DEEPHEALTH

The modernization of HPC applications for the cloud era

Prof. Marco Aldinucci
Head of the Parallel Computing group at Univ. of Torino
Director of HPC-KTT national lab at CINI
Italian delegate at the EuroHPC GB

Iacopo Colonnelli
PhD student at the Univ. of Torino, Italy

cini national lab HPC Key Technologies & Tools

HPC4AI

Highlights

StreamFlow, Jupyter-Workflow & JaaS: portability of modern HPC+AI applications

- **Workflows for HPC, AI, and the cloud - who is approaching the others?**
  - Example: The “universal pipeline” for AI-assisted COVID-19 pneumonia diagnosis
- **The StreamFlow Workflow Management System**
  - Open standard Workflows focusing on portability and performance portability
  - A tools for engineering portable AI (and not AI) applications on HPC+cloud with containers
- **Towards usability, interactivity and portability: Jupyter-as-a-Service**
  - From the cloud to HPC : JupyterHub and Jupyter enterprise gateway
- **Jupyter-workflow**
  - A sequentially-equivalent, Streamflow-based distributed and parallel implementation of Jupyter

Orthodox HPC approach

Concepts

- **Primary metric: performance at any cost (time and price)**
  - Strong and weak scalability, FLOPS, ...
- **Methodology: Co-design**
  - In principle co-developing <HW, dev-tools, apps> - in practice expensive performance tuning
- **Execution model: Batch Jobs**
  - Batch (SLURM, ...) on large homogenous or modular clusters with heterogenous nodes (multicore+GPU)
  - Mostly in memory - I/O-bound applications typically critical
- **Programming model: low-level task & data parallel**
  - MPI+x (OpenMP, CUDA, OpenACC, ...) - the programmer has full knowledge of data partitions
  - Locally synchronous data parallel kernels + coarse grain tasks (complex workflows)

Heretical HPC approach

Modern concepts

- **Primary metric: performance portability**
  - Helps in preserving the value of investment of applications and upgrading HW
- **Methodology: Segregation+Composition**
  - Helps modular design, correctness, robustness, reuse
- **Execution model: Batch+Interactive Jobs, multi-tenant services**
  - Interactive, multi-platform with data-location awareness, tasks and long standing services
  - Provably secure and accountable data on shared infrastructure
- **Programming model: high-level data-centric + task & data parallel**
  - Spark, Tensorflow, Pytorch, MPI+x (OpenMP, CUDA, OpenACC, ...), ...
  - Workflows of easily composable tasks and services

HPC at a cross-road

More industry, much more applications and frameworks, multi-tenancy, ...

- **Novel applicative pipelines (as digital twins)**
  - Use traditional kernels + AI/ML algorithms: data movements are more important than FLOPS
  - Requires many different node configurations, frameworks and long-standing services
  - Should exploit rapidly evolving HW accelerators to reach peak performance
  - Most of the frameworks available via OS containers

- **HPC platforms are more and more owned/used by industries**
  - They cannot employ hundreds of highly skilled programmers to maintain the code
  - Single-tenant batch workloads don't fit the needs of large industries (with many divisions)

From HPC to the cloud or vice-versa?

HPC is fated to meet AI and the cloud is the bridge

- **HPC managed as-a-service**
  - HPC becomes an OTT business (unless for “exotic” platforms)
  - Datacenter-as-a-Service
  - Open standards and interpretability are the way for avoiding tech lock-in & security concerns

- **Cloud methods adopted by on-prem HPC platforms**
  - Multi-tenancy, containers/VMs, batch and interactive usage… but managed with HPC SLAs
  - Security easier to manage: defence, finance, oil&gas, space, …
  - Composition and reuse: reduced SW development costs of traditional & data-centric applications
  - Middleware tools are the picklocks for changing working routines

HPC4AI

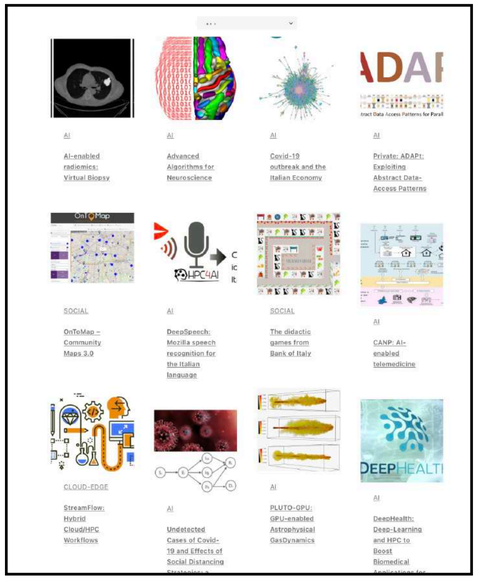
Two systems “**DevOps**” (Production+Dev) each made by two clusters
1 Cloud/OpenStack + 1 HPC/SLURM

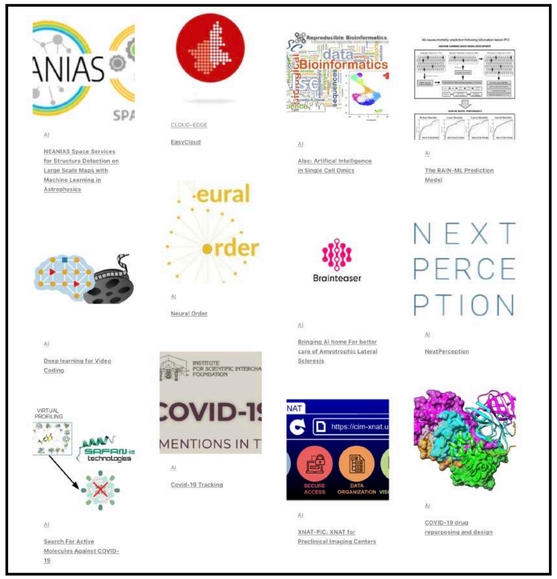
Overall **11k+ cores** (Intel, ADM), **120 GPUs** (T4/V100/A40/A100), 4 novel **Arm+A100+BlueField** Nvidia-DEV platforms
3PB in 5 systems: CEPH (SSD/rep3, Rot/ErasureCode), EMC2 (Unity+Avamar)

40+ research projects running (HPC, AI, ...), mostly with critical data (medical)
Several in-house PaaS: SLURM-calendar integration, Kubernetes-as-a-Service, **Jupyter-as-a-Service**, ...

Our “own” **250KVA Tier-3** datacenter - 1.04 P.PUE design (**PUE 1.1/year** measured)
Production system: 21 months with 0 “system-down” incidents

We develop it and manage it! Our “products” is knowledge and skilled students





Example

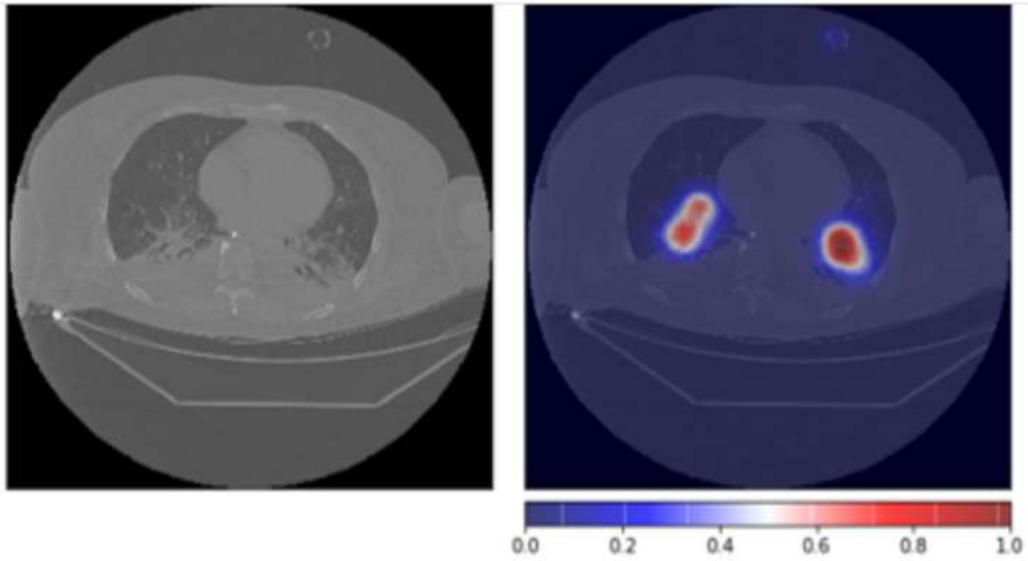
The COVID-19 universal pipeline

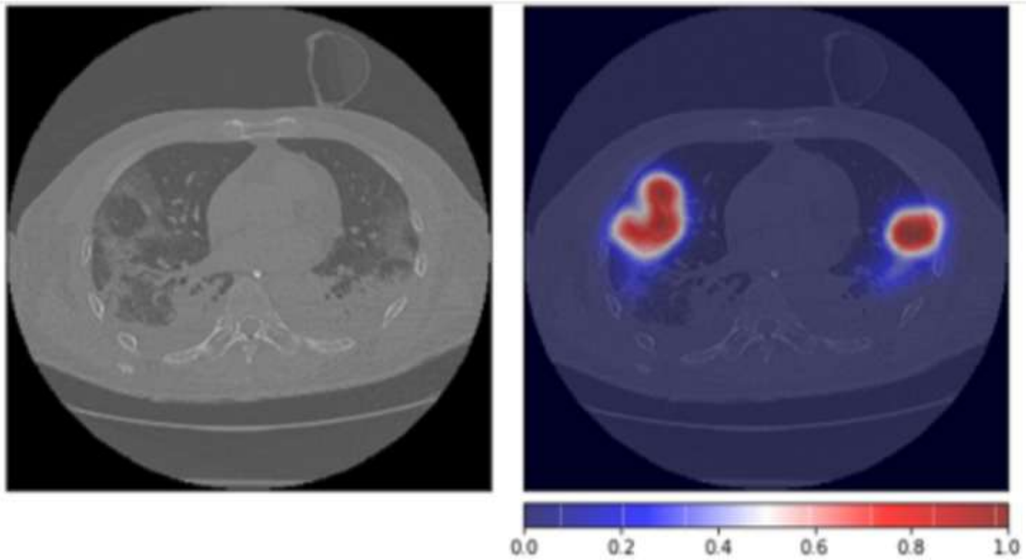
developed by the CLAIRE taskforce on COVID-19

CLAIRE Confederation of Laboratories for Artificial Intelligence Research in Europe

Assessing a baseline for AI-assisted diagnosis of COVID-19 pneumonia

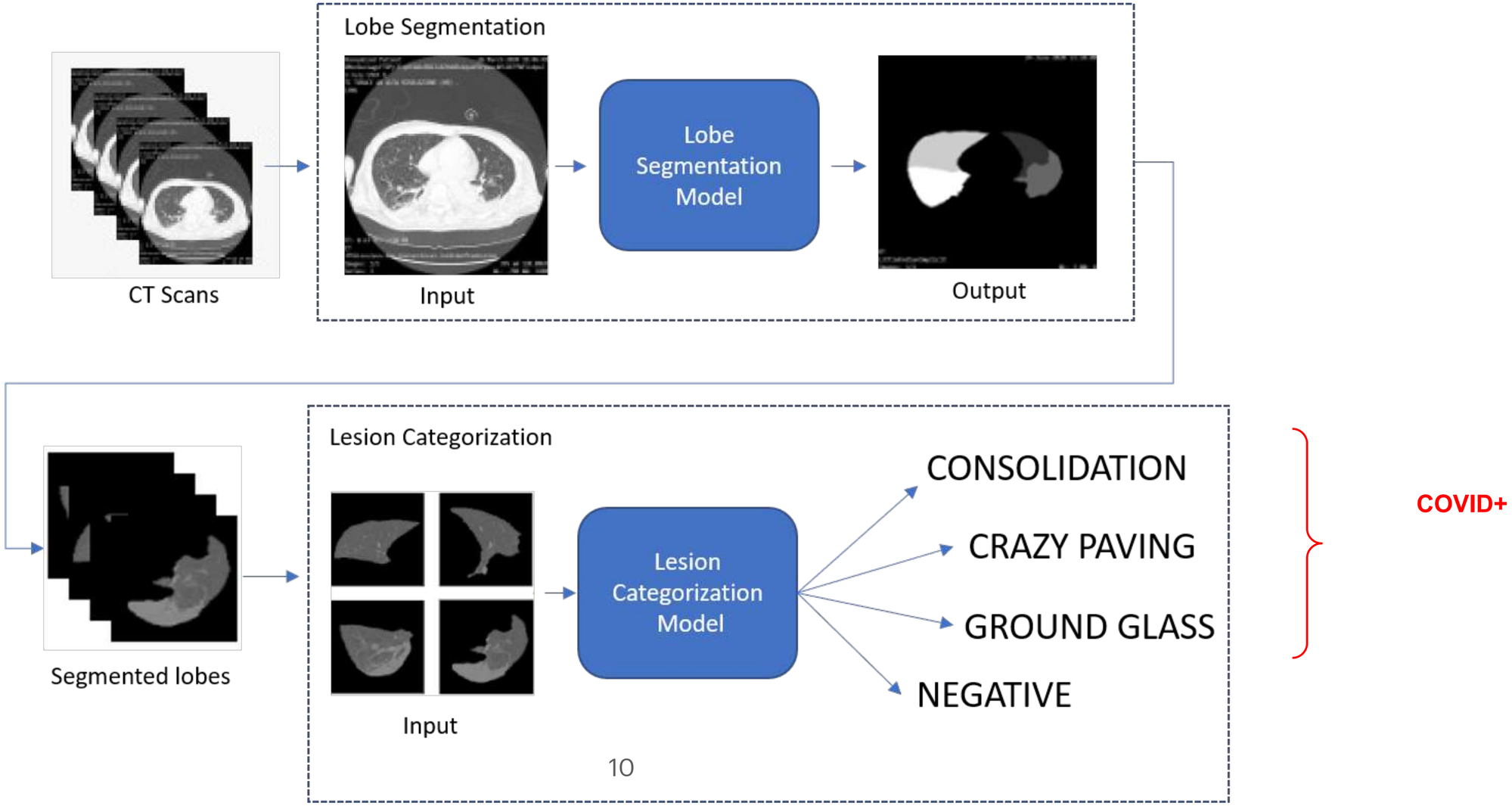
- CT scans
  - Many papers on “DNN for classification” published under fast track
  - Relatively simple: many existing DNN succeed in recognising COVID-19 pneumonia: AlexNet, DenseNet, ResNet, ...
- X-ray images
  - Usage of X-ray simplify the medical practice: cheaper, smaller, movable, easier to use ...
  - Too low resolution to really see COVID-19 pneumonia lesions





AI-assisted diagnosis of COVID-19 pneumonia

A common diagnosis pipeline



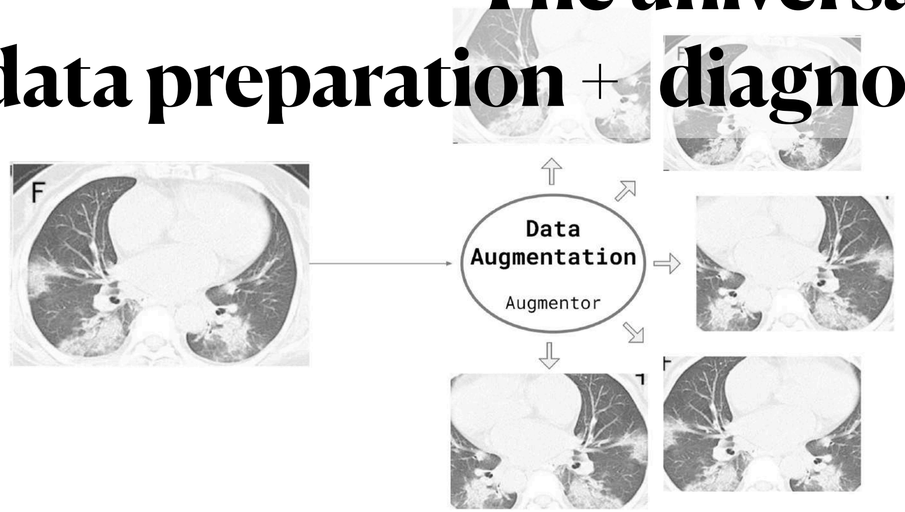
Universal

Reproducible, parametric, portable with performance

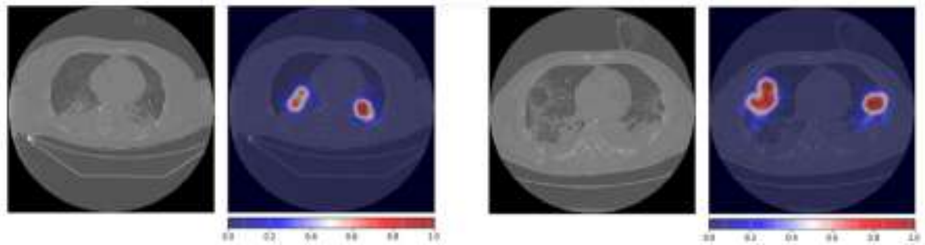
- **We do not want to produce a solution, there are already many around**
  - That is the problem of AI:
    `1) One DNN, one dataset, one hyperparam conf, goto 1`

- **We want instead to capture and abstract a broad class of solutions (i.e. universal)**
  - Fully reproducible ➜ fully formalized as a workflow
  - Expressing a broad class of pipelines ➜ code is another hyper-parameter
  - Portable on different platforms ➜ described with a new generation of workflow
  - Performance is another hyper-parameter

The universal pipeline: data preparation + diagnosis & metrics assessment



Rotations, translations, reflection, ...



