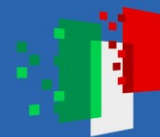




Finanziato  
dall'Unione europea  
NextGenerationEU



Ministero  
dell'Università  
e della Ricerca



Italiadomani  
PIANO NAZIONALE  
DI RIPRESA E RESILIENZA



# AccelerateRL: scaling beam dynamics simulations on INFN Cloud for Reinforcement Learning training

Davide Marcato, Daniel Lupu, Massimo Biasotto, Sergio Fantinel, Michele Gulmini  
INFN-LNL

20 - 24 maggio 2024

Workshop sul Calcolo nell'I.N.F.N.—Palau (Sassari)



Istituto Nazionale di Fisica Nucleare  
Laboratori Nazionali di Legnaro



Finanziato  
dall'Unione europea  
NextGenerationEU



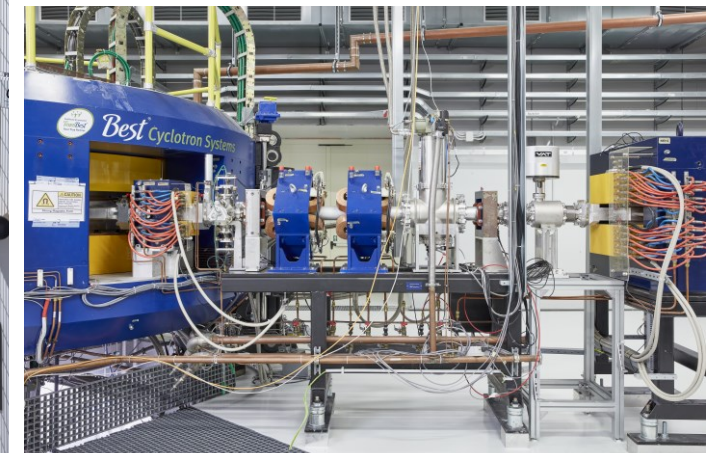
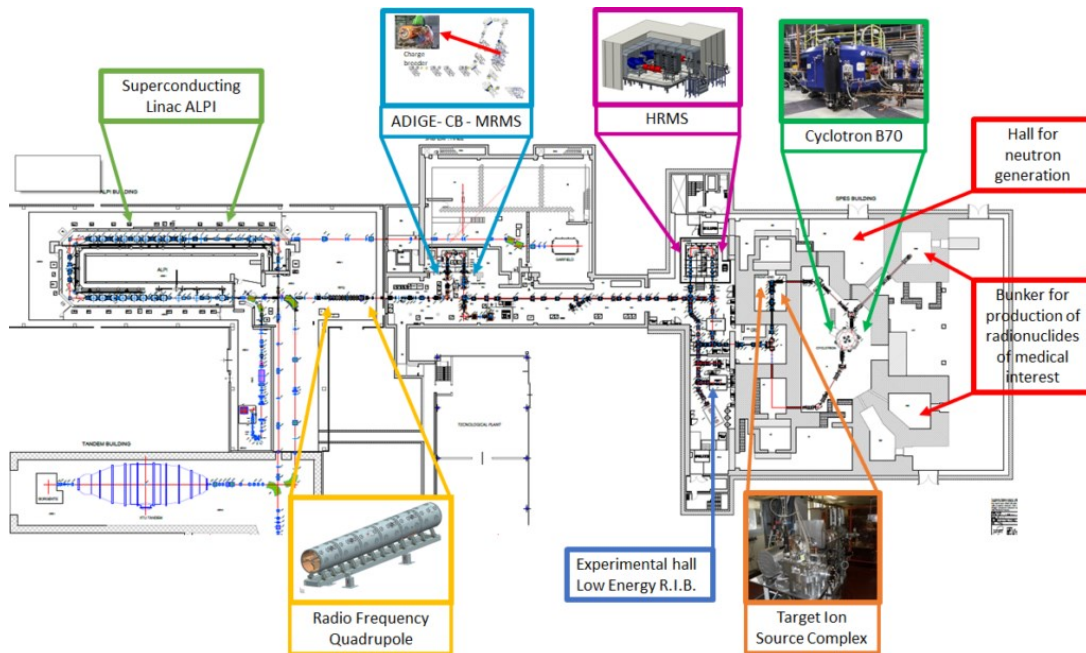
Ministero  
dell'Università  
e della Ricerca



