CWL in the HPC Ecosystem

Iacopo Colonnelli

Assistant Professor (RTDA), Computer Science Dept., University of Torino, Italy
Member of the CWL Technical Team
Co-lead of the CWL4HPC Working Group

UNIVERSITÀ
DI TORINO

UNIVERSITÀ DI TORINO

CWL in the HPC Ecosystem: How To

- Build HPC support **inside** CWL: the CWL4HPC Working Group
  1. Identify workflow patterns in HPC
  2. Implement them in the CWL semantics
  3. Validate the new features with real WMSs on real use cases
  4. Extend CWL with the new features

- Build HPC support **around** CWL: the StreamFlow WMS
  1. Couple standard CWL workflows with distributed execution semantics
  2. Develop workflow-aware and location-aware techniques (scheduling, fault tolerance, data movement, ...)
  3. Extend the StreamFlow WMS with custom plugins
  4. Let StreamFlow orchestrate large-scale distributed workflows on cloud+HPC and cross-HPC environments

Build HPC support inside CWL

The CWL4HPC Working Group

The CWL4HPC Working Group

The **Common Workflow Language for High-Performance Computing** (CWL4HPC) Working Group aims to identify **workflow patterns** capable of modelling **large-scale scientific applications** and implement the related **CWL enhancement proposals**

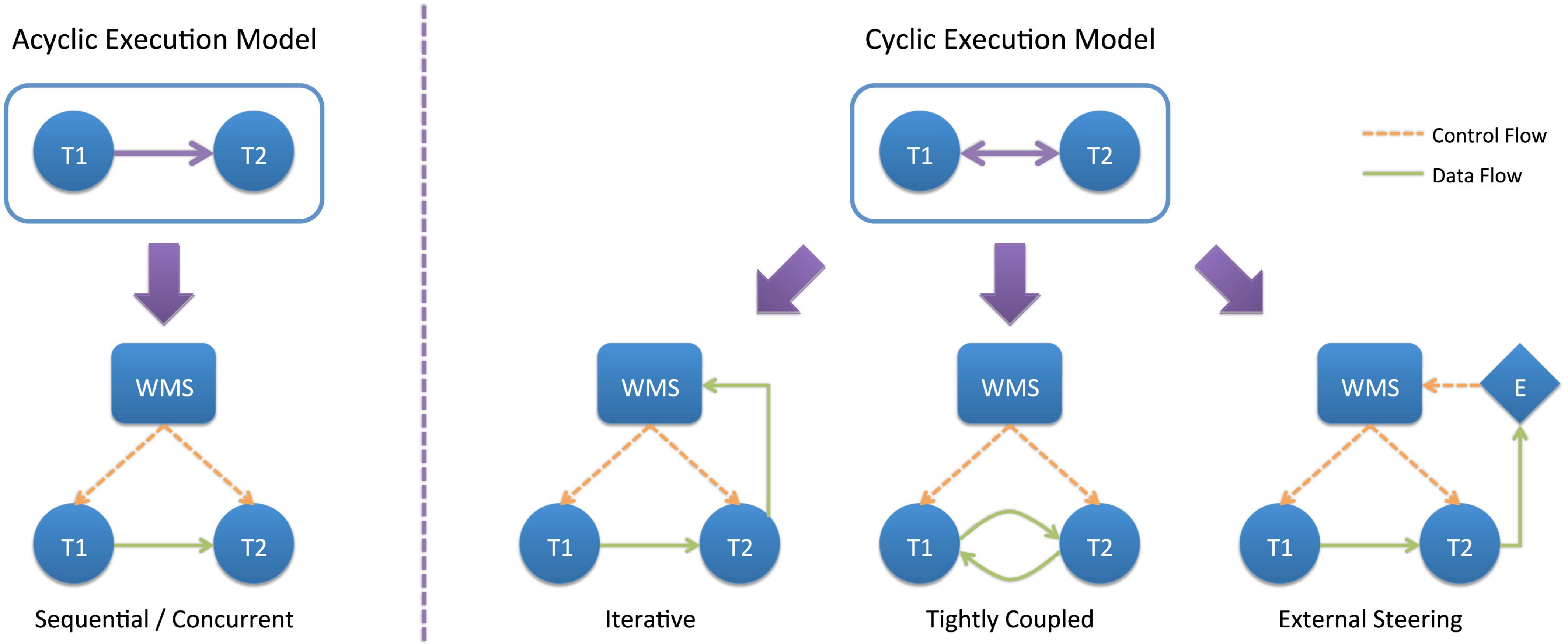
UNIVERSITÀ DI TORINO

The CWL4HPC Working Group

For each CWL enhancement proposal, the CWL4HPC group aims to:

1. Motivate it with **two real use cases** that would benefit from the proposed feature
2. Agree on a **first draft of the syntax and semantics** of the proposed feature
3. Implement it as a **CWL extension** on `cwltool` and at least another CWL-compliant workflow system, together with a suite of **conformance tests**
4. Validate it on **at least two existing CWL workflows** where the proposal is applied, or with new example workflows created on purpose

Workflow Patterns in HPC



R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications," *Future Generation Computer Systems*, vol 75, pp. 228-238, 2017. doi: 10.1016/j.future.2017.02.026

Workflow Patterns in CWL: Sequential

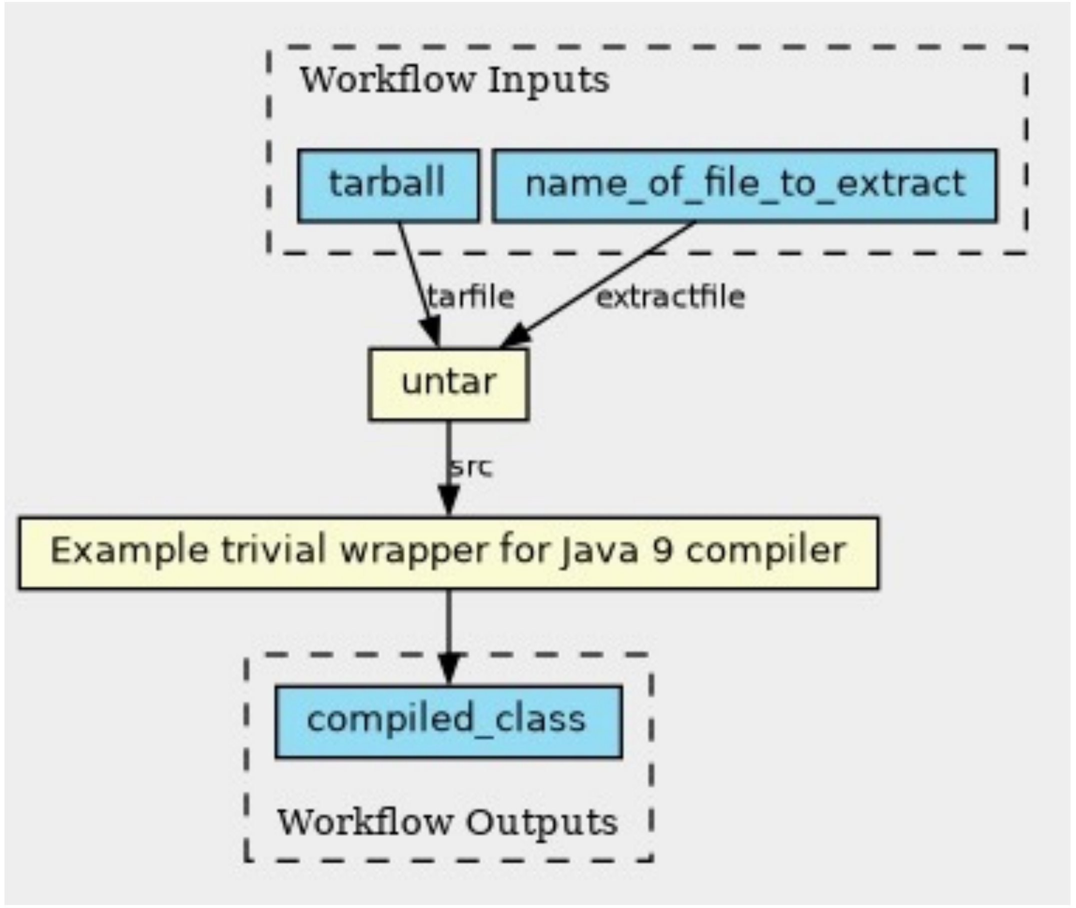
```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow

inputs:
  tarball: File
  name_of_file_to_extract: string

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

steps:
  untar:
    run: tar-param.cwl
    in:
      tarfile: tarball
      extractfile: name_of_file_to_extract
    out: [extracted_file]
  compile:
    run: arguments.cwl
    in:
      src: untar/extracted_file
    out: [classfile]
```



UNIVERSITÀ DI TORINO

Workflow Patterns in CWL: Parallel

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: Workflow

requirements:
  ScatterFeatureRequirement: {}

inputs:
  bam: File
  chromosomes: string[]

outputs:
  HaplotypeCaller_VCFs:
    type: File[][]
    outputSource: GATK_HaplotypeCaller/vcf

steps:
  GATK_HaplotypeCaller:
    run: GATK_HaplotypeCaller.cwl
    scatter: [intervals, input_bam]
    scatterMethod: flat_crossproduct
    in:
      input_bam: bam
      intervals: chromosomes
    out: [vcf]
```

CWL supports the scatter/gather data parallel patterns at the step level since v1.0

If the `scatter` field declares more than one input parameter, the `scatterMethod` field describes how to decompose the input into a discrete set of jobs (`dotproduct`, `nested_crossproduct`, or `flat_crossproduct`)

Workflow Patterns in CWL: Iterative

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.3.0-dev1
class: Workflow

requirements:
  InlineJavascriptRequirement: {}

inputs:
  i1: int

outputs:
  o1:
    type: int
    outputSource: subworkflow/o1

steps:
  subworkflow:
    run: sum.cwl
    when: $(inputs.i1 < 10)
    loop:
      i1: o1
    outputMethod: last
    in:
      i1: i1
    out: [o1]
```

A `cwltool:Loop` extension has been introduced to support iterative workflows, which is currently being evaluated for **inclusion in CWL v1.3**.