Explainable (with heat map) classifications of COVID-19 lesions, e.g. crazy paving, consolidation, ground glass.

Dataset, e.g., BIMCV-COVID19 120k images from 1300+ patients

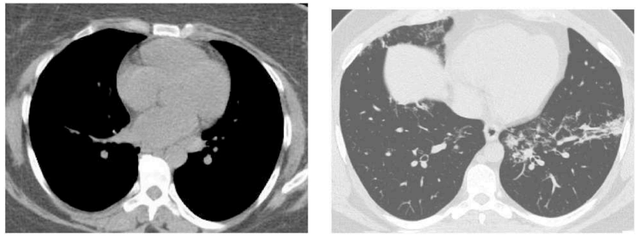
DNN

DNN

DNN

TC scans → Pre-processing → Segmentation → Augmentation / Pre-training → Explainability / Explainability Classification → Cross-Validation → Performance metrics

Noise reduction: contrast, normalisation



Segmentation of region of interest

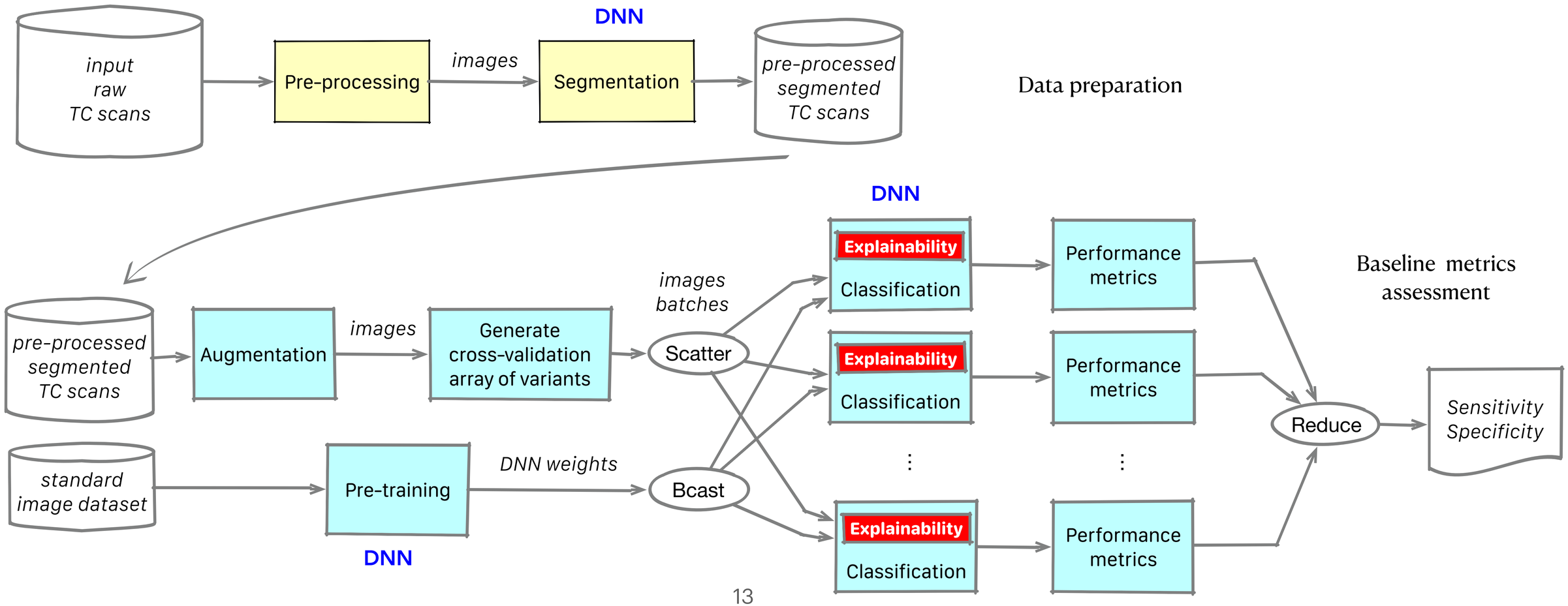
Original CT scan | Lung segmentation | Lobe segmentation

Primary metrics
- Sensitivity (true and false positives)
- Specificity (true and false negatives)

Derived metrics
- Accuracy
- AUC of TPR vs FPR
- Diagnostic Odds Ratio

The universal pipeline

Data preparation (run once) and Metrics assessment



Main variants

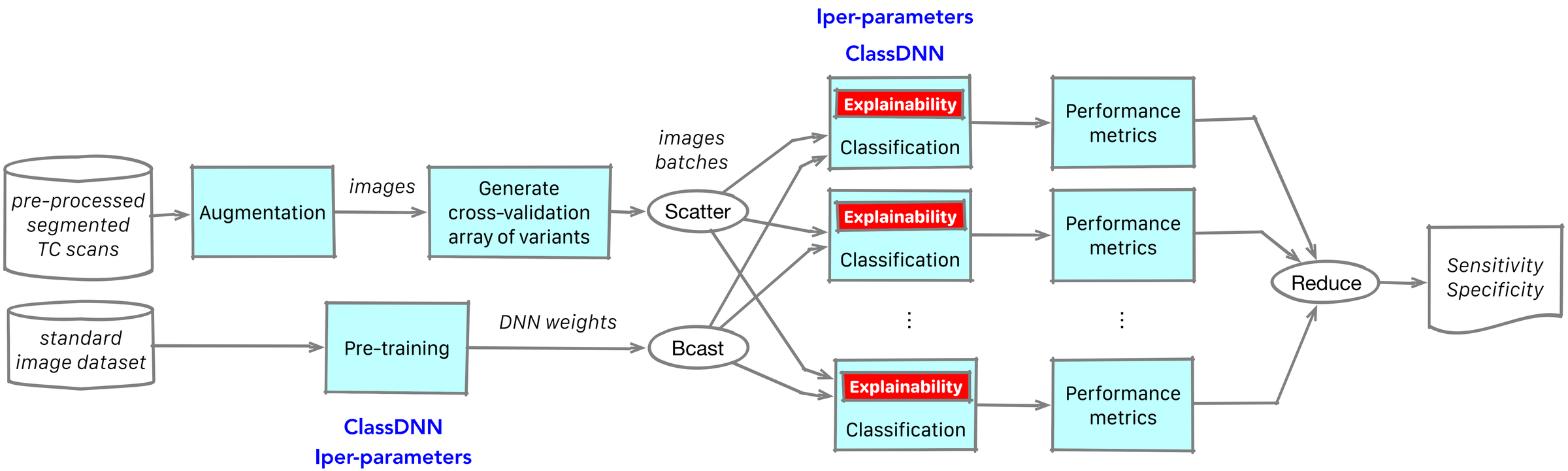
state of the art DNNs

Iper-parameters (9 variants)

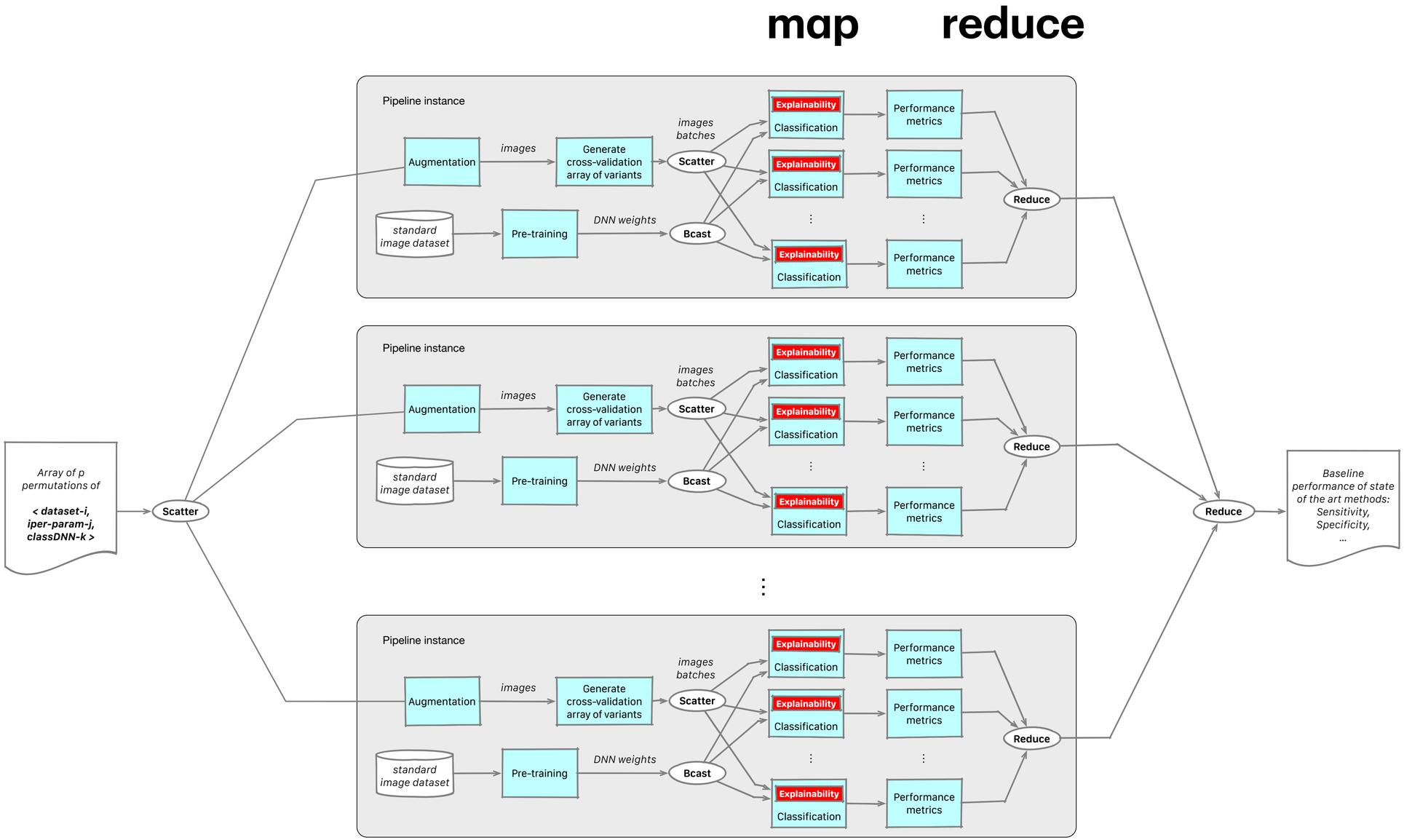
A. Learning rate (3 valori)
B. Weight decay (3 valori)
C. Learning rate decay step (3 valori)

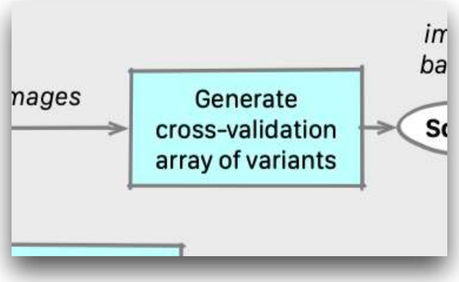
ClassDNN (11 variants)

1. AlexNet
2. ResNet (18, 34, 50, 101, 152)
3. DenseNet (121, 169, 201)
4. SqueezeNet
5. ResNeXt



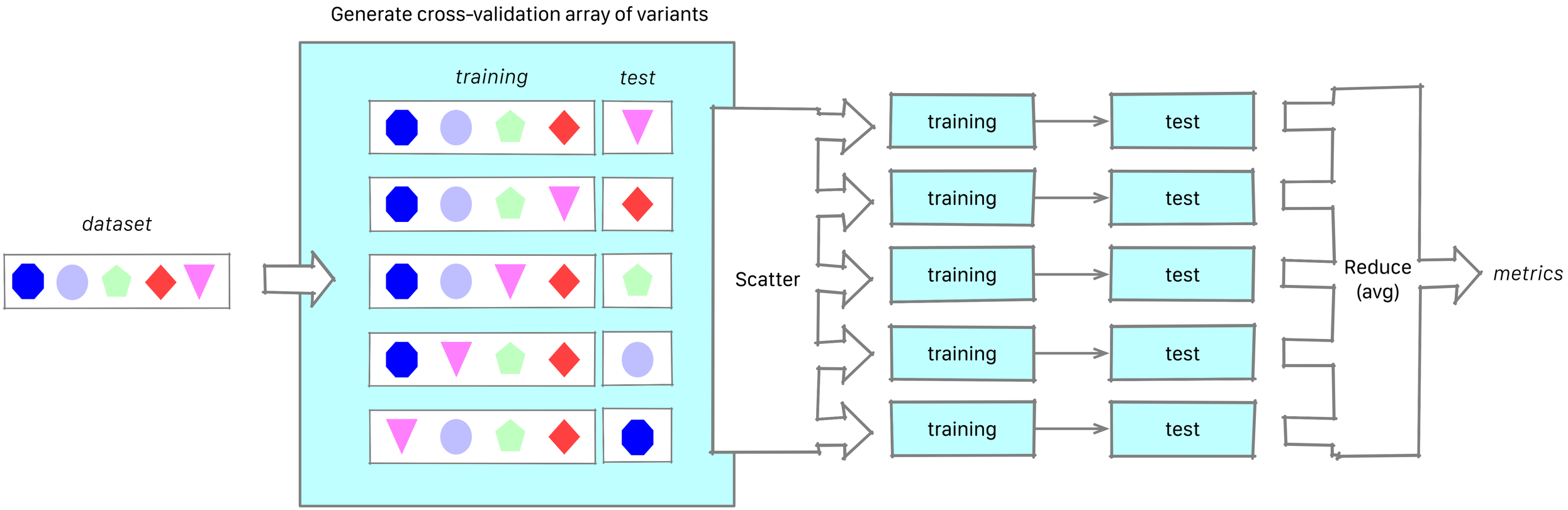
The universal pipeline (unfolded)





Graceful refinement of parallelism

Further parallelisation of cross-validation stage



Comparing all the variants is a HPC task

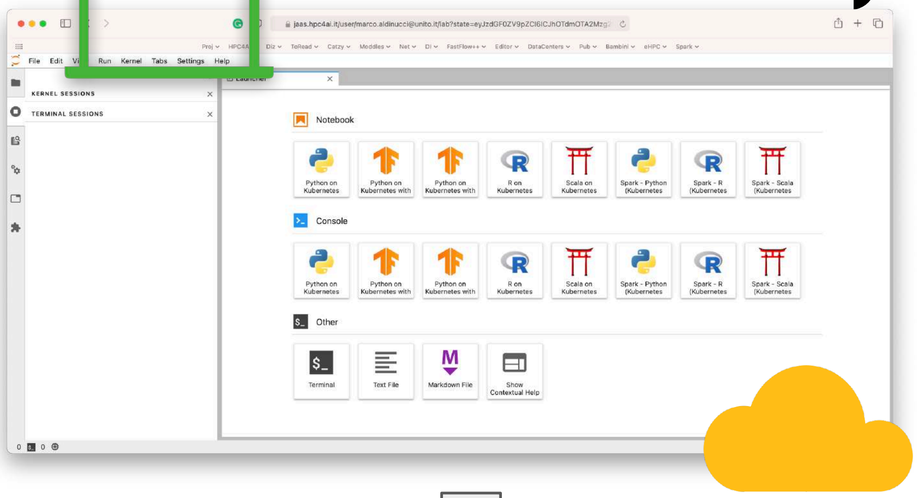
The best variant is the baseline for the future developments

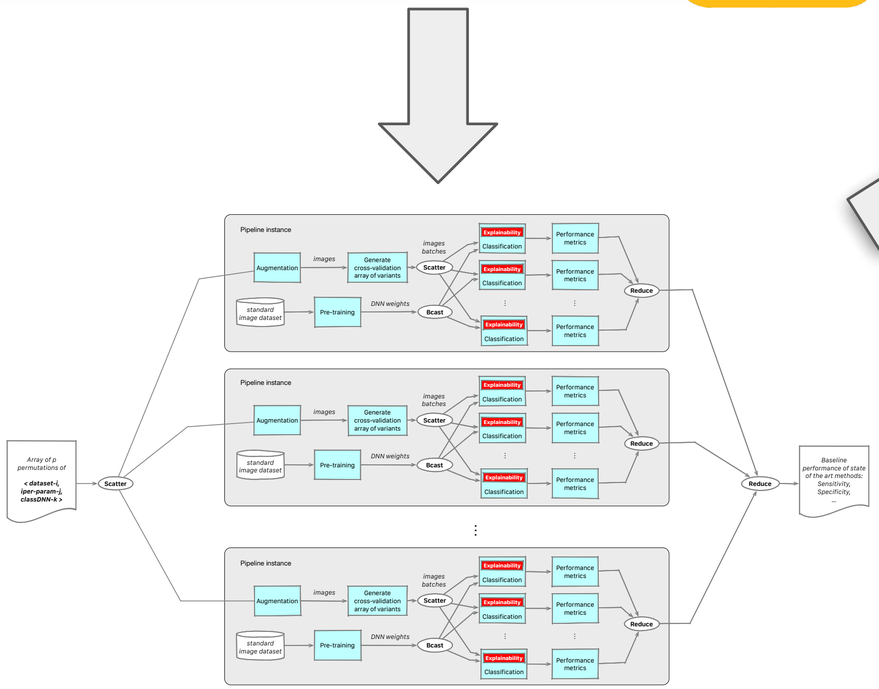
- **1 variant needs 52+ training hours on NVidia GPU V100**
  - 5 min avg per epoch, trimming training at 50 epochs, 5 cross-validations = 1250 min = 52 hours
  - 9 x 11 variants x 52 hours = 215 days for a single run

