ADMIRE
malleable data solutions for HPC

Adaptive multi-tier intelligent data manager for Exascale

EuroHPC Joint Undertaking

admire-eurohpc.eu

The CAPIO Middleware

Massimo Torquati
CINI - University of Pisa
massimo.torquati@unipi.it

ADMIRE User Day

12 December, BSC, Barcelona, Spain

cini consorzio interuniversitario nazionale per l'informatica

Università di Pisa

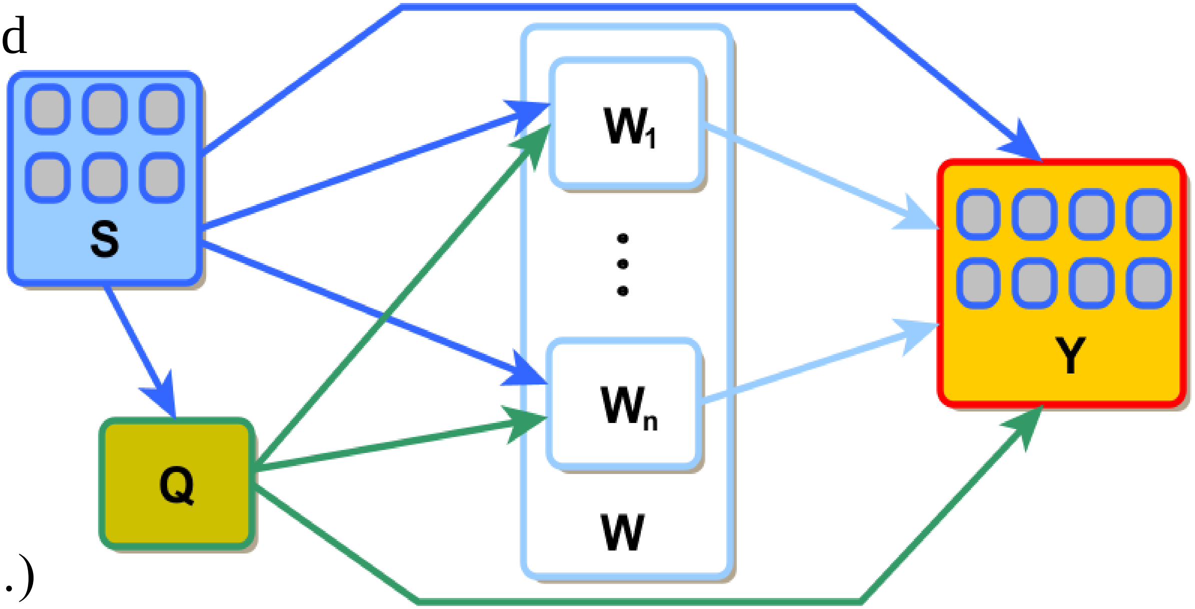
Grant Agreement number: 956748 — ADMIRE — H2020-JTI-EuroHPC-2019-1

Outline

- ❑ Context: workflows, traditional vs in situ
- ❑ Why CAPIO?
- ❑ CAPIO features and its software architectures
- ❑ Evaluation through simple I/O benchmarks and the WRF-Visualization workflow part of the ENVapp
- ❑ Conclusions

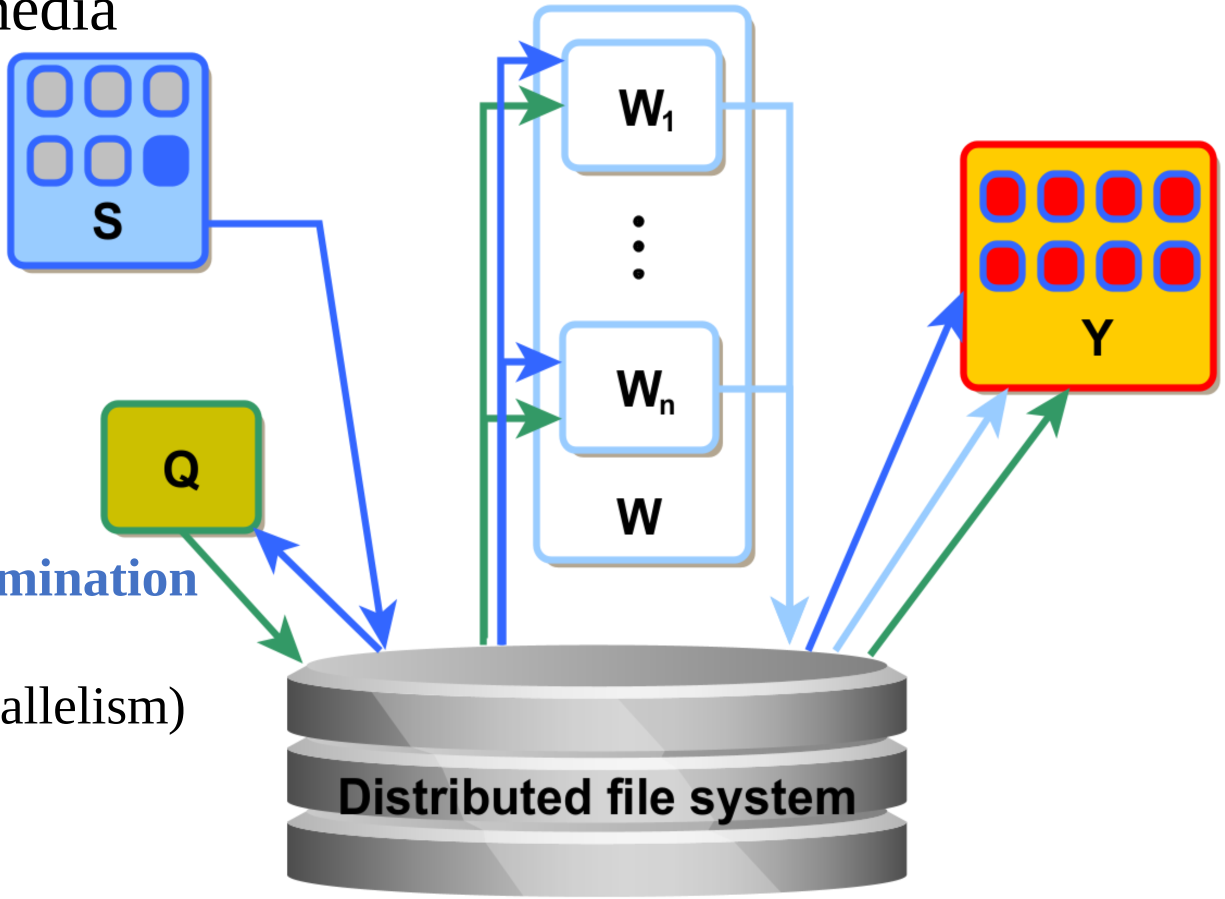
Scientific Workflow in a nutshell

- ❑ Workflows are typically structured as DAGs of SW components (called jobs, tasks or steps)
  - Workflow Management Systems (WMSs) deploy and coordinate the execution of ready steps
- ❑ S, Q, W, and Y are workflow steps
  - S and Y are internally parallel (e.g., MPI+OpenMP)
  - W is sequential, but it can be replicated
  - Q is inherently sequential
- ❑ Arrows are data dependencies
  - W needs data from S and Q to start executing
  - Several **data models** available (e.g., HDF5, VTK, …)
  - Several **data transports** options (e.g., files, direct messaging, data staging)
  - Multiple back-ends for data communication (e.g., ADIOS2)



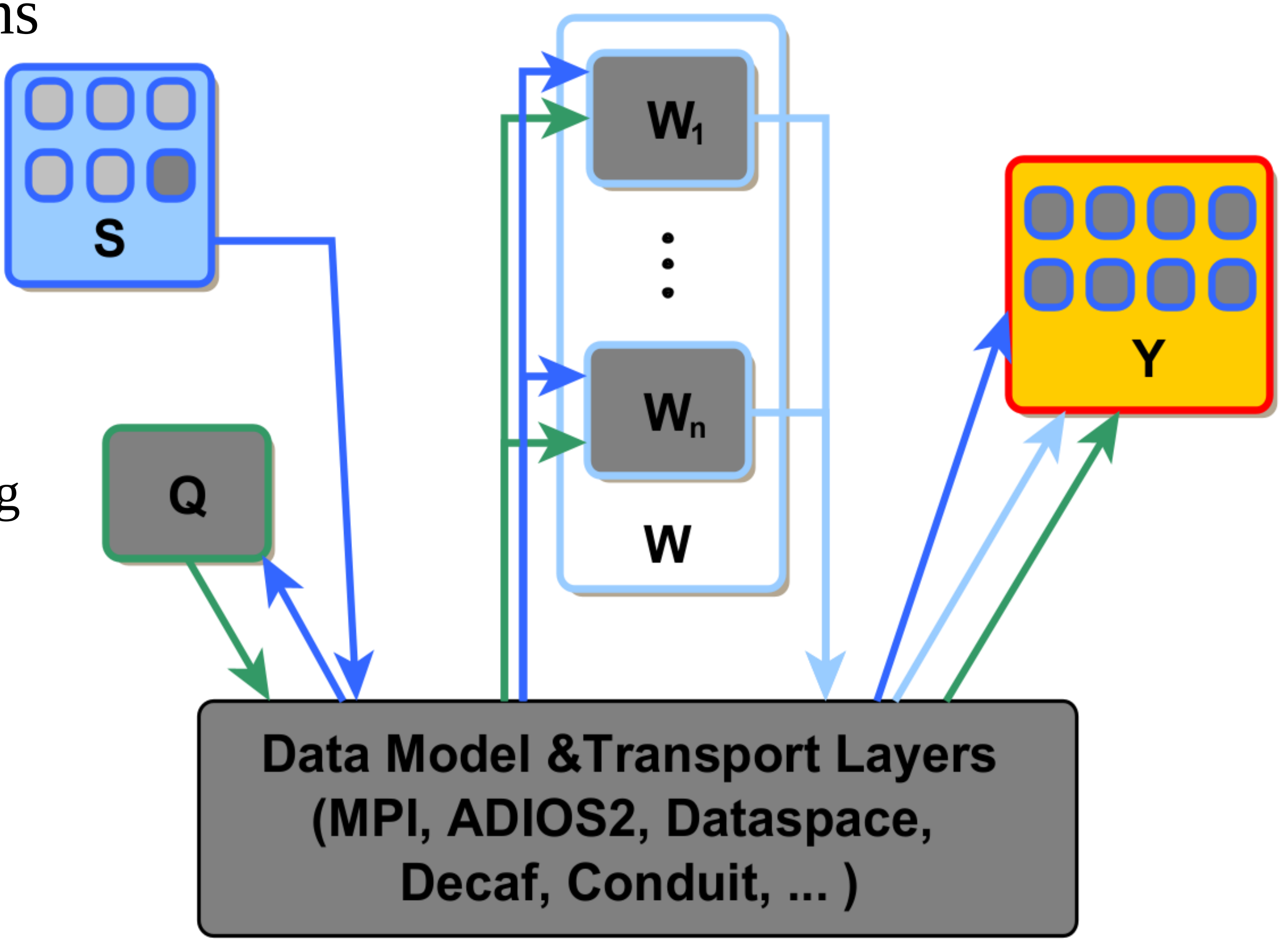
Traditional scientific workflows

- ❑ The distributed FS acts as a communication media
- ❑ WMSs enforce data dependencies leveraging **files** for data transport
- ❑ Pros:
  - Portable data transport
  - No changes to the user's code when moving from standalone to workflow execution
- ❑ Cons:
  - The semantics for synchronization is strict: **on-termination** (e.g., before starting Q, S must be finished)
  - No data streaming between steps (i.e., pipeline parallelism) even if it would be possible
  - The DFS may limit the performance



