The Computing Revolution in Biosciences

BU-ISCIII

Unidades Comunes Científico Técnicas – SGSAFI-ISCIII

28-31 Octubre 2024, 6ª 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

Healthcare IT News GLOBAL EDITION TO

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

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

Healthcare IT News GLOBAL EDITION TO

Microsoft, Google invest in precision medicine startup DNAnexus

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

Computing in Biosciences I

Research used to focus on 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

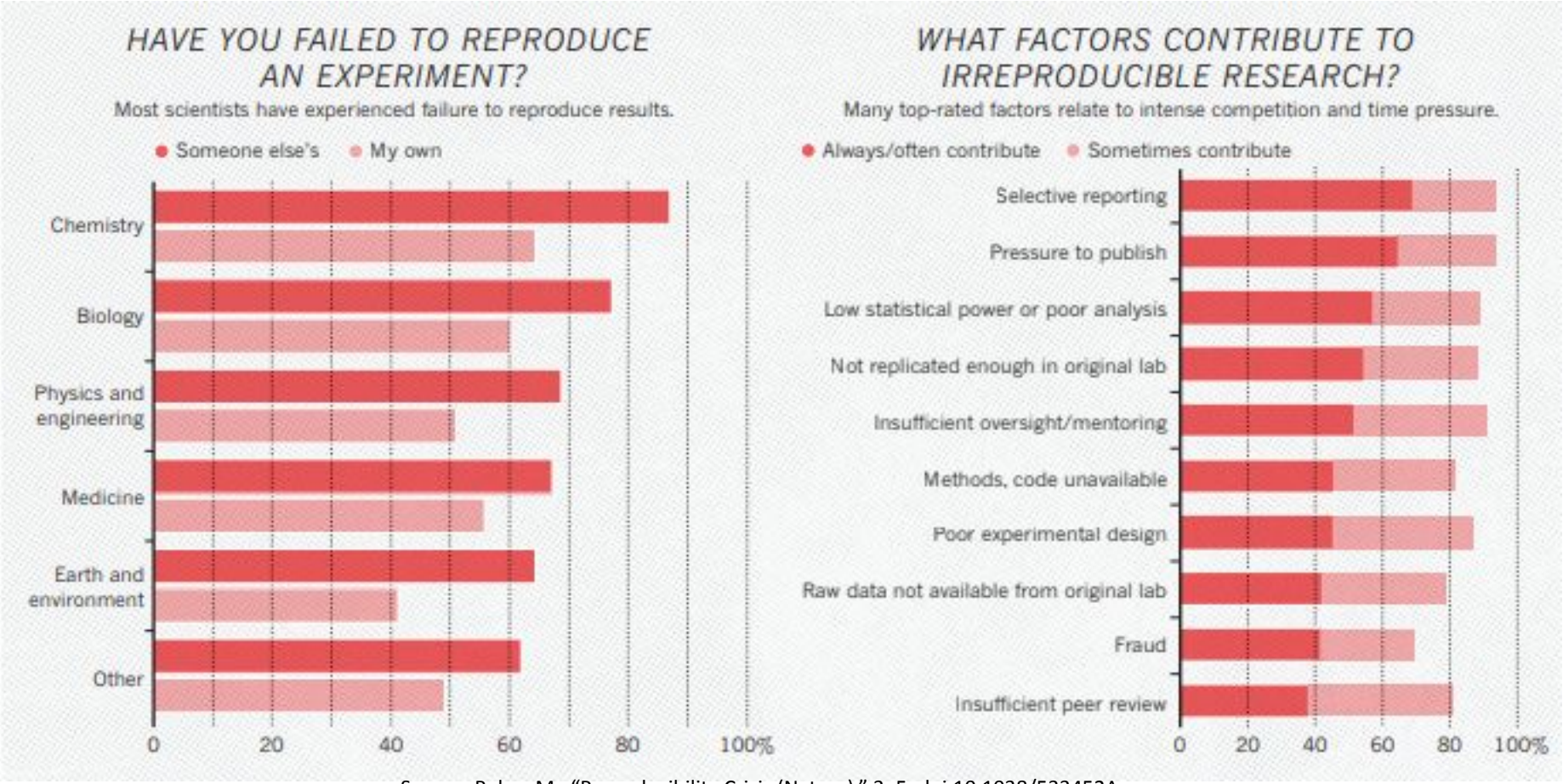
Pros	Cons
No need to storage intermediate files	Your data is in someone else's computer No backups or data management schemes
No need to install software Partial control over installed software	No control over installed software, versions and future availability
Graphic interface	No control over hidden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