Italiadomani  
PIANO NAZIONALE  
DI RIPRESA E RESILIENZA



# SPES and the accelerator complex at Legnaro National Laboratories

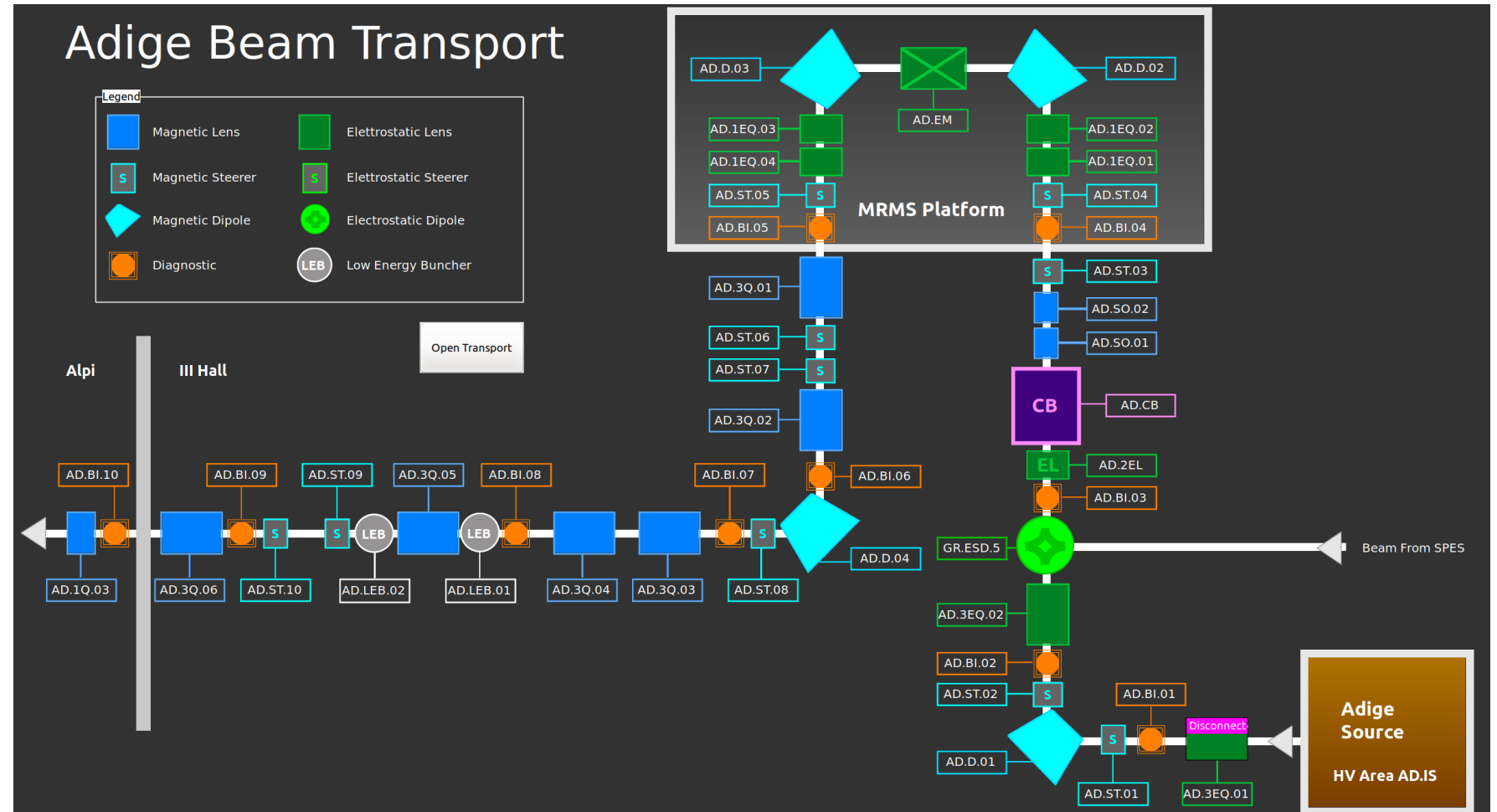




# Beam Dynamics Optimization

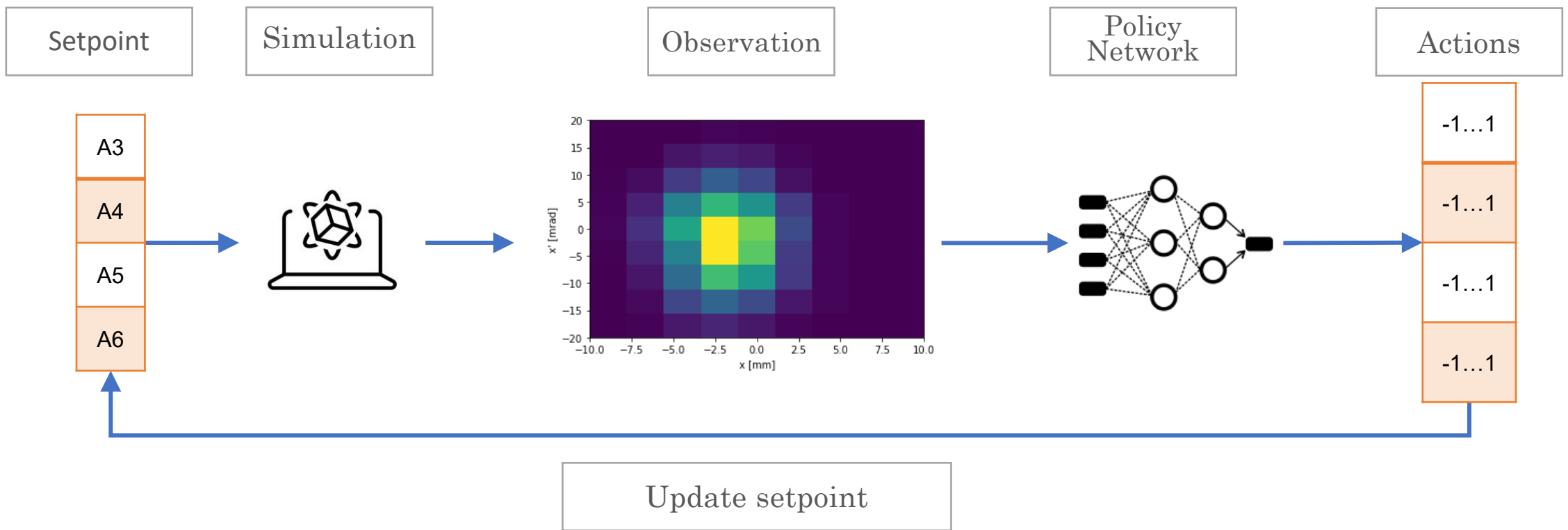
Find the optimal setpoint of all the control system variables on a beam line

Use case: SPES@LNL



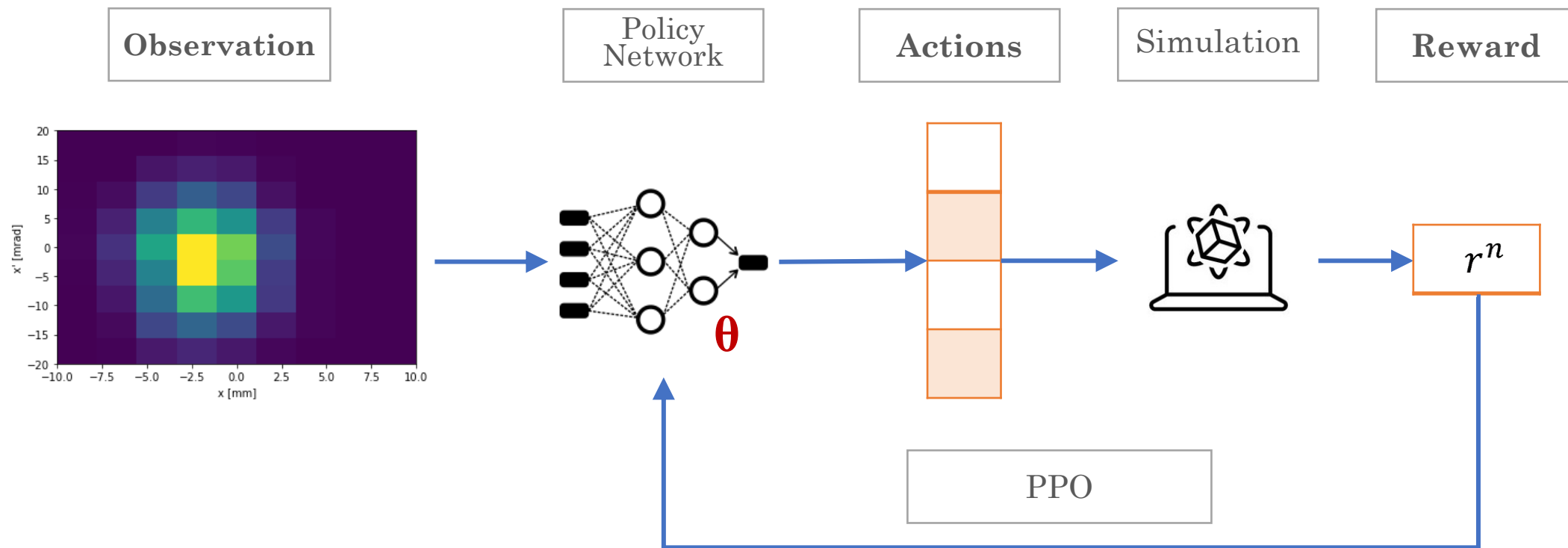


# Reinforcement Learning for Beam Dynamics Optimization





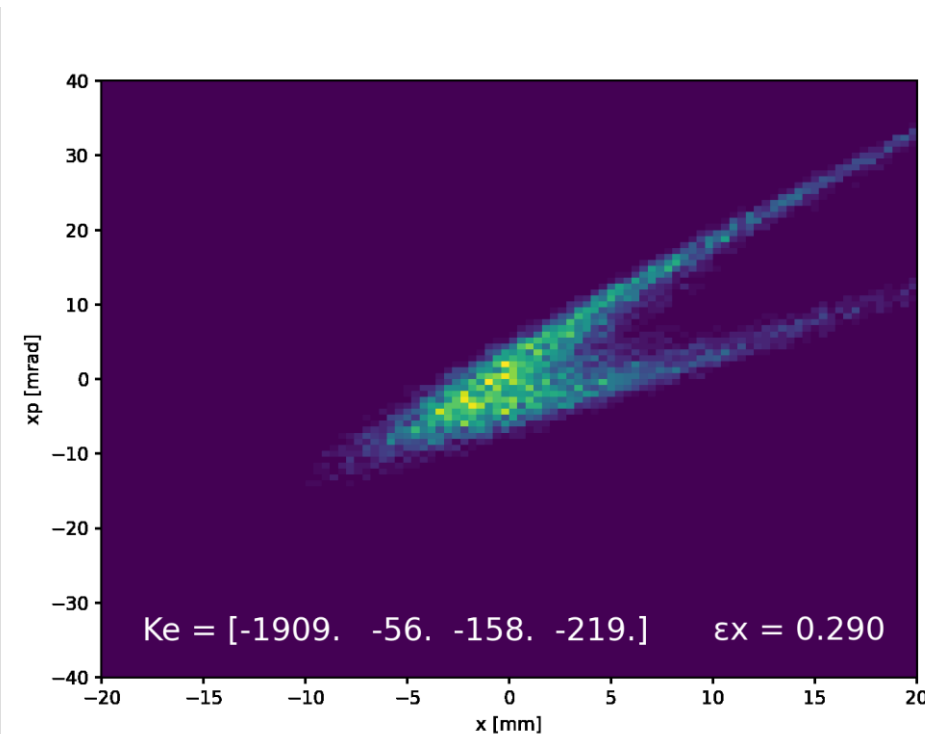
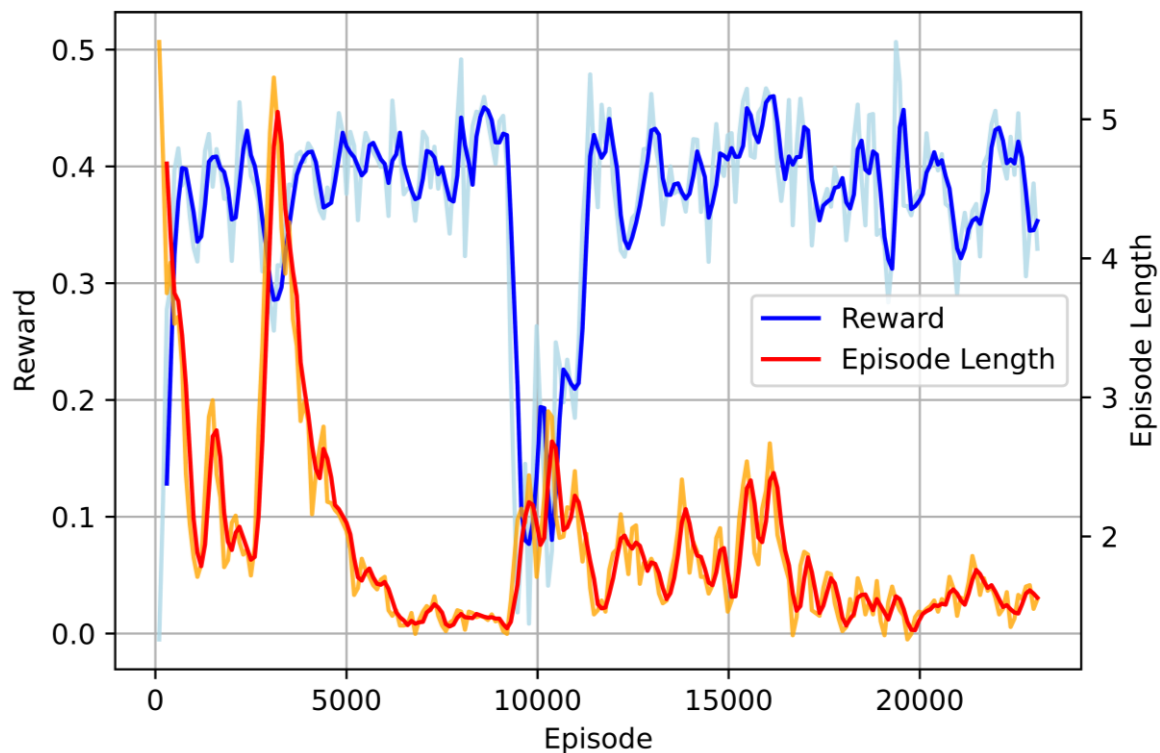
# Reinforcement Learning Training