The `outputMethod` field determines the value to be propagated to subsequent steps of the workflow: just the output of the last iteration (`last`) or the ordered set containing the output of all loop iterations (`all`).

This is the first ongoing activity promoted by the **CWL4HPC Working Group**. There is still time to discuss improvements and modifications to the proposed syntax.

https://matrix.to/#/#cwl4hpc:matrix.org

UNIVERSITÀ DI TORINO

Workflow Patterns in CWL: Concurrent

Concurrent workflow steps require **streaming** capabilities. CWL itself does not (yet) support an explicit **Stream type** to connect input and output ports

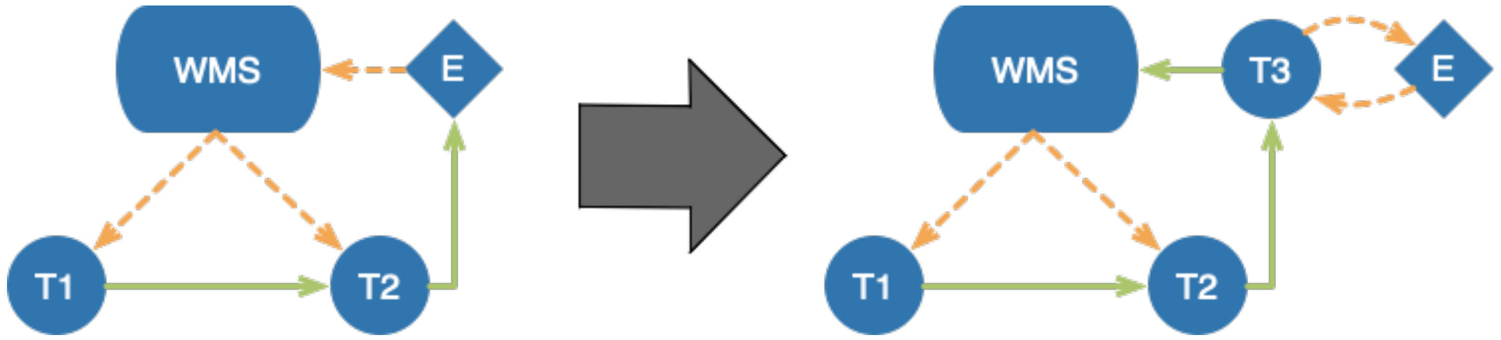
CWL structure is regular enough to allow some **automatic conversions** of arrays into streams (e.g., **Gather+Scatter**, **Loop+Scatter**), but not all cases can be safely optimized by the compiler/WMS (e.g., in case of `valueFrom` fields)

A **Channel CWL extension** has been proposed, but no WMS supports it by now. It will probably be one of the next focuses in the CWL4HPC group

https://github.com/common-workflow-language/common-workflow-language/issues/939

Workflow Patterns in CWL: Concurrent

Note that the **External Steering** pattern can always be modelled as a **Concurrent Iteration** pattern, enclosing the external interaction in a step's internal logic.



The **CWL Operation** class can model workflow steps as custom processes. This mechanism could be used to embed interactive steps in a CWL Workflow (e.g., modelling the cells of a Jupyter Notebook)

I. Colonnelli, M. Aldinucci, B. Cantalupo, L. Padovani, S. Rabellino, C. Spampinato, R. Morelli, R. Di Carlo, N. Magini and C. Cavazzoni, "Distributed workflows with Jupyter", *Future Generation Computer Systems*, vol. 128, pp. 282-298, 2022. doi: [10.1016/j.future.2021.10.007](https://doi.org/10.1016/j.future.2021.10.007)

Workflow Patterns in CWL: Coupling

Support for step coupling requires several features:

- Co-scheduling and potentially co-location of multiple workflow steps
- Communication channels between in/out ports of different workflow steps
- Application-agnostic communication protocols between multiple steps (e.g., POSIX-based pipes)

CWL does not support co-scheduling, co-location, and channel types. Plus, it offers (limited) support for streaming File contents (through the `streamable` field). A new extension to enrich the CWL data streaming capabilities is in plan

Alberto Riccardo Martinelli, Massimo Torquati, Marco Aldinucci, Iacopo Colonnelli, Barbara Cantalupo. "CAPIO: a Middleware for Transparent I/O Streaming in Data-Intensive Workflows", *2023 IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC)*, IEEE, Goa, India, 2023.

The CWL4HPC Working Group

The CWL4HPC Working Group aims to build, refine, and validate high-quality CWL enhancement proposals in the HPC ecosystem.

The CWL4HPC proposal process is **iterative**. If some criticalities of further enhancements emerge during the implementation or validation phases, the syntax and semantics can be refined and the process restarts. After reaching a sufficient level of maturity, the group agrees to present the proposal to the CWL community for **inclusion in the following standard version**

Domain experts, HPC administrators, workflow designers and maintainers, and workflow system implementers are **welcome to join the discussion**, but the group is open to anyone who wants to contribute

https://www.commonwl.org/working-groups/cwl4hpc

UNIVERSITÀ DI TORINO

Outside CWL4HPC

The CWL4HPC Working Group is not (and neither aims to be) the only place where to develop HPC-oriented CWL extensions. Other examples of CWL Extensions for the HPC ecosystem are:

- The **MPIRequirement** extension, supported by the `cwltool` WMS, which allows users to specify the number of processes that should run an MPI command
- The **CUDARequirement** extension, supported again by the `cwltool` WMS, that targets heterogeneous HPC facilities equipped with NVIDIA GPUs