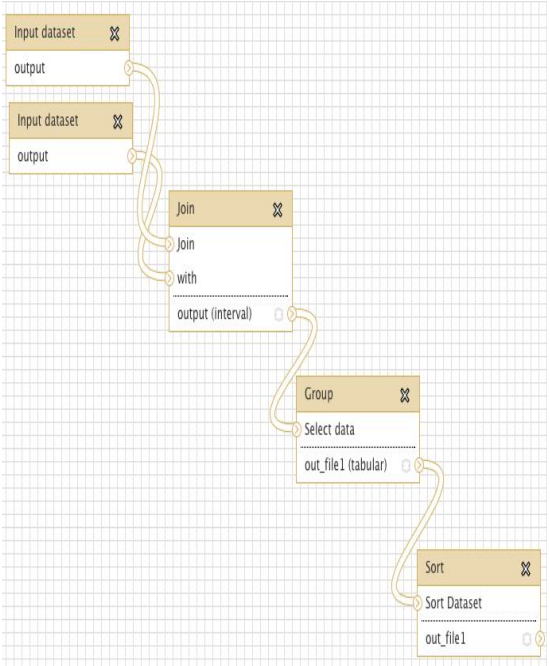
Source: Baker, M. "Reproducibility Crisis (Nature)," 3–5. doi:10.1038/533452A.

Change of Paradigm I

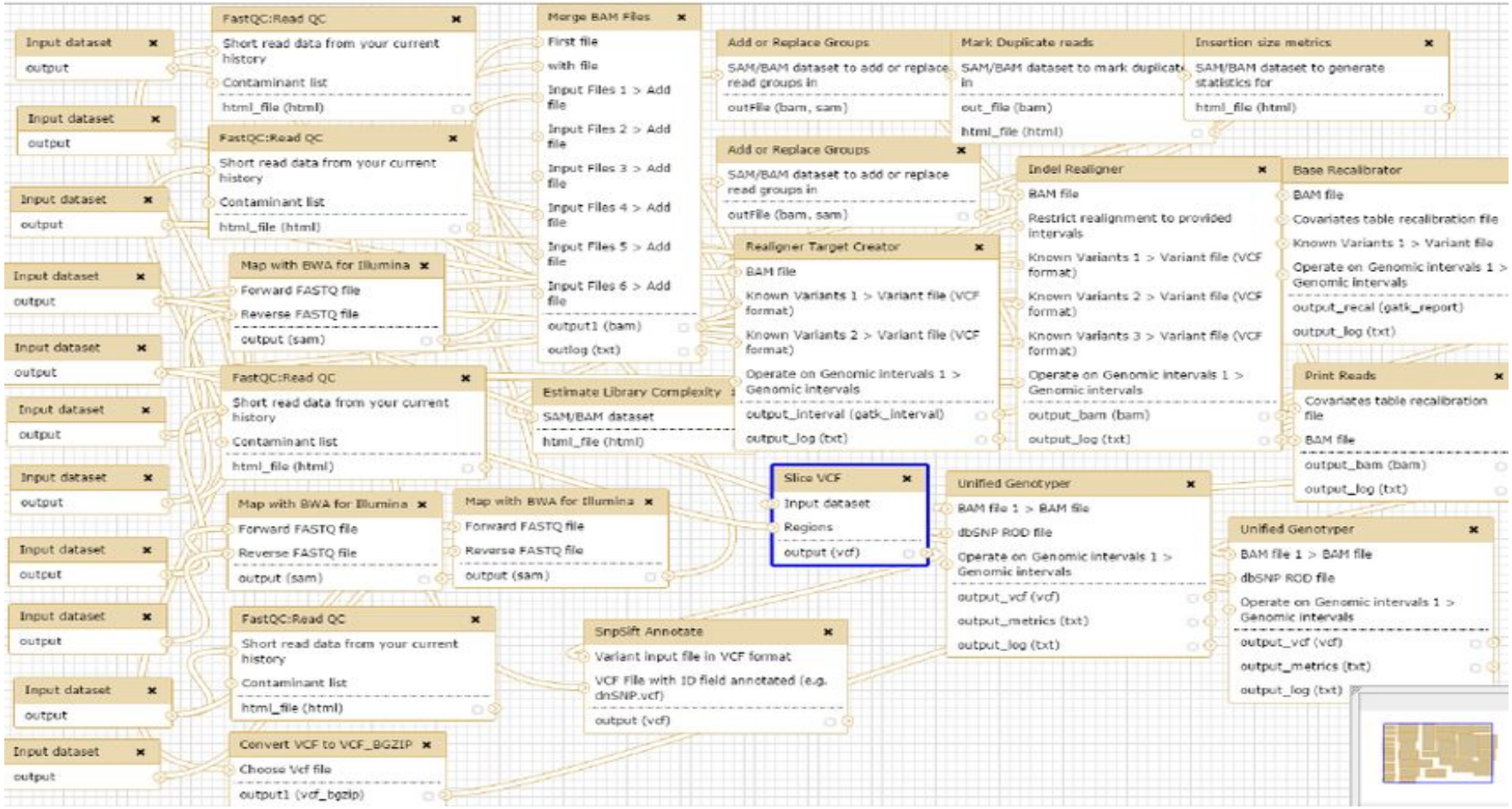
1 sample	multiple samples
Research only: NGS was still a new thing, no applications 10 years ago	**Many applications:** research, clinical, industrial, forensic, military, …
Reproducibility is not needed: Why would anyone reanalyse this?	**Reproducibility, scalability , portability and standardisation 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**