D. Marcatò et al., "Demonstration of Beam Emittance Optimization using Reinforcement Learning", in Proc. IPAC'23, Venice, Italy, May 2023, pp. 2881-2884. doi:10.18429/JACoW-IPAC2023-WEPA100

## Results



**Success rate: 97% Mean number of steps per episode: 2.2**



Finanziato  
dall'Unione europea  
NextGenerationEU



Ministero  
dell'Università  
e della Ricerca



Italiadomani  
PIANO NAZIONALE  
DI RIPRESA E RESILIENZA



## Similar Research Projects

### RL4AA Collaboration

- Reinforcement Learning for Autonomous Accelerator Collaboration

### Some example papers:

- *Learning to Do or Learning While Doing: Reinforcement Learning and Bayesian Optimisation for Online Continuous Tuning*
- *Ultra fast reinforcement learning demonstrated at CERN AWAKE*
- *Towards automatic setup of 18 MeV electron beamline using machine learning*
- *Orbit Correction Based on Improved Reinforcement Learning Algorithm*

Reinforcement Learning for Autonomous Accelerators **RL4AA**

# RL4AA'24 COLLABORATION WORKSHOP

5-7 February 2024  
Salzburg, Austria

**JOIN NOW**

**KEYNOTE  
SPEAKERS**

**Reinforcement learning** is a powerful learning paradigm of machine learning that holds a lot of promise for **particle accelerators**. Join us in our yearly workshop to learn, discuss, and network!

- Facility overview talks**  
Summaries from different particle accelerator facilities about their work in optimization and control
- Contributed talks & posters**  
Targeted and specialized talks and posters on different RL applications
- Student talks**  
Dedicated session for master and first years doctoral students to present their progress
- Programming tutorial**  
Python tutorial for advanced RL concepts

**Antonin Raffin**  
Research Engineer in Robotics and Machine Learning  
German Aerospace Center  
Maintainer of Stable Baselines3

**Felix Berkenkamp**  
Lead Research Scientist  
Bosch Center for AI  
Safe exploration

**CALL FOR ABSTRACTS  
IS OPEN!**

You can submit for:

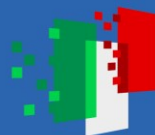
- Poster
- Facility talk
- Contributed talk
- Student talk

**Register here!**

**RL4AA'24 Organizing Committee**  
Simon Hirrlinger (University of Salzburg)  
Andrea Santomaria Garcia (Karlsruhe Institute of Technology)  
Annika Eicher (DESY)  
Sabrina Pochavez (University of Salzburg)  
Jan Kaiser (DESY)  
Chenran Xu (Karlsruhe Institute of Technology)

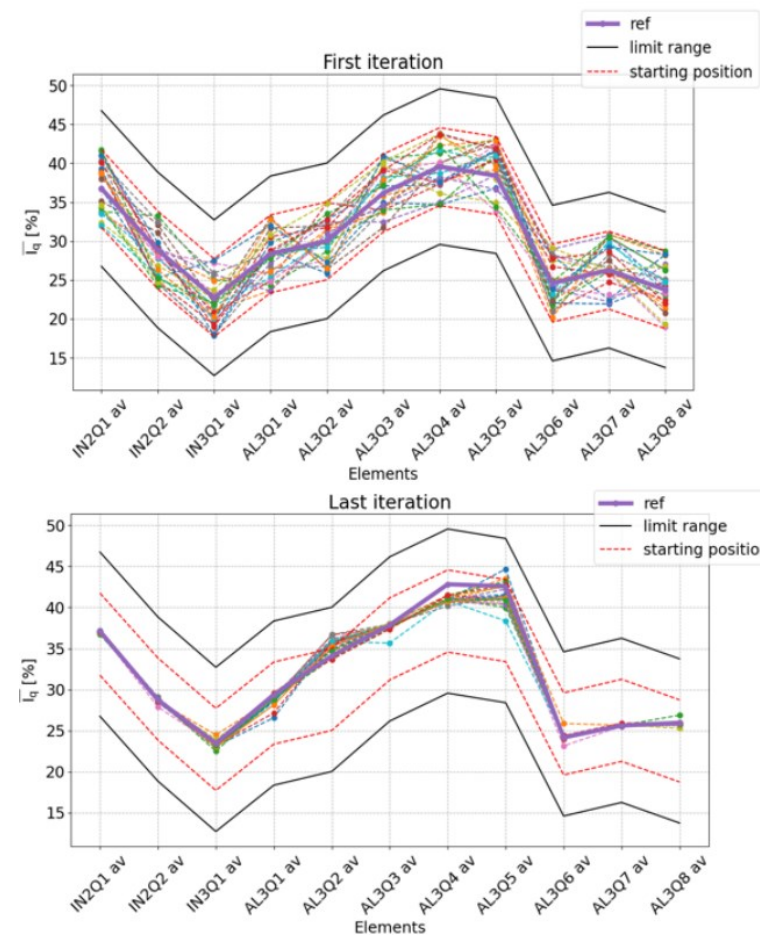
**Register here!**

<https://rl4aa.github.io/RL4AA24/>



## Other approaches

- Swarm optimization algorithms
- Bayesian optimization
- Deep Learning surrogate models of the beam line
- Luca Bellan et al., [New Techniques Method for Improving the Performance of the ALPI Linac](#), Published in: JACoW HB2023 (2024), FRA1C1 - DOI: 10.18429/JACoW-HB2023-FRA1C1
- Luca Bellan et. al., [New techniques for the LNL superconductive linac ALPI beam dynamics simulations and commissioning](#), Published in: JACoW IPAC2023 (2023), TUODA3 - DOI: 10.18429/JACoW-IPAC2023-TUODA3







Finanziato  
dall'Unione europea  
NextGenerationEU



Ministero  
dell'Università  
e della Ricerca



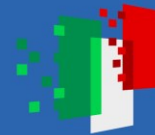
Italiadomani  
PIANO NAZIONALE  
DI RIPRESA E RESILIENZA



All these research projects require extensive use of  
beam dynamics simulations

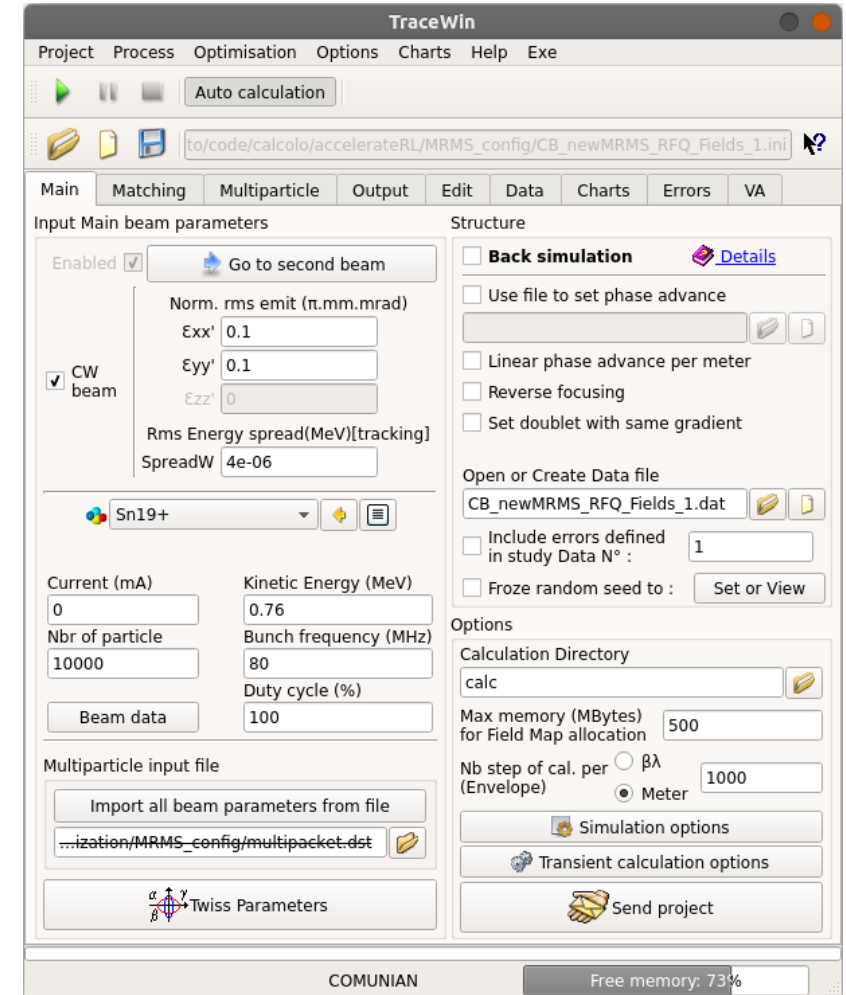
The background features a deep blue gradient. On the left side, there is a vertical axis of light trails and dots that create a sense of depth and motion, resembling a data stream or a digital tunnel. The trails are composed of many thin, parallel lines that converge towards a central point, with small, bright blue dots scattered along them. The overall effect is futuristic and high-tech.