Both features are under evaluation to be **included in CWL 1.3**

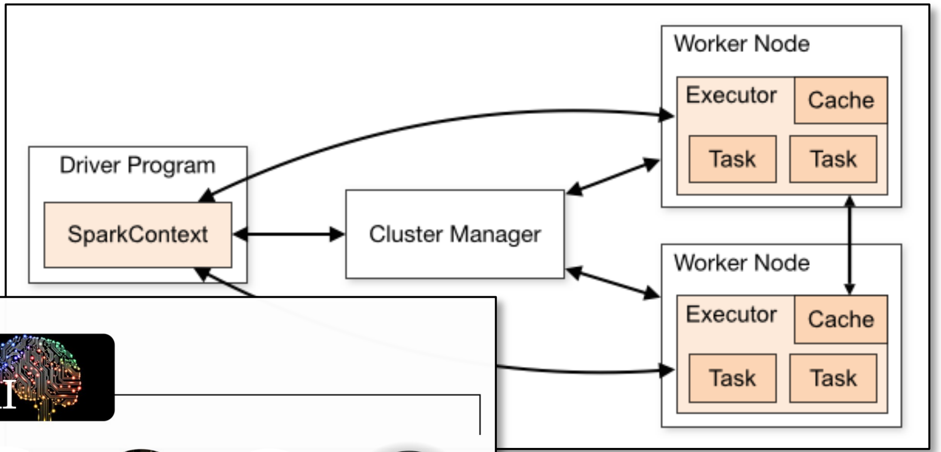
Build HPC support around CWL

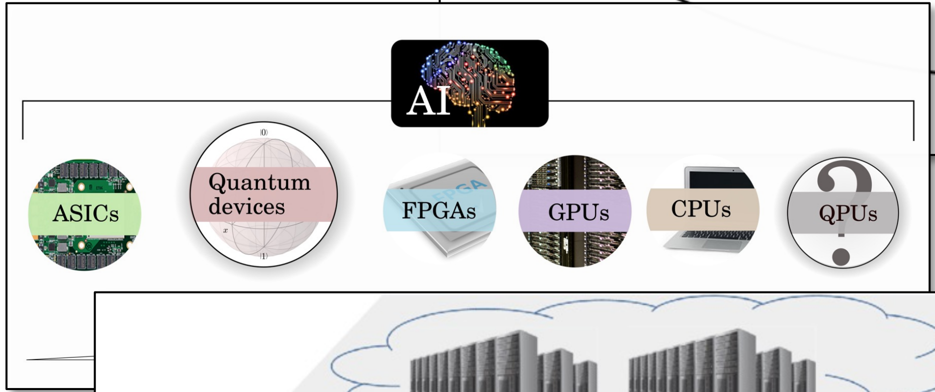
Hybrid Workflows and the StreamFlow WMS

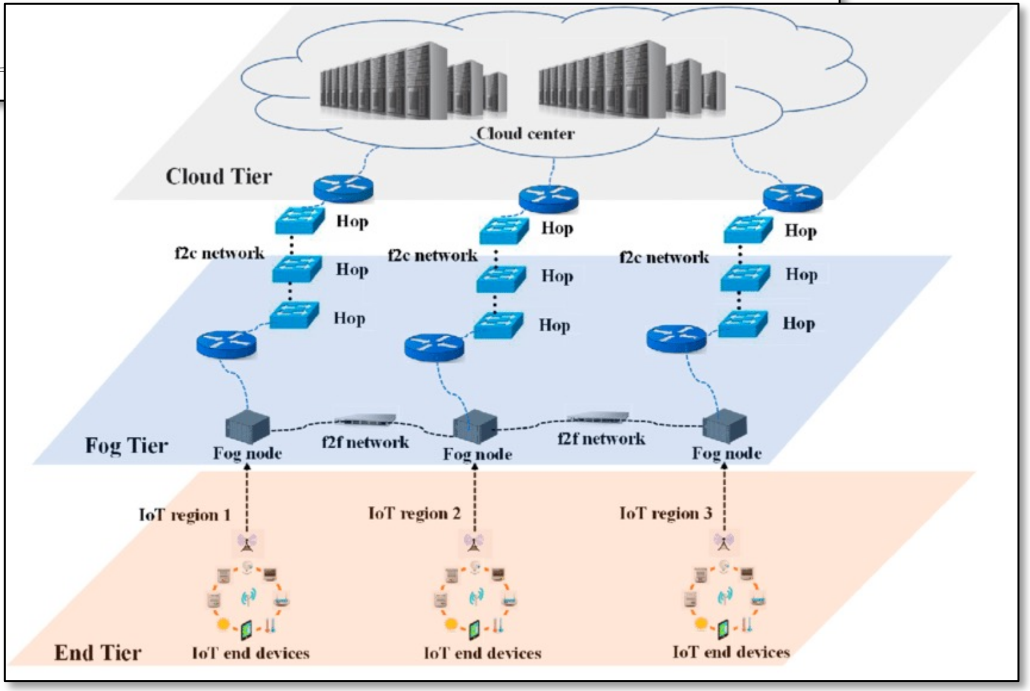
UNIVERSITÀ
DI TORINO

Large-Scale Workflows

- Each step of a distributed application can require **multiple intercommunicating agents** (e.g., a Spark cluster or a micro-services architecture)
- Large-scale architectures can be **heterogeneous** (e.g., Cloud+HPC environments and Classical+Quantum computing)
- Large-scale architectures can be **modular**, and modules can be **independent** of each other (e.g., modular HPC and infrastructure federations)

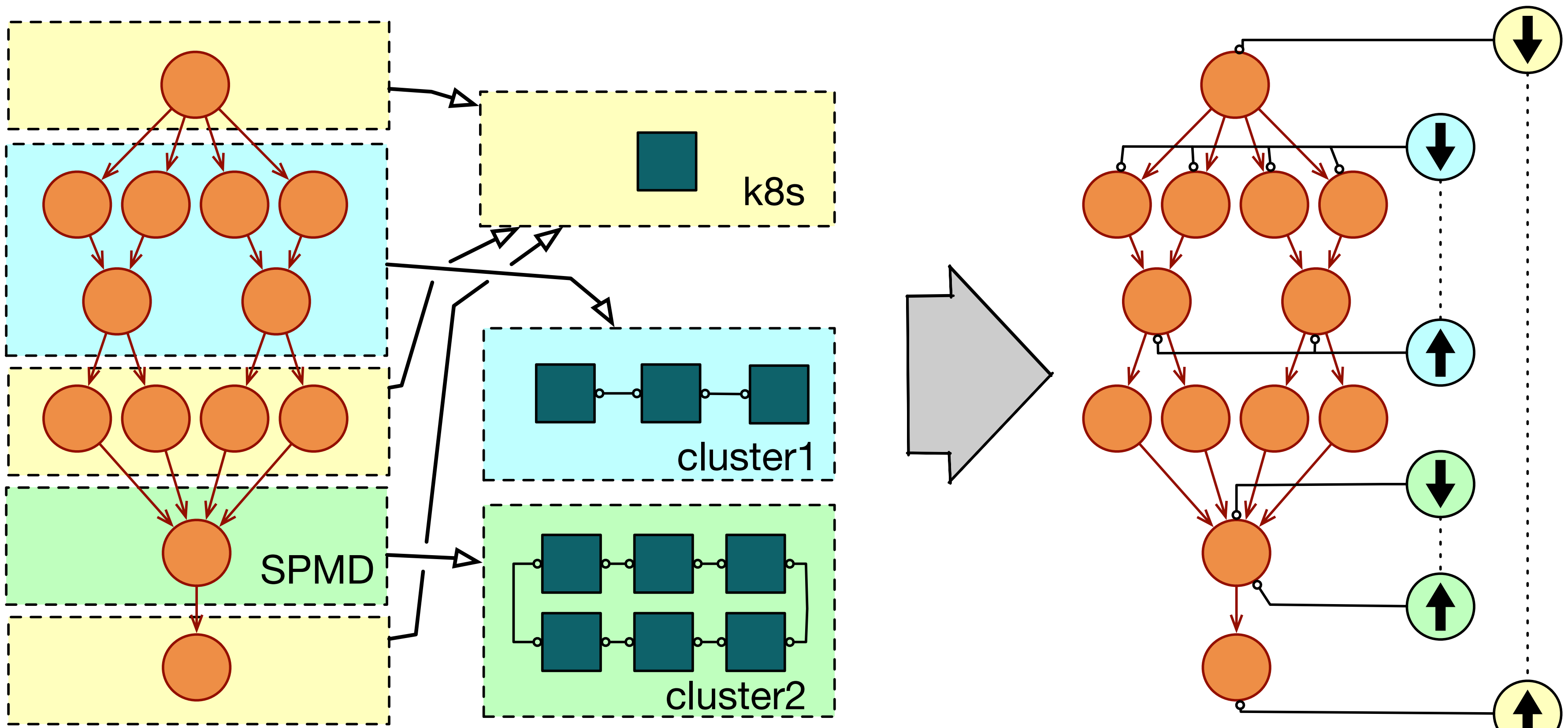






Hybrid Workflows

A **hybrid workflow** is a workflow whose steps can span **multiple**, **heterogeneous**, and **independent** computing infrastructures.



Hybrid Workflows

Workflow model

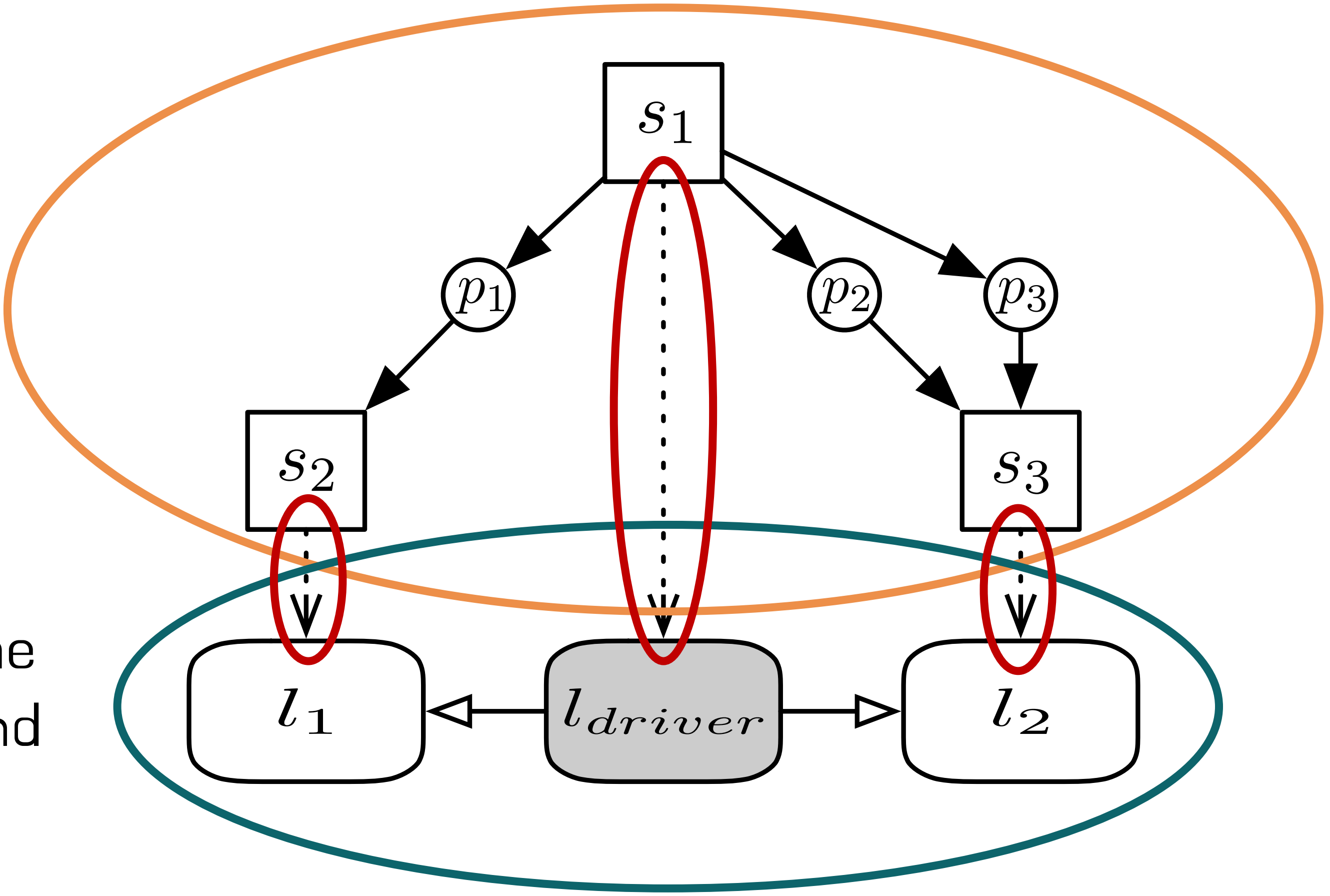
A directed bipartite graph encoding executable **steps**, data **ports** and **dependencies** between them