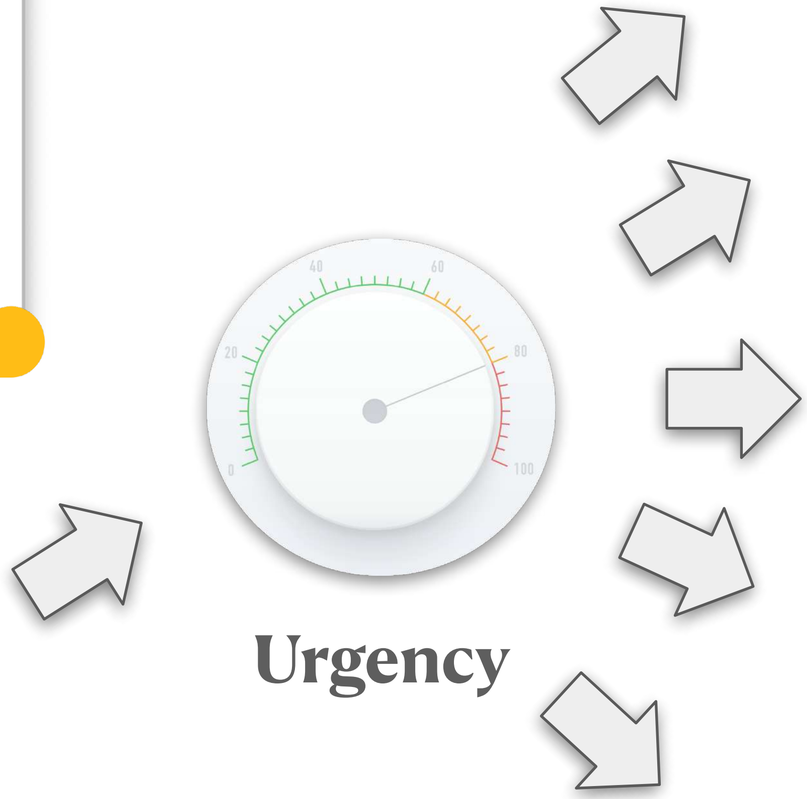
- **From the HPC standpoint the problem is "embarrassingly parallel"**
  - All variants can be run in any order. In principle all at the same time on 99 x 5 GPUs (in ~4 hours)
  - Or with *N* GPU: T= (495/N) x 4 hours. E.g., *N*=16 => *T* = ~5 days
  - Possible to trade-off urgency and scalability of the execution (number of used GPUs)
  - Fault-tolerance is crucial: in many days of compute everything that can happen will happen

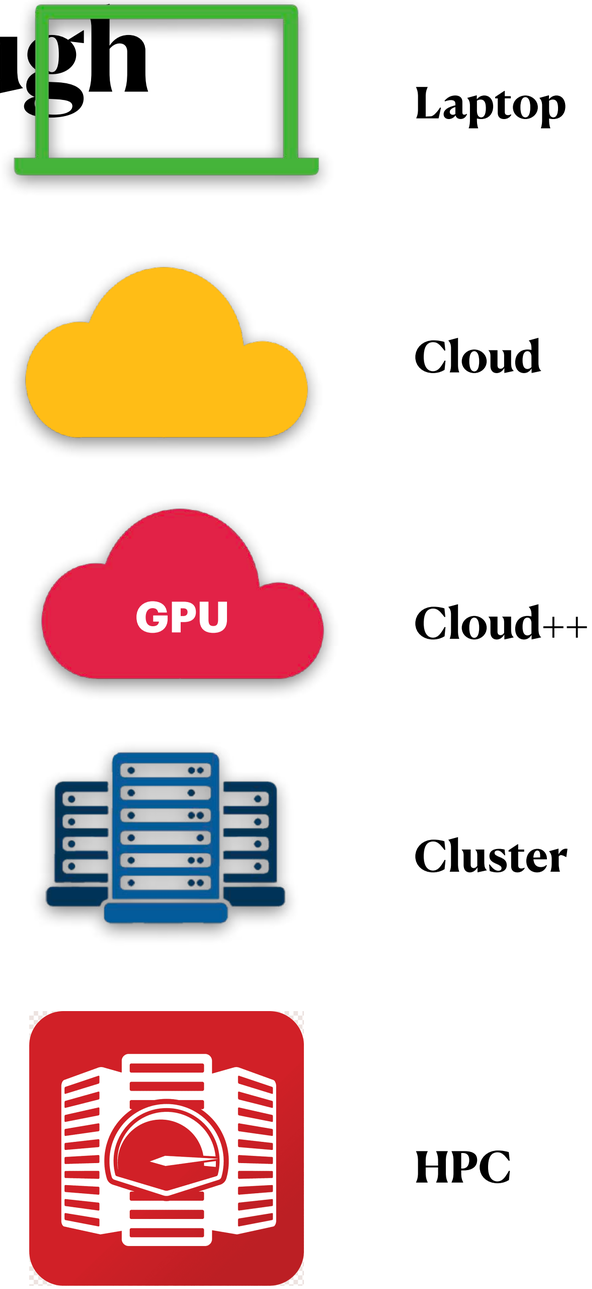
Elasticity is no longer enough

Portability of deployments









Laptop

Cloud

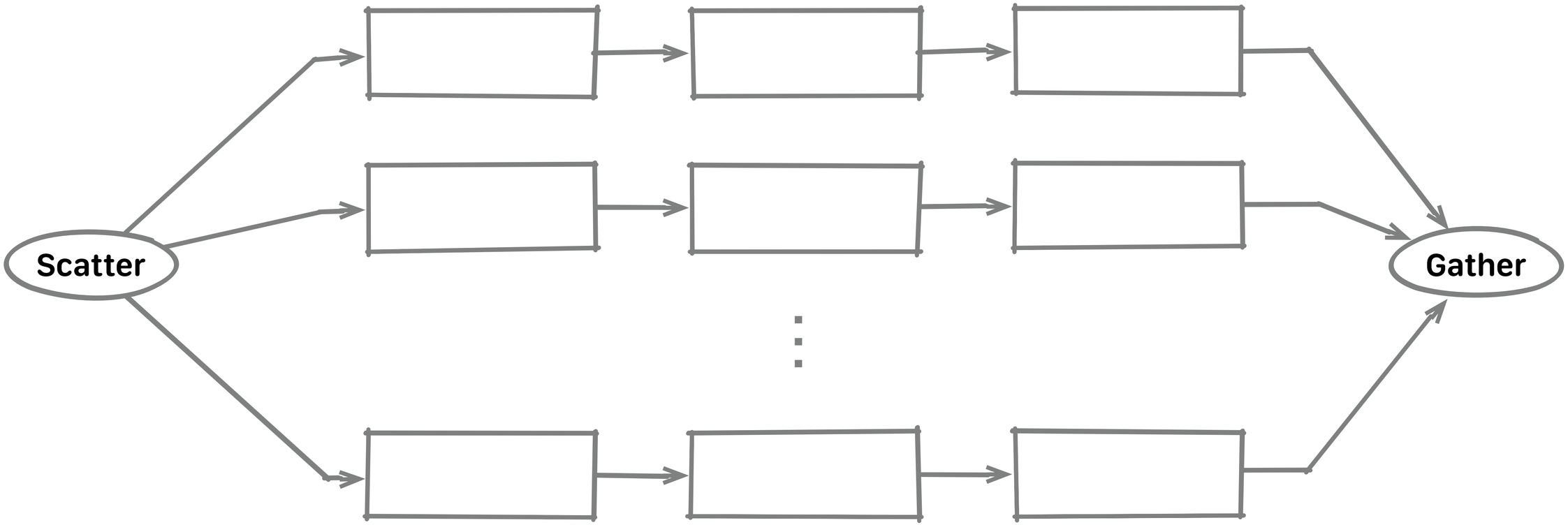
Cloud++

Cluster

HPC

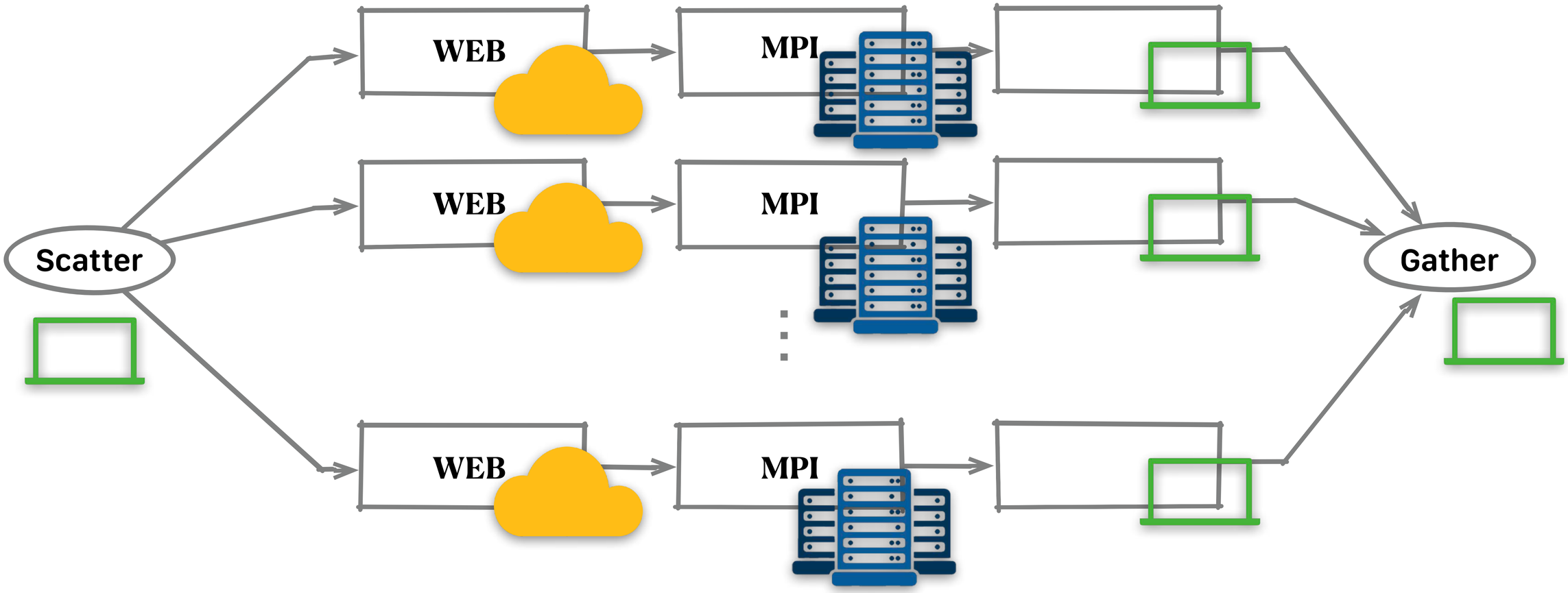
Portability

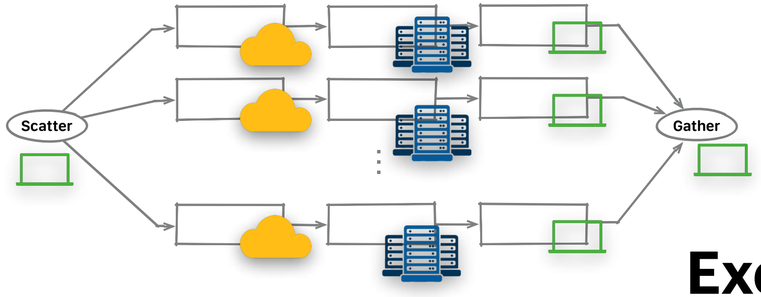
Example: functional mapping



Portability

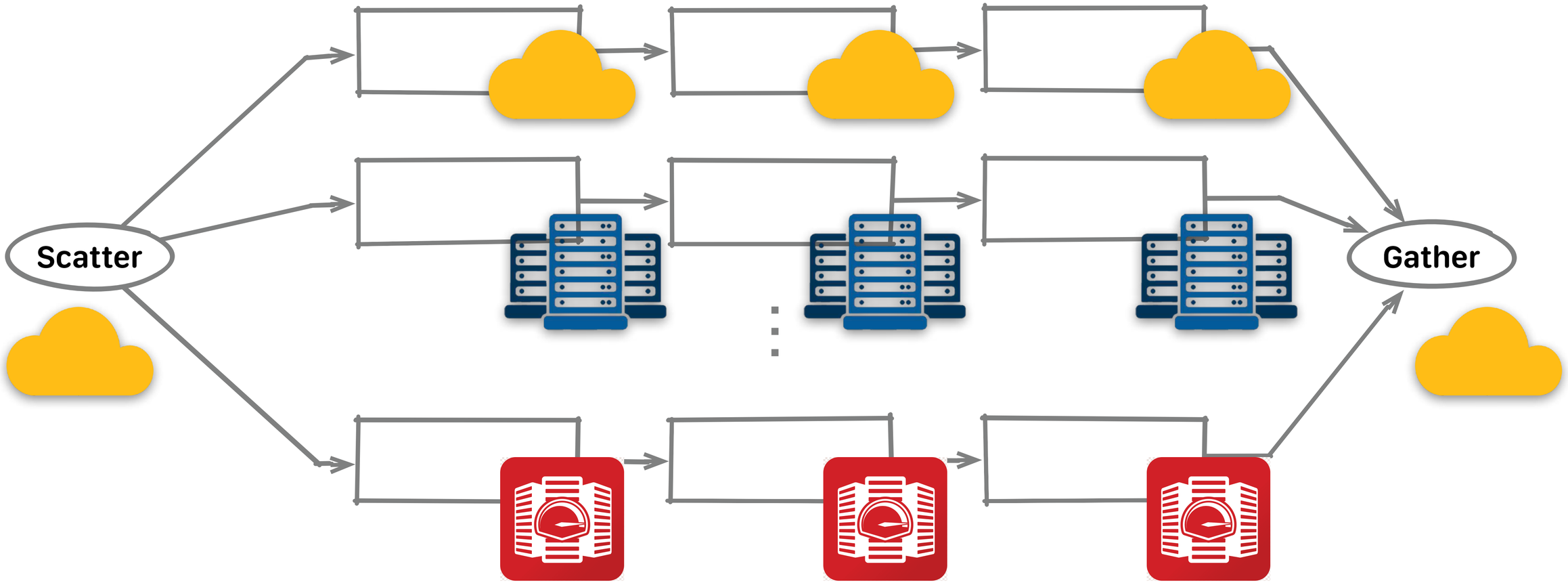
Example: functional mapping





Portability

Example: HPC boosting of cloud services



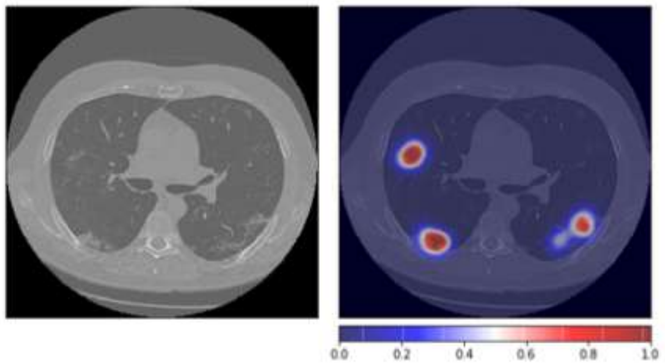
Results

M. Pennisi et al. An explainable AI system for automated COVID-19 assessment and lesion categorization from CT-scans. Artificial Intelligence in Medicine, page 102114, 2021. doi:10.1016/j.artmed.2021.102114

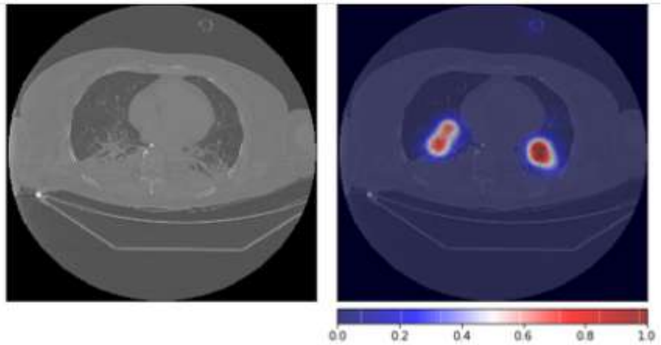
Model	Variant	Sensitivity	Specificity	Accuracy
AlexNet	–	74.2%	93.5%	83.9%
ResNet	18	83.9%	87.1%	85.5%
	34	83.9%	90.3%	87.1%
	50	83.9%	90.3%	87.1%
	101	74.2%	93.5%	83.9%
	152	70.9%	93.5%	82.2%
DenseNet	121	80.7%	90.3%	85.5%
	169	77.4%	90.3%	83.9%
	201	**90.3%**	**93.5%**	**91.9%**
SqueezeNet	–	77.4%	93.5%	85.5%
ResNeXt	–	80.6%	93.5%	87.1%