**TraceWin**



## TraceWin Overview

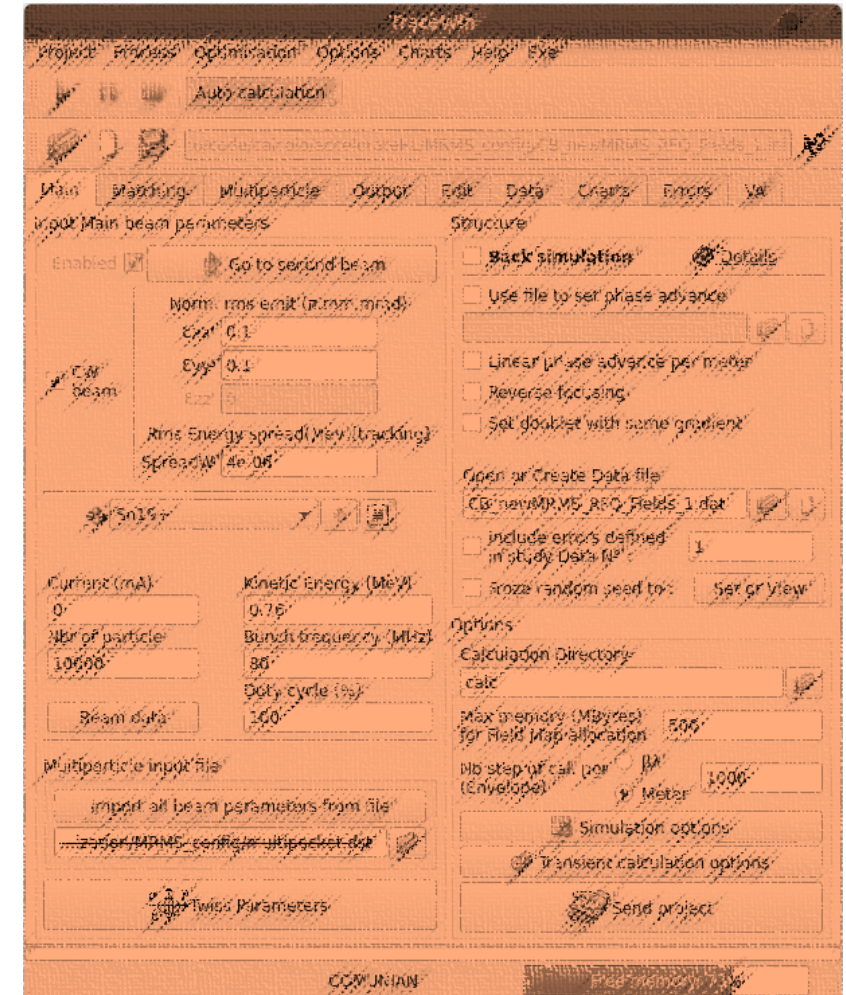
- Simulator of beam dynamics in particle accelerators
- Developed by CEA in France
- Second order momentum or/and by a macro-particle distribution
- Field maps
- GUI-first program (QT-based)
- Windows – Linux – MacOS
- Feature-rich, great physics engine
  - Many graphs available out-of-the-box
  - Some optimization algorithms
  - Error study
- Multicore computing
  - CPU only





## TraceWin shortcomings

- GUI-first
  - Command line interface available
  - Some properties only available on the GUI
- Not extensible
  - No API/library for any programming language
  - Input and outputs through files
- Licensed
- Closed source





## TraceWin Usage

- Configuration file ([project\\_xyz.ini](#)) contains all simulation parameters
  - Including the description of the beam line
  - Initially configured through GUI, then saved/loaded
- Three executables available (for linux)

Executable	GUI	Multicore	Paths
TraceWin	V	V	V
TraceWin_noX11	X	V	V
tracelx64	X	X	X

```

FIELD_MAP_PATH C:\Campi
uscita CB
DRIFT 1e-008 150 0 0 0
DRIFT 214 150 0 0 0
DRIFT 112 150 0 0 0
AD.SO.01 : SOLENOID 320 0.375536 65
DRIFT 425 150 0 0 0
DRIFT 308.5 150 0 0 0
AD.SO.02 : SOLENOID 320 0.134319 65
DRIFT 88.5 150 0 0 0
DRIFT 190.5 150 0 0 0

;THIN_STEERING 0 0 150 0 ; starebbe dentro la field map
DRIFT 200 150 0 0 0
AD.AT.01 : ELECTROSTA_ACC 120000 500 0.817421 50
DRIFT 200 150 0 0 0
DRIFT 42.5 150 0 0 0
DRIFT 42.5 150 0 0 0

Dia : DRIFT 1e-008 150 0 0 0
DRIFT 150 150 0 0 0
DRIFT 100 150 0 0 0
AD.ST.04 : THIN_STEERING 0 0 150 0
DRIFT 150 150 0 0 0
AD.1EQ.01 : QUAD_ELE 200 -9530.98 50 0 0 0 0 0
DRIFT 513.209 150 0 0 0
AD.1EQ.02 : QUAD_ELE 200 -2262.8 50 0 0 0 0 0
DRIFT 155.77 350 0 0 0
; mappa del dipolo 1
SUPERPOSE_MAP_OUT 1050 1050.0 0.0 0 0 -90
AD.D.02 : FIELD_MAP 70 1400 0 550 -0.200708 0 0 0 sd1b

DRIFT 4 350 0 0 0
; mappa multipolo è data da 4 mappe delle componenti sovrapposte:
superpose_map 0
AD.EM : FIELD_MAP 7 1200 0 150 0 -1230.72 0 0 MpoloV6b
superpose_map 0
AD.EM : FIELD_MAP 7 1200 0 150 0 -24.4137 0 0 MpoloV8b
superpose_map 0
AD.EM : FIELD_MAP 7 1200 0 150 0 -1.99093 0 0 MpoloV10b
superpose_map 0
AD.EM : FIELD_MAP 7 1200 0 150 0 -83.4286 0 0 MpoloV12b
DRIFT 4 350 0 0 0
; mappa del dipolo 2
SUPERPOSE_MAP_OUT 1050 1050.0 0.0 0 0 -90
AD.D.03 : FIELD_MAP 70 1400 0 550 0.200708 0 0 0 sd2b

DRIFT 4 350 0 0 0
DRIFT 151.77 350 0 0 0
AD.1EQ.03 : QUAD_ELE 200 -2262.8 50 0 0 0 0 0
DRIFT 513.209 150 0 0 0
AD.1EQ.04 : QUAD_ELE 200 -9530.98 50 0 0 0 0 0
DRIFT 50 150 0 0 0
DRIFT 50 150 0 0 0
DRIFT 50 150 0 0 0
AD.ST.05 : THIN_STEERING 0 0 150 0
DRIFT 50 150 0 0 0
DRIFT 50 150 0 0 0
DRIFT 50 150 0 0 0
DRIFT 100 150 0 0 0
dia : DRIFT 1e-008 150 150 0 0
DIAG_TWISS 22 0.0001 0.3 0.0001 0.3
DIAG_WAIST 20 0.001
DIAG_EMIT 30 0.1
end
  
```



## TraceWin Usage

- Run the simulation
  - Execution time depends on complexity and length of beam line

```
$ TraceWin project.ini path_cal=outdir hide nbr_part1=10000 ...
```

- Read the output files from the output folder

Output file	Type	Description
partran1.out	csv	Main beam properties at each beam line element
part_dtl1.dst	binary	Distributions of beam properties at the end of the beam line
dtl1.plt	binary	Distributions of beam properties at each beam line element
Many more...		