Topology of deployment locations

A directed graph where the nodes are the **locations** in charge of executing steps and the links are directed **communication channels** between locations

Mapping relations

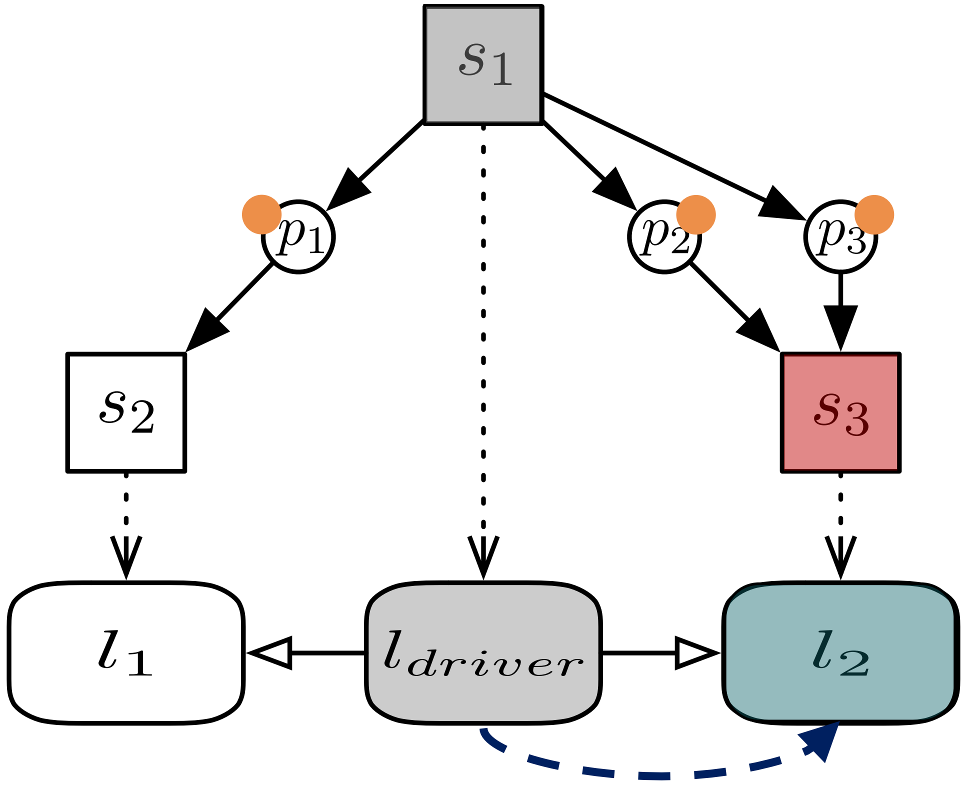
Many-to-many relations stating which locations are in charge of executing each workflow step



Model Interpretation

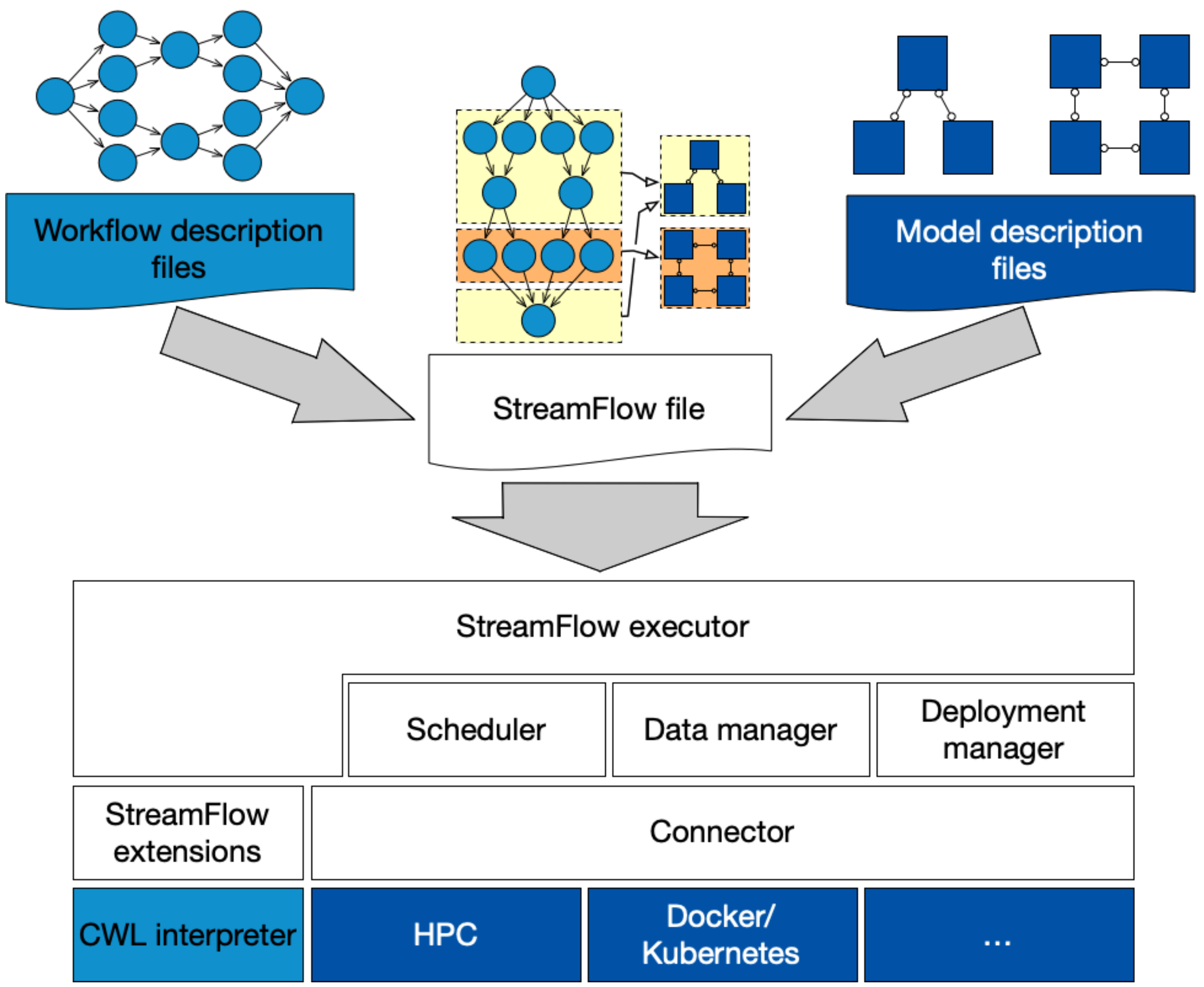
A step s becomes **fireable** (ready for execution) when:

- Each input port $In(s)$ contains the right number of **tokens**
- Its related location is **deployed**
- All its input data have been **transferred** on that location