In situ scientific Workflows

- ❑ **In situ** workflow bypasses the DFS in favor of faster in-memory or network communications (i.e., direct messaging, data staging)
- ❑ Pros:
  - Overlapping computation with I/O
  - Possibility to use more efficient data format
  - Moving only needed data between steps
  - In transit computation (e.g., compression)
  - Enabling pipeline parallelism (i.e., data streaming between steps)
- ❑ Cons:
  - **Need to modify and adapt the steps of the workflow**


S
Q
W_1
W_n
W
Y
Data Model &Transport Layers (MPI, ADIOS2, Dataspace, Decaf, Conduit, ...)

In situ approach to Workflows

- ❑ In situ approach to Workflows is the best from the performance standpoint
  - Many tools developed in the last years (e.g., ADIOS2, Conduit, ParaView Catalyst, DataSpace, Bredala)

- ❑ However, **it is not always possible to rewrite/patch all (or some) workflow steps**
  - Legacy modules
  - Not enough expertise
  - Modifications not accepted in the master branch by application module maintainers
  - Different in situ tools used from distinct steps of the same workflow
  - …..

What can we do in traditional workflows to introduce (some of) in situ optimizations?

The CAPIO Middleware overview

- ❑ CAPIO: Cross-Applications Programmable I/O

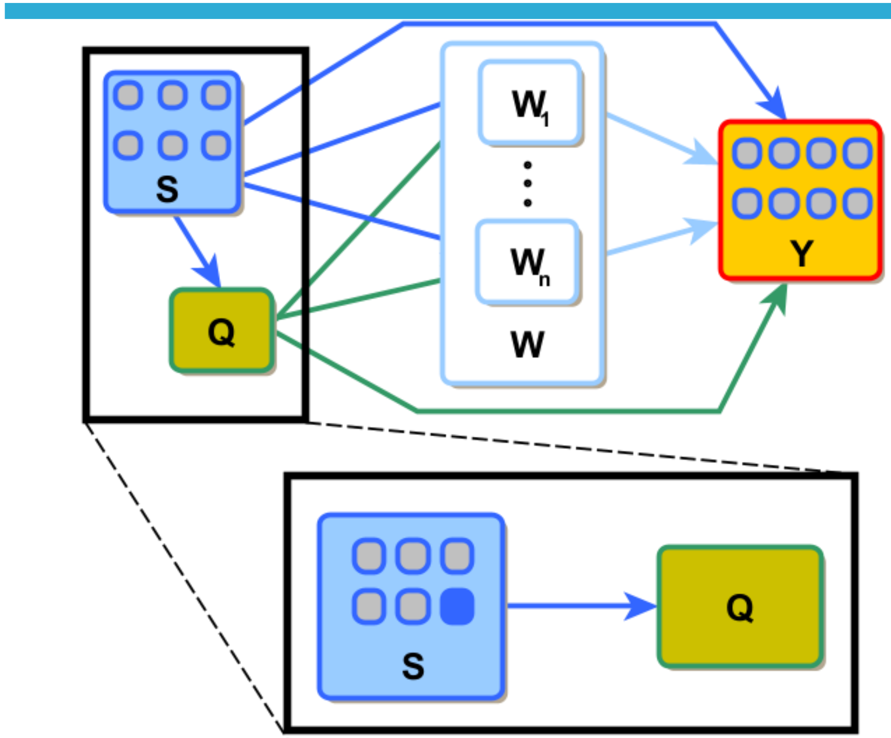
CAPIO
Cross Application Programmable I/O

- ❑ User-space **data transport middleware** enabling I/O coordination in scientific Workflows
- ❑ Pros:
  - ➢ No need to modify the workflow’s steps
  - ➢ POSIX I/O System Calls (read/write/seek/stat/…) are transparently intercepted using dynamic linker features (i.e., LD_PRELOAD)
    - ❖ Current version limitations: no file locking nor mmap/munmap support
- ❑ Cons: no visibility of high-level metadata annotations (i.e., file data structure)
- ❑ **The producer-consumer synchronization semantics related to files is controlled by the user through a separate coordination language (JSON file)**
- ❑ CAPIO can be coupled with existing WMSs to enable data streaming in file data movement
  - we are working to integrate CAPIO with StreamFlow

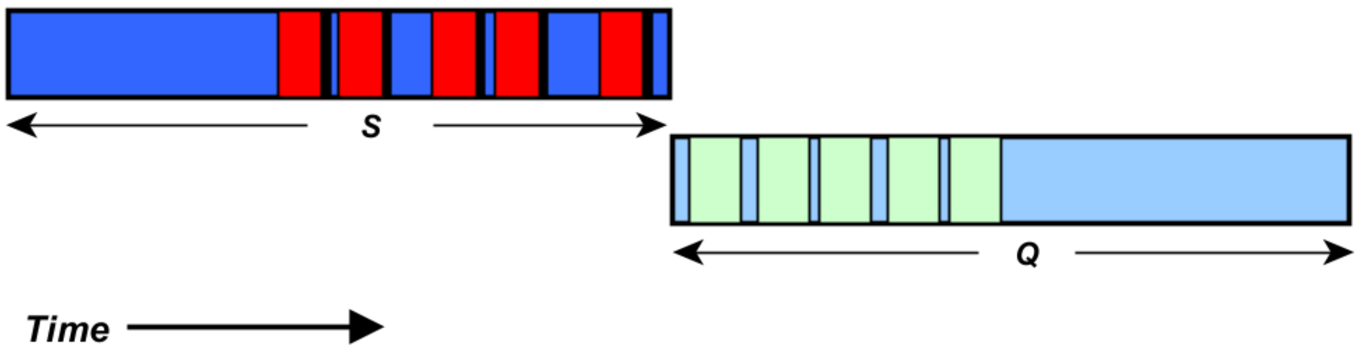
StreamFlow
Towards Cloud-HPC Continuum

Dagonstar

CAPIO: transparent in situ processing



Without CAPIO (batch execution of S and Q)

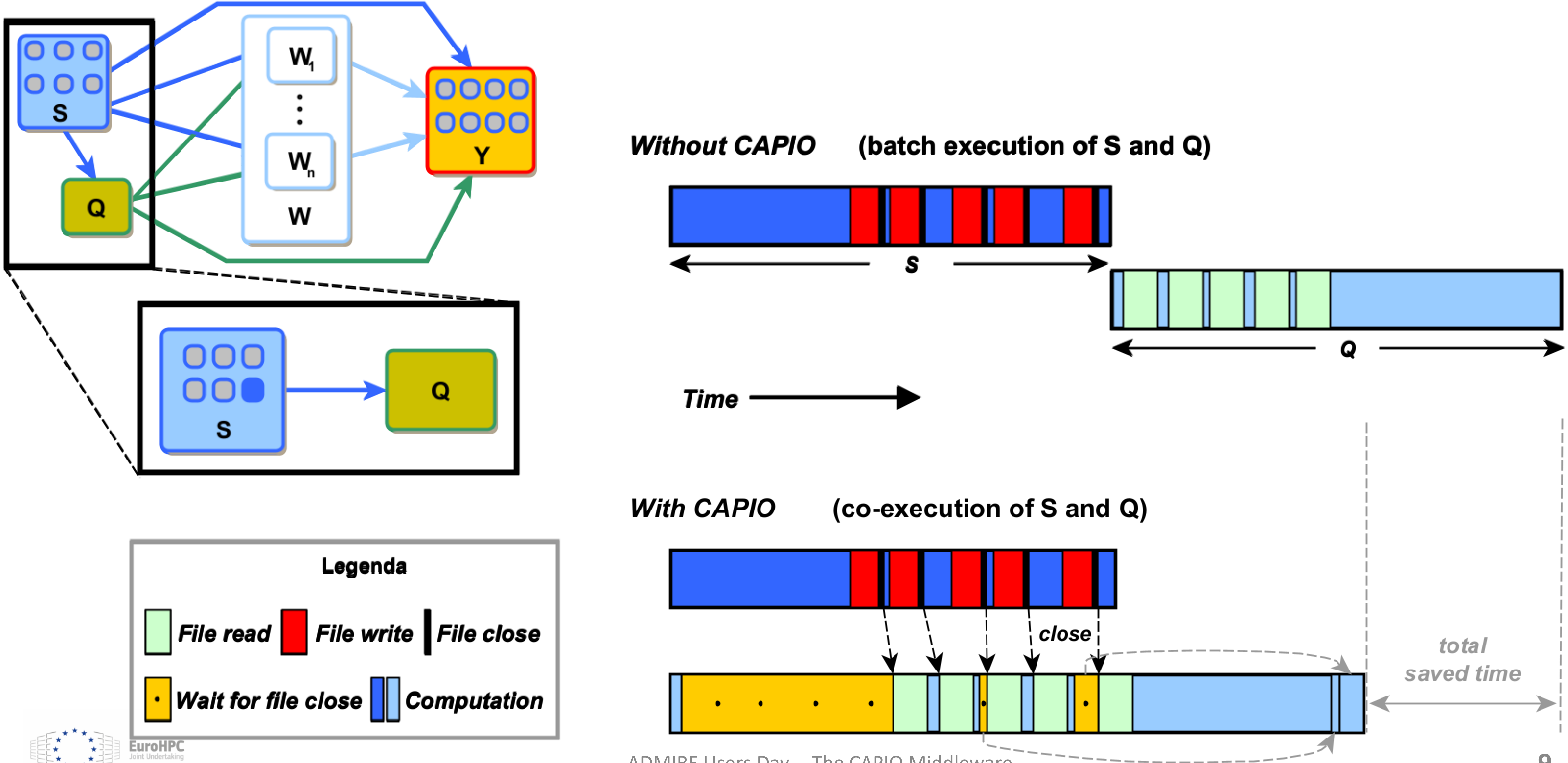


Legenda

🟩 *File read* 🟥 *File write* ▌ *File close*

🟨 *Wait for file close* ▌▌ *Computation*

CAPIO: transparent in situ processing



Without CAPIO (batch execution of S and Q)