<i>Ele[n][v]</i>	Change the v <sup>th</sup> parameter of for the n <sup>th</sup> element
<i>hide</i>	Hide the GUI, or cancel console output (no parameter)
<i>tab_file</i>	Save to file the data sheet at the end of calcul (only GUI version)
<i>Synoptic_file</i>	Save the geometric layout at (entance (=1), middle (=2), exit (=3) of elements. (See "Synoptic" tools for file name) .
<i>nbr_thread</i>	Set the max. number of core/thread used
<i>path_cal</i>	Calculation directory
<i>dat_file</i>	Full name of structure file
<i>dst_file1</i>	Full name Input dst of main beam (*)
<i>dst_file2</i>	Full name Input dst of second beam (*)
<i>current1</i>	Input beam current (mA) of main beam
<i>current2</i>	Input beam current (mA) of second beam
<i>nbr_part1</i>	Number of particle of main beam
<i>nbr_part2</i>	Number of particle of second beam
<i>energy1</i>	Input kinetic energy (MeV) of main beam
<i>energy2</i>	Input kinetic energy (MeV) of second beam
<i>etnx1</i>	Input XX' emittance (mm.mrad) of main beam
<i>etnx2</i>	Input XX' emittance (mm.mrad) of second beam
<i>etny1</i>	Input YY' emittance (mm.mrad) of main beam
<i>etny2</i>	Input YY' emittance (mm.mrad) of second beam
<i>eln1</i>	Input ZZ' emittance (mm.mrad) of main beam
<i>eln2</i>	Input ZZ' emittance (mm.mrad) of second beam
<i>freq1</i>	Input beam frequency (MHz) of main beam
<i>freq2</i>	Input beam frequency (MHz) of second beam
<i>duty1</i>	Duty cycle of main beam
<i>duty2</i>	Duty cycle of second beam
<i>mass1</i>	Input beam mass (eV) of main beam
<i>mass2</i>	Input beam mass (eV) of second beam
<i>charge1</i>	Input particle charge state of main beam
<i>charge2</i>	Input particle charge state of second beam

TraceWin manual - CEA/SACLAY - DRF/Irfu/DACM - Didier URIOT

The background is a deep blue gradient. On the left side, there are numerous light trails and dots in shades of cyan and white, creating a sense of depth and movement, similar to a data visualization or a futuristic tunnel. The text 'pytracewin' is positioned on the right side of the image.

**pytracewin**



## Goal

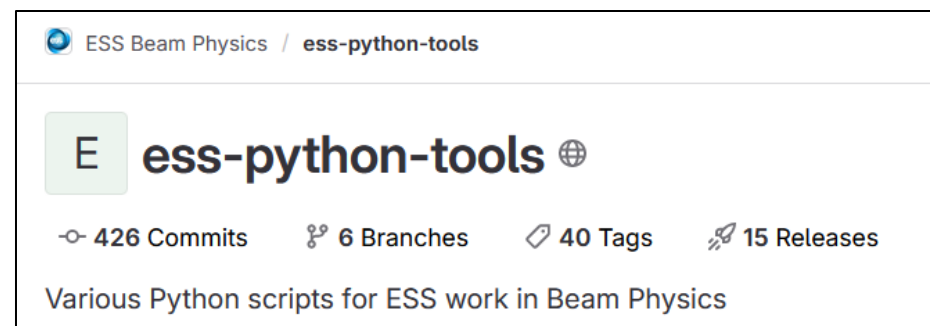
- Python wrapper for TraceWin executables
- Easy to use
- Run simulations with parameters
- Parsing of the output files
- Combine physics simulations with algorithms or ML libraries
- Do not require GUI by default
  - To be run on remote servers
- Easy to install via pip
  - Do not include proprietary TraceWin executables





## Implementation

- **TraceWin** class to interact with the simulator
- Using the **subprocess** package to run the shell command
  - Handle runtime parameters
  - Raise exception when simulation fails
  - Impose timeout to avoid blocking calls
- Using **ess-python-tools** package to decode binary output files
  - From European Spallation Source (ESS)
  - dst, plt
- Using **pandas** to parse the csv result file
- Hide mode
- Debug mode





## Usage

```
from pytracewin import TraceWin

tracewin = TraceWin("MRMS_config/CB_newMRMS_RFQ_Fields_1.ini",
                    executable="TraceWin",
                    hide=True, debug=False)

tracewin.run(params={"nbr_part1": 10000, "ele[27][6]": -1200},
             timeout=30)

tracewin.dst()
tracewin.plt()
tracewin.results()
```

✓ 11.6s

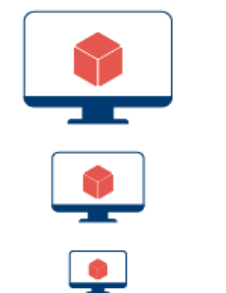
Python

##	z(m)	gama-1	x0	y0	p0	x'0	y'0	W0	SizeX	...
0	0	-0.000000	0.000006	1.245505e-08	4.161037e-08	-0.017999	5.401000e-09	1.686846e-07	-5.822923e-08	4.475022 ...
1	1	0.000000	0.000006	1.245505e-08	4.161037e-08	-0.017999	5.401000e-09	1.686846e-07	-5.822923e-08	4.475022 ...
2	2	0.214000	0.000006	1.361086e-08	7.770887e-08	2.677506	5.401000e-09	1.686846e-07	-5.822923e-08	8.968381 ...

## Scaling to the Cloud

- pytracewin lets you run simulations from python
- This is useful to test optimization algorithms
- The limiting factor becomes the computing power of your local machine
- You can run them on *fat* servers
  - But this doesn't scale much
- Often the simulations can be run independently between one another
  - Embarrassingly parallel workload
- Scale to the cloud
  - You need a coordination mechanism
  - Send jobs and retrieve results
  - Simple user interface
  - Backend-agnostic

Vertical Scaling  
Increase size of instance



Horizontal Scaling  
Add more instances

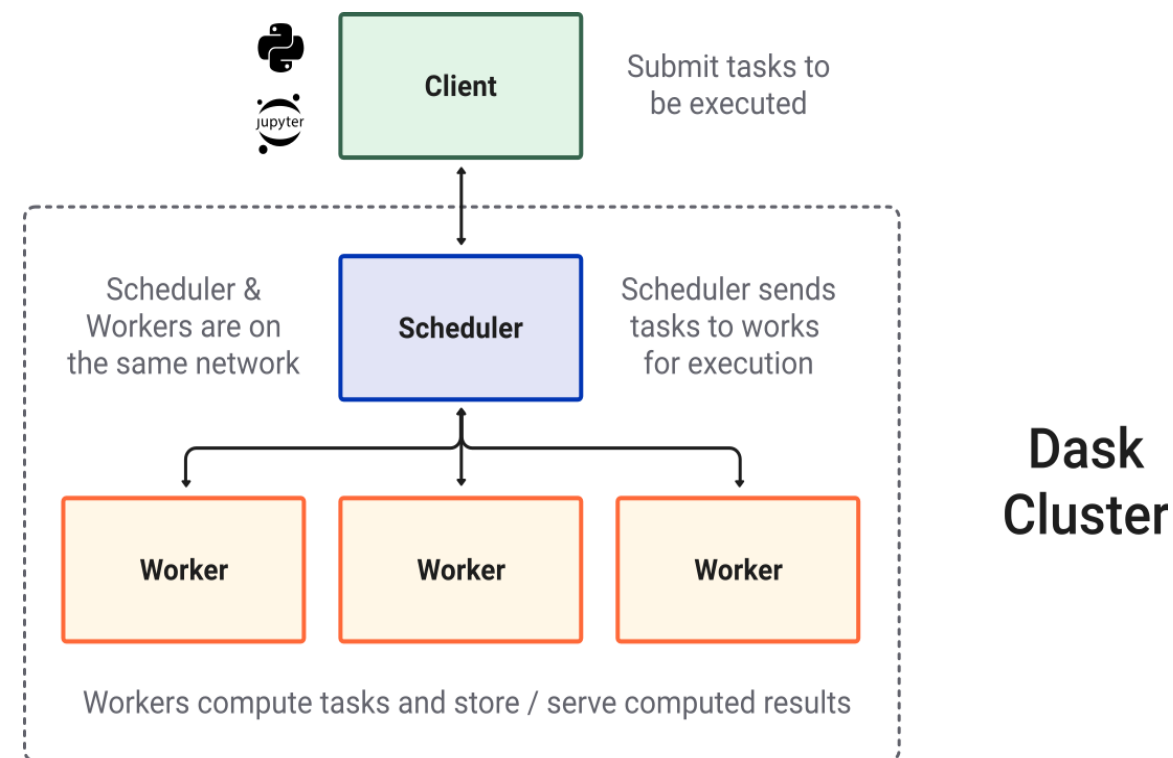




**Dask**

## Dask Cluster

- Dask is an open-source Python library for parallel computing
- Can be deployed anywhere:
  - Local, ssh
  - Cloud (AWS, GCP, Azure)
  - SLURM
  - Kubernetes
- Dask is a task scheduler
- Enables embarrassingly parallel computing
- Works on native python data types
- Graph task lazily evaluated
- Diagnostic dashboard



## Dask Cluster with SSH

SSHCluster: solution for manually managed machines on a local network

- Deploys a Dask Scheduler and Workers on the set of provided machines

We tested it on a couple of local VMs

- Validate the functionality of Dask+pytracewin

Does not scale easily

- All the VMs must be pre-configured with the environment for TraceWin
- Hard to deploy and maintain