UNIVERSITÀ DI TORINO

The StreamFlow WMS

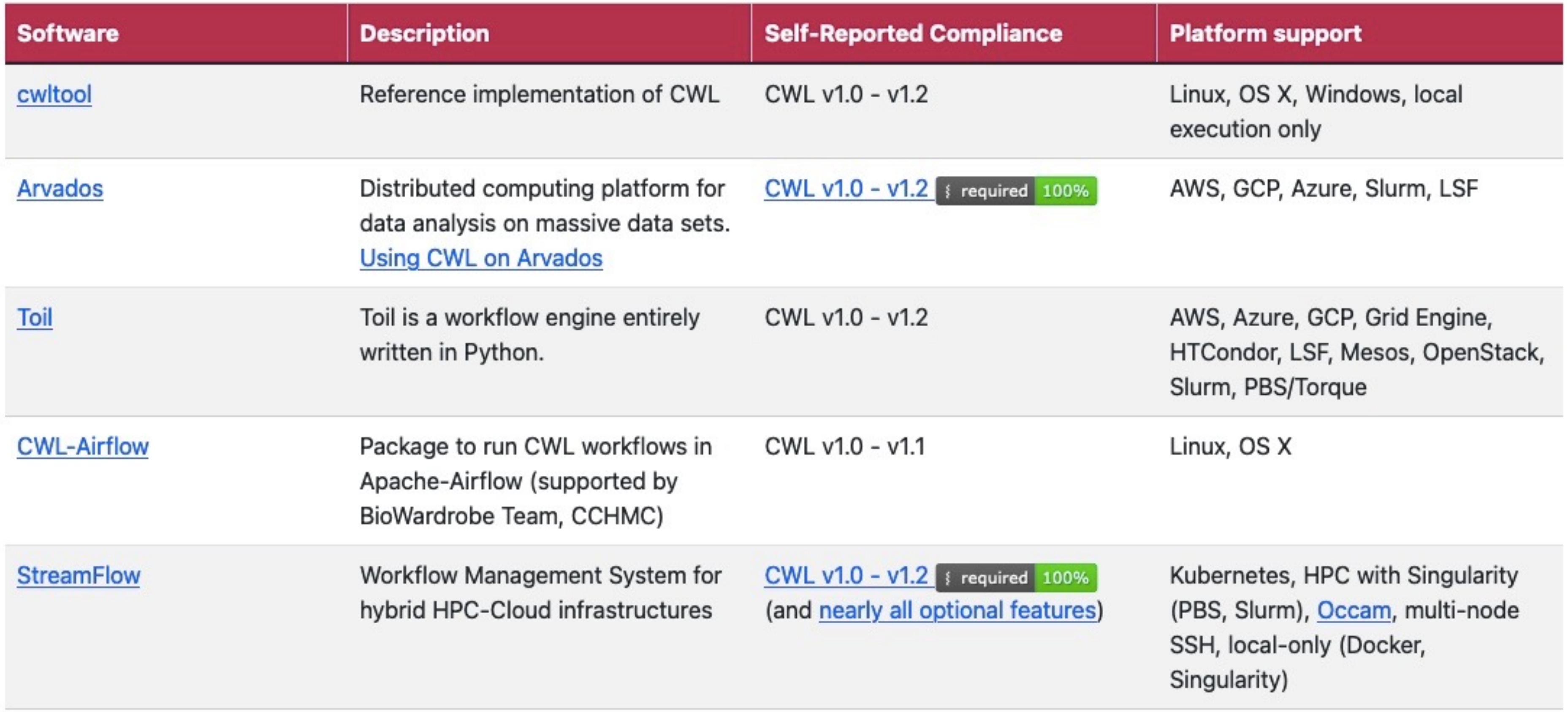


https://streamflow.di.unito.it

I. Colonnelli, B. Cantalupo, I. Merelli and M. Aldinucci, "StreamFlow: cross-breeding cloud with HPC," in *IEEE Transactions on Emerging Topics in Computing*, vol. 9, iss. 4, p. 1723-1737, 2021. doi: 10.1109/TETC.2020.3019202.

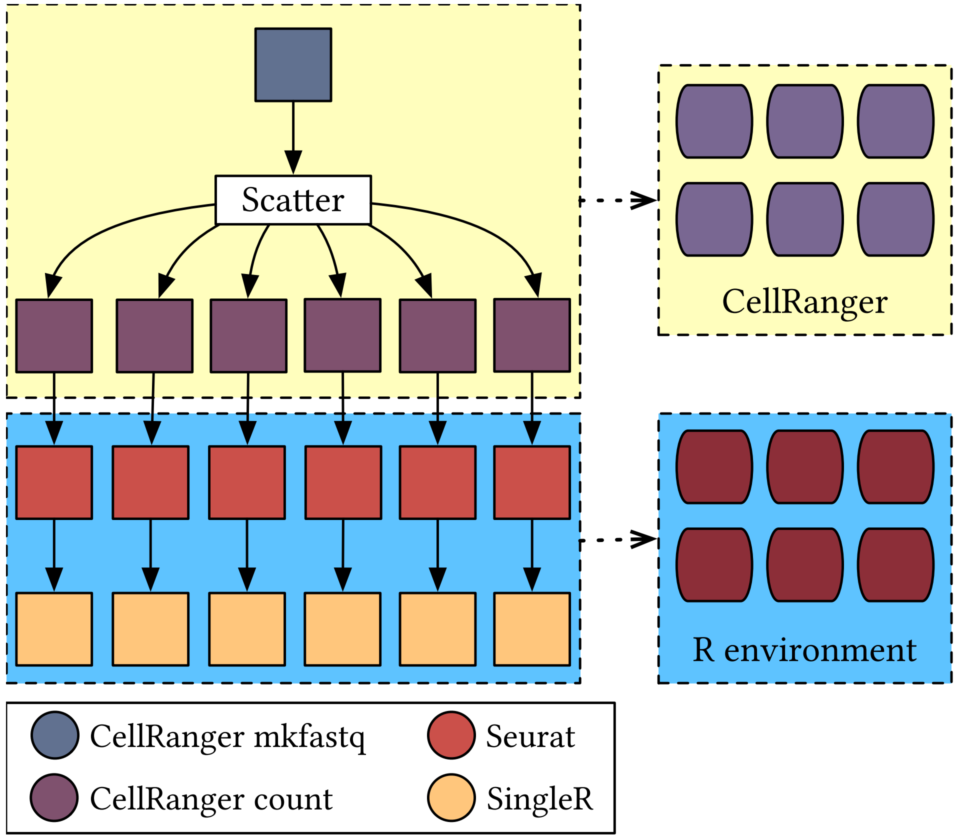
The StreamFlow WMS

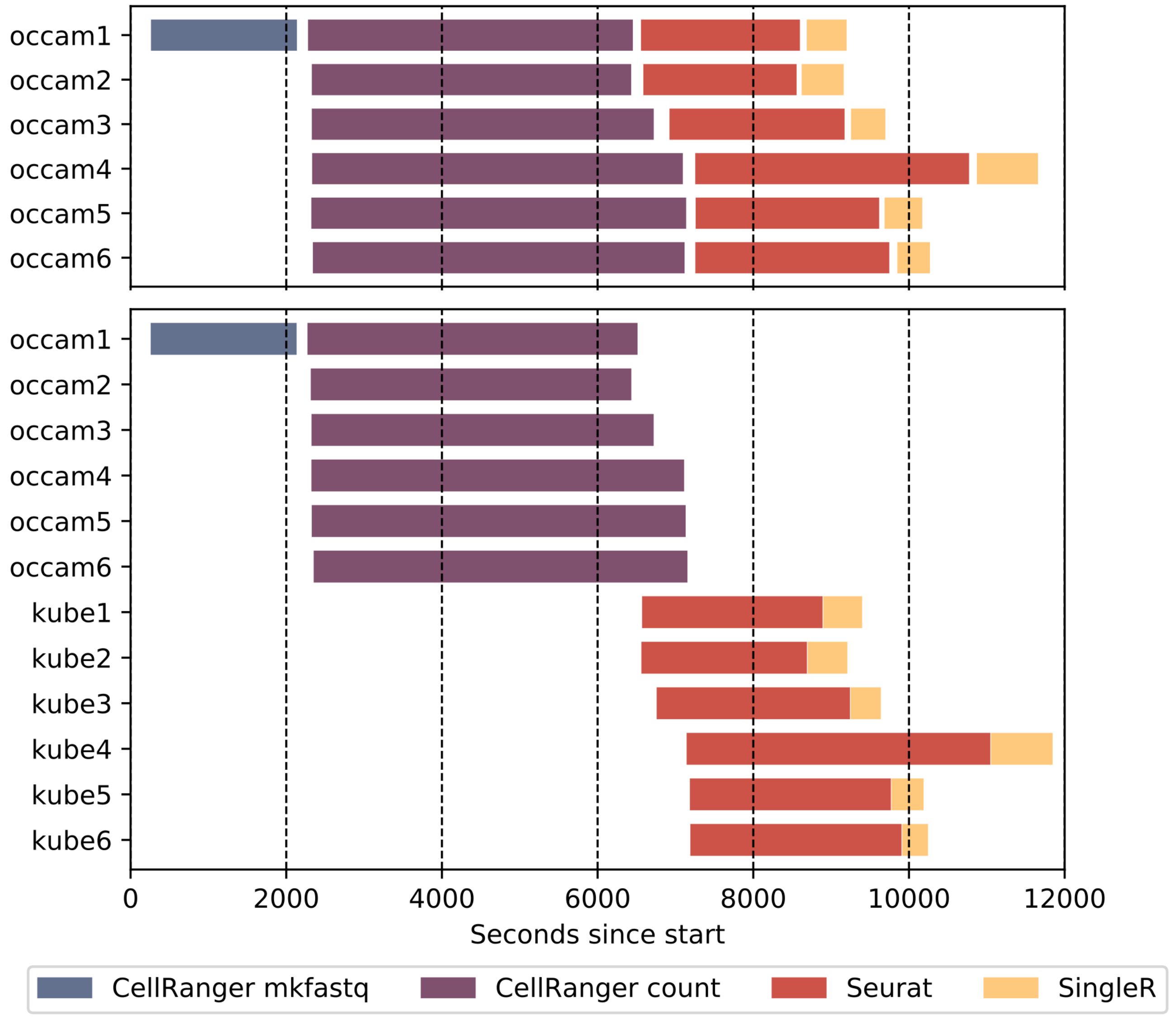
StreamFlow is listed as a **production-ready implementation** of CWL. It has also been used as a software laboratory to experiment new CWL extensions in the **CWL4HPC Working Group** (e.g., the Loop extension for iterative workflows)



