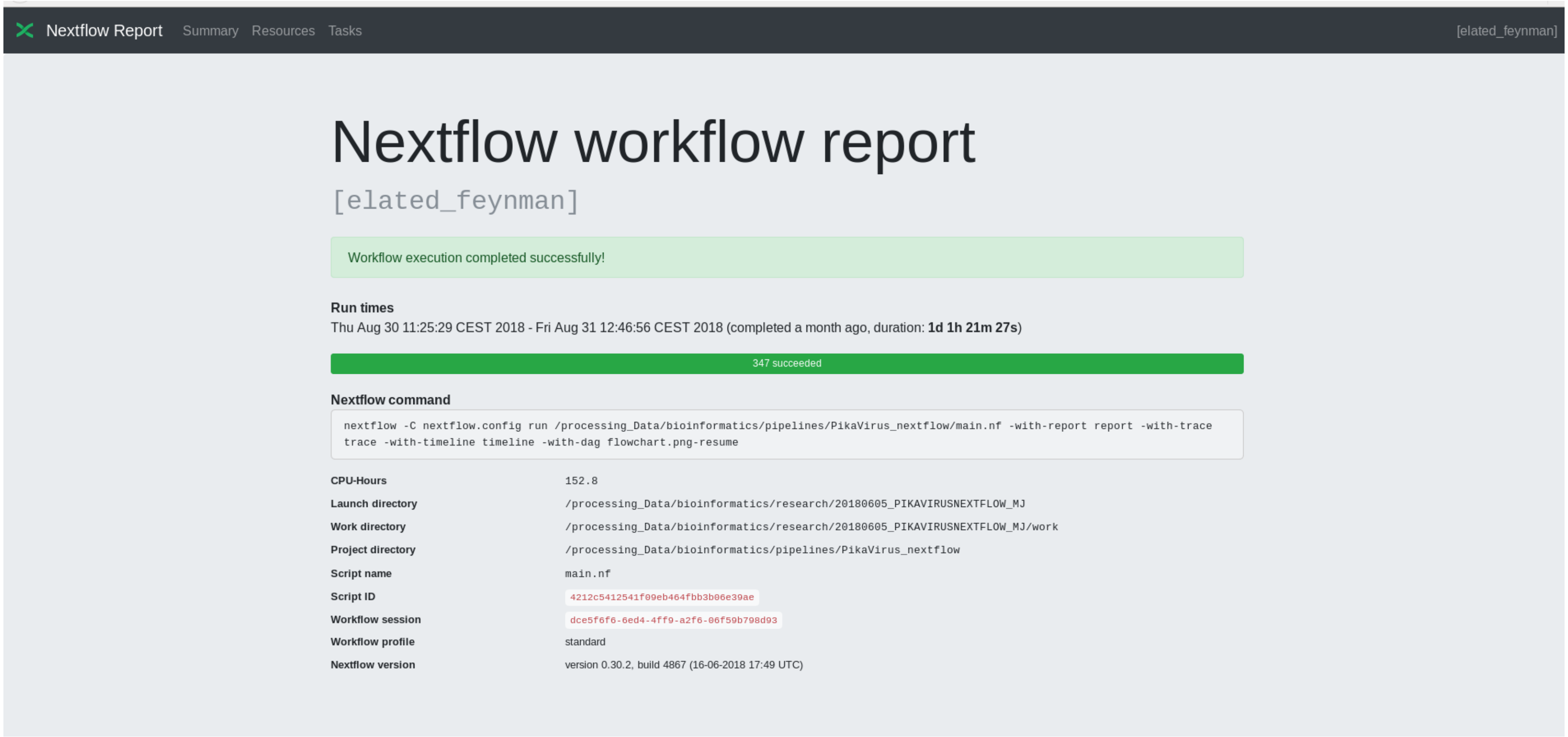
Parallelisation is implicitly defined by the processes input and output declarations. The resulting applications are inherently parallel and can scale-up or scale-out, transparently, without having to adapt to a specific platform architecture.

Stream oriented

Nextflow extends the Unix pipes model with a fluent DSL, allowing you to handle complex stream interactions easily.

It promotes a programming approach, based on functional composition, that results in resilient and easily reproducible pipelines.

Nextflow III



Nextflow workflow report

[elated_feynman]

Workflow execution completed successfully!