Time →

With CAPIO (co-execution of S and Q)

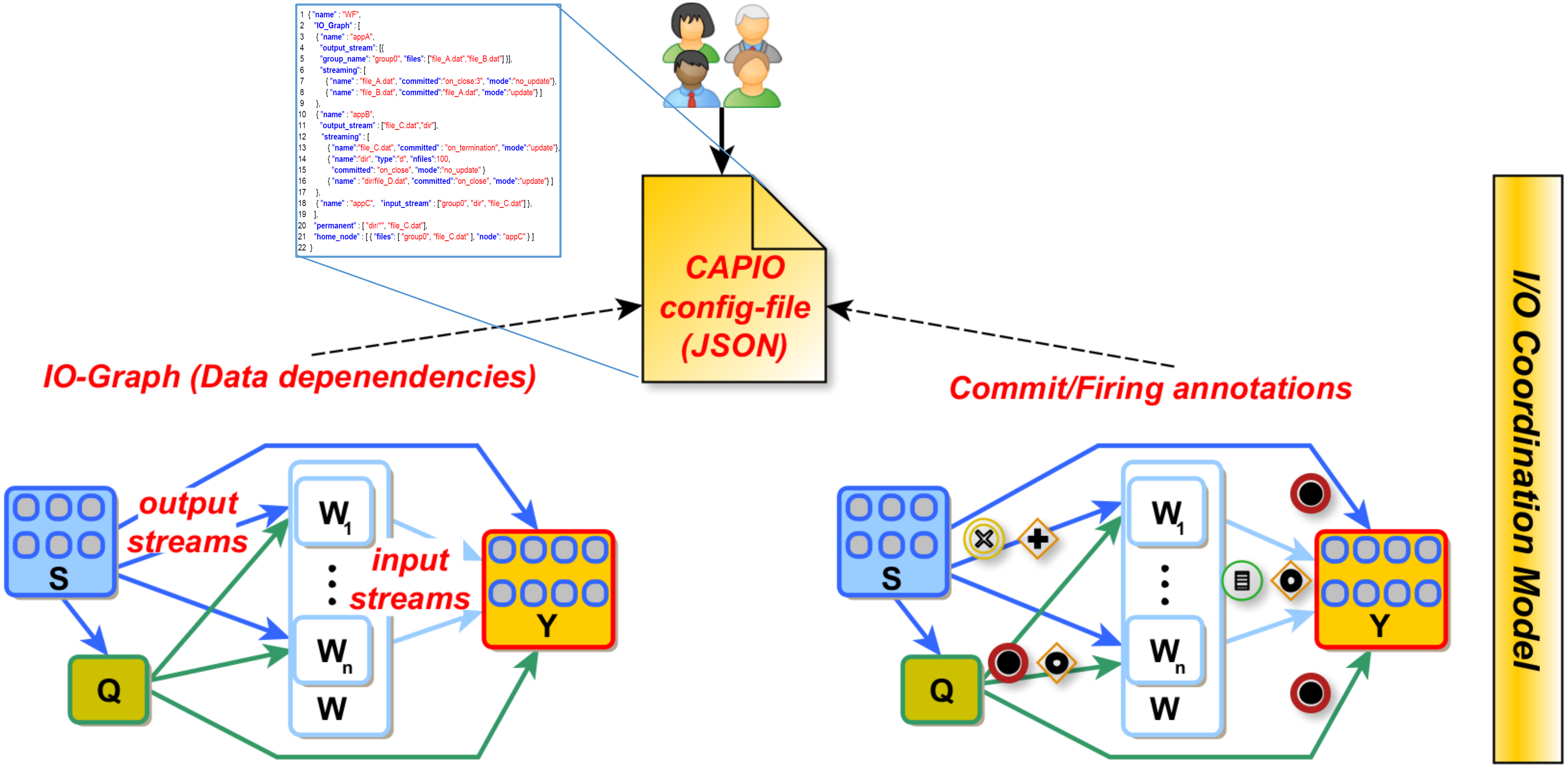
total saved time

Legenda

🟩 *File read* 🟥 *File write* | *File close*

🟨 *Wait for file close* ▮▮ *Computation*

The CAPIO coordination model

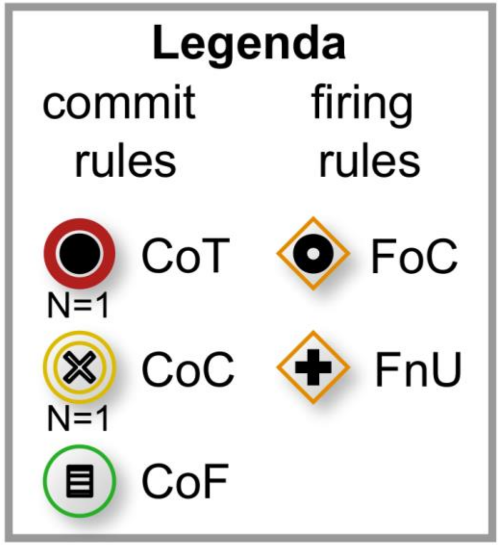


```
{ "name" : "WF",
  "IO_Graph" : [
   { "name" : "appA",
     "output_stream": [{
     "group_name": "group0", "files": ["file_A.dat","file_B.dat"] }],
     "streaming": [
       { "name" : "file_A.dat", "committed":"on_close:3", "mode":"no_update"},
       { "name" : "file_B.dat", "committed":"file_A.dat", "mode":"update"} ]
   },
   { "name" : "appB",
     "output_stream" : ["file_C.dat","dir"],
      "streaming" : [
        { "name":"file_C.dat", "committed" : "on_termination", "mode":"update"},
        { "name":"dir", "type":"d", "nfiles":100,
          "committed": "on_close", "mode":"no_update" }
        { "name" : "dir/file_D.dat", "committed":"on_close", "mode":"update"} ]
   },
   { "name" : "appC",   "input_stream" : ["group0", "dir", "file_C.dat"] },
  ],
  "permanent" : [ "dir/*", "file_C.dat"],
  "home_node" : [ { "files": [ "group0", "file_C.dat" ], "node": "appC" } ]
}
```

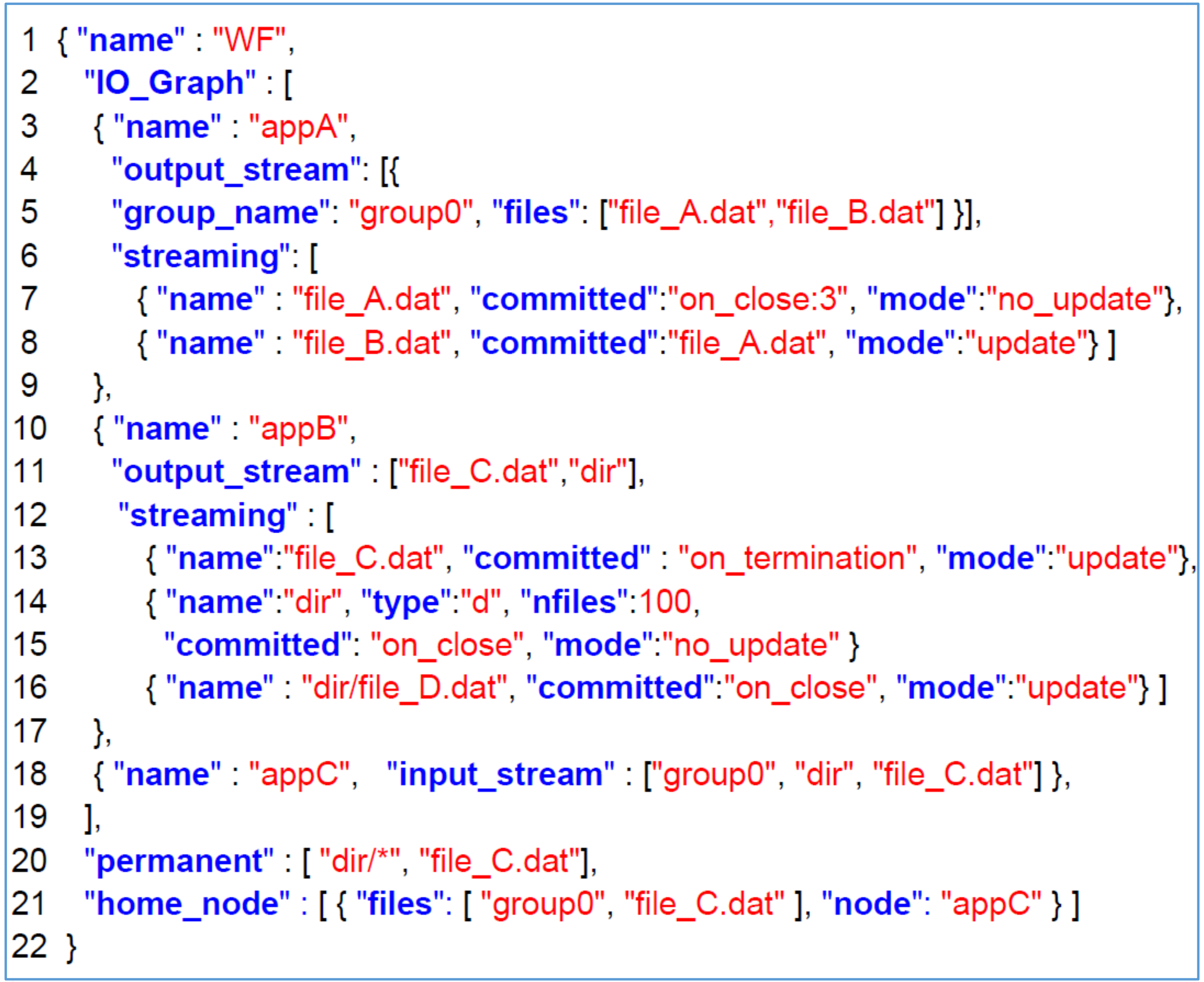
Annotations define when the file's contents can be accessed and when its data stream is complete.