Software	Description	Self-Reported Compliance	Platform support
cwltool	Reference implementation of CWL	CWL v1.0 - v1.2	Linux, OS X, Windows, local execution only
Arvados	Distributed computing platform for data analysis on massive data sets. Using CWL on Arvados	CWL v1.0 - v1.2 required 100%	AWS, GCP, Azure, Slurm, LSF
Toil	Toil is a workflow engine entirely written in Python.	CWL v1.0 - v1.2	AWS, Azure, GCP, Grid Engine, HTCondor, LSF, Mesos, OpenStack, Slurm, PBS/Torque
CWL-Airflow	Package to run CWL workflows in Apache-Airflow (supported by BioWardrobe Team, CCHMC)	CWL v1.0 - v1.1	Linux, OS X
StreamFlow	Workflow Management System for hybrid HPC-Cloud infrastructures	CWL v1.0 - v1.2 required 100% (and nearly all optional features)	Kubernetes, HPC with Singularity (PBS, Slurm), Occam, multi-node SSH, local-only (Docker, Singularity)

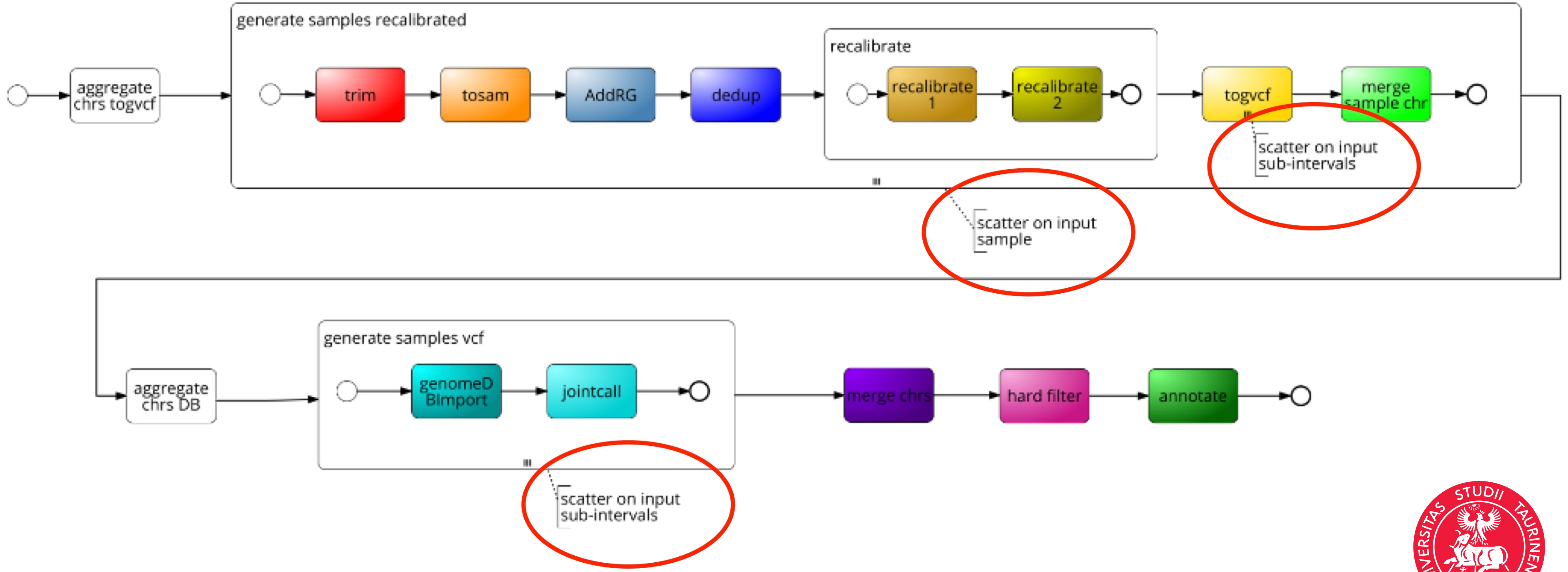
Case study: Single-Cell Pipeline





I. Colonnelli, B. Cantalupo, I. Merelli and M. Aldinucci, "StreamFlow: cross-breeding cloud with HPC," in *IEEE Transactions on Emerging Topics in Computing*, vol. 9, iss. 4, p. 1723-1737, 2021. doi: 10.1109/TETC.2020.3019202.

Case study: Variant Calling Pipeline

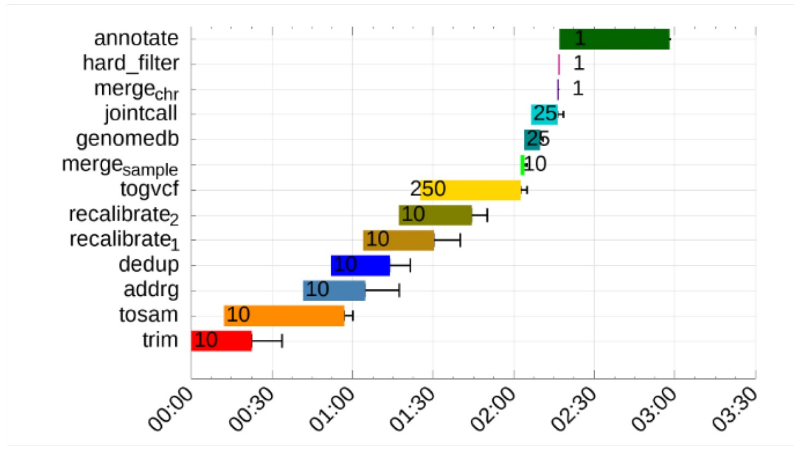


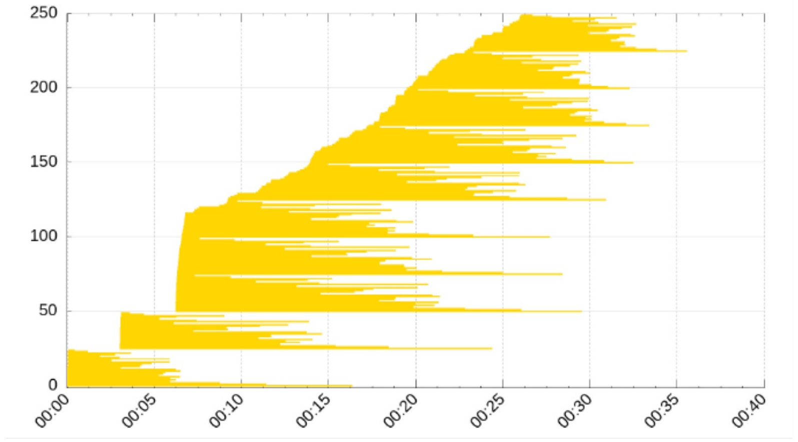
A. Mulone, S. Awad, D. Chiarugi, and M. Aldinucci, “Porting the variant calling pipeline for NGS data in cloud-HPC environment,” in *47th IEEE annual computers, software, and applications conference, COMPSAC 2023*, Torino, Italy, 2023, p. 1858–1863. doi:10.1109/COMPSAC57700.2023.00288

UNIVERSITÀ DI TORINO

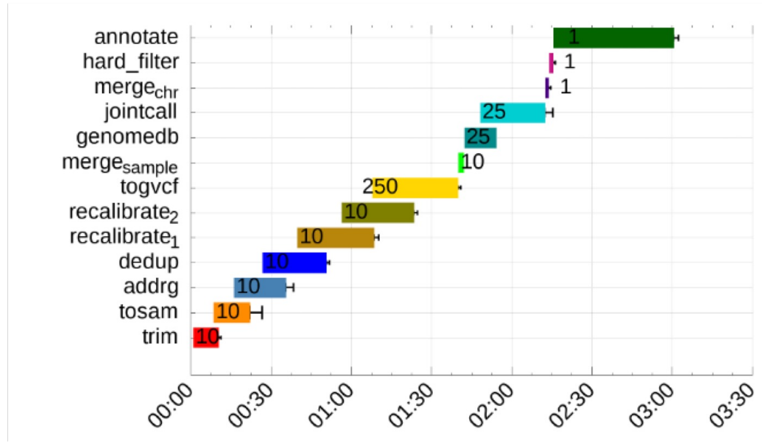
Case study: Variant Calling Pipeline

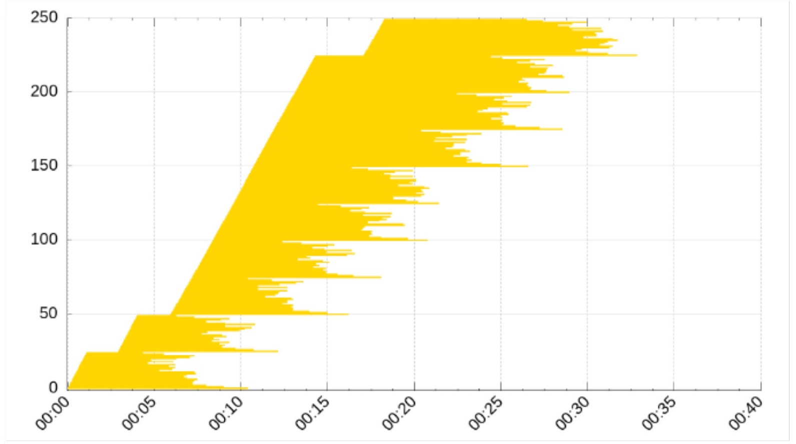
Cloud VM
(96 cores)