- Scalability almost ideal but ...
  - Small dataset 76GB (45k images from 124 patients after data preparation)
  - Independent parallel tasks only (no Distributed SGD)
    - Waiting time in the SLURM queue depends on machine load
    - Coarse grain is generally needed

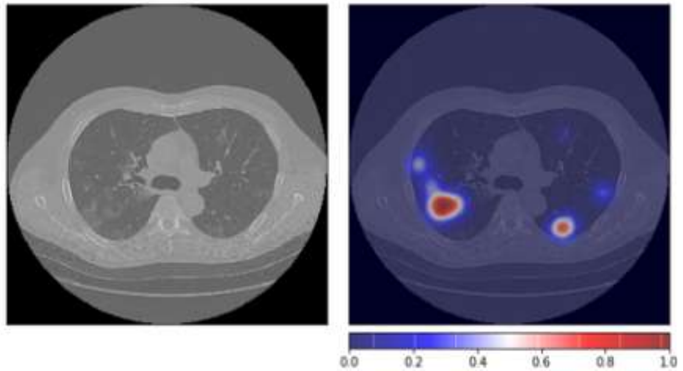
Crazy paving



Consolidation

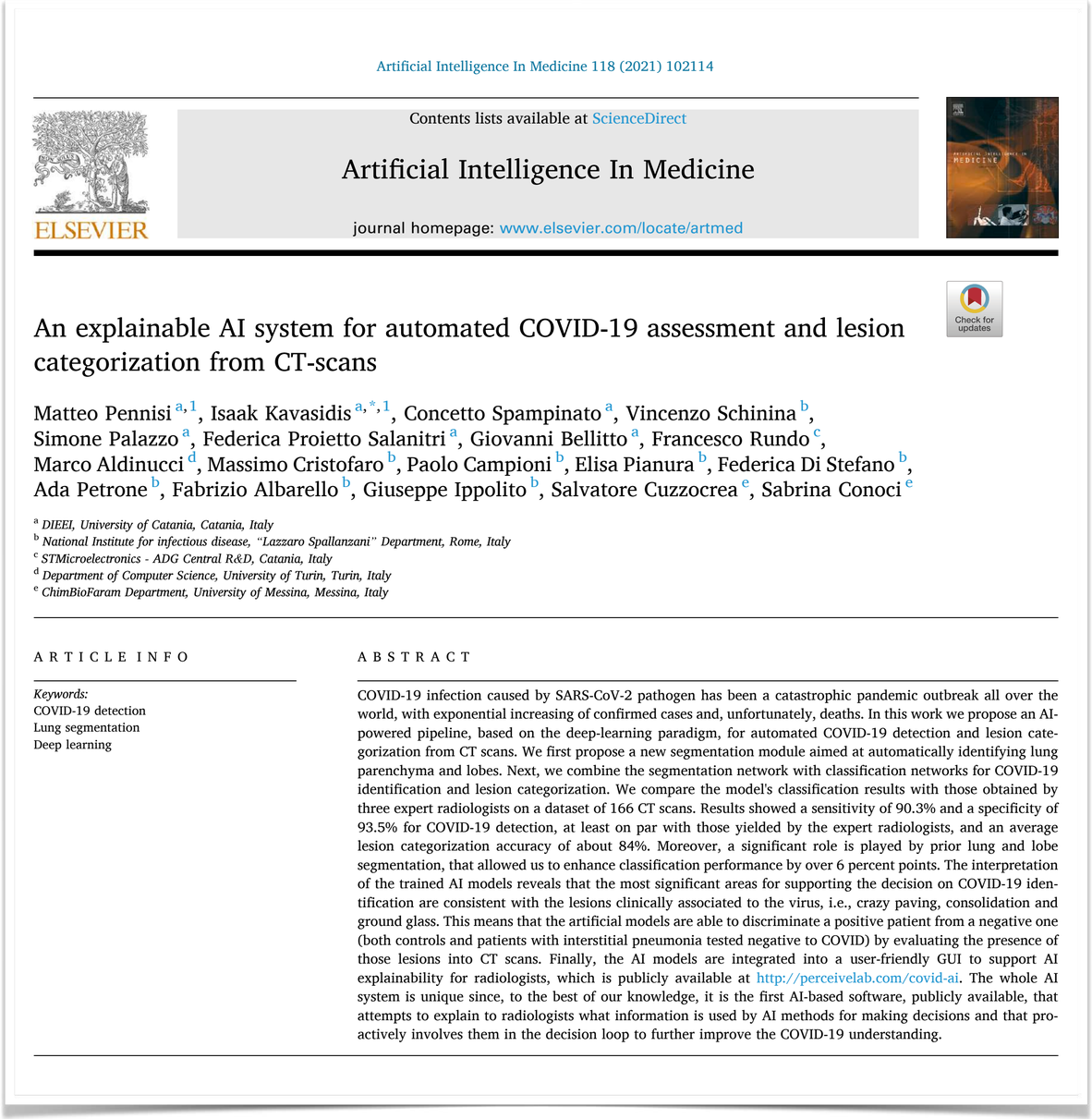


Ground glass



References

- M. Pennisi, I. Kavasidis, C. Spampinato, V. Schinina, S. Palazzo, F. P. Salanitri, G. Bellitto, F. Rundo, M. Aldinucci, M. Cristofaro, et al. **An explainable AI system for automated COVID-19 assessment and lesion categorization from CT-scans.** Artificial Intelligence in Medicine, page 102114, 2021. doi:10.1016/j.artmed.2021.102114
- I. Colonnelli, B. Cantalupo, C. Spampinato, M. Pennisi, and M. Aldinucci. **Bringing ai pipelines onto cloud-HPC: setting a baseline for accuracy of covid-19 diagnosis.** in Enea cresco in the fight against covid-19, 2021. doi:10.5281/zenodo.5151511
- M. Aldinucci. **High-performance computing and AI team up for COVID- 19 diagnostic imaging**. AIHUB, Jan. 2021.
- G. Bontempi, R. Chavarriaga, H. De Canck, E. Girardi, H. Hoos, I. Kilbane-Dawe, T. Ball, A. Nowé, J. Sousa, D. Bacciu, M. Aldinucci, M. De Domenico, A. Saffiotti, and M. Maratea. **The CLAIRE COVID- 19 initiative: approach, experiences and recommendations.** Ethics and Information Technology, Feb. 2021. doi:10.1007/s10676-020-09567-7
- I. Colonnelli, B. Cantalupo, R. Esposito, M. Pennisi, C. Spampinato, and M. Aldinucci. **HPC Application Cloudification: The StreamFlow Toolkit.** In proc of the 12th PARMA-DITAM workshop, volume 88 of OASIcs, pages 5:1– 5:13, Dagstuhl, Germany, 2021. doi:10.4230/OASIcs.PARMA-DITAM.2021.5



Artificial Intelligence In Medicine 118 (2021) 102114

Contents lists available at ScienceDirect

Artificial Intelligence In Medicine

journal homepage: www.elsevier.com/locate/artmed

An explainable AI system for automated COVID-19 assessment and lesion categorization from CT-scans

Matteo Pennisi [a,1], Isaak Kavasidis [a,*,1], Concetto Spampinato [a], Vincenzo Schinina [b], Simone Palazzo [a], Federica Proietto Salanitri [a], Giovanni Bellitto [a], Francesco Rundo [c], Marco Aldinucci [d], Massimo Cristofaro [b], Paolo Campioni [b], Elisa Pianura [b], Federica Di Stefano [b], Ada Petrone [b], Fabrizio Albarello [b], Giuseppe Ippolito [b], Salvatore Cuzzocrea [e], Sabrina Conoci [e]

[a] DIEEI, University of Catania, Catania, Italy
[b] National Institute for infectious disease, "Lazzaro Spallanzani" Department, Rome, Italy
[c] STMicroelectronics - ADG Central R&D, Catania, Italy
[d] Department of Computer Science, University of Turin, Turin, Italy
[e] ChimBioFaram Department, University of Messina, Messina, Italy

ARTICLE INFO

Keywords:
COVID-19 detection
Lung segmentation
Deep learning

ABSTRACT

COVID-19 infection caused by SARS-CoV-2 pathogen has been a catastrophic pandemic outbreak all over the world, with exponential increasing of confirmed cases and, unfortunately, deaths. In this work we propose an AI-powered pipeline, based on the deep-learning paradigm, for automated COVID-19 detection and lesion categorization from CT scans. We first propose a new segmentation module aimed at automatically identifying lung parenchyma and lobes. Next, we combine the segmentation network with classification networks for COVID-19 identification and lesion categorization. We compare the model's classification results with those obtained by three expert radiologists on a dataset of 166 CT scans. Results showed a sensitivity of 90.3% and a specificity of 93.5% for COVID-19 detection, at least on par with those yielded by the expert radiologists, and an average lesion categorization accuracy of about 84%. Moreover, a significant role is played by prior lung and lobe segmentation, that allowed us to enhance classification performance by over 6 percent points. The interpretation of the trained AI models reveals that the most significant areas for supporting the decision on COVID-19 identification are consistent with the lesions clinically associated to the virus, i.e., crazy paving, consolidation and ground glass. This means that the artificial models are able to discriminate a positive patient from a negative one (both controls and patients with interstitial pneumonia tested negative to COVID) by evaluating the presence of those lesions into CT scans. Finally, the AI models are integrated into a user-friendly GUI to support AI explainability for radiologists, which is publicly available at http://perceivelab.com/covid-ai. The whole AI system is unique since, to the best of our knowledge, it is the first AI-based software, publicly available, that attempts to explain to radiologists what information is used by AI methods for making decisions and that proactively involves them in the decision loop to further improve the COVID-19 understanding.

www.commonwl.org

COMMON WORKFLOW LANGUAGE

Getting Started ▾ Implementations ▾ Community Projects ▾

Common Workflow Language

gitter join chat | stars 1.3k | CWL Community Support | Donate

Getting Started
Vision for the CWL Project
Support, Community and Contributing
Specification
Implementations
Repositories of CWL Tools and Workflows
Software for working with CWL
Projects the CWL community is participating in
Contributors and Governance

The Common Workflow Language (CWL) is an open standard for describing analysis workflows and tools in a way that makes them portable and scalable across a variety of software and hardware environments, from workstations to cluster, cloud, and high performance computing (HPC) environments. CWL is designed to meet the needs of data-intensive science, such as Bioinformatics, Medical Imaging, Astronomy, High Energy Physics, and Machine Learning.

CWL is developed by a multi-vendor working group consisting of organizations and individuals aiming to enable scientists to share data analysis workflows. The CWL project is maintained on Github and we follow the Open-Stand.org principles for collaborative open standards development. Legally CWL is a member project of Software Freedom Conservancy and is formally managed by the elected CWL leadership team, however every-day project decisions are made by the CWL community which is open for participation by anyone.

open stand — the modern paradigm for standards

The Streamflow WMS

A cloud-HPC implementation of the CWL open standard

https://streamflow.di.unito.it

CWL 2021 Mini-Conference

The CWL 2021 Mini-Conference was on February 8-10th. All videos are accessible online via YouTube and ConfTube.

Vision for the CWL Project

What this community hopes to have accomplished in the future

Researchers, scientists, and analysts share their batch data analysis workflows without technical barriers using an open standard. Sharing workflows this way is a usual occurrence and seen as a typical way of working. The workflows are complete and run on a variety of environments; and people re-use shared workflow descriptions and build new workflows from them. No vendor dominates the ecosystem

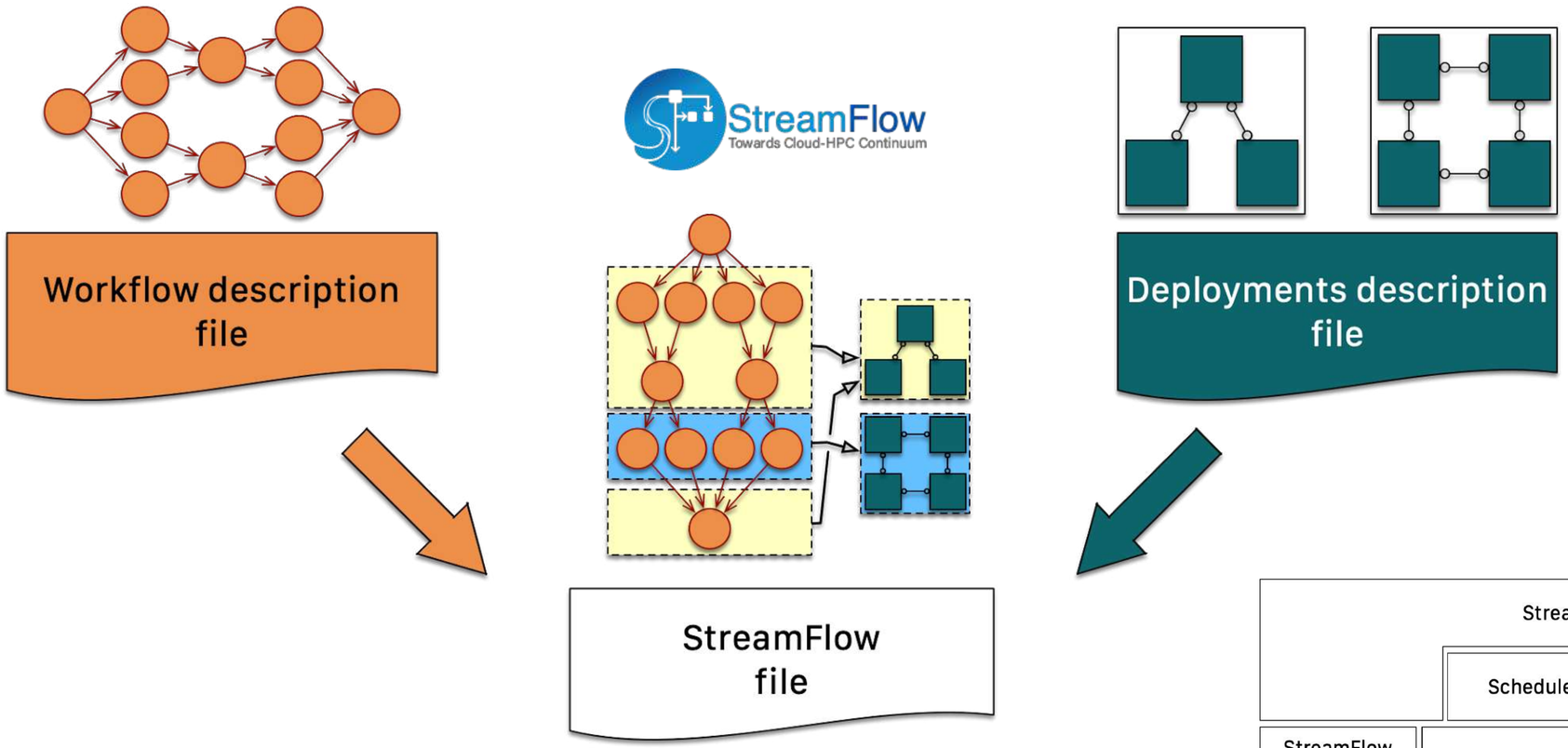
Mission of the CWL Project

StreamFlow WMS

I. Colonnelli, B. Cantalupo, I. Merelli, and M. Aldinucci, "Streamflow: cross-breeding cloud with HPC," IEEE Transactions on Emerging Topics in Computing, 2020. doi:10.1109/TETC.2020.3019202

Es. K8S@ HPC4AI

Es. SLURM@ Marconi100



StreamFlow executor			
	Scheduler	Data manager	Deployment manager
StreamFlow extensions	Connector		
CWL interpreter	Docker compose	Helm	SLURM

https://streamflow.di.unito.it

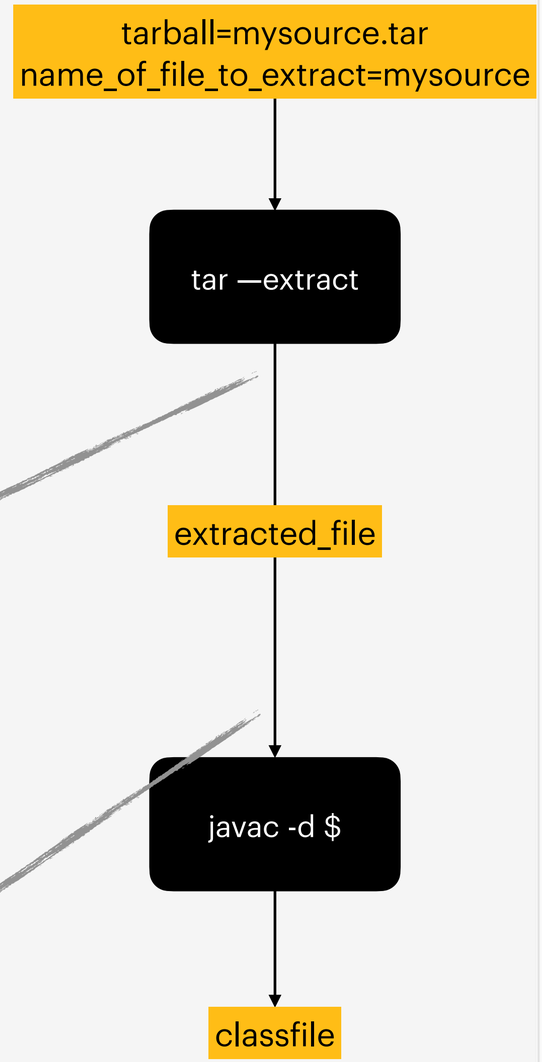
Streamflow

A CWL open standard implementation

- Portable & Interoperable
- Data movement automatic
  - Don't need a shared FS
- Deployment are plug-ins
  - k8s (w docker), SLURM (singularity or modules), ssh, air-gapped platforms, ...
- Task & data parallel
    - Tasks, streams (*), QuantumC (*)
    - Scatter, Gather, Broadcast