Commit and firing annotations

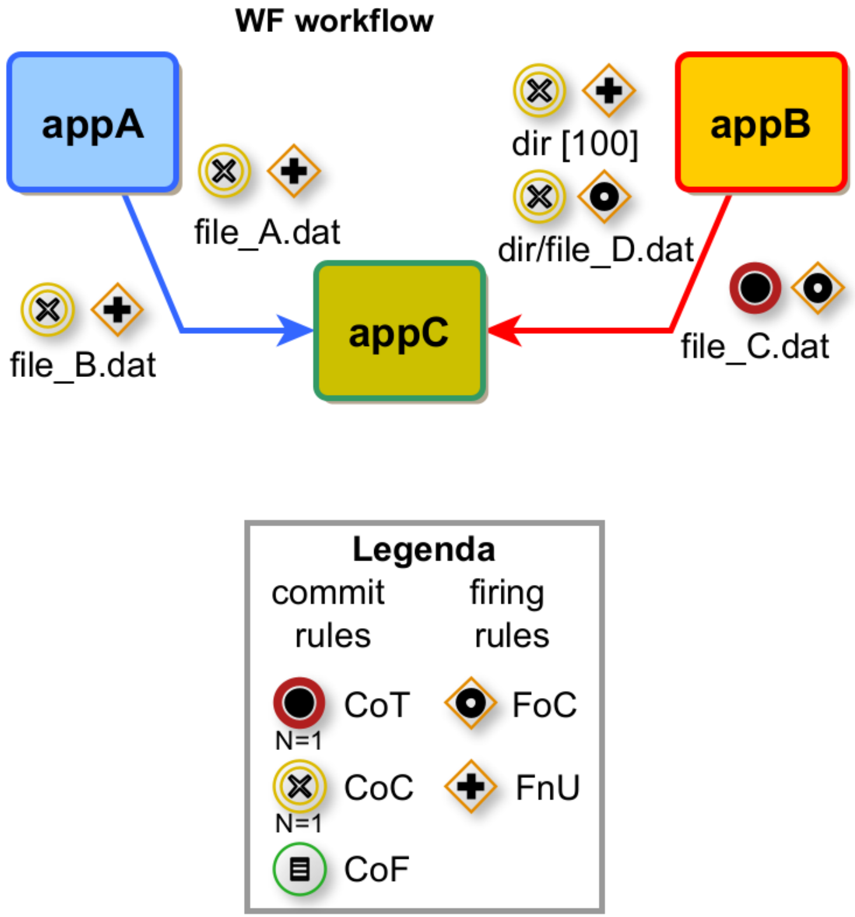
- □ Annotations define Producer-Consumer synchronization semantics on files and directories
  - A file is seen by CAPIO as a bounded stream of data bytes
- □ **Commit annotation** defines when there are no more updates to the file (i.e., End-of-Stream)
  - **Commit on-Termination** (**CoT**): all producers are terminated
  - **Commit on-Close** (**CoC**): all producers finished operating on the file (all of them definitely close the file)
  - **Commit on-File** (**CoF**): the commit semantics of the file being annotated depends on the commit semantics of another file
- □ **Firing annotation** defines when a consumer can start accessing the file content
  - **Firing on-Commit** (**FoC**): when the commit rule holds (i.e., the file content could be updated by producers)
  - **Firing no-Update** (**FnU**): as soon data is produced (i.e., the file content is never updated, e.g., write append mode)



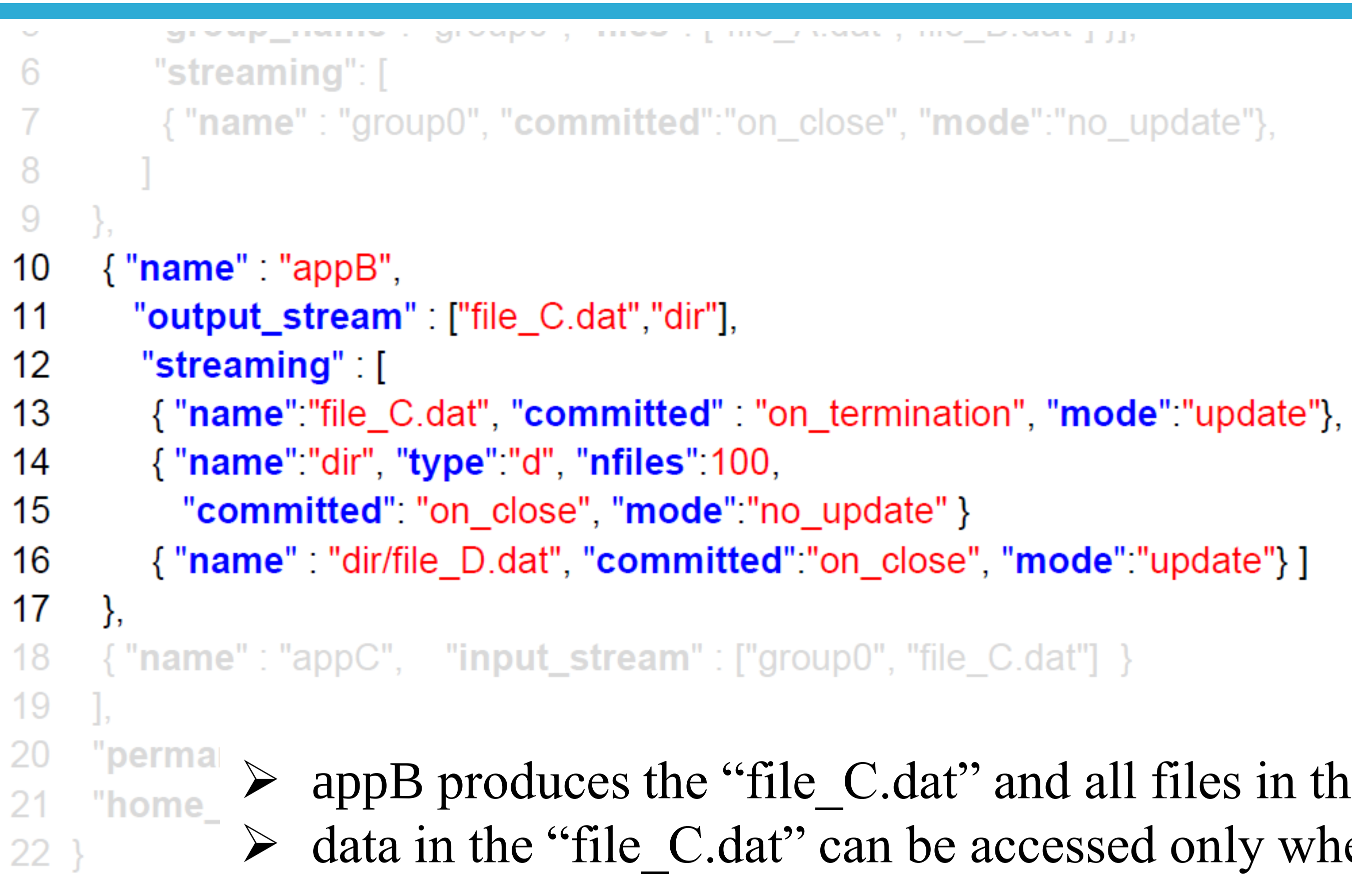
The Configuration File (example)



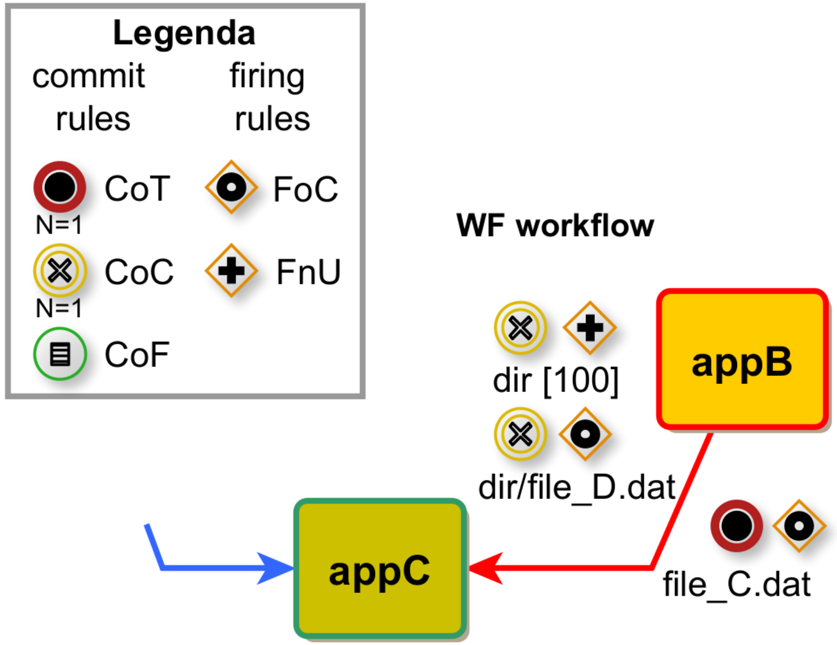
```
{ "name" : "WF",
  "IO_Graph" : [
   { "name" : "appA",
     "output_stream": [{
     "group_name": "group0", "files": ["file_A.dat","file_B.dat"] }],
     "streaming": [
        { "name" : "file_A.dat", "committed":"on_close:3", "mode":"no_update"},
        { "name" : "file_B.dat", "committed":"file_A.dat", "mode":"update"} ]
   },
   { "name" : "appB",
     "output_stream" : ["file_C.dat","dir"],
      "streaming" : [
         { "name":"file_C.dat", "committed" : "on_termination", "mode":"update"},
         { "name":"dir", "type":"d", "nfiles":100,
           "committed": "on_close", "mode":"no_update" }
         { "name" : "dir/file_D.dat", "committed":"on_close", "mode":"update"} ]
   },
   { "name" : "appC",   "input_stream" : ["group0", "dir", "file_C.dat"] },
  ],
  "permanent" : [ "dir/*", "file_C.dat"],
  "home_node" : [ { "files": [ "group0", "file_C.dat" ], "node": "appC" } ]
}
```



The Configuration File (example)