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



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

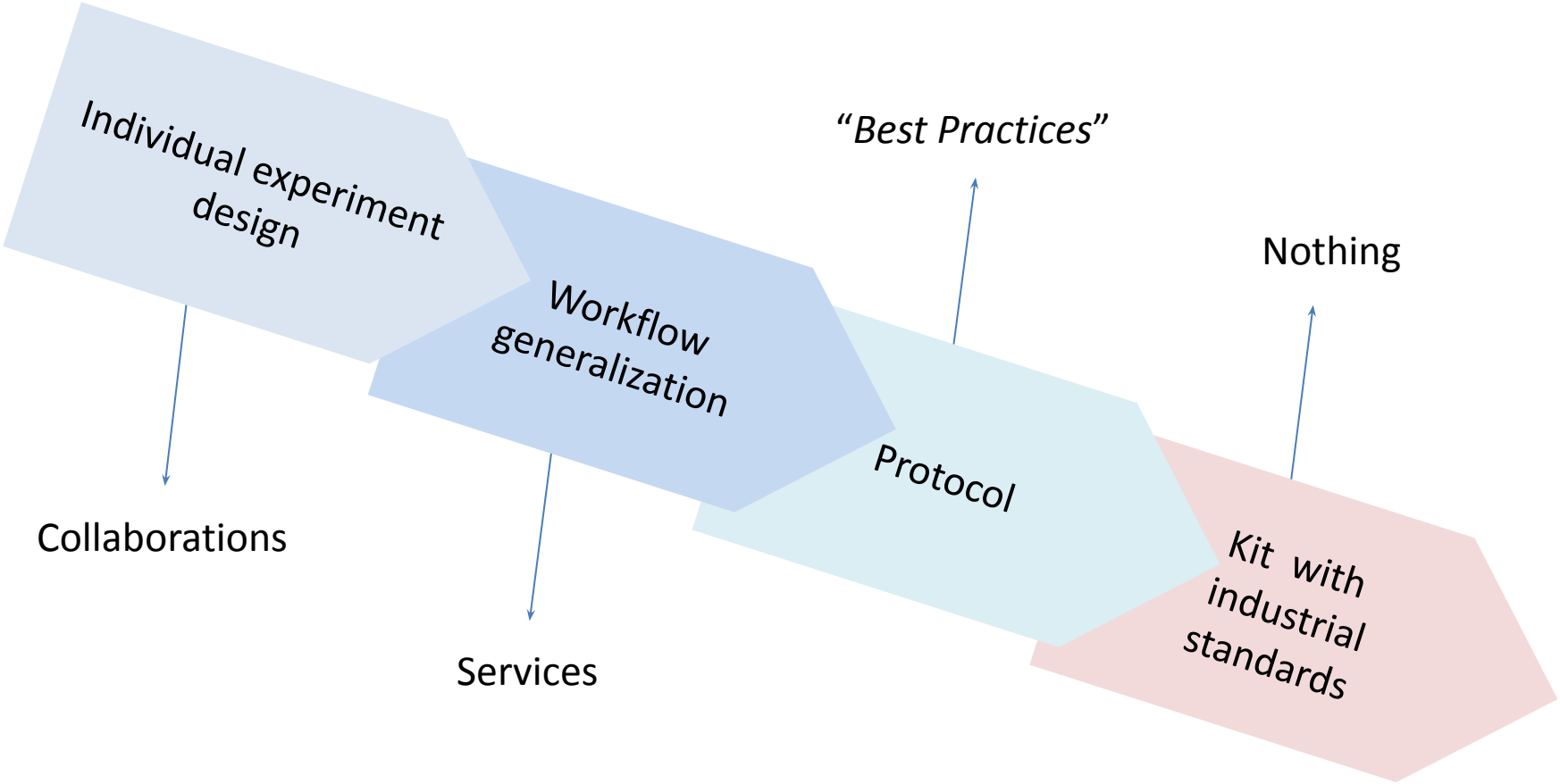
- **Portability**

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.

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.

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.

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.

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.

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 Report Summary Resources Tasks [elated_feynman]