(*) Under development

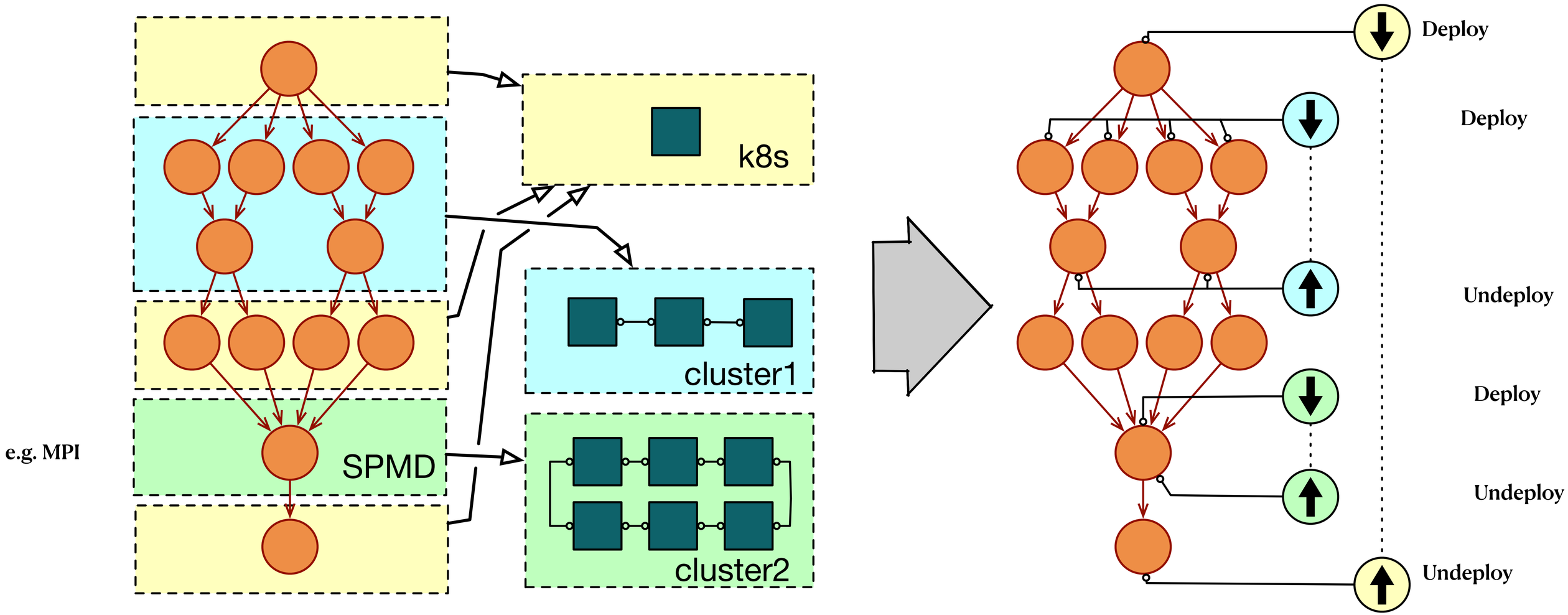
```yaml
cwlVersion: v1.2
class: Workflow
inputs:
  tarball: File
  name_of_file_to_extract: string

outputs:
  compiled_class:
    type: File
    outputSource: compile/classfile

steps:
  untar:
    run:
      class: CommandLineTool
      baseCommand: [tar, --extract]
      inputs:
        tarfile:
          type: File
          inputBinding:
           prefix: --file
      outputs:
        example_out:
          type: File
          outputBinding:
            glob: hello.txt
    in:
      tarfile: tarball
      extractfile: name_of_file_to_extract
    out: [extracted_file]

  compile:
    run:
      class: CommandLineTool
      baseCommand: javac
      arguments: ["-d", $(runtime.outdir)]
      inputs:
        src:
          type: File
          inputBinding:
            position: 1
      outputs:
        classfile:
          type: File
          outputBinding:
            glob: "*.class"
    in:
      src: untar/extracted_file
    out: [classfile]
```



Example

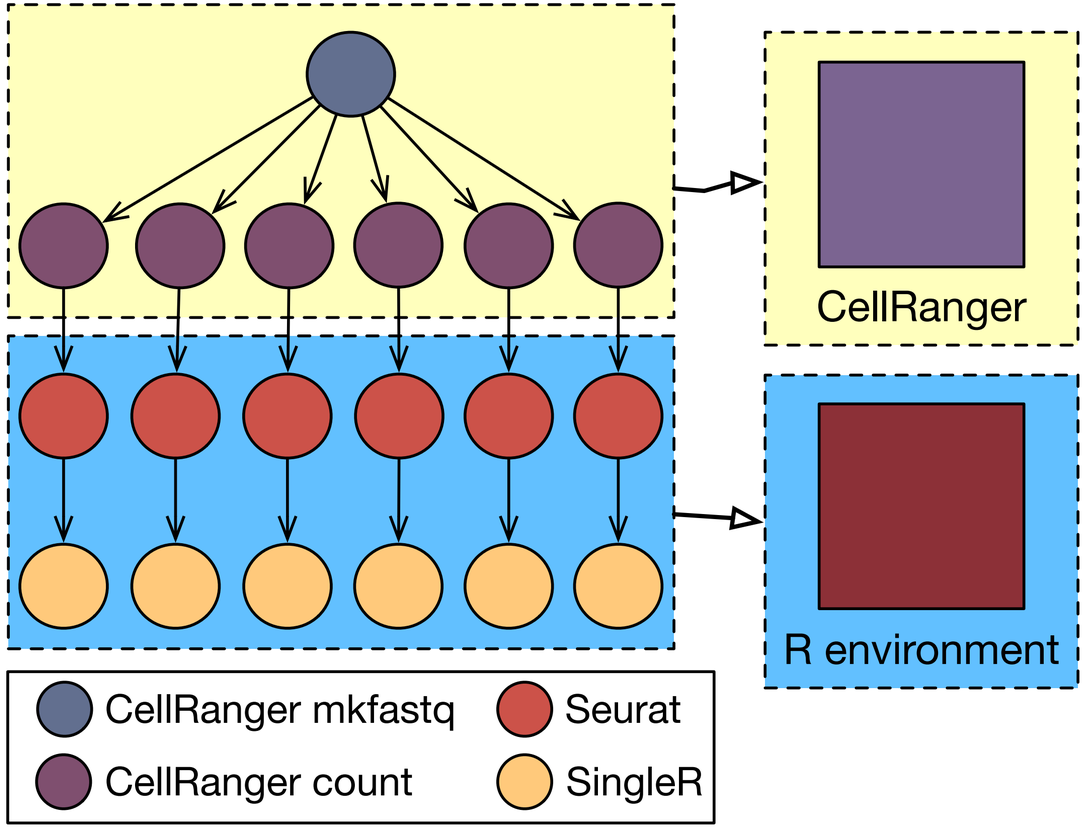
Cloud/docker/k8s + HPC/MPI/SLURM

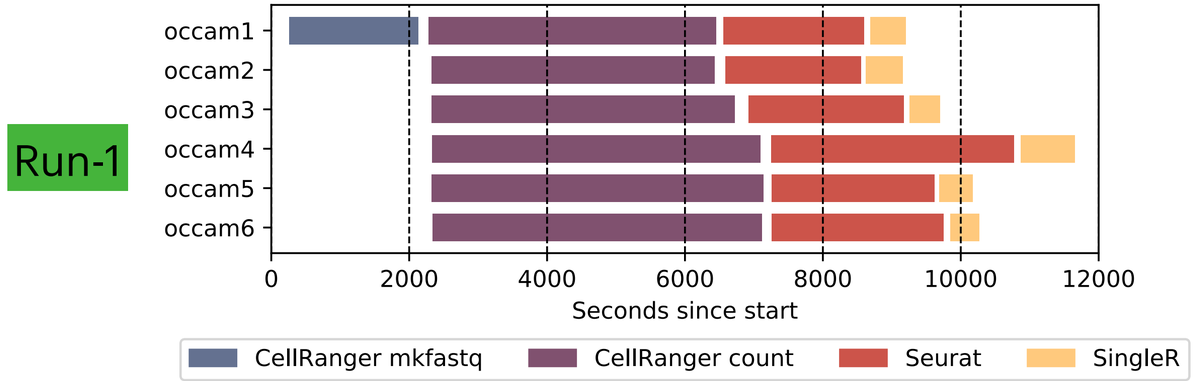
Workflow + Deployment models = StreamFlow

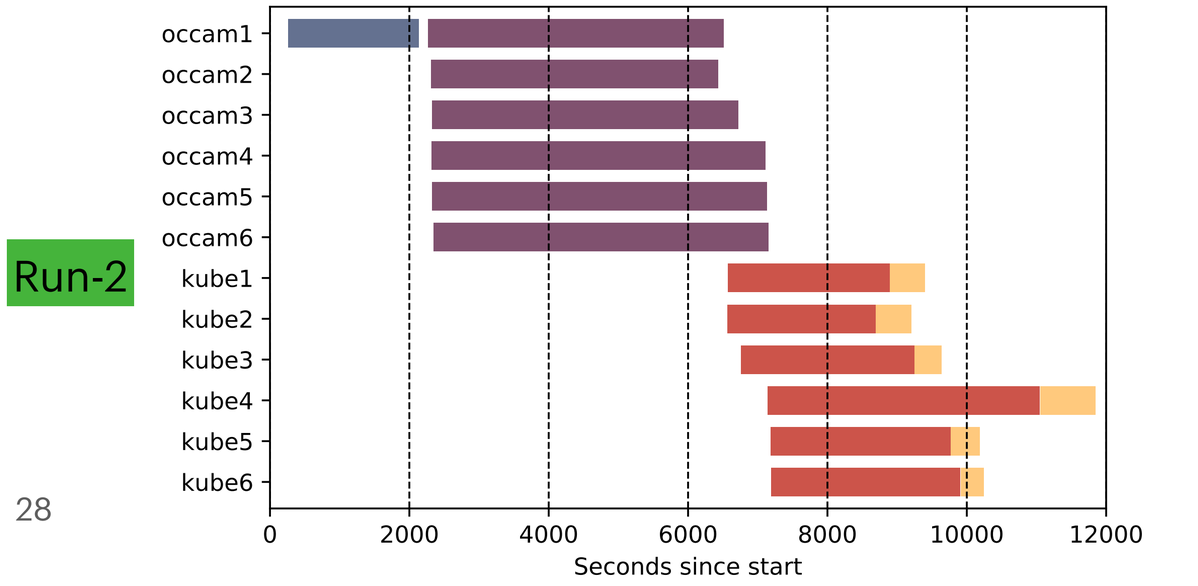


Single-cell alignment pipeline

I. Colonnelli, B. Cantalupo, I. Merelli, and M. Aldinucci, "Streamflow: cross-breeding cloud with HPC," IEEE Transactions on Emerging Topics in Computing, 2020. doi:10.1109/TETC.2020.3019202







Jupyter-as-a-Service

R

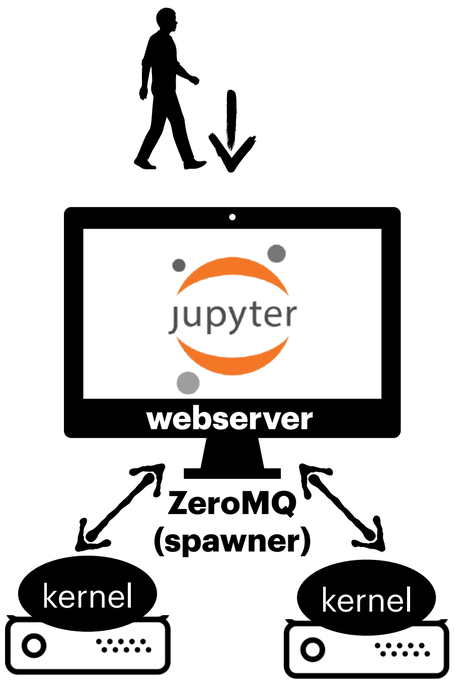
JupyterHub
enterprise edition

https://jaas.hpc4ai.unito.it

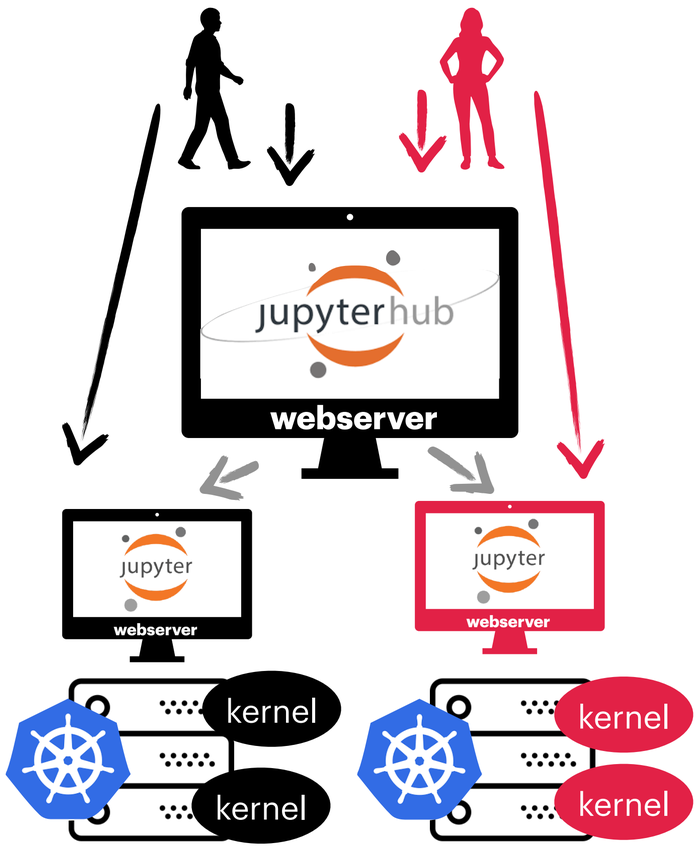
Julia

Python

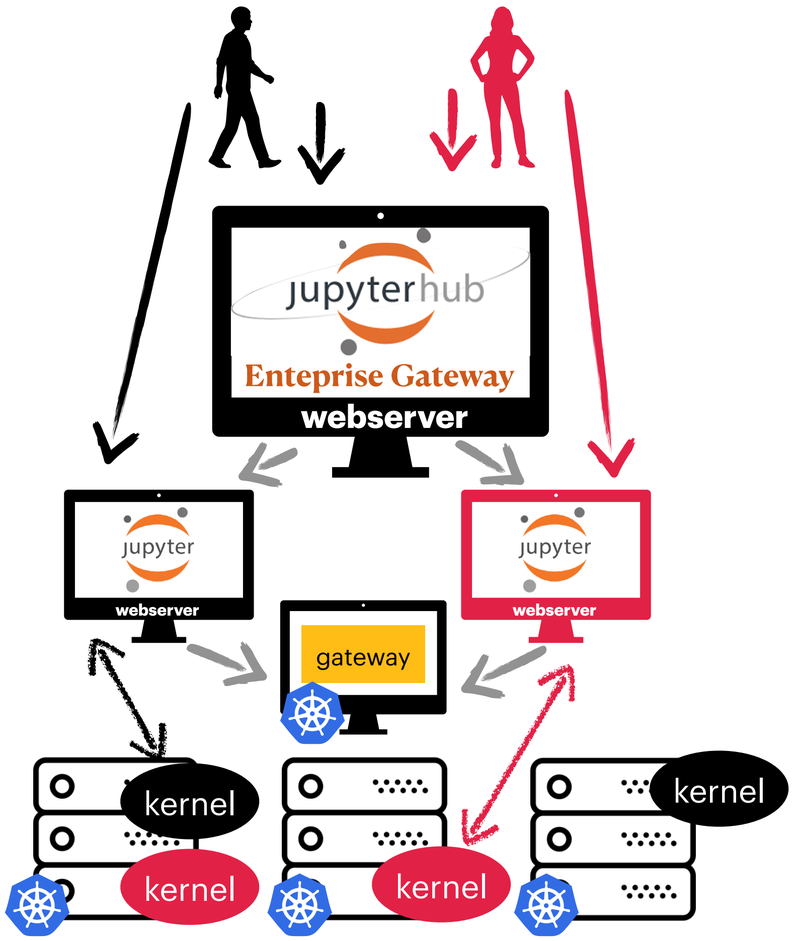
Notebooks: from personal to multi-tenant to managed



Single user
Single node (*)



Multi-tenant
Single node per user (*)