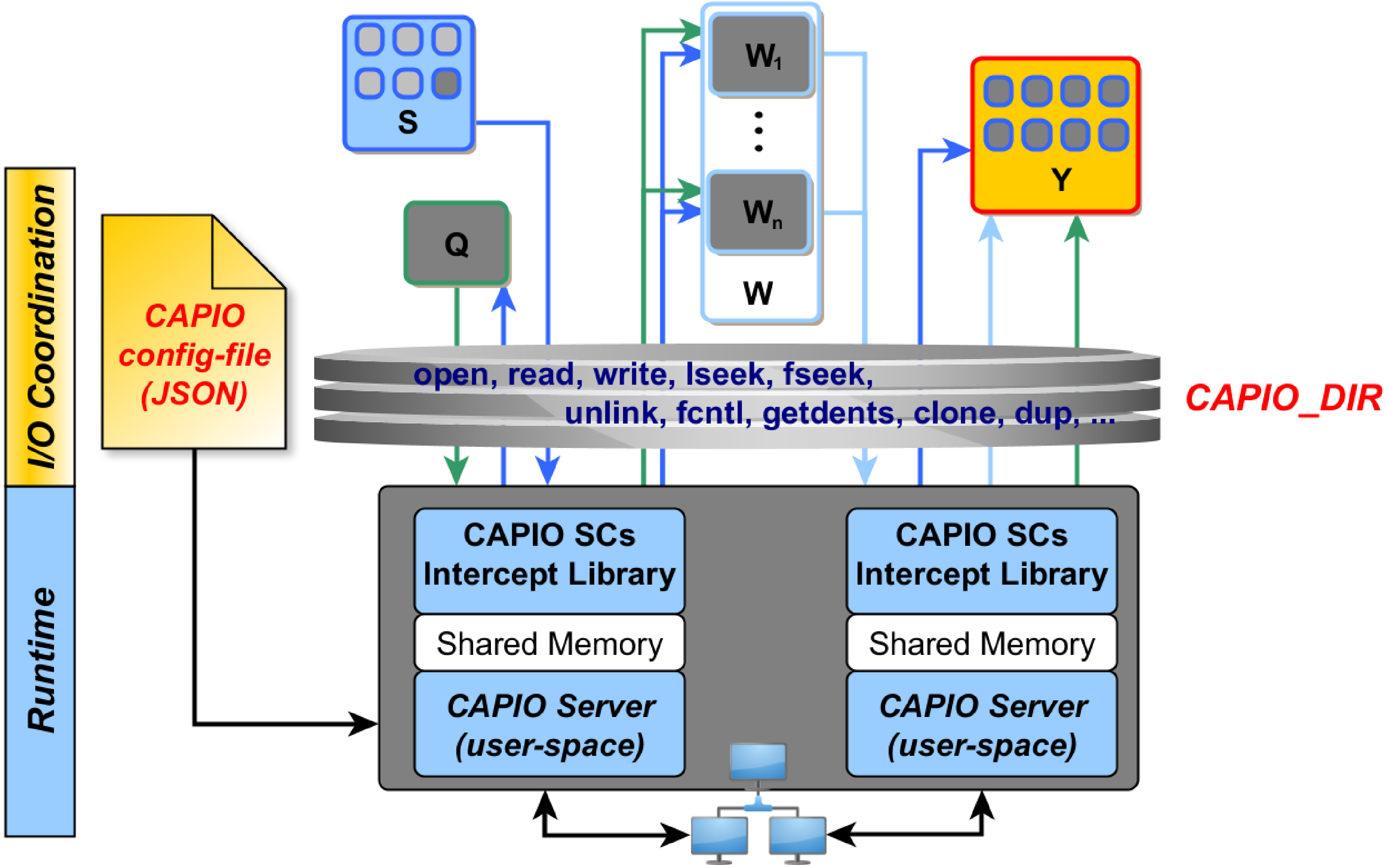


```
      "group_name" : "group0", "files" : ["file_A.dat", "file_B.dat"]}],
      "streaming": [
       { "name" : "group0", "committed":"on_close", "mode":"no_update"},
      ]
    },
    { "name" : "appB",
      "output_stream" : ["file_C.dat","dir"],
       "streaming" : [
        { "name":"file_C.dat", "committed" : "on_termination", "mode":"update"},
        { "name":"dir", "type":"d", "nfiles":100,
           "committed": "on_close", "mode":"no_update" }
        { "name" : "dir/file_D.dat", "committed":"on_close", "mode":"update"} ]
    },
    { "name" : "appC",    "input_stream" : ["group0", "file_C.dat"]  }
   ],
   "perma
   "home_
}
```

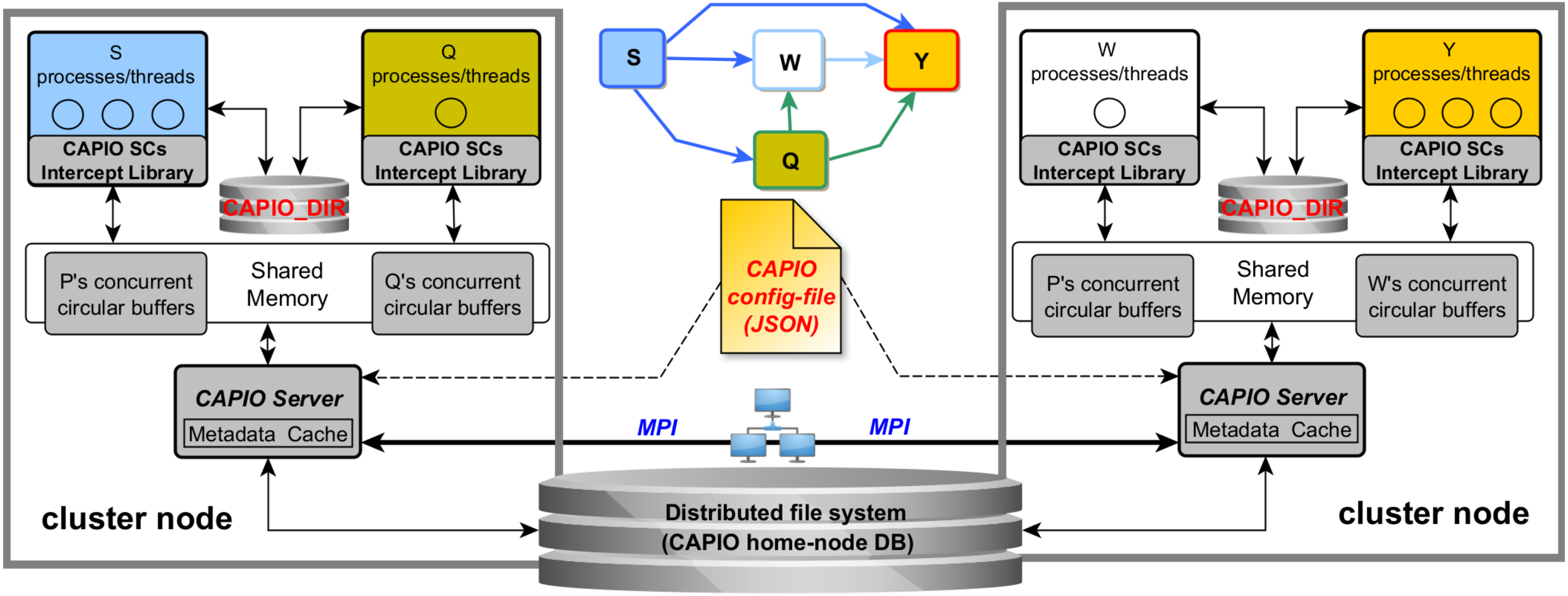


- ➢ appB produces the "file_C.dat" and all files in the directory "dir" (100 files)
- ➢ data in the "file_C.dat" can be accessed only when appB finishes (i.e., **CoT, FoC rules**)
- ➢ for all 100 files in the directory "dir", but "file_D.dat", data can be accessed as soon as is produced (i.e., **CoC, FnU rules**)
- ➢ the data in "dir/file_D.dat" can be accessed only when the file is closed (i.e., **CoC, FoC rules**)
- ➢ the annotation **nfiles=100** is an upper-bound hint for the number of files in "dir"

Conceptual SW architecture


S
Q
W1
⋮
Wn
W
Y
I/O Coordination
Runtime
CAPIO config-file (JSON)
open, read, write, lseek, fseek,
unlink, fcntl, getdents, clone, dup, ...
CAPIO_DIR
CAPIO SCs Intercept Library
Shared Memory
CAPIO Server (user-space)
CAPIO SCs Intercept Library
Shared Memory
CAPIO Server (user-space)

Capio deployment (2 cluster node example)



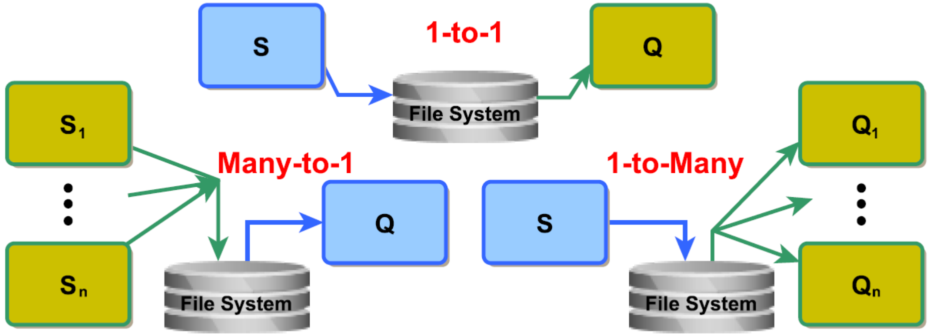
Evaluation Environments

❑ Tests conducted on two clusters:

- ➢ Galileo100 (@CINECA, Italy) Tire-I supercomputer
  - ❖ CascadeLake nodes, 2 x Intel® Xeon® 8260, @2.4GHz 24 core each, IB 100Gbit/s network
  - ❖ LUSTRE parallel file system
  - ❖ https://www.hpc.cineca.it/systems/hardware/galileo100/

- ➢ HPC4AI Cluster (@C3S Turin, Italy) Tire-III supercomputer
  - ❖ Broadwell nodes 2 x Intel® Xeon® E5-2697 v4 @2.3GHz 18 core each, OPA 100Gbit/s network
  - ❖ BeeGFS (and also LUSTRE) parallel file system(s)
  - ❖ https://hpc4ai.unito.it/

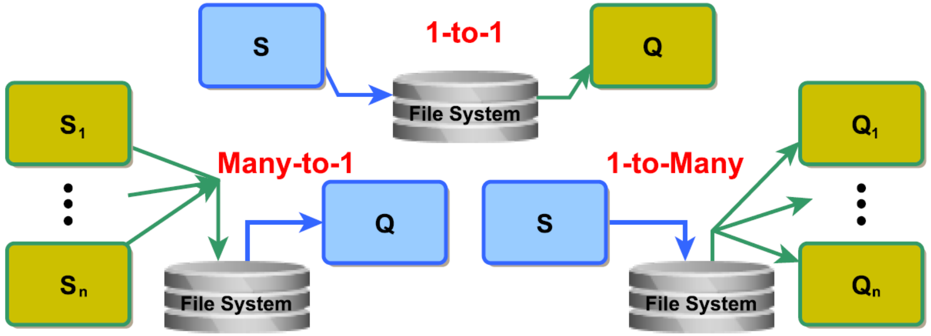
Benchmarks on Galileo100



CAPIO config-file (JSON)

```
{ "name" : "benchmarks",
  "IO_Graph" : [
    {"name" : "S",   "output_stream": [ { "files": ["file*.dat"] } ],
     "streaming": [ { "name" : "file*.dat", "committed":"on_close", "mode":"MODE"}] },
    {"name" : "Q",   "input_stream" : ["file*.dat"] }
  ]
}
```