Run times

Thu Aug 30 11:25:29 CEST 2018 - Fri Aug 31 12:46:56 CEST 2018 (completed a month ago, duration: **1d 1h 21m 27s**)

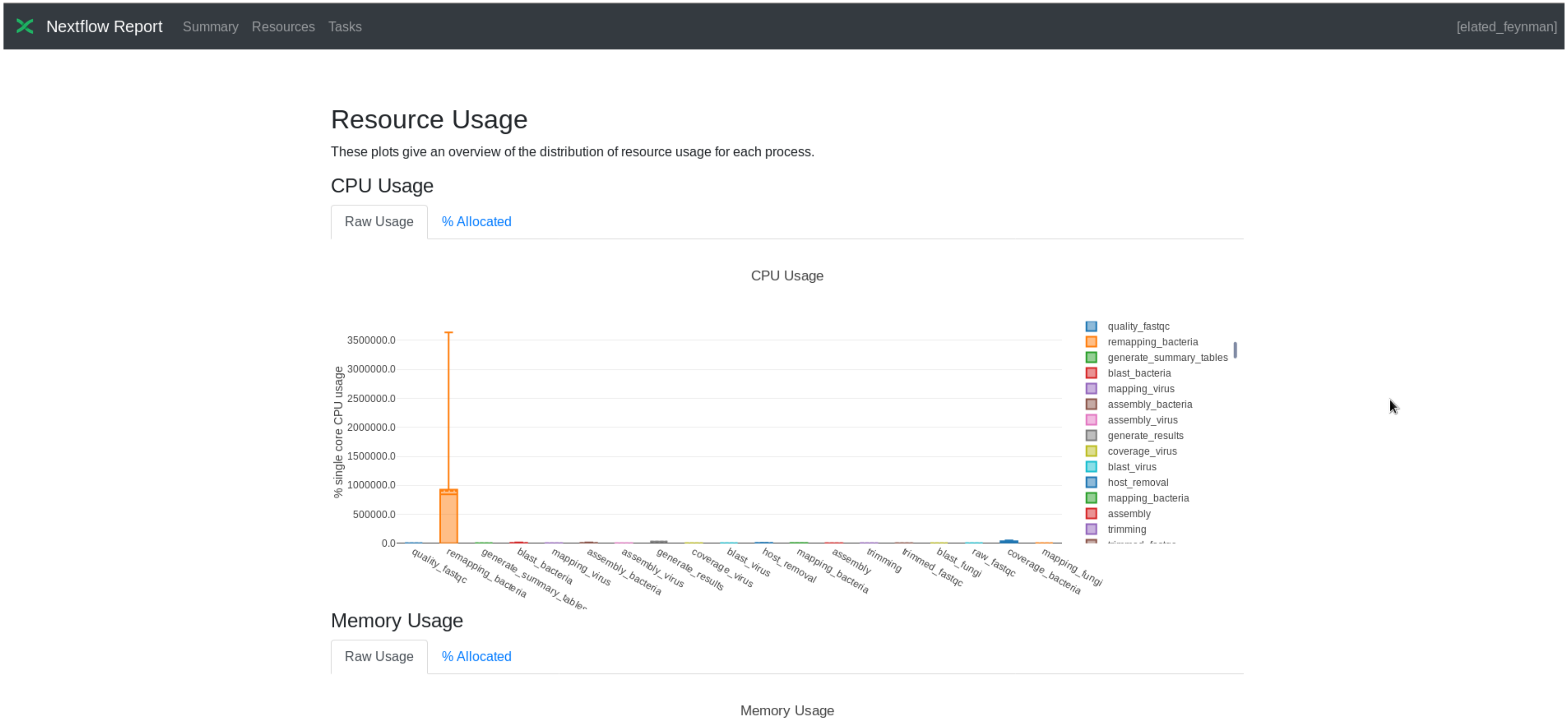
347 succeeded

Nextflow command

```
nextflow -C nextflow.config run /processing_Data/bioinformatics/pipelines/PikaVirus_nextflow/main.nf -with-report report -with-trace
trace -with-timeline timeline -with-dag flowchart.png-resume
```

CPU-Hours	152.8
Launch directory	/processing_Data/bioinformatics/research/20180605_PIKAVIRUSNEXTFLOW_MJ
Work directory	/processing_Data/bioinformatics/research/20180605_PIKAVIRUSNEXTFLOW_MJ/work
Project directory	/processing_Data/bioinformatics/pipelines/PikaVirus_nextflow
Script name	main.nf
Script ID	4212c5412541f09eb464fbb3b06e39ae
Workflow session	dce5f6f6-6ed4-4ff9-a2f6-06f59b798d93
Workflow profile	standard
Nextflow version	version 0.30.2, build 4867 (16-06-2018 17:49 UTC)