Multi-tenant
Multiple nodes per user

jaas.hpc4ai.unito.it/user/marco.aldinucci@unito.it/lab/tree/claire-covid.ipynb

Untitled1.ipynb - JupyterLab | JupyterLab

File Edit View Run Kernel Tabs Settings Help

Filter files by name

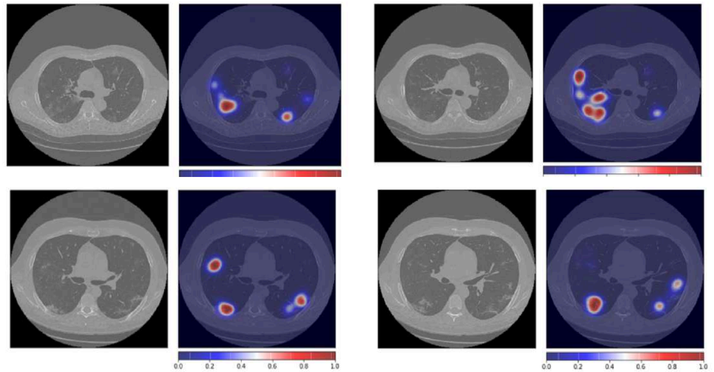
/

Name	Last Modified
lost+found	19 hours ago
claire-covi...	8 minutes ago

claire-covid.ipynb

Code | Python on Kubernetes

```
    train_loader = torch.utils.data.DataLoader(dataset1,**train_kwargs)
    test_loader = torch.utils.data.DataLoader(dataset2, **test_kwargs)

    model = Net().to(device)
    optimizer = optim.Adadelta(model.parameters(), lr=args.lr)

    scheduler = StepLR(optimizer, step_size=1, gamma=args.gamma)
    for epoch in range(1, args.epochs + 1):
        train(args, model, device, train_loader, optimizer, epoch)
        test(model, device, test_loader)
        scheduler.step()

    if args.save_model:
        torch.save(model.state_dict(), "mnist_cnn.pt")


if __name__ == '__main__':
    main()
```

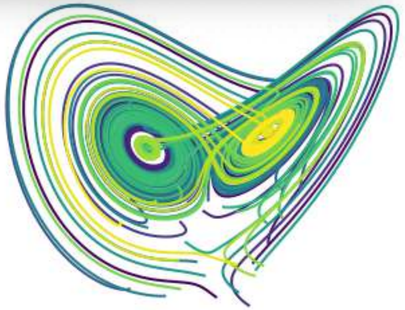


Simple 0 1 Python on Kubernetes | Idle Mode: Command Ln 137, Col 11 claire-covid.ipynb

Python 3

trajectories.

o=(0.0,50.0))



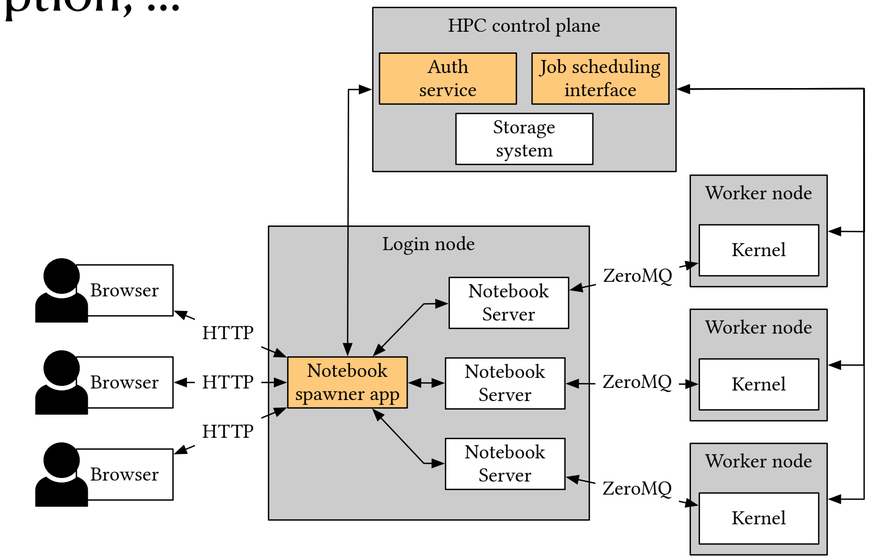
lorenz.py

```
def solve_lorenz(sigma=10.0, beta=8./3, rho=28.0):
    """Plot a solution to the Lorenz differential equations."""

    max_time = 4.0
    N = 30

    fig = plt.figure()
    ax = fig.add_axes([0, 0, 1, 1], projection='3d')
    ax.axis('off')

    # prepare the axes limits
    ax.set_xlim((-25, 25))
    ax.set_ylim((-35, 35))
    ax.set_zlim((5, 55))

    def lorenz_deriv(x_y_z, t0, sigma=sigma, beta=beta, rho=rho):
        """Compute the time-derivative of a Lorenz system."""
        x, y, z = x_y_z
        return [sigma * (y - x), x * (rho - z) - y, x * y - beta * z]

    # Choose random starting points, uniformly distributed from -15 to 15
    np.random.seed(1)
    x0 = -15 + 30 * np.random.random((N, 3))

    # Solve for the trajectories
    t = np.linspace(0, max_time, int(250*max_time))
    x_t = np.asarray([integrate.odeint(lorenz_deriv, x0i, t)
                      for x0i in x0])

    # choose a different color for each trajectory
    colors = plt.cm.viridis(np.linspace(0, 1, N))

    for i in range(N):
        x, y, z = x_t[i,:,:].T
        lines = ax.plot(x, y, z, '-', c=colors[i])
        plt.setp(lines, linewidth=2)
    angle = 104
    ax.view_init(30, angle)
```

The “common” JaaS implementation

Distributed - not yet HPC

- **Kernels are still long-term cloud services - not HPC jobs**
  - Kernels’ life cycle not managed - No suspend-resume, preemption, ...
  - IPython not serialisable
  - Kernels talk TCP/IP (ZeroMQ) with the Interface
  - Difficult integration with Batch workload managers
- **Kernels segmented on user-basis not on SLA**
  - SLA is the key of cloud-HPC integration
  - How ask: run this kernel on a node with a GPU
- **A single notebook is not distributed**
  - It runs a single node



HPC4AI

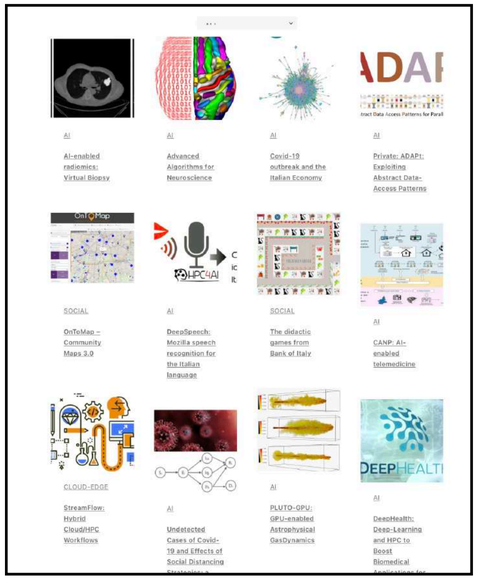
Two systems “**DevOps**” (Production+Dev) each made by two clusters
1 Cloud/OpenStack + 1 HPC/SLURM

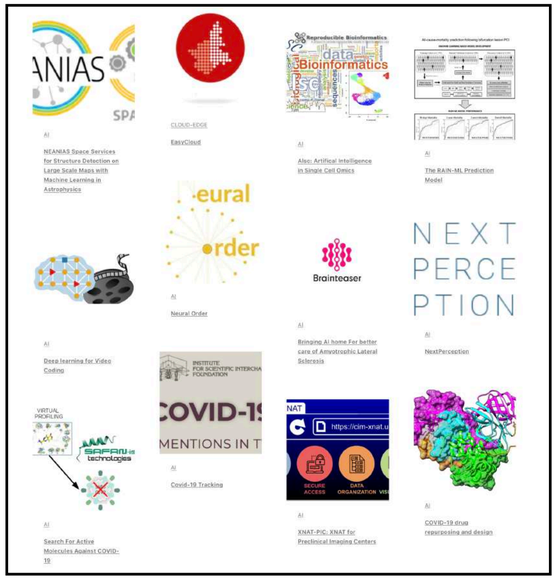
Overall **5k+ cores** (Intel, ADM), **100 GPUs** (T4/V100), 4 novel **Arm+A100+BlueField** Nvidia-DEV platforms
3PB in 5 systems: CEPH (SSD/rep3, Rot/ErasureCode), EMC2 (Unity+Avamar)

40+ research projects running (HPC, AI, ...), mostly with critical data (medical)
Several in-house PaaS: SLURM-calendar integration, Kubernetes-as-a-Service, **Jupyter-as-a-Service**, ...

Our “own” **250KVA Tier-3** datacenter - 1.04 P.PUE design (**PUE 1.1/year** measured)
Production system: 21 months with 0 “system-down” incidents

We develop it and manage it! Our “products” is knowledge and skilled students





jaas.hpc4ai.unito.it/user/marco.aldinucci@unito.it/lab

COVID-Sic Proj Diz HPC4AI Catzy ToRead Moddles Net login DI eHPC FastFlow++ Editor DataCenters Pub Bambini Spark AI

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/

Name	Last Modified
lost+found	2 hours ago

Launcher

Notebook

Python on Kubernetes | Python on Kubernetes with | Python on Kubernetes with | R on Kubernetes | Scala on Kubernetes | Spark - Python (Kubernetes | Spark - R (Kubernetes | Spark - Scala (Kubernetes

Console

Python on Kubernetes | Python on Kubernetes with | Python on Kubernetes with | R on Kubernetes | Scala on Kubernetes | Spark - Python (Kubernetes | Spark - R (Kubernetes | Spark - Scala (Kubernetes

Other

Terminal | Text File | Markdown File | Python File | R File | Show Contextual Help

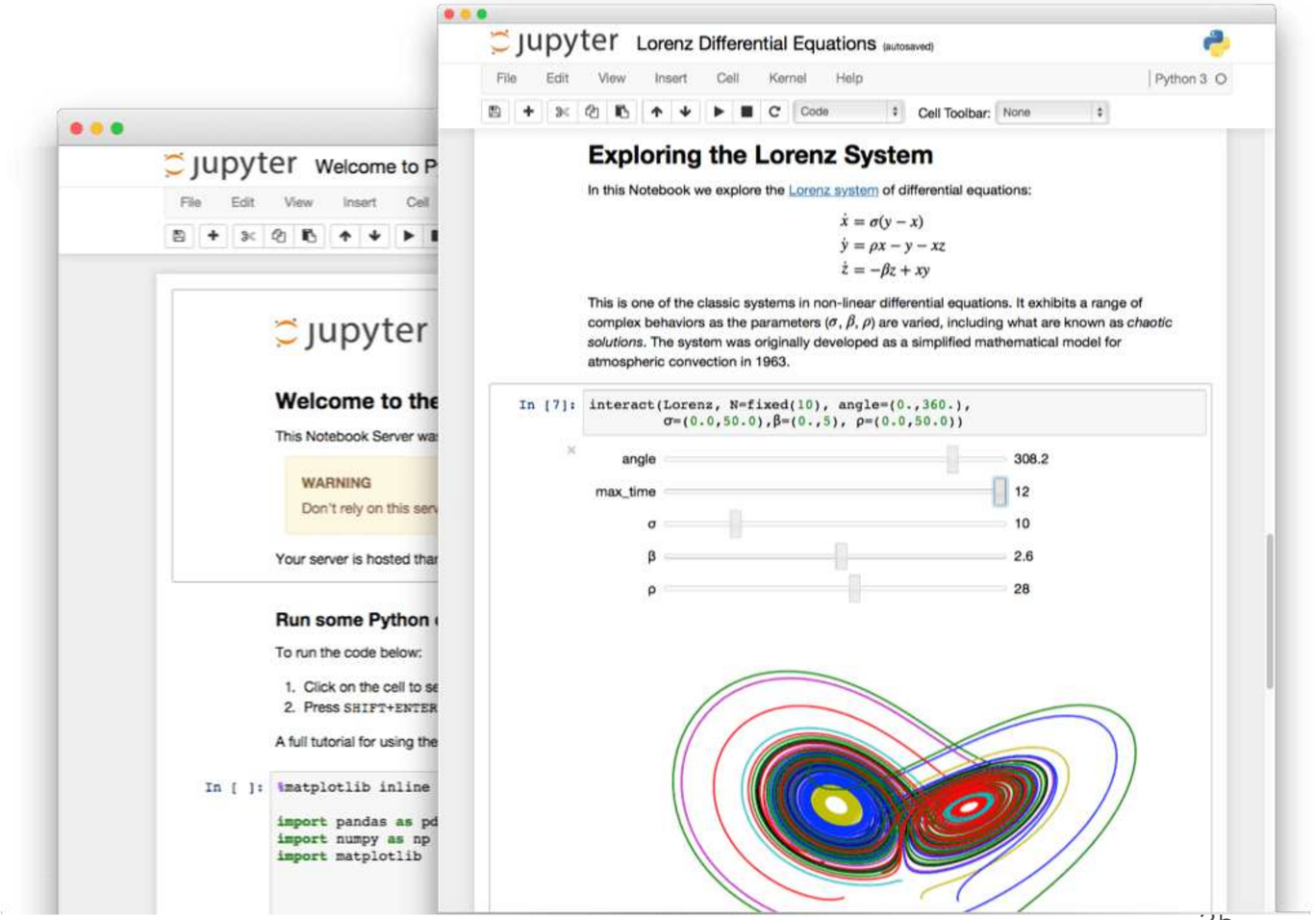
HPC4AI@UNITO

https://jaas.hpc4ai.unito.it

Simple 0 0 Launcher

Jupyter-workflow

https://github.com/alpha-unito/jupyter-workflow



Listing 1: Jupyter Notebook document cell format (general and code)

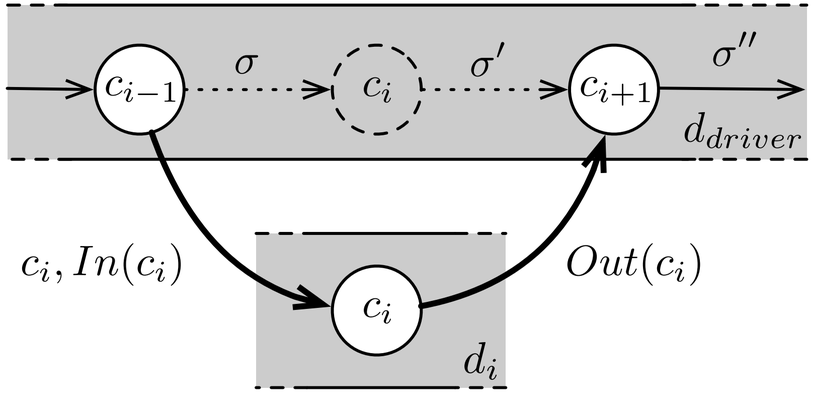
```
# Generic cell metadata
{
    "cell_type": "type",
    "metadata": {},
    "source": "single string or [list, of, strings]",
}

# Code cell metadata
{
    "cell_type": "code",
    "execution_count": 1,# Integer or null
    "metadata": {
    "collapsed": True, # Whether the output of the cell is collapsed
    "scrolled": False, # Any of true, false or "auto"
    },
    "source": "[some multi-line code]",
    "outputs": [{
    # List of output dicts (described below)
    "output_type": "stream",
    ...
    }],
}
```

A concurrent semantics for Jupyter

For both interactive and bulk execution

Sequential-**local** Semantics (default)

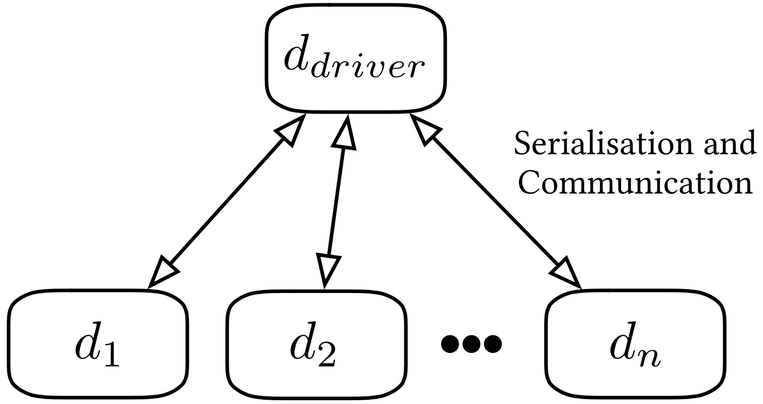
$$\langle c_1; \ldots; c_n, \sigma_1 \rangle \rightarrow \sigma_{n+1}$$
$$\text{if } \langle c_i, \sigma_i \rangle \rightarrow \sigma_{i+1} \text{ for every } 1 \leq i \leq n$$



Sequential-**global** Semantics (jupyter-workflow)

$$\langle [c_1, d_1]; \ldots; [c_n, d_n], \sigma_1 \rangle \rightarrow \sigma_{n+1}$$
$$\text{if } \langle [c_i, d_i], \sigma_i \rangle \rightarrow \sigma_{i+1} \text{ for every } 1 \leq i \leq n$$

Default Notebook's kernel (driver)



Kernels deployed on other Processing Elements or other processes of the driver

Bernstein's conditions relaxed

There are always clashes in the output the Notebook output

- $Out(c_i) \cap In(c_j) \neq \emptyset$. In this case we have a *true data dependency*, whereby c_i modifies an identifier read by c_j;
- ~~$Out(c_i) \cap Out(c_j) \neq \emptyset$. In this case we have an *output dependency*, whereby c_i and c_j modify the same identifier;~~
- $In(c_i) \cap Out(c_j) \neq \emptyset$. This is a so-called *anti-dependency*, whereby c_j modifies an identifier read by c_i.