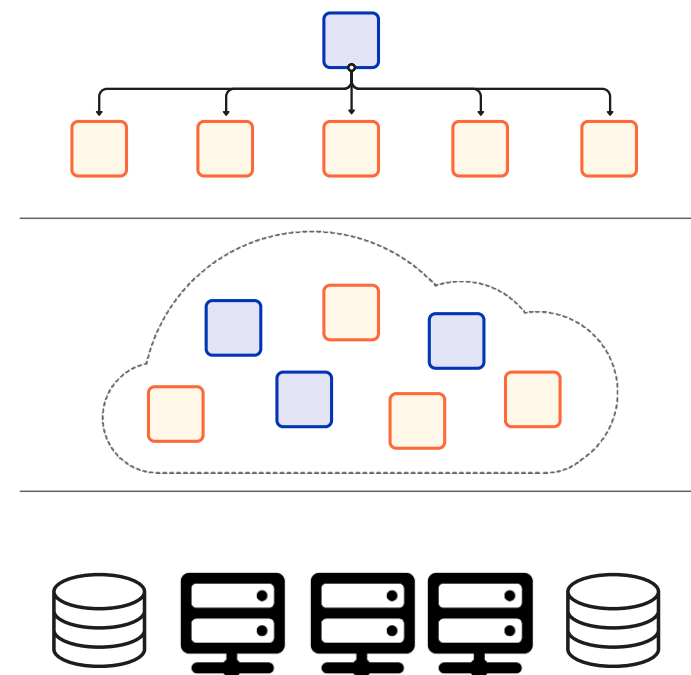


## Dask Cluster on Kubernetes

- Automatic setup
  - Service described by YAML files
- **Scale** to a large number of worker nodes
  - Dynamic cluster
- Exploit the available service on INFN-Cloud
- Common interface
  - Do not depend on the VM provider
  - **Portable**
- Docker image with TraceWin environment



kubernetes





## Custom Docker image

- Derived from *dask:latest*
  - Includes python, dask, conda
- Adding **USER**
  - Required by the software license
- Adding headless **tracelx64** executable
- Installing **pytracewin**
- 350MB

```
1 FROM ghcr.io/dask/dask:latest
2
3 ARG USER
4
5 RUN groupadd -g 1000 ${USER} && \
6     useradd -ms /bin/bash -g ${USER} -u 1000 ${USER}
7
8 COPY TraceWin/tracelx64 /usr/local/bin/tracelx64
9
10 USER ${USER}
11 WORKDIR /home/${USER}
12
13 COPY --chown=${USER}:${USER} pytracewin accelerateRL/pytracewin
14 COPY --chown=${USER}:${USER} pyproject.toml accelerateRL/pyproject.toml
15
16 RUN pip install --no-cache-dir --upgrade pip && \
17     pip install --no-cache-dir ./accelerateRL
```

```
$ docker build -t baltig.infn.it:4567/marcato/accelerated-rl/dask_tracewin:1.1.0  
--build-arg USER=marcato
```

```
$ docker push baltig.infn.it:4567/marcato/accelerated-rl/dask_tracewin:1.1.0
```





## Dask Operator

A set of custom resources and a controller that runs on your Kubernetes cluster and allows you to create and manage your Dask clusters as Kubernetes resources

```
$ helm install --repo https://helm.dask.org --create-namespace -n dask-operator  
--generate-name dask-kubernetes-operator
```

```
> █
```

## Dask Cluster definition

### dask\_cluster.yaml

- Number of workers
- Custom image to be used
  - Credentials to download the custom image
- CPU/memory requests and limits
  - Horizontal scaling
- nthreads=1
  - Force a single TraceWin execution per container
  - Avoid conflicts on I/O files

```
1 # dask_cluster.yaml
2 apiVersion: kubernetes.dask.org/v1
3 kind: DaskCluster
4 metadata:
5   name: accelerate-rl
6 spec:
7   worker:
8     replicas: 40
9     spec:
10      containers:
11      - name: worker
12        image: "baltig.infn.it:4567/marcat0/accelerated-rl/dask_tracewin
13        imagePullPolicy: "IfNotPresent"
14        args:
15        - dask-worker
16        - --name
17        - ${DASK_WORKER_NAME}
18        - --nthreads
19        - "1"
20        - --dashboard
21        - --dashboard-address
22        - "8788"
23        ports:
24        - name: http-dashboard
25          containerPort: 8788
26          protocol: TCP
27        resources:
28          limits:
29            cpu: "1"
30            memory: "2Gi"
31          requests:
32            cpu: "0.8"
33            memory: "1Gi"
34        imagePullSecrets:
35        - name: regcred
```



## Dask Cluster definition

### dask\_cluster.yaml

- Dask Scheduler
- Same docker image as the workers
- Scheduler and Dashboard port 8786, 8787
- *Fat* scheduler pod
- Service NodePort for connectivity to the scheduler

```
$ kubectl apply -f dask_cluster.yaml
$ kubectl port-forward svc/accelerate-rl-scheduler 8786:8786
$ kubectl port-forward svc/accelerate-rl-scheduler 8787:8787
```

```
6 spec:
36 scheduler:
37   spec:
38     containers:
39     - name: scheduler
40       image: "baltig.infn.it:4567/marcato/accelerated-rl/dask_tracewin
41       imagePullPolicy: "IfNotPresent"
42     args:
43     - dask-scheduler
44     ports:
45     - name: tcp-comm
46       containerPort: 8786
47       protocol: TCP
48     - name: http-dashboard
49       containerPort: 8787
50       protocol: TCP
51     readinessProbe:
52     httpGet:
53     port: http-dashboard
54     path: /health
55     initialDelaySeconds: 5
56     periodSeconds: 10
57     livenessProbe:
58     httpGet:
59     port: http-dashboard
60     path: /health
61     initialDelaySeconds: 15
62     periodSeconds: 20
63     resources:
64     limits:
65     cpu: "8"
66     memory: "16Gi"
67     requests:
68     cpu: "7.5"
69     memory: "14Gi"
70     imagePullSecrets:
71     - name: regcred
72
73   service:
74     type: NodePort
75     selector:
76     dask.org/cluster-name: accelerate-rl
77     dask.org/component: scheduler
78     ports:
79     - name: tcp-comm
80     protocol: TCP
81     port: 8786
82     targetPort: "tcp-comm"
83     - name: http-dashboard
84     protocol: TCP
85     port: 8787
86     targetPort: "http-dashboard"
```



## Basic Usage

```
client = Client("tcp://localhost:8786")

# Upload project file and license key
client.register_plugin(UploadDirectory("MRMS_config"))

def simulate(*args):
    tracewin = TraceWin("/tmp/dask-scratch-space/MRMS_config/CB_newMRMS_RFQ_Fields_1.ini")
    completed_proc = tracewin.run(*args)
    results = tracewin.results()
    return [completed_proc, results]
```

[ ]

```
# Run the simulation
client.submit(simulate, {"ele[26][6]": -1400}).result()
```

[4] ✓ 11.9s

... [CompletedProcess(args=['tracelx64', 'CB\_newMRMS\_RFQ\_Fields\_1.ini', 'hide', 'ele[26][6]=-1400'], returncode=0),

	##	z(m)	gama-1	x0	y0	p0	\
0	0	-0.000000	0.000006	2.896655e-09	-2.276210e-08	-0.017999	
1	1	0.000000	0.000006	2.896655e-09	-2.276210e-08	-0.017999	
2	2	0.214000	0.000006	1.181234e-08	-5.706434e-08	2.677344	
3	3	0.326000	0.000006	1.647850e-08	-7.501692e-08	4.087991	
4	4	0.646000	0.000006	-7.434411e-08	1.198164e-07	4.088095	