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)

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.

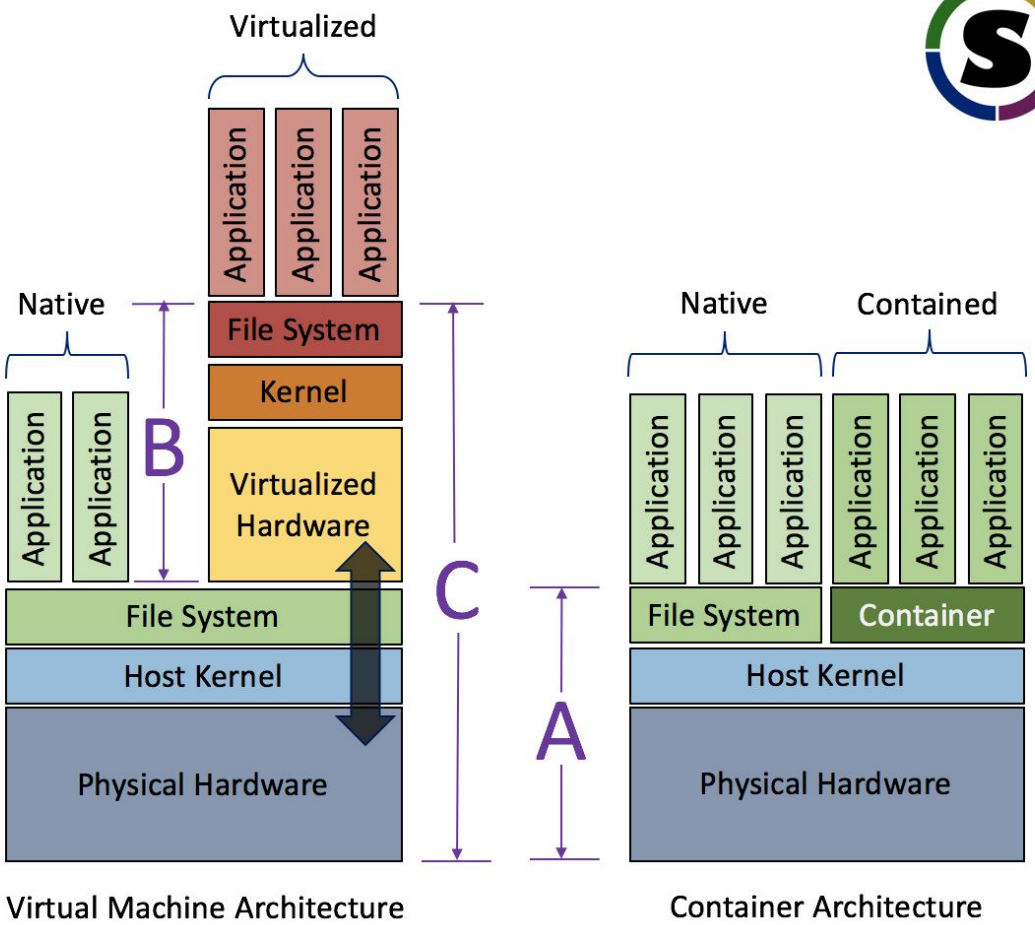
One of the main uses of Singularity is to bring containers and **reproducibility to scientific computing** and the HPC world.

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



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 singularity
```

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