In all other cases, namely when

$$Out(c_i) \cap In(c_j) = Out(c_i) \cap Out(c_j) = In(c_i) \cap Out(c_j) = \emptyset$$

A concurrent semantics for Jupyter

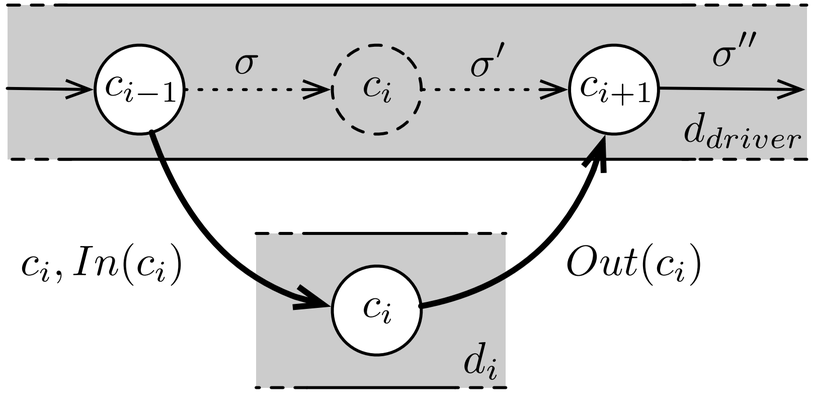
Task parallel parallelisation of bulk execution - driver outside the

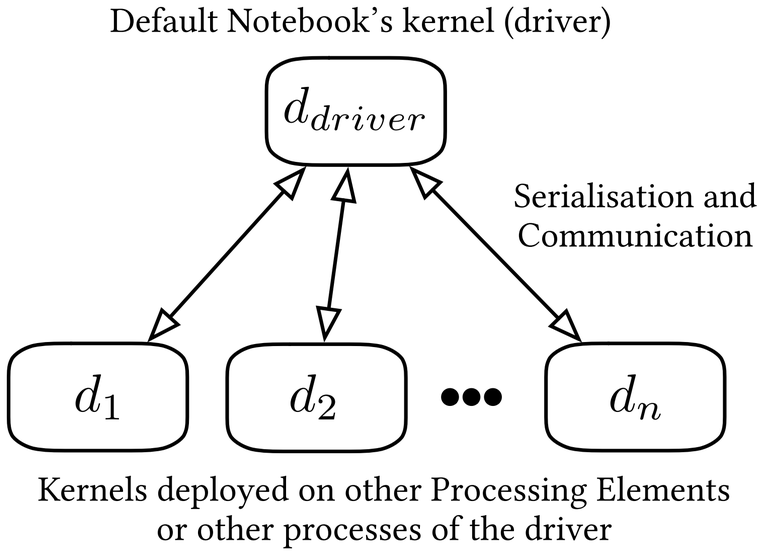
Global-**parallel** Semantics (jupyter-workflow)

$$c_1; (c_2 \mid (c_3; c_4))$$

$$\langle [c_1, d_1] \mid \cdots \mid [c_n, d_n], \sigma \rangle \rightarrow \uplus_{1 \leq i \leq n} \sigma'_i$$
$$\text{if } \langle [c_i, d_i], \sigma_i \rangle \rightarrow \sigma'_i \text{ for every } 1 \leq i \leq n$$

$$(\sigma \uplus \sigma')(x) = \begin{cases} \sigma(x) & \text{if } x \in \mathrm{dom}(\sigma) \setminus \mathrm{dom}(\sigma') \\ \sigma'(x) & \text{if } x \in \mathrm{dom}(\sigma') \setminus \mathrm{dom}(\sigma) \\ \sigma(x) \uplus \sigma'(x) & \text{if } x \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\sigma') \end{cases}$$





A concurrent semantics for Jupyter

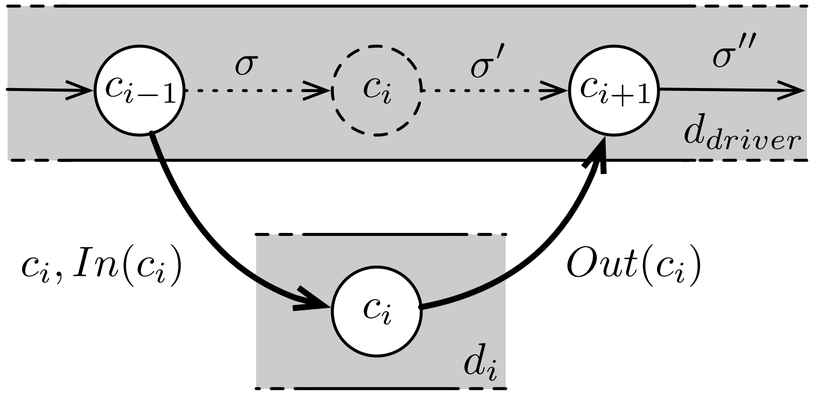
Task parallel parallelisation of bulk execution - driver outside the data path

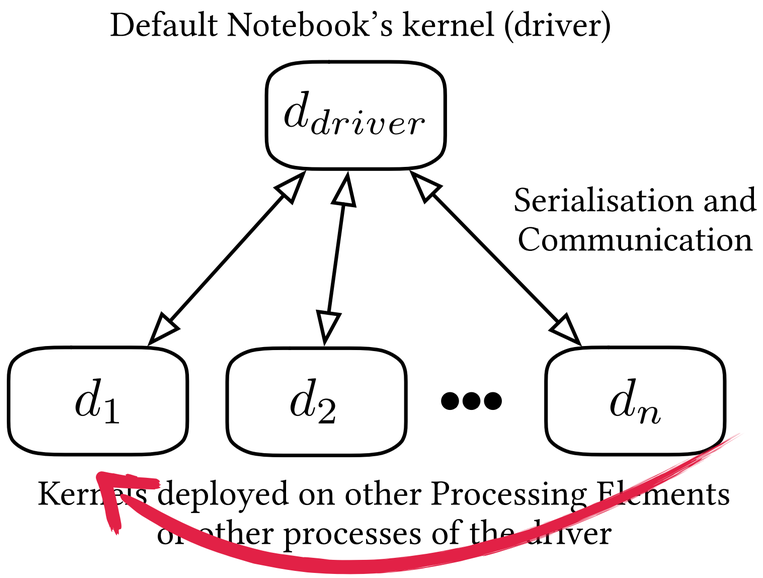
Global-**parallel** Semantics (jupyter-workflow)

$$c_1; (c_2 \mid (c_3; c_4))$$

$$\langle [c_1, d_1] \mid \cdots \mid [c_n, d_n], \sigma \rangle \rightarrow \uplus_{1 \leq i \leq n} \sigma'_i$$
$$\text{if } \langle [c_i, d_i], \sigma_i \rangle \rightarrow \sigma'_i \text{ for every } 1 \leq i \leq n$$

$$(\sigma \uplus \sigma')(x) = \begin{cases} \sigma(x) & \text{if } x \in \mathrm{dom}(\sigma) \setminus \mathrm{dom}(\sigma') \\ \sigma'(x) & \text{if } x \in \mathrm{dom}(\sigma') \setminus \mathrm{dom}(\sigma) \\ \sigma(x) \uplus \sigma'(x) & \text{if } x \in \mathrm{dom}(\sigma) \cap \mathrm{dom}(\sigma') \end{cases}$$





A concurrent semantics for Jupyter

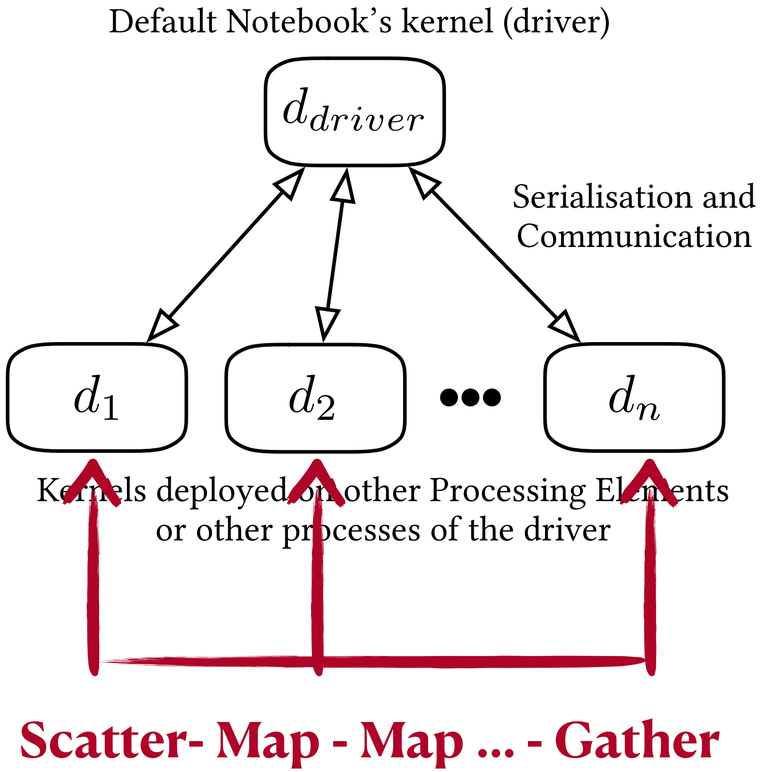
Data parallel explicit parallelisation with traditional optimisations

Explicit data-parallelism (to be declared in the cell meta-data)

$$\langle [Map(c), d], \sigma \rangle \rightarrow Gather(\sigma'_1, \ldots, \sigma'_n)$$
$$\text{if } \langle [c_1, d_i], \sigma_i \rangle \rightarrow \sigma'_i \text{ for every } 1 \leq i \leq n$$
$$\text{where } Scatter(\sigma) = [\sigma_1, \ldots, \sigma_n]$$

Map lifting

$$\langle Map(c_1); Map(c_2), \sigma \rangle = \langle Map(c_1; c_2), \sigma \rangle$$



Densenet 240 training on HPC4AI+Marconi100

localhost+SLURM - (with automatic serialization between x86 and P9/V100)

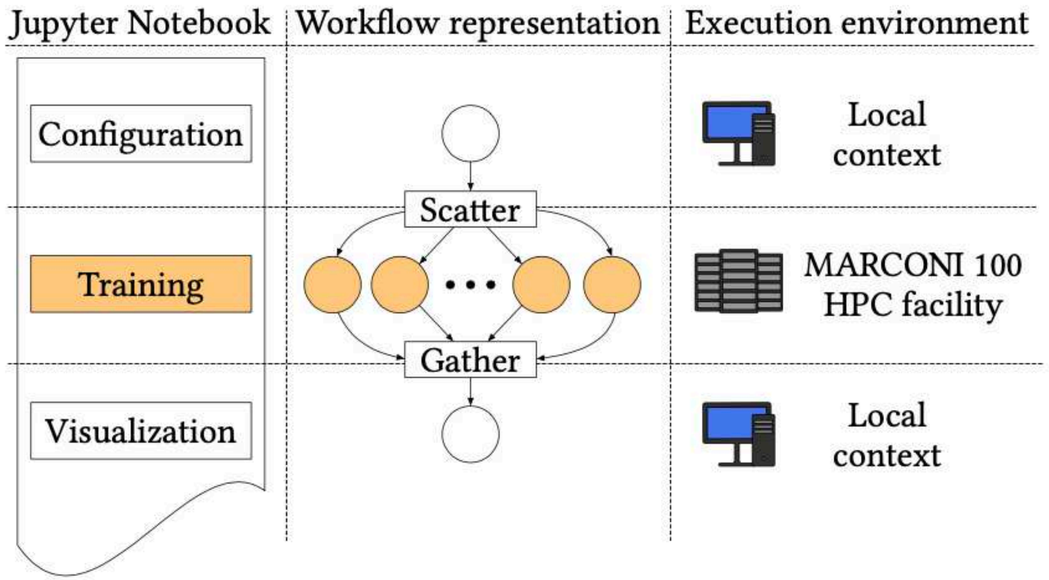


Figure 4: Graphical representation of the Jw execution plan for the CLAIRE-COVID19 Notebook.

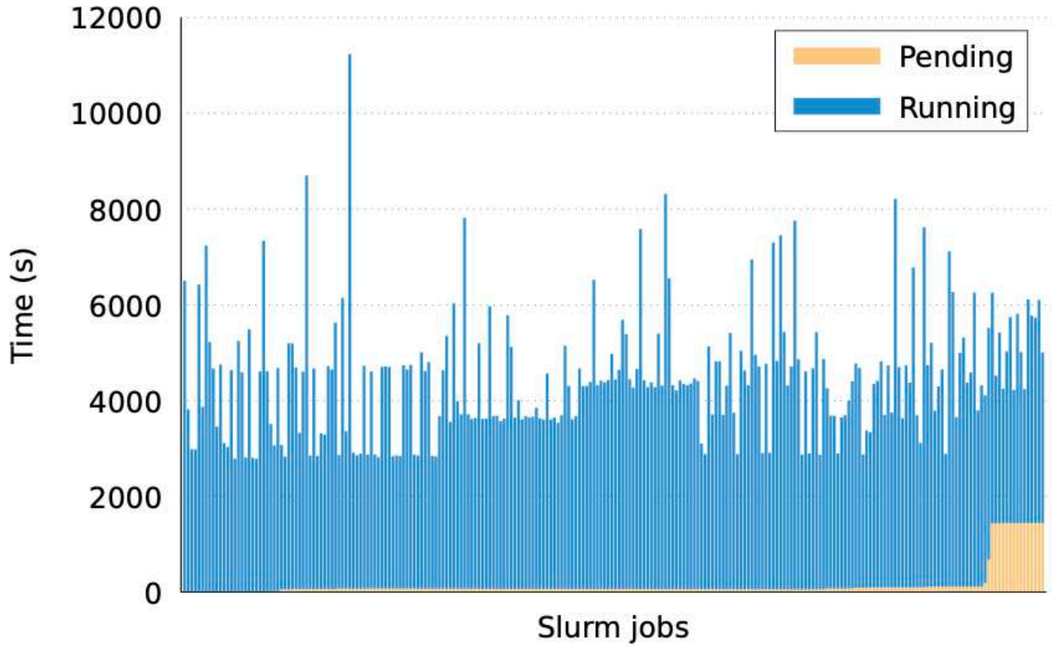
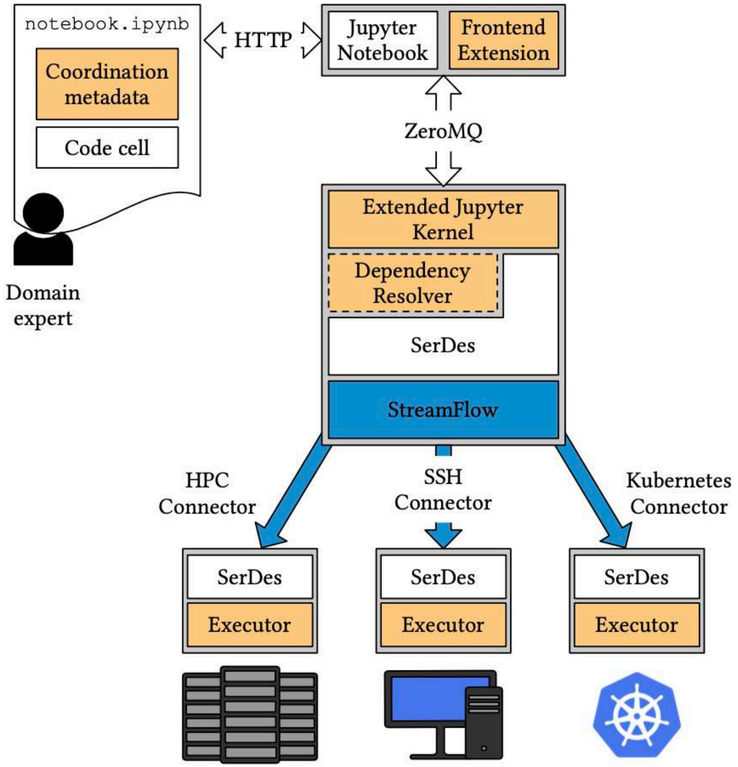
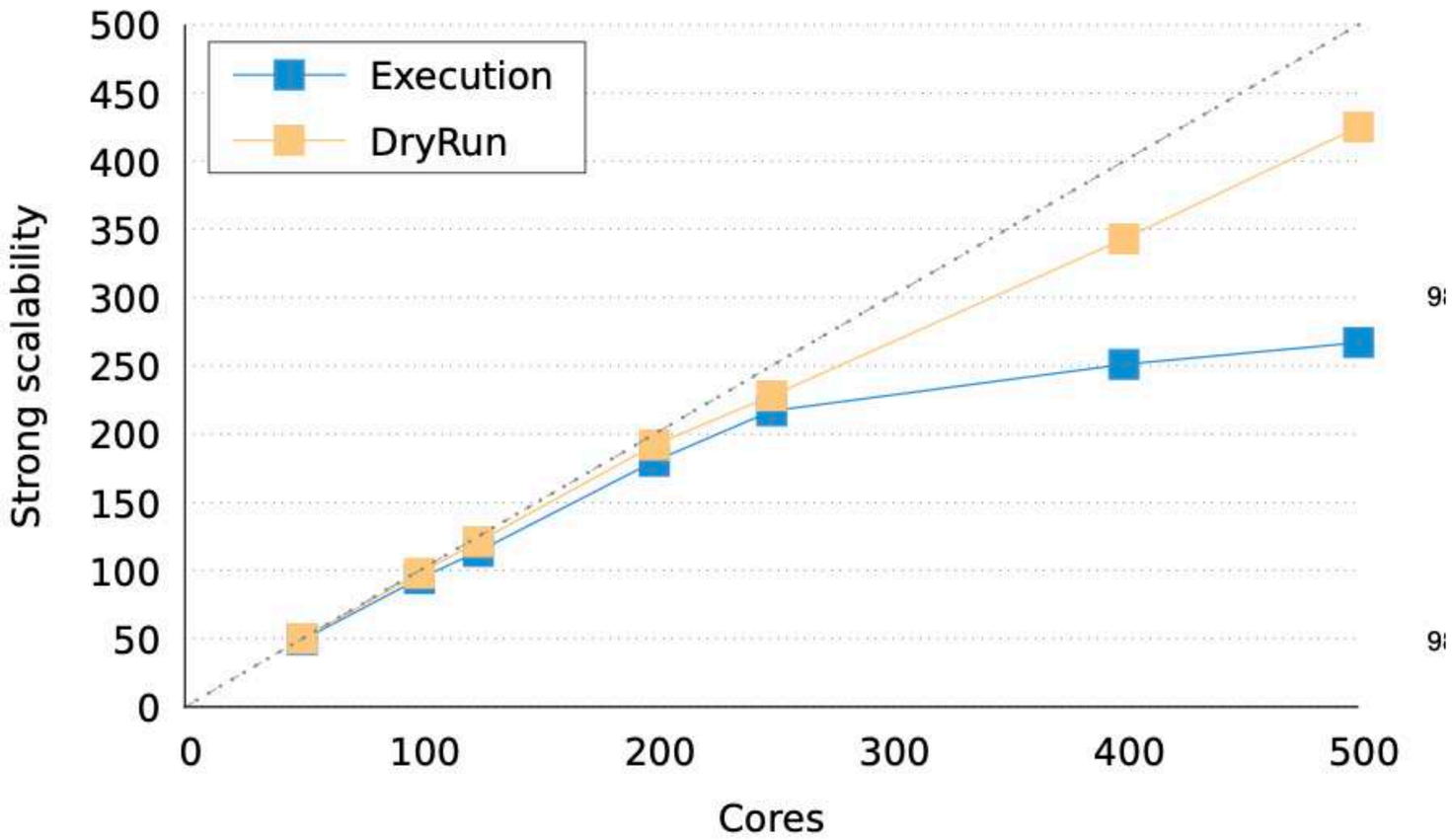


Figure 5: Execution times for the 240 DenseNet training experiments running in parallel on 240 GPU-equipped nodes of the CINECA MARCONI 100 facility.