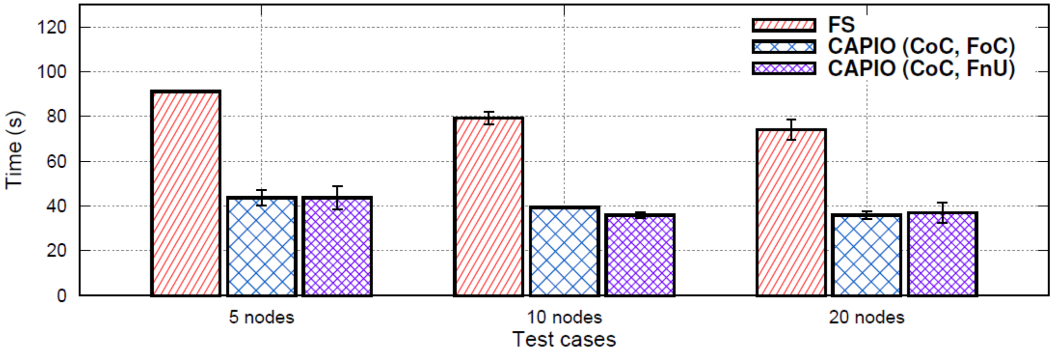
Benchmarks on Galileo100



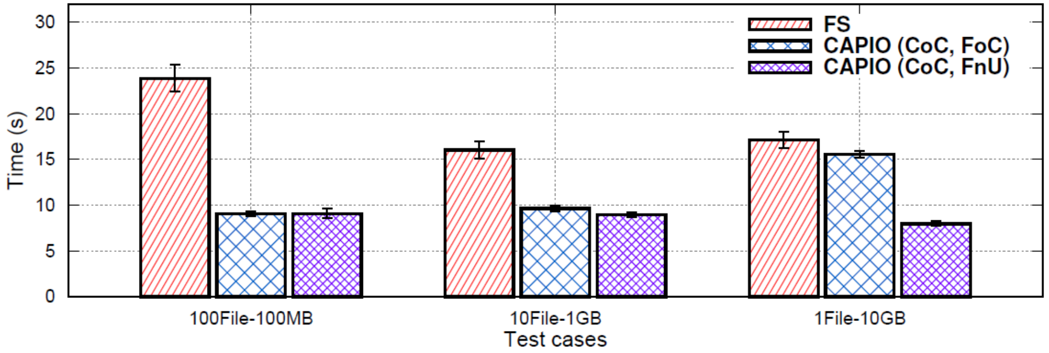
```
{ "name" : "benchmarks",
  "IO_Graph" : [
   {"name" : "S",   "output_stream": [ { "files": ["file*.dat"] } ],
    "streaming": [ { "name" : "file*.dat", "committed":"on_close", "mode":"MODE"}] },
   {"name" : "Q",   "input_stream" : ["file*.dat"] }
  ]
}
```

CAPIO config-file (JSON)

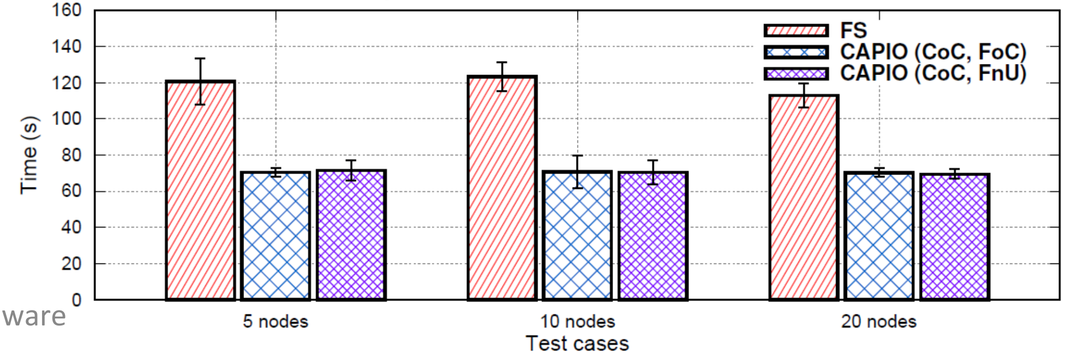
1-to-Many, 100 files of 1GB data, ws=1MB



1-to-1, 2 nodes, 10GB dataset, ws=1MB

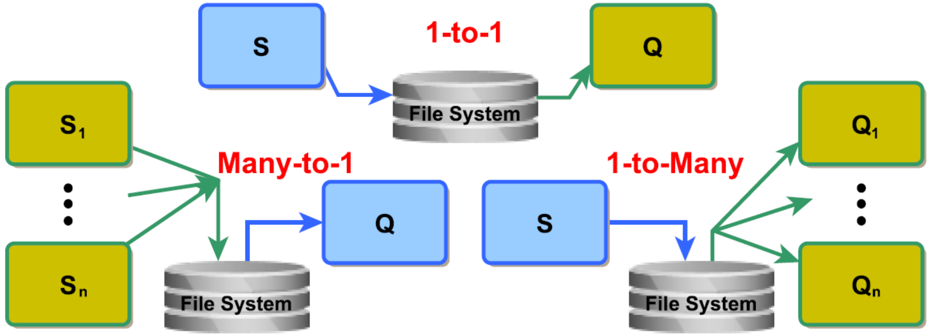


Many-to-1, 100 files of 1GB data, ws=1MB



Benchmarks on HPC4AI; testing ADIOS2

- **100 files of 1 GB** (flat binary data)
- Write throughput: **1 file/s**
- Write granularity: ws=1 MB (**CHUNK_SIZE**)
- **BeeGFS** File System (FS)
- CAPIO with firing rule FnU (i.e, **no_update)**
- ADIOS2 (ver. 2.9), BP5 engine



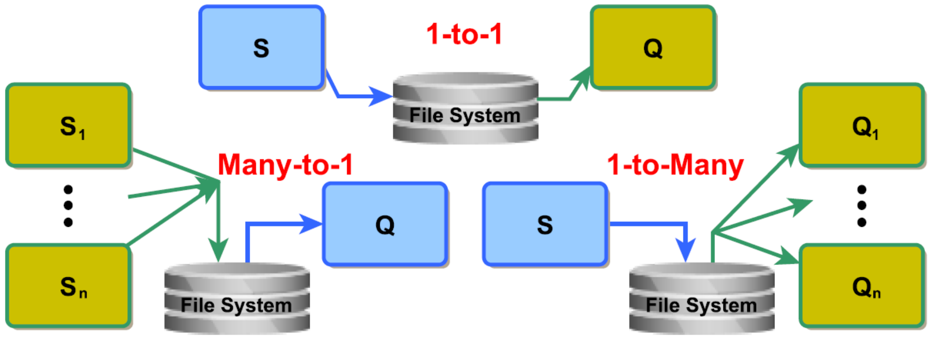
Benchmarks on HPC4AI; testing ADIOS2

- **100 files of 1 GB** (flat binary data)
- Write throughput: **1 file/s**
- Write granularity: ws=1 MB (**CHUNK_SIZE**)
- **BeeGFS** File System (FS)
- CAPIO with firing rule FnU (i.e, **no_update)**
- ADIOS2 (ver. 2.9), BP5 engine

1-to-1 -- 2 nodes			
	FS	**CAPIO**	**ADIOS2**
Total time	219s	163s	140s

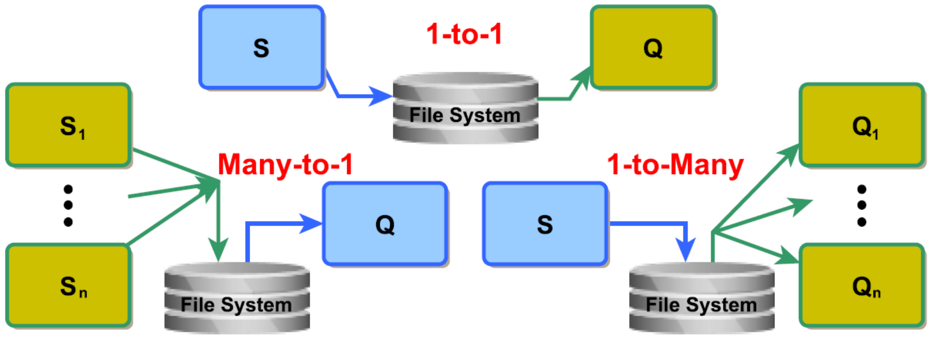
1-to-Many -- 21 nodes			
	FS	**CAPIO**	**ADIOS2**
Total time	189s	163s	205s

Many-to-1 -- 21 nodes			
	FS	**CAPIO**	**ADIOS2**
Total time	200s	133s	108s



Benchmarks on HPC4AI; testing ADIOS2

- **100 files of 1 GB** (flat binary data)
- Write throughput: **1 file/s**
- Write granularity: ws=1 MB (**CHUNK_SIZE**)
- **BeeGFS** File System (FS)
- CAPIO with firing rule FnU (i.e, **no_update)**
- ADIOS2 (ver. 2.9), BP5 engine