HPC system
(CINECA Galileo100)



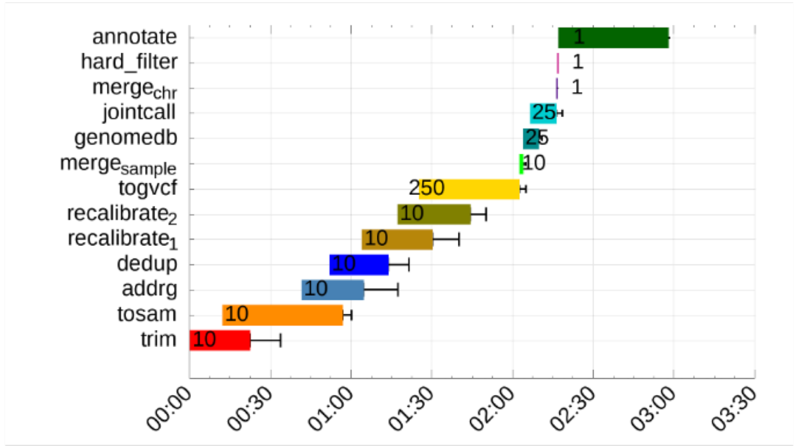


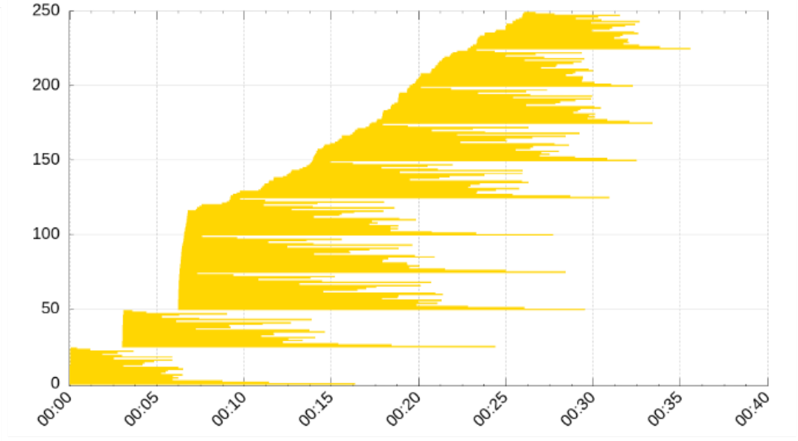
A. Mulone, S. Awad, D. Chiarugi, and M. Aldinucci, "Porting the variant calling pipeline for NGS data in cloud-HPC environment," in *47th IEEE annual computers, software, and applications conference, COMPSAC 2023*, Torino, Italy, 2023, p. 1858–1863. doi:10.1109/COMPSAC57700.2023.00288

UNIVERSITÀ DI TORINO

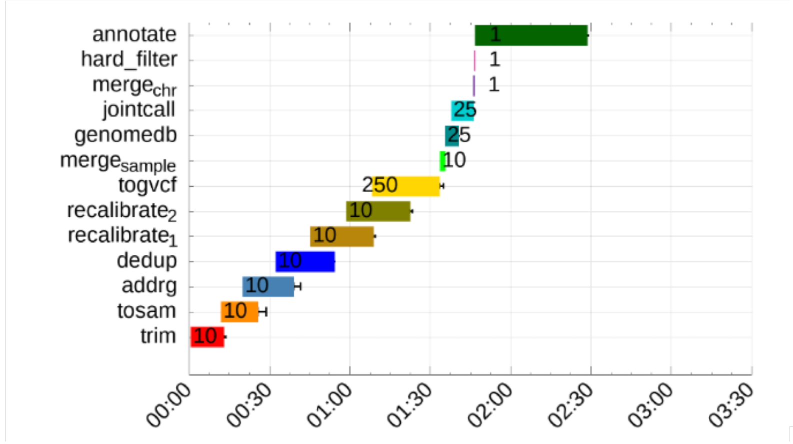
Case study: Variant Calling Pipeline

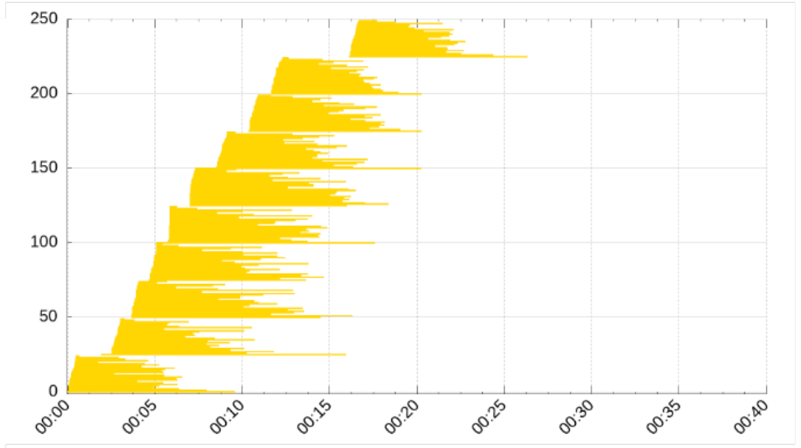
Cloud VM
(96 cores)





Hybrid
(Cloud + HPC)