1000-genome workflow on HPC4AI

Openstack + Kubernetes + HPC





QuantumEspresso “primordial soup” on Da-Vinci-1

SLURM (Openstack+SLURM on air-gapped facilities under design)

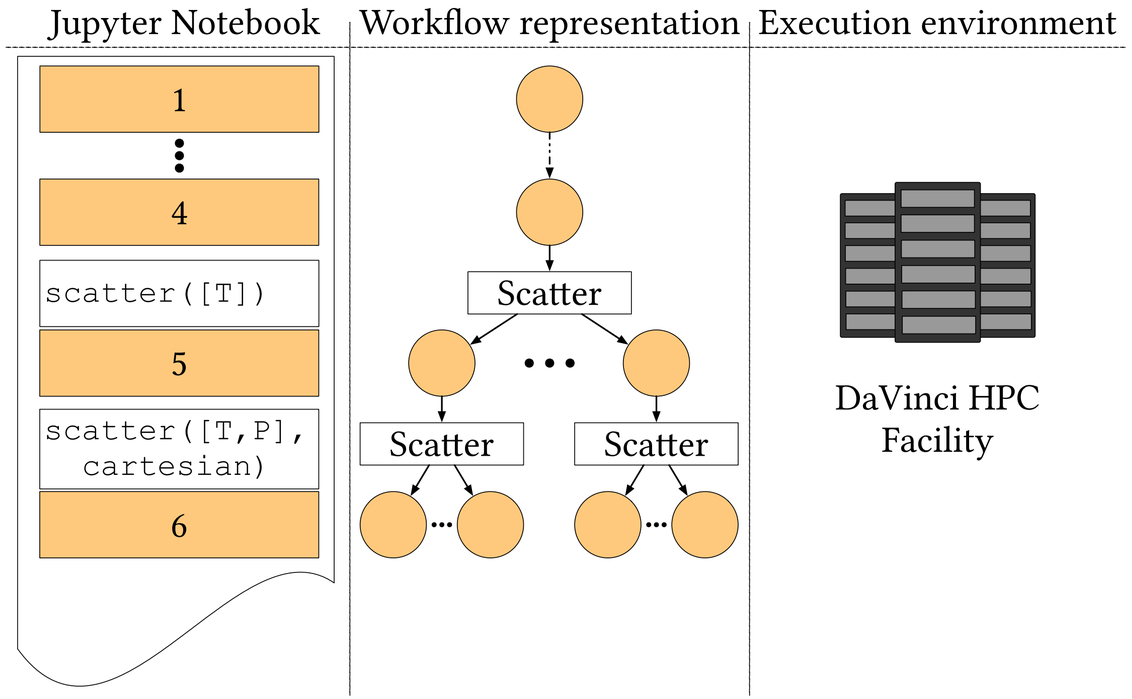


Table 1: Weak scalability for the Quantum ESPRESSO Notebook when executed manually on PBS and in both Jw execution modes.

Step	Nodes	PBS (s)	Jw Interactive (s)	Jw Bulk (s)
Cell 5	2	408	461	413
	4	407	490	415
	8	407	553	418
Cell 6	2	459	536	465
	8	459	706	469
	32	461	1861	474

Primordial Broth: explore the phase space to find where C-H, O-H and N-H bonds break up

Air-gapped facility

The “HPC Key Technologies and Tools” national lab at at the at the National Interuniversity Consortium for Informatics

M. Aldinucci et al. “The Italian research on HPC key technologies across EuroHPC,” in ACM Computing Frontiers, Virtual Conference, Italy, 2021. http://dx.doi.org/10.1145/3457388.3458508

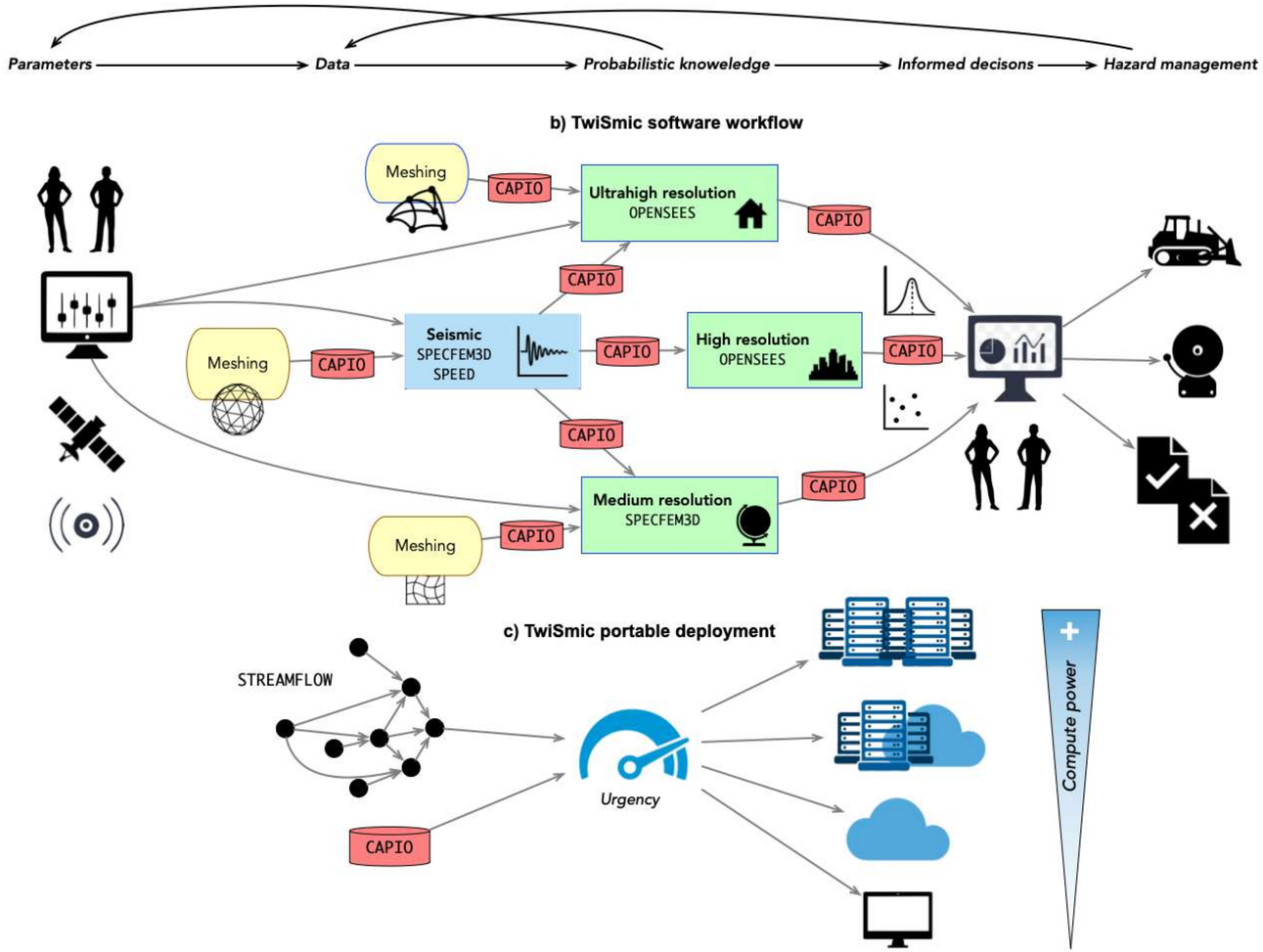
- Gathers 27 Italian Universities and 8 research institutes with a specific interest in HPC and cloud
- Formed during the outbreak - started early 2021
- Co-founder and director: Marco Aldinucci (UNITO)
- Team up to a scale to play at the international level
  - Manage internal competition in large EU and international initiatives
  - Promotes national strategy for HPC, cloud and related technolgies
- During first year of life get 5 EU EuroHPC projects funded (total cost 95M€)
  - ADMIRE, ACROSS, TEXTAROSSA, EUPEX, The European Pilot
- Three-pillars vision : Infrastructures, **Technologies & Tools** (we are here!), Applications

cini national lab HPC Key Technologies & Tools

We are hiring!

Universality enables digital twins

Scientific domain is just another hyper-parameter

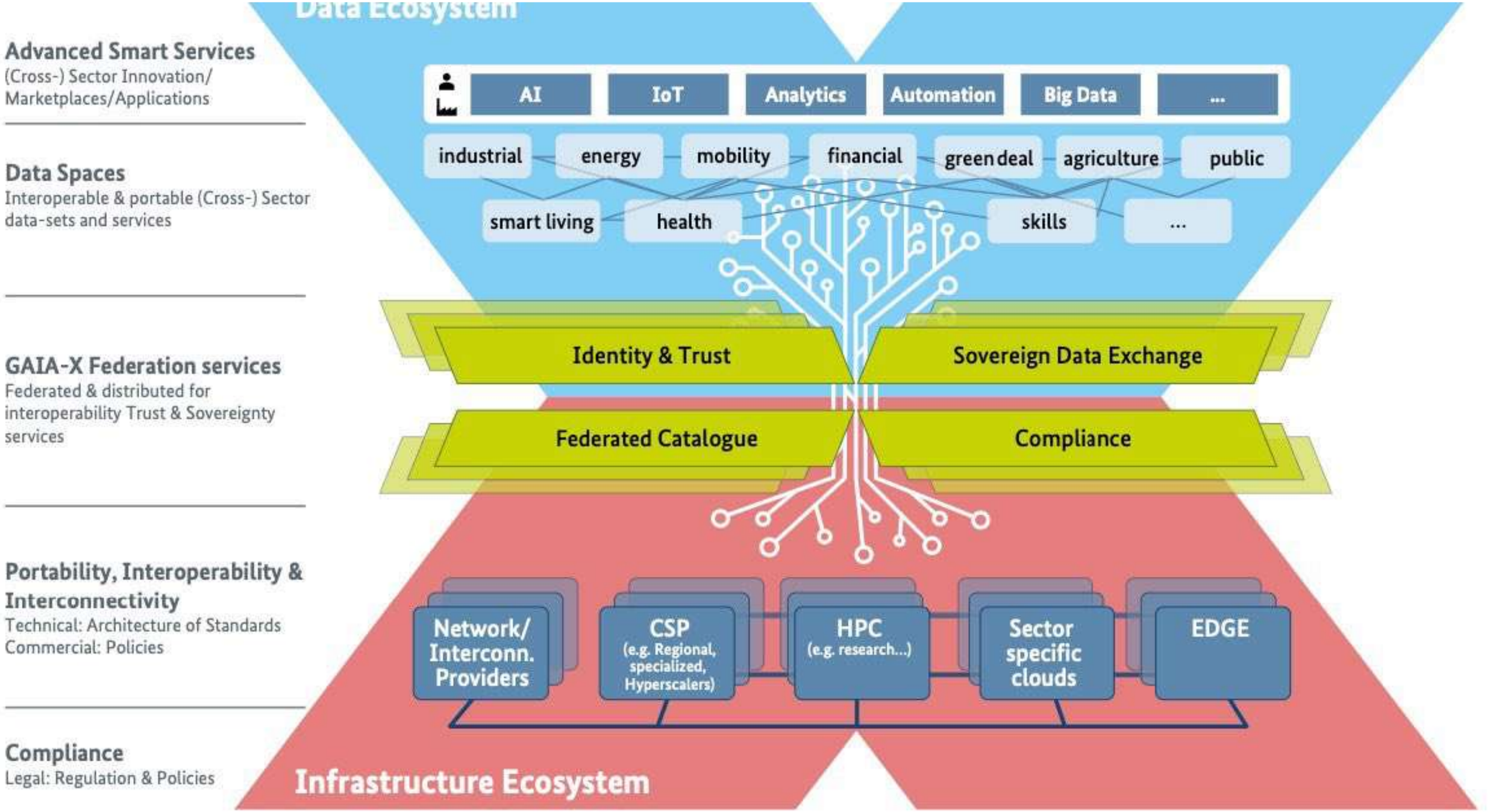


- The EUPEX project (41M€) (EUropean Pilot for EXascale)
- Starting mid 2022
- TwiSmic: High-Performance Digital-twin for Seismic Resiliency
- Another scientific domain, another kind of possible outbreak, same need to support urgent decisions, almost the same technologies likely to be exploited

HPC and AI are fated to meet, the cloud is the liaison

I. Colonnelli, B. Cantalupo, R. Esposito, M. Pennisi, C. Spampinato, and M. Aldinucci, "HPC Application Cloudification: The StreamFlow Toolkit," in 12th PARMA-DITAM workshop (co-located with HiPEAC 2021), Dagstuhl, Germany, 2021, p. 5:1–5:13. doi:10.4230/OASIcs.PARMA-DITAM.2021.5

- **AI (DL) needs HPC high-performance in computing and data movement (HPC)**
  - Accuracy depends on the number of examples; DNN are increasingly large
  - Actually, DL started to seriously work with GPUs (i.e. enough FLOPS)
- **HPC at scale needs GPUs (for performance/watt) and AI are a natural workload**
  - AI enables success in scientific challenges, where traditional HPC techniques have failed
- **Nevertheless, supercomputers are still rarely used for AI**
  - Supercomputers are not multi-tenant and not equipped to support specific AI services. AI researchers are not used to the HPC batch execution model
- **The HPC Modernisation, i.e. the shift of HPC toward cloud & services provisioning model is already undergoing**


Data Ecosystem
Advanced Smart Services
(Cross-) Sector Innovation/
Marketplaces/Applications
AI
IoT
Analytics
Automation
Big Data
...
Data Spaces
Interoperable & portable (Cross-) Sector
data-sets and services
industrial
energy
mobility
financial
green deal
agriculture
public
smart living
health
skills
...
GAIA-X Federation services
Federated & distributed for
interoperability Trust & Sovereignty
services
Identity & Trust
Sovereign Data Exchange
Federated Catalogue
Compliance
Portability, Interoperability &
Interconnectivity
Technical: Architecture of Standards
Commercial: Policies
Network/
Interconn.
Providers
CSP
(e.g. Regional,
specialized,
Hyperscalers)
HPC
(e.g. research...)
Sector
specific
clouds
EDGE
Compliance
Legal: Regulation & Policies
Infrastructure Ecosystem

The modernization of HPC applications for the cloud era

Marco Aldinucci, University of Torino

Abstract. Workflows are among the most commonly used tools in a variety of execution environments. Many of them target a specific environment; few of them make it possible to execute an entire workflow in different environments, e.g., clouds, supercomputers, and both of them. We present a novel approach to workflow execution, called StreamFlow, that complements the workflow graph with the declarative description of potentially complex execution environments (such as Kubernetes and SLURM), making it possible to execute onto multiple sites not sharing a common data space. Streamflow clearly distinguishes it from many other workflow management systems because it decouples the data dependencies from the deployment of (containerized) workflow steps. Streamflow also leverages CAPIO (Cross-Application Programmable I/O) to move data from one step to another efficiently. CAPIO captures the POSIX file system and streams it in parallel and in-memory to the workflow's next step, possibly enabling in-transit data filtering.