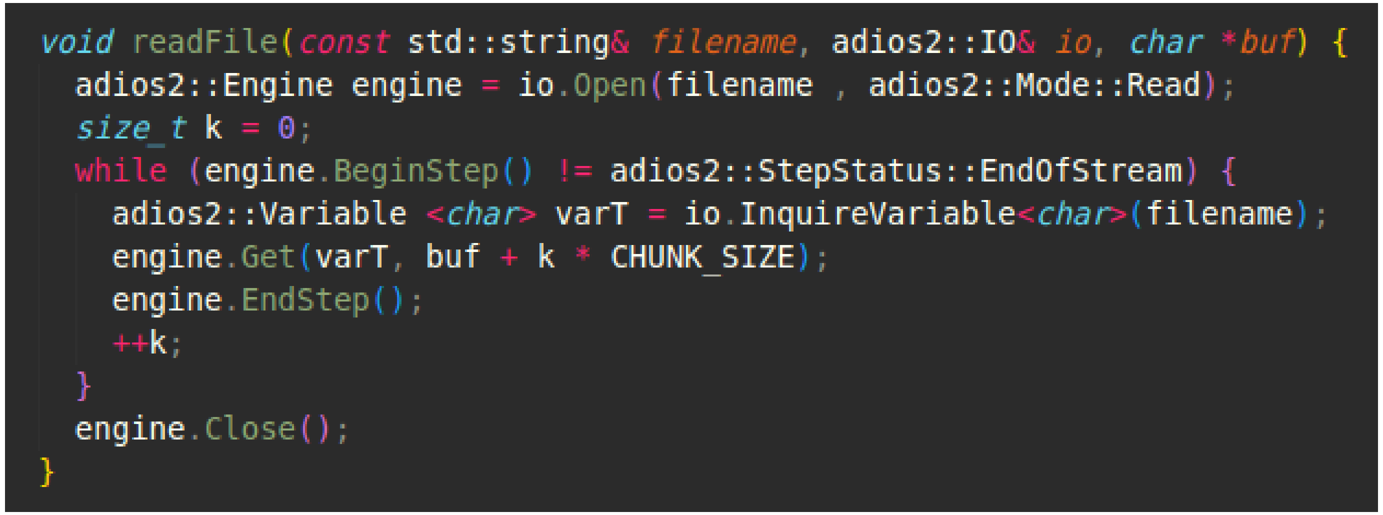


1-to-1 -- 2 nodes			
	FS	CAPIO	ADIOS2
Total time	219s	163s	140s

1-to-Many -- 21 nodes			
	FS	CAPIO	ADIOS2
Total time	189s	163s	205s

Many-to-1 -- 21 nodes			
	FS	CAPIO	ADIOS2
Total time	200s	133s	108s

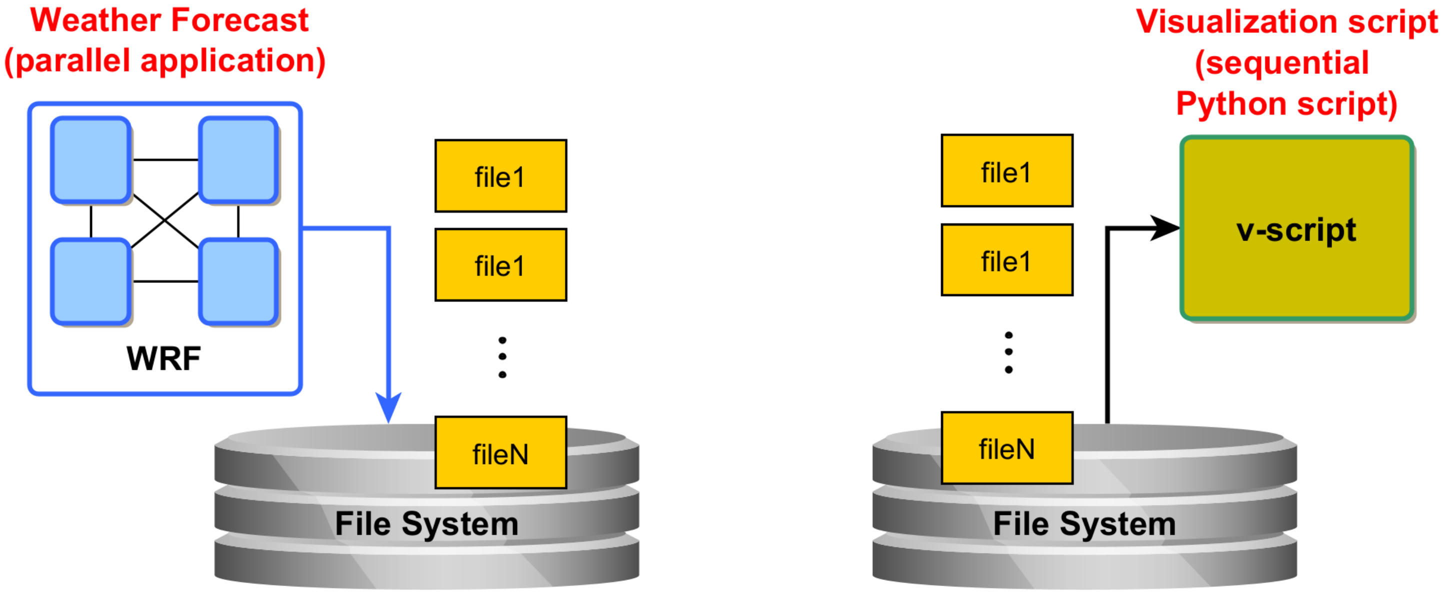
Code snippet, one way for reading a binary file, chunk by chunk, in ADIOS2



```
void readFile(const std::string& filename, adios2::IO& io, char *buf) {
  adios2::Engine engine = io.Open(filename , adios2::Mode::Read);
  size_t k = 0;
  while (engine.BeginStep() != adios2::StepStatus::EndOfStream) {
    adios2::Variable <char> varT = io.InquireVariable<char>(filename);
    engine.Get(varT, buf + k * CHUNK_SIZE);
    engine.EndStep();
    ++k;
  }
  engine.Close();
}
```

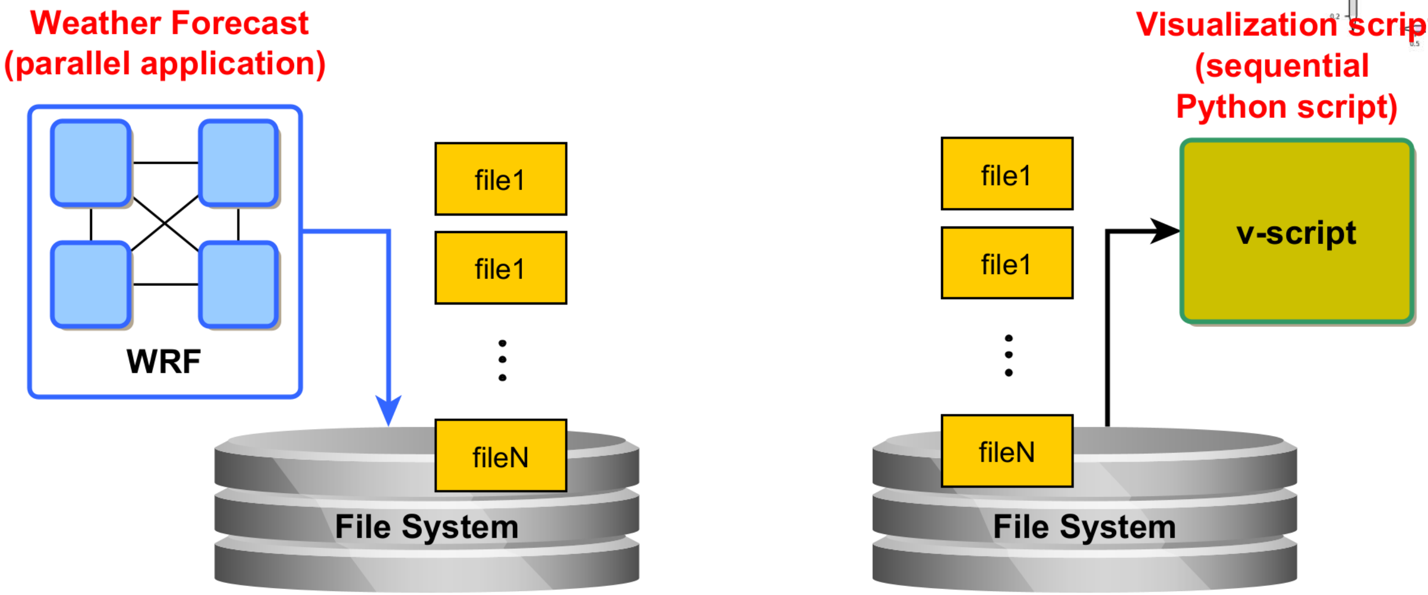
Weather Forecast Visualization Workflow on HPC4AI

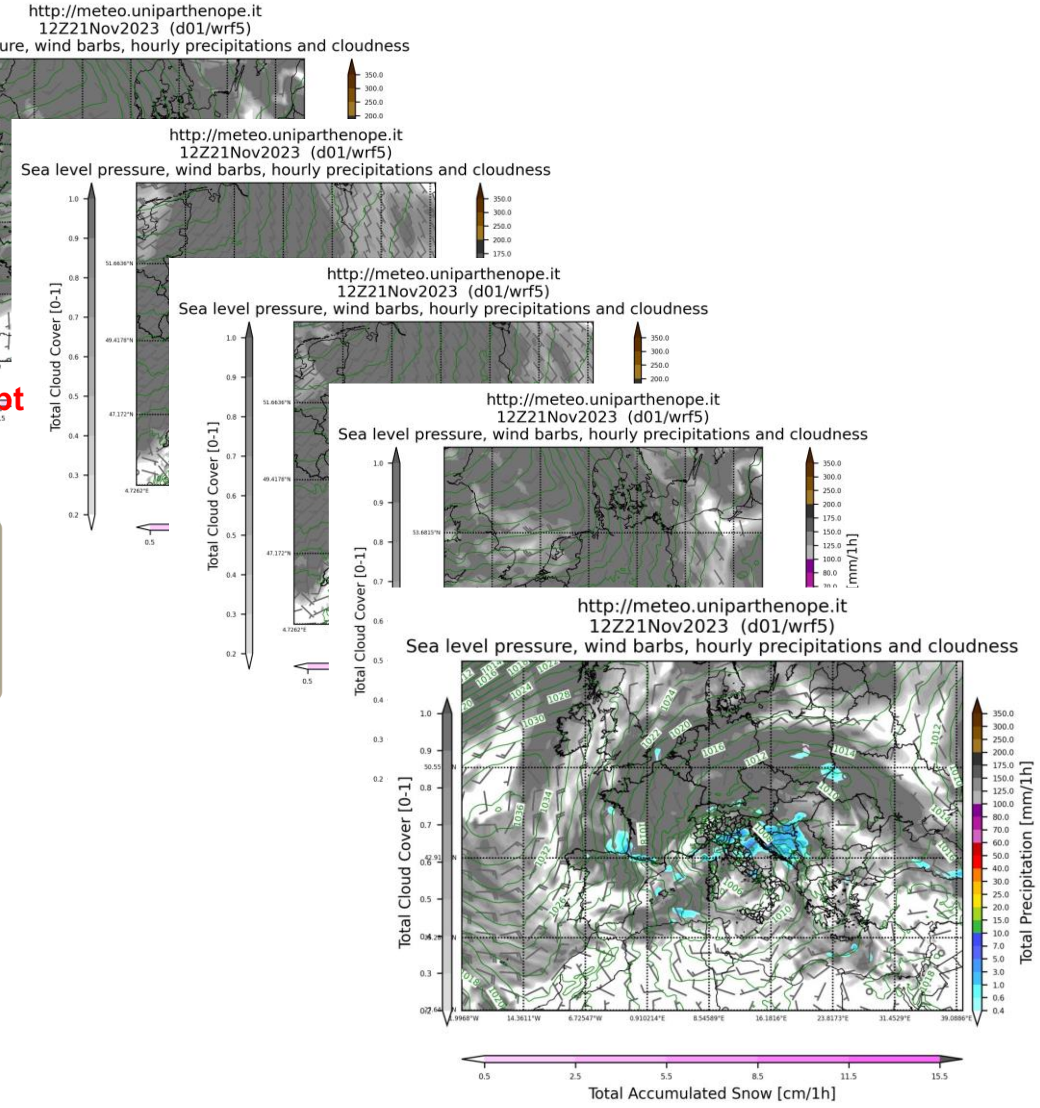
- Traditional workflow
- First WRF produces all output files of the simulation
- Then a Python script generates a PNG image from each file