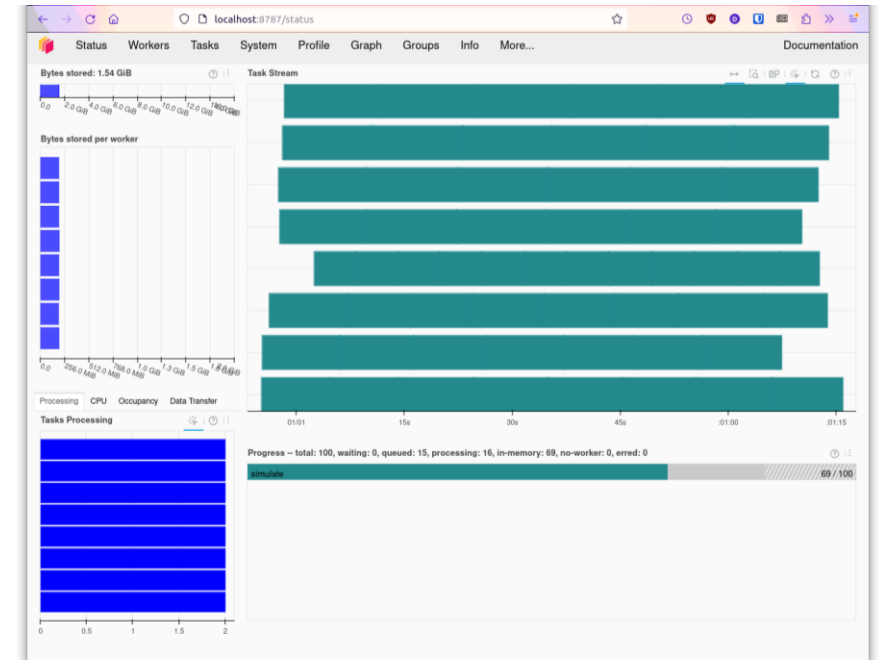
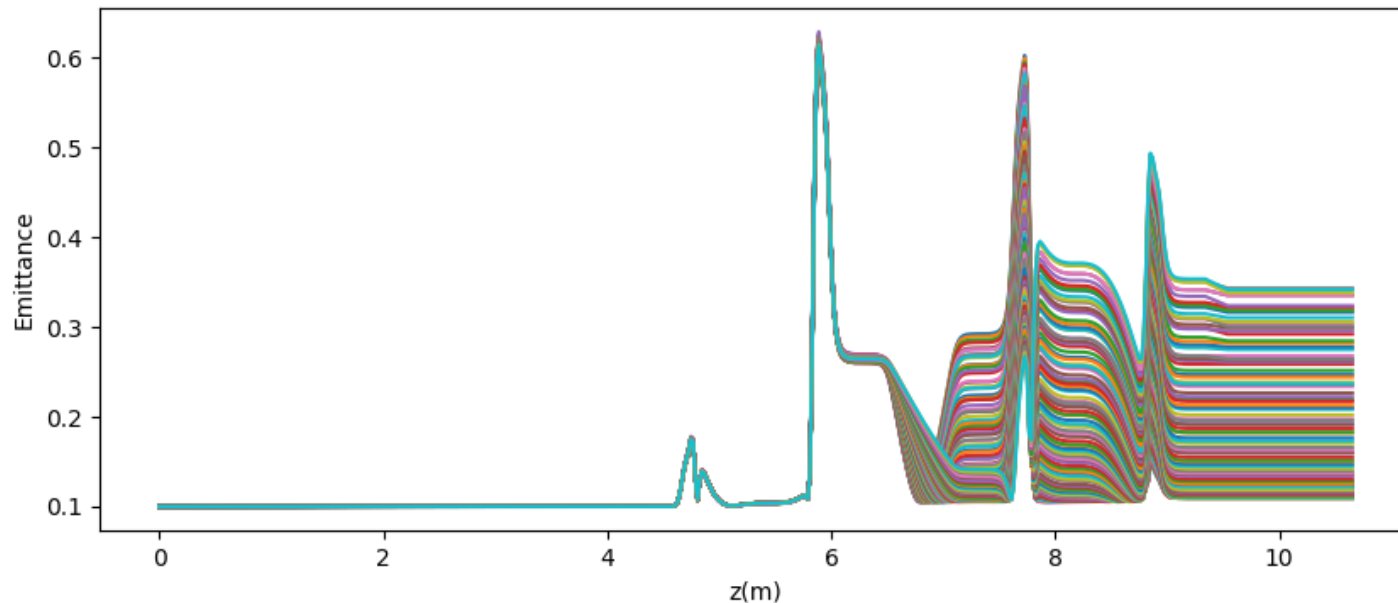


## Parallel job submission

```
runs = []
for i in range(100):
    runs.append(client.submit(simulate, {"ele[26][6]": -1400+i*10}))

data = client.gather(runs, errors="skip")
```

[15] ✓ 54.9s



More advanced Dask API can be used

`dask.DataFrame`

The background is a deep blue gradient. On the left side, there are numerous thin, curved lines of light that appear to be data paths or fiber optic cables, converging towards the center. Interspersed among these lines are small, bright blue dots of varying sizes, some of which are slightly blurred, giving a sense of depth and motion. The overall effect is a futuristic, high-tech aesthetic.

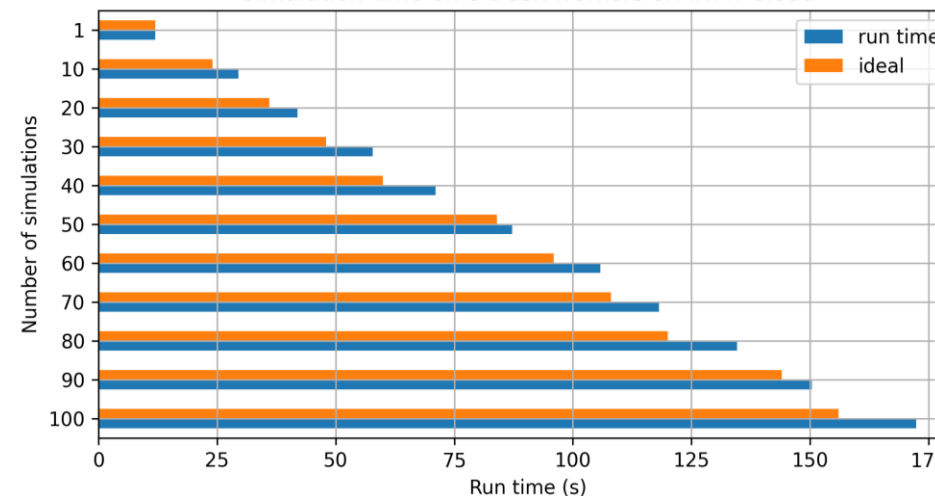
**Testing:  
INFN Cloud  
CloudVeneto**



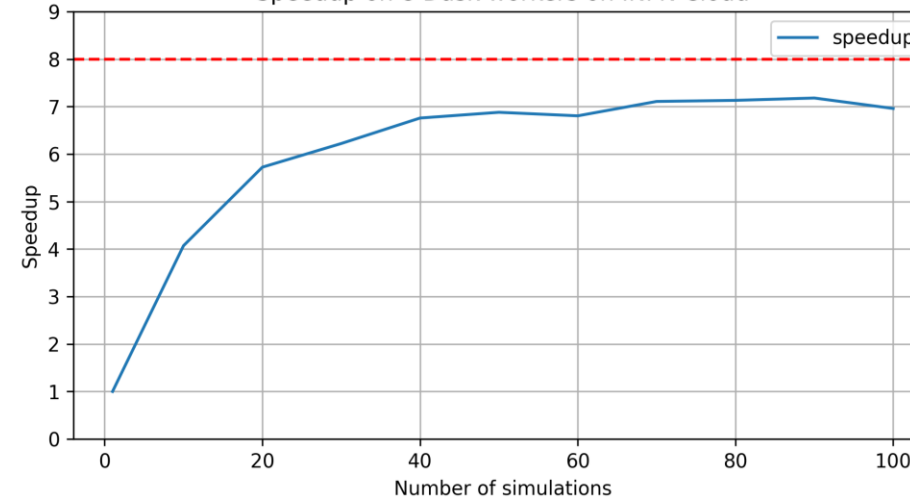
## INFN Cloud

- Using Kubernetes-as-a-Service
  - Deployed on RECAS-BARI
  - Resources from the admin/catchall group
  - 1 k8s Master (4CPU, 8GB)
  - 4 k8s Nodes (2CPU, 4GB)
  - Open ports 8786, 8787
- Dask Cluster
  - 1 Scheduler (4CPU, 8GB)
  - **8 Workers** (1CPU, 1GB)
- Simulating ADIGE beam line @ LNL
  - From charge breeder to emittance meter
  - Including MRMS

Simulation time on 8 Dask workers on INFN Cloud



Speedup on 8 Dask workers on INFN Cloud





## CloudVeneto CaaS

- Managed k8s cluster
  - Avoid instantiation/maintenance
- Large pool of resources available to us via SPES project
- Thanks to Lisa Zangrando for the support
  - Helm operator installation
- Using OSNodes on openstack
  - VMs using resources from own tenant quota

```
kubectl apply -f osnodes.yaml
```

```
1  ---
2  apiVersion: osnode.infn.it/v1
3  kind: OpenStackNode
4  metadata:
5  |   name: k8s-osn-00
6  spec:
7  |   flavor: cloudveneto.xlarge
8  |   keyPair: pc1815
9  |   policy: shared
10 ---
11 apiVersion: osnode.infn.it/v1
12 kind: OpenStackNode
13 metadata:
14 |   name: k8s-osn-01
15 spec:
16 |   flavor: cloudveneto.medium
17 |   keyPair: pc1815
18 |   policy: shared
```