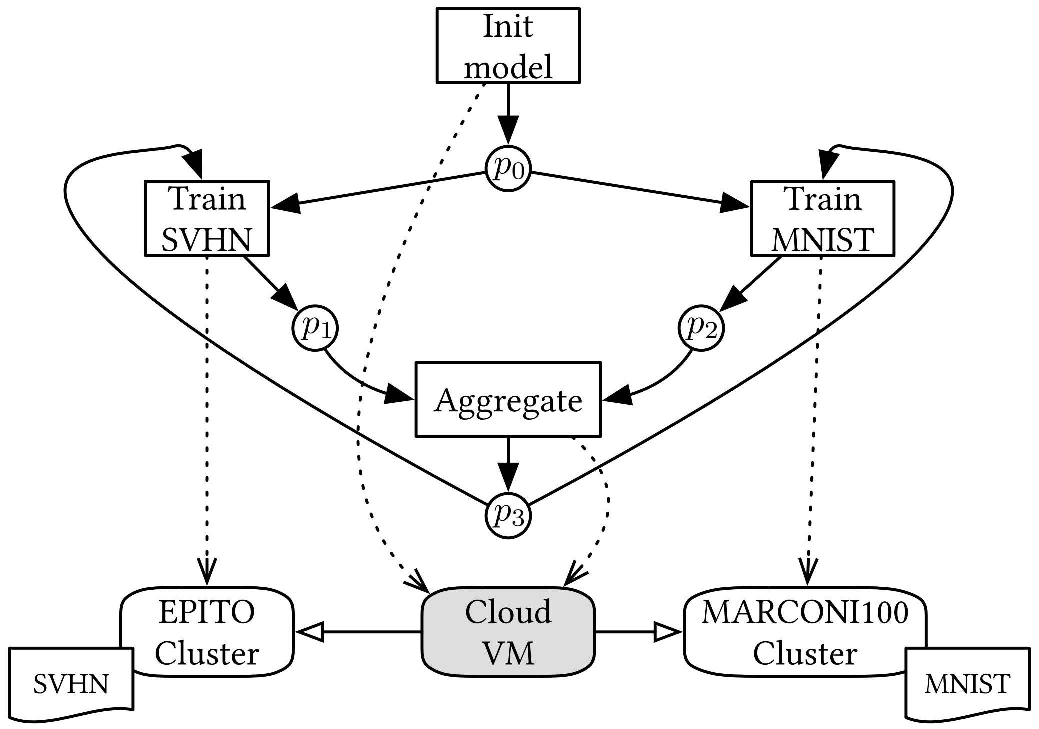




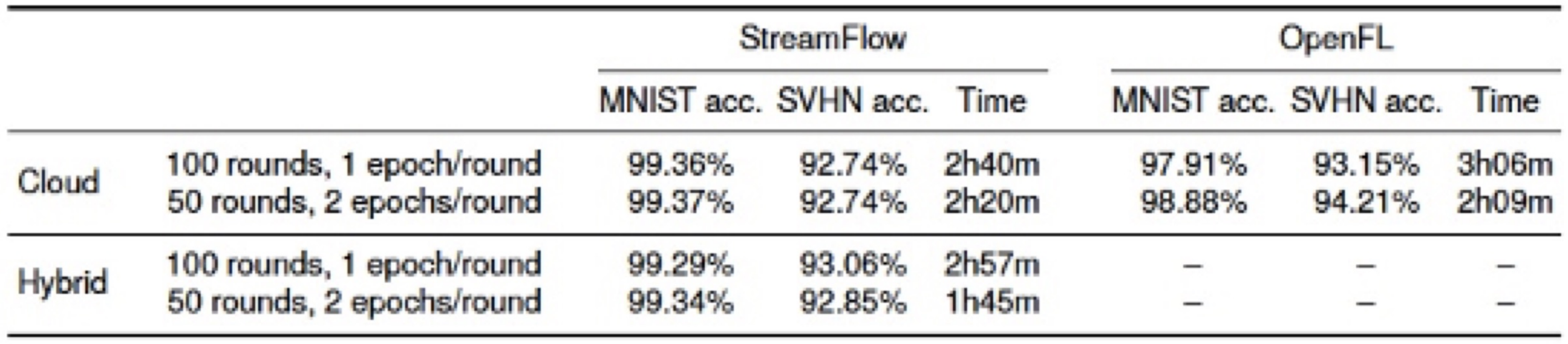
A. Mulone, S. Awad, D. Chiarugi, and M. Aldinucci, “Porting the variant calling pipeline for NGS data in cloud-HPC environment,” in *47th IEEE annual computers, software, and applications conference, COMPSAC 2023*, Torino, Italy, 2023, p. 1858–1863. doi:10.1109/COMPSAC57700.2023.00288

UNIVERSITÀ DI TORINO

Case study: Cross-Facility Federated Learning

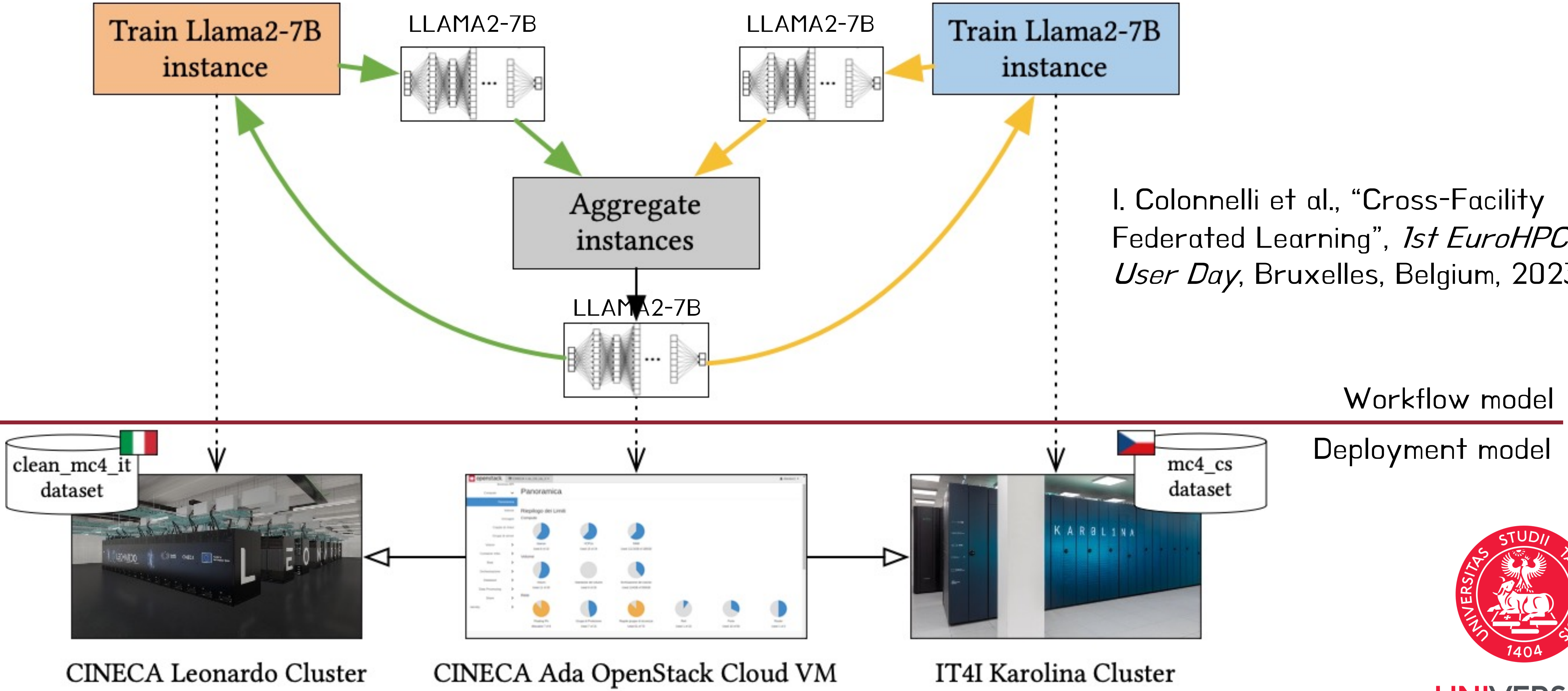


I. Colonnelli, B. Casella, G. Mittone, Y. Arfat, B. Cantalupo, R. Esposito, A. R. Martinelli, D. Medić and M. Aldinucci, "Federated Learning meets HPC and cloud," in *Astrophysics and Space Science Proceedings*, vol 60, 2023, p. 193-199. doi: 10.1007/978-3-031-34167-0_39



		StreamFlow			OpenFL		
		MNIST acc.	SVHN acc.	Time	MNIST acc.	SVHN acc.	Time
Cloud	100 rounds, 1 epoch/round	99.36%	92.74%	2h40m	97.91%	93.15%	3h06m
	50 rounds, 2 epochs/round	99.37%	92.74%	2h20m	98.88%	94.21%	2h09m
Hybrid	100 rounds, 1 epoch/round	99.29%	93.06%	2h57m	–	–	–
	50 rounds, 2 epochs/round	99.34%	92.85%	1h45m	–	–	–

Case study: Cross-Facility Federated Learning



I. Colonnelli et al., "Cross-Facility Federated Learning", *1st EuroHPC User Day*, Bruxelles, Belgium, 2023.

StreamFlow Adoption: European Projects

StreamFlow orchestrated distributed workflows in the **OpenDeepHealth platform**, a Kubernetes-based Deep Learning and Inference platform realised in the context of the **DeepHealth European project** (G. A. 825111, 36 months, 14.8M€)

StreamFlow is the **high-level workflow coordination tool** of the HPC+cloud Orchestration architecture in the **ACROSS European Project** (G. A. 955648, 36 months, 8M€)

StreamFlow has been chosen as one of the three **exploitable results** of the **EUPEX European Project** (G.A. 101033975, 48 months, 41M€), where it is used to orchestrate large-scale workflows on top of modular HPC architectures

StreamFlow will be used to develop **next generation large-scale workflows**, targeting modular Exascale HPC facilities, in the **SPACE Center of Excellence** (2023, 48 months. total cost 8M€, G.A. 101093441)

StreamFlow Adoption: Other Use Cases

StreamFlow coordinates distributed data-oriented workflows on top of OKD and HPC facilities in the EBRAINS Workflow Platform, the European Brain ReseArch InfrastructureS federation

StreamFlow has been evaluated to submit large-scale distributed workflows on the Amazon AWS Batch cloud service by the NASA Jet Propulsor Laboratory

StreamFlow is being evaluated by the ECMWF Research Centre to serve as the orchestration engine for large-scale weather forecasting workflows in the Earth Observation for European Union (EO4EU) European Project

StreamFlow is being evaluated by ASTRON, the Netherlands Institute for Radio Astronomy, to orchestrate large-scale astrophysics workflow at the LOFAR radio telescope

StreamFlow is part of the workflow orchestration software stack in the first Spoke (Future HPC & BigData) of the National Research Centre for High Performance, Big Data and Quantum Computing

EBRAINS

Jet Propulsion Laboratory
California Institute of Technology

ECMWF

ASTRON

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

UNIVERSITÀ DI TORINO

Conclusion

Build HPC support **inside** CWL: the CWL4HPC Working Group

Goal: make CWL a first-class citizen in HPC workflow modelling

Non-goals:

- Sacrifice ease of usage/support in the name of performance
- Privilege HPC over other CWL application fields
- Design new features without involving CWL community in the process

Roadmap:

- [✓] Support iterative patterns
- [ ] Support concurrent patterns
- [ ] Support coupling patterns
- [ ] Evaluate your proposals https://matrix.to/#/#cwl4hpc:matrix.org

UNIVERSITÀ DI TORINO

Conclusion

Build HPC support **around** CWL: the StreamFlow WMS

Goals:

- Experiment with CWL-based hybrid workflows on complex environments
- Easily test new HPC-oriented CWL extensions at scale
- Allow workflow researcher to integrate and test their ideas through plugins

Roadmap:

- ✓ 100% CWL compliance
- ✓ Support iterative patterns
- ☐ Support concurrent and coupling patterns
- ☐ Support your research https://github.com/alpha-unito/streamflow

UNIVERSITÀ DI TORINO