Nextflow IV



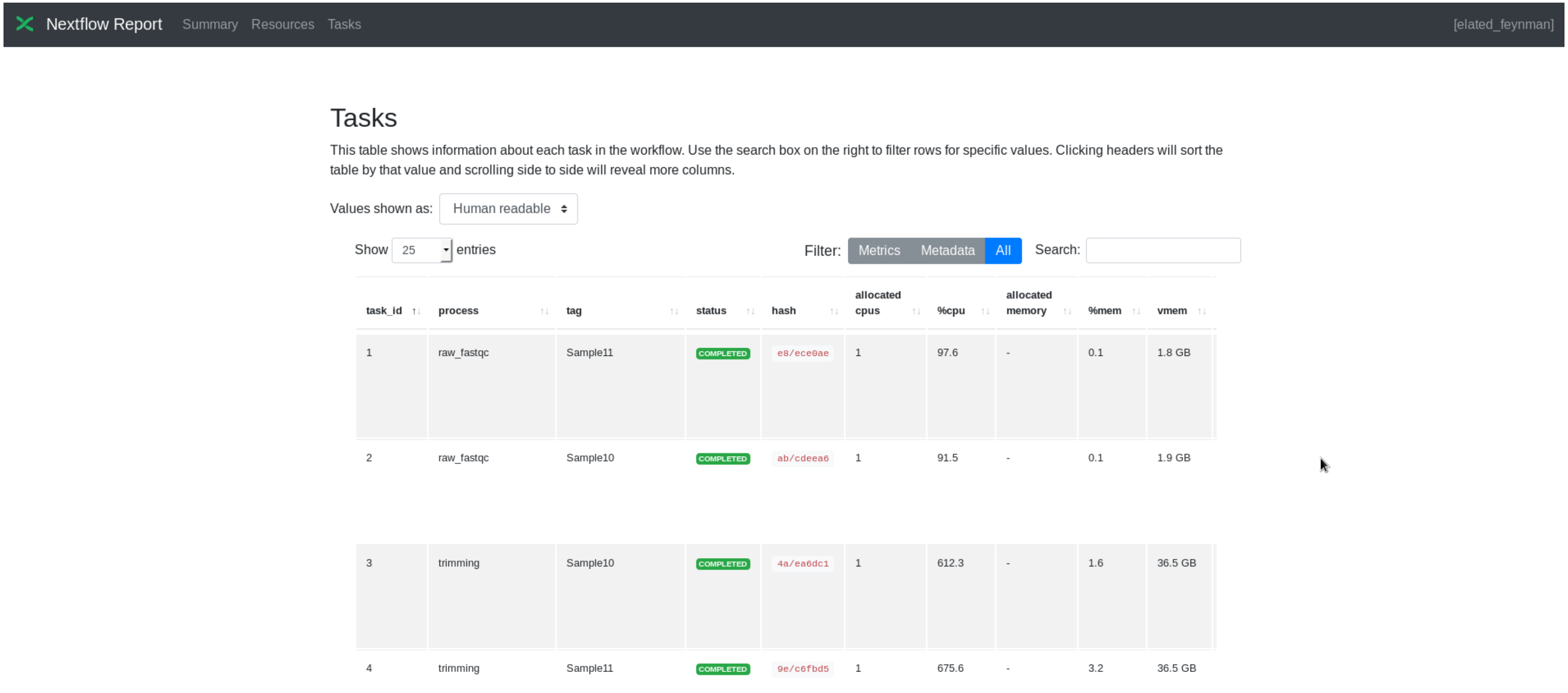
Resource Usage

These plots give an overview of the distribution of resource usage for each process.

CPU Usage

Memory Usage

Nextflow V



Tasks

This table shows information about each task in the workflow. Use the search box on the right to filter rows for specific values. Clicking headers will sort the table by that value and scrolling side to side will reveal more columns.

Values shown as: Human readable

Show 25 entries

Filter: Metrics Metadata All Search:

task_id	process	tag	status	hash	allocated cpus	%cpu	allocated memory	%mem	vmem
1	raw_fastqc	Sample11	COMPLETED	e8/ece0ae	1	97.6	-	0.1	1.8 GB
2	raw_fastqc	Sample10	COMPLETED	ab/cdeea6	1	91.5	-	0.1	1.9 GB
3	trimming	Sample10	COMPLETED	4a/ea6dc1	1	612.3	-	1.6	36.5 GB
4	trimming	Sample11	COMPLETED	9e/c6fbd5	1	675.6	-	3.2	36.5 GB

Containers I

Linux containers is a generic term for an implementation of operating system-level virtualization for the Linux operating system.

Containers allow us to **port** pipelines and **replicate** their exact execution environments across different hardware.

Currently, a number of such implementations exist, and they are all based on the **virtualization, isolation, and resource management mechanisms provided by the Linux kernel.**

Containers II

Singularity is a free, cross-platform and open-source computer program that performs operating-system-level virtualization.