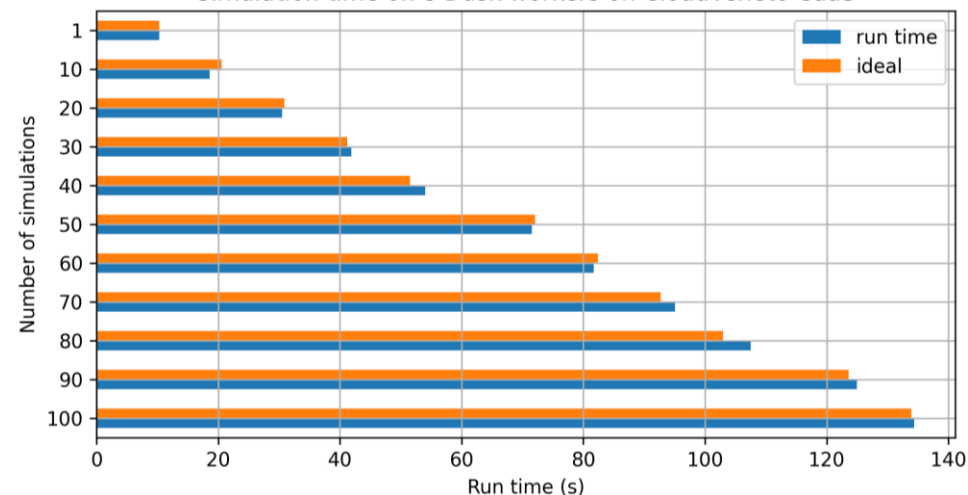




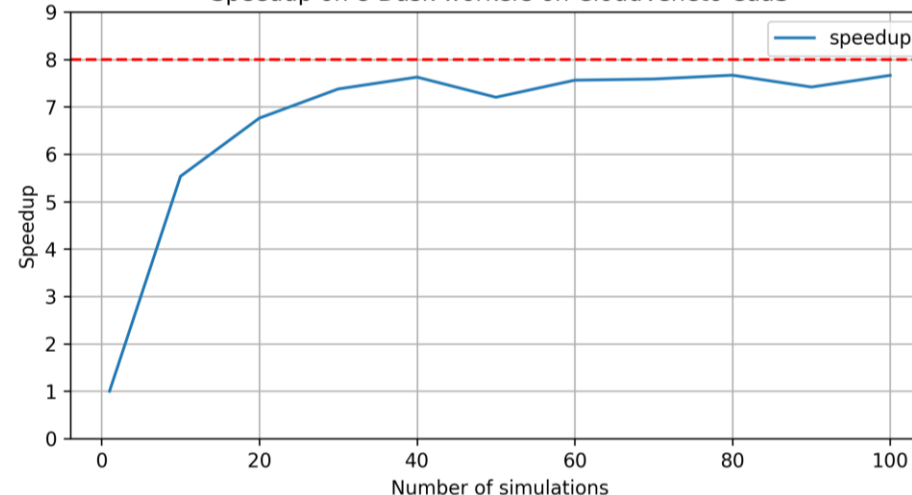
## CloudVeneto

- CaaS cluster
  - 1 OSN xlarge (8CPU, 16GB)
  - 30 OSN medium (2CPU, 4GB)
- Dask Cluster
  - 1 Scheduler (8CPU, 16GB)
  - **Up to 40 Workers** (1CPU, 1GB)
- Limited by the memory on the scheduler
  - Used to distribute the input files

Simulation time on 8 Dask workers on CloudVeneto CaaS

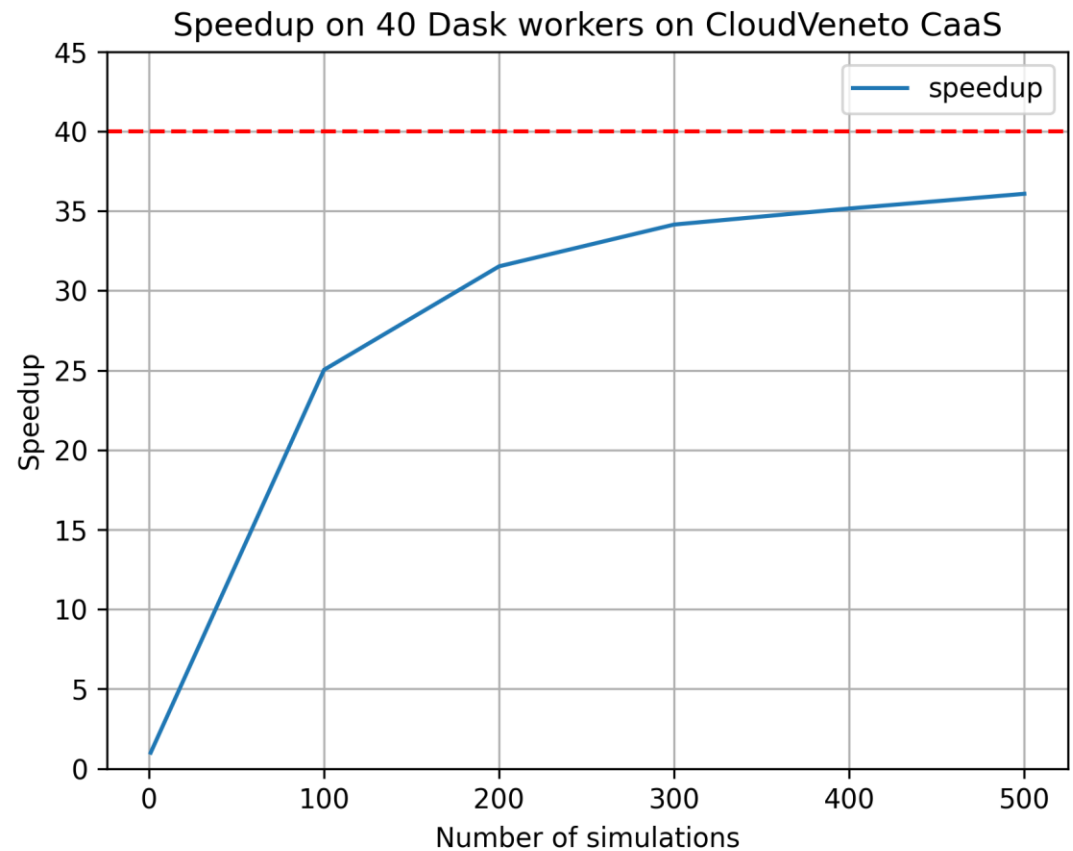
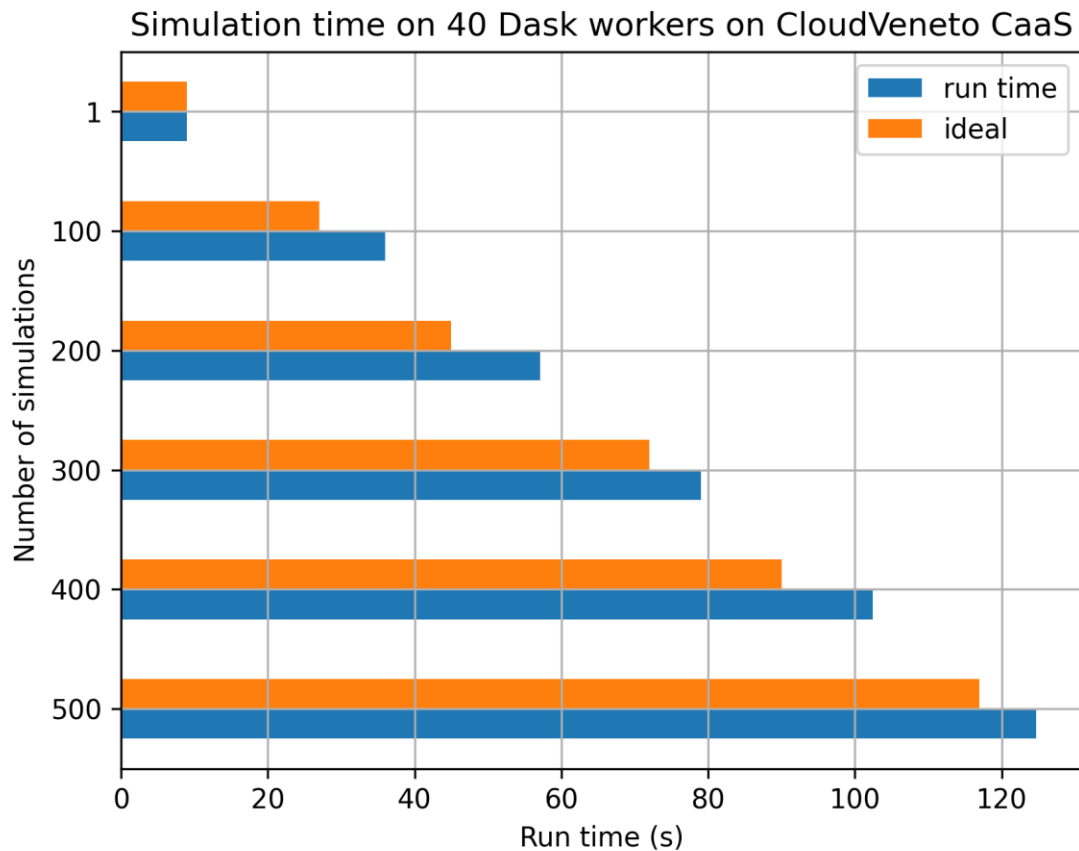


Speedup on 8 Dask workers on CloudVeneto CaaS





## Scaling to 40 workers



The background features a vibrant blue gradient. On the left side, there are numerous light trails and dots in shades of cyan and white, creating a sense of depth and movement, reminiscent of a data visualization or a futuristic tunnel. The right side is a solid, deep blue.

**Conclusion**



## What's next

- Manage distribution of I/O files
  - Kubernetes volumes
  - In-memory volumes to get fast I/O
- Improve accessibility for end users
  - Dask Cluster as a service?
  - Authentication?
- Better debug messages,
  - e.g. failed computation for TraceWin
- Being used in combination with off-policy RL algorithms!



The background is a deep blue gradient. On the left side, there are numerous bright blue light trails and particles that appear to be moving towards the center, creating a sense of depth and motion. The trails are composed of many small, bright blue dots connected by thin, glowing lines. The overall effect is reminiscent of a data visualization or a futuristic digital environment.

**Thank You**

[davide.marcato@lnl.infn.it](mailto:davide.marcato@lnl.infn.it)

[daniel.lupu@lnl.infn.it](mailto:daniel.lupu@lnl.infn.it)