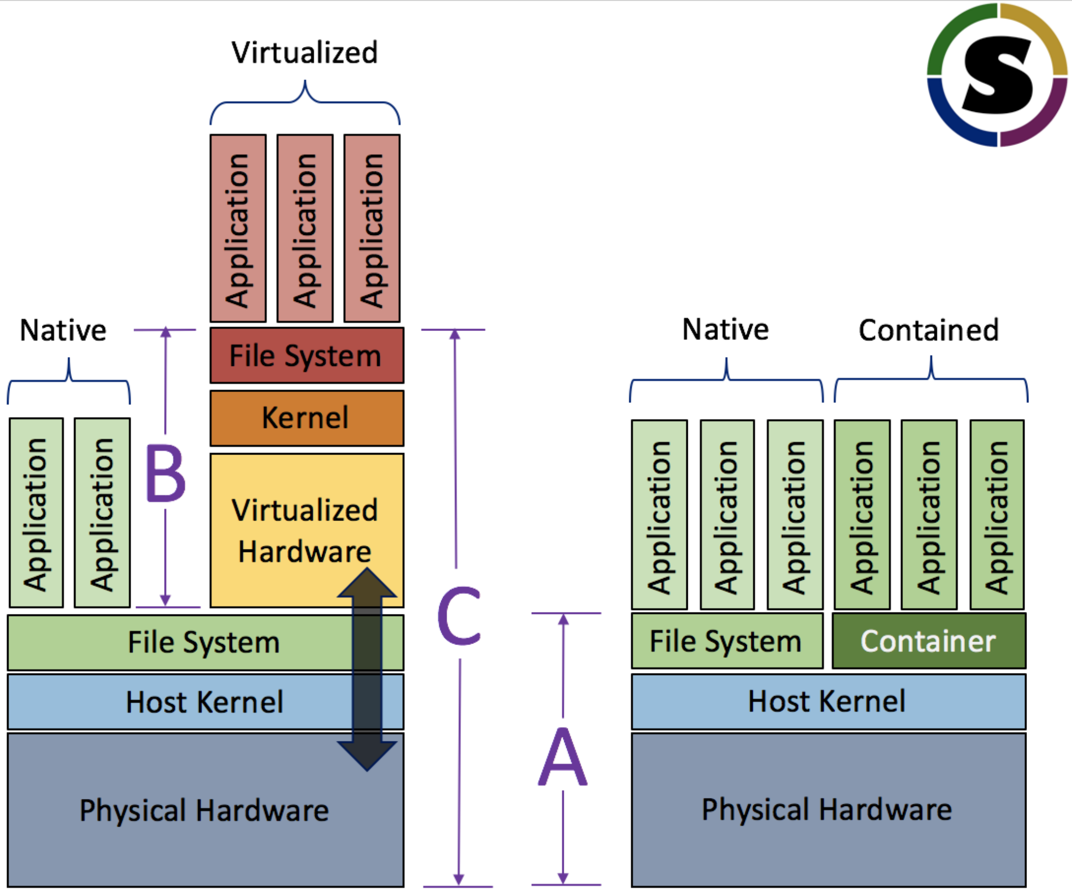
One of the main uses of Singularity is to bring containers and **reproducibility to scientific computing** and the HPC world.

While Docker is broadly used, Singularity is **fully compatible with Docker**, plus Singularity does **not require root permissions** to be executed.

Singularity I

VIRTUAL
MACHINES
VS.
CONTAINERS

- Applications running within a container will always be "closer" to the physical hardware
  - Notice how close to native a container behaves
- Applications running through a virtual machine will always have multiple levels of indirection
- The container's proximity to the physical hardware equates to less overhead, higher performance and lower latency



Virtual Machine Architecture

Container Architecture

Singularity II

Singularity image runs on the same level as the OS, directly above the kernel, and can access all hardware in the machine.

Not needing to virtualise the hardware and run a kernel again makes this kind of virtualisation really effective.

Filesystem is shared, and some paths are automatically mounted (/tmp and /home), while the others are optional.

Files of the host system can be created, modified and deleted from the image in the mounted folders.

Results I

Before the implementation of these combination of framework, software and guidelines, executing a workflow in an HPC environment consisted in the following steps:

- Loading input data
- Asking sysadmin to install dependencies
- Load references
- Estimate and book computational resources
- Manually execute each step of the pipeline, or automate it with a script
- Wait with no control over the process status until finished

Results II

Now a simple command works out of the box in any machine:

```
nextflow run //buisciii/main.nf –profile sigularity
```

Plus, it give us:

- Dependency and computing automatization
- Resource usage statistics
- Easy to share and maintain
- Reproducibility and re-entrancy
- Transparency

Thanks for your attention!

And this is only the tip of the iceberg...
Check this if you wanna know what’s really going under the hood:

https://github.com/BU-ISCIII