Weather Forecast Visualization Workflow on HPC4AI

- ➢ Traditional workflow
- ➢ First WRF produces all output files of the simulation
- ➢ Then a Python script generates a PNG image from each file

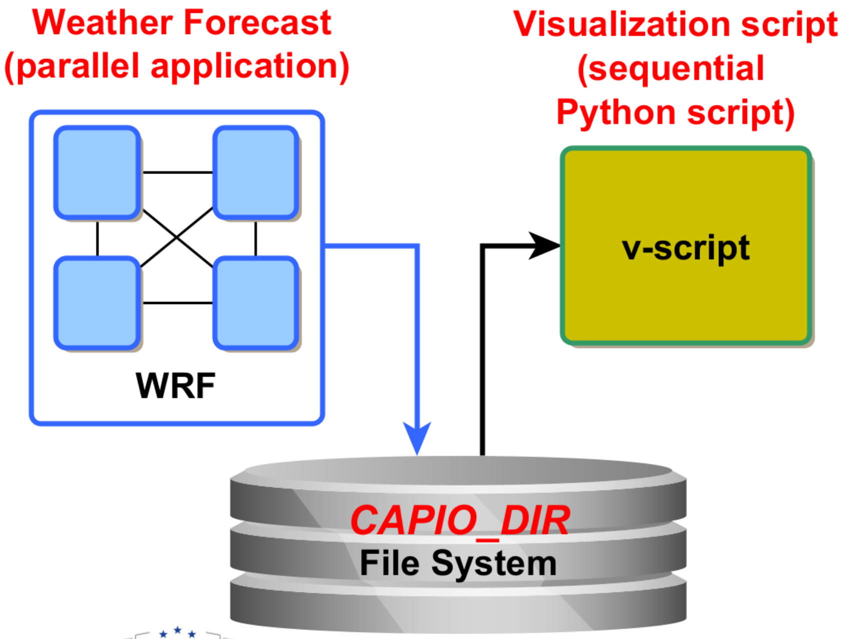



http://meteo.uniparthenope.it
12Z21Nov2023 (d01/wrf5)
Sea level pressure, wind barbs, hourly precipitations and cloudness
Total Cloud Cover [0-1]
Total Precipitation [mm/1h]
Total Accumulated Snow [cm/1h]

Weather Forecast Visualization Workflow on HPC4AI

CAPIO config-file (JSON)

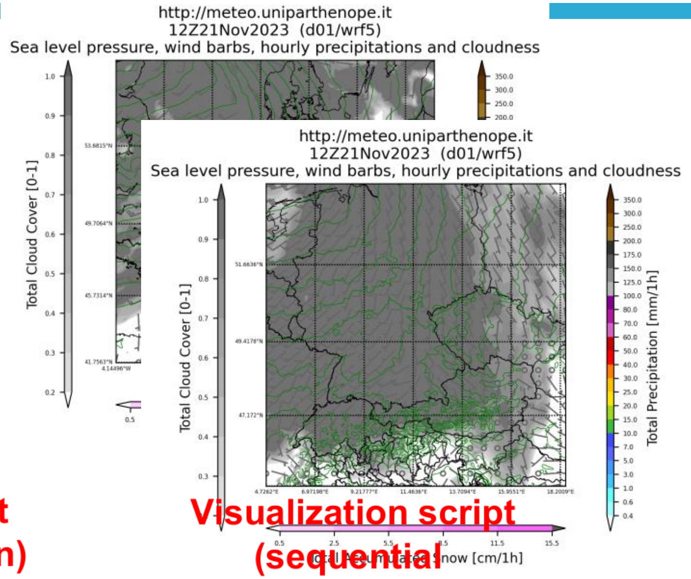
```
{ "name" : "WRF_WORKFLOW",
  "IO_Graph" : [
   { "name" : "WRF",
     "output_stream": [ "wrfoutdir", "wrfoutdir/*" ],
     "streaming": [
       { "name" : "wrfoutdir”, "type":"d",
                    "committed":"on_termination", "mode" : "no_update"},
       { "name":"wrfoutdir/*", "committed" : "on_close", "mode" : "update"} ]
   },
   { "name" : "v-script",
     "input_stream" : ["wrfoutdir/*”]}
  ],
}
```


Weather Forecast (parallel application)
Visualization script (sequential Python script)
WRF
v-script
CAPIO_DIR
File System

ADMIRE
malleable data solutions for HPC

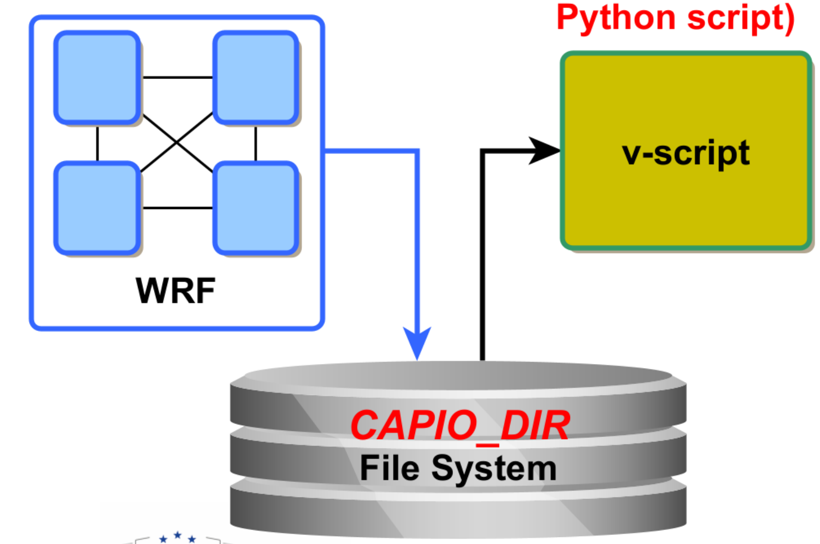
Weather Forecast Visualization Workflow on HPC4AI

cini consorzio interuniversitario nazionale per l'informatica
Università di Pisa


http://meteo.uniparthenope.it
12Z21Nov2023 (d01/wrf5)
Sea level pressure, wind barbs, hourly precipitations and cloudness
Total Cloud Cover [0-1]
Total Precipitation [mm/1h]
Total accumulated snow [cm/1h]

Weather Forecast (parallel application)

Visualization script (sequential Python script)

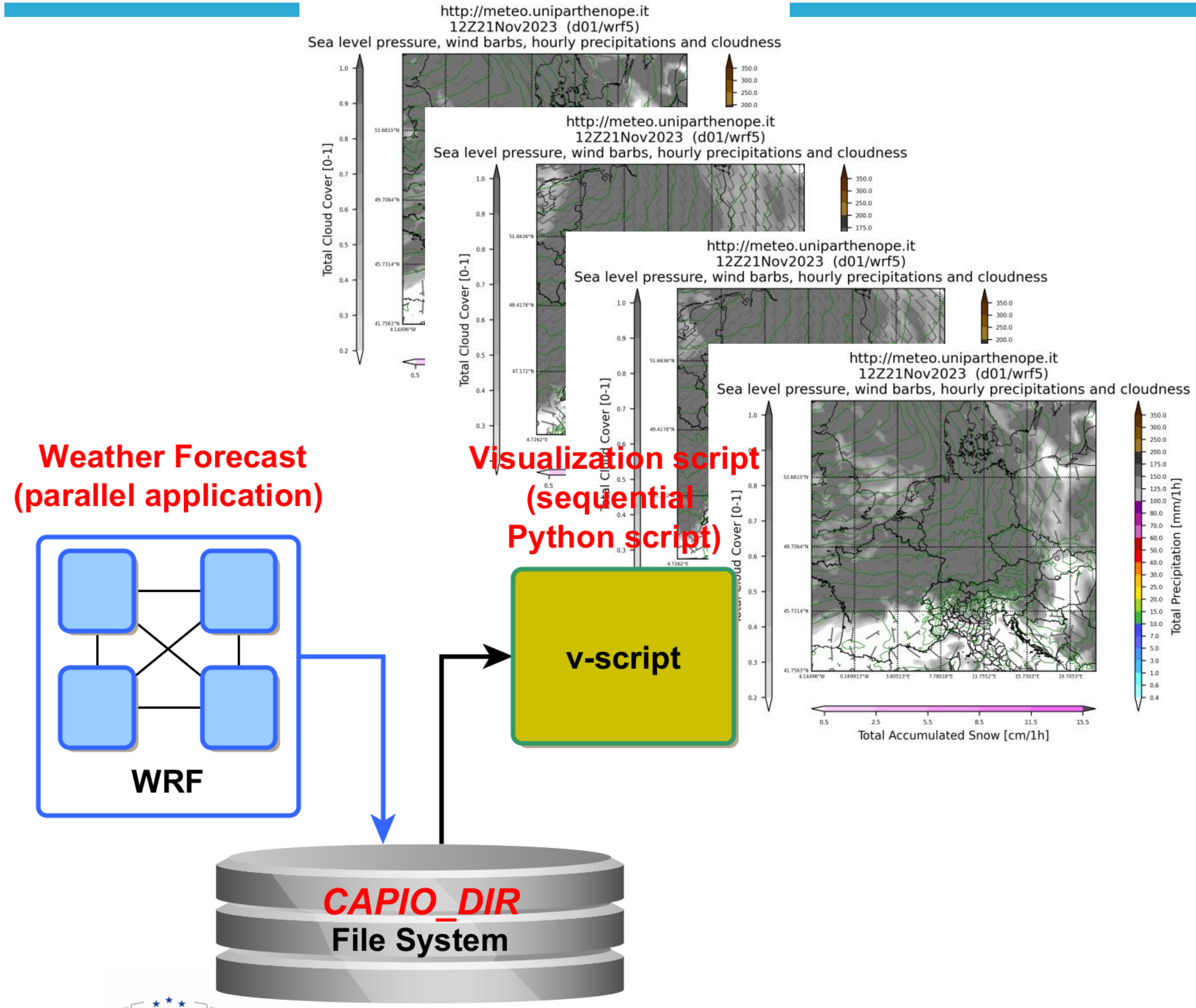


CAPIO config-file (JSON)

```
{ "name" : "WRF_WORKFLOW",
  "IO_Graph" : [
   { "name" : "WRF",
     "output_stream": [ "wrfoutdir", "wrfoutdir/*" ],
     "streaming": [
       { "name" : "wrfoutdir”, "type":"d",
                        "committed":"on_termination", "mode" : "no_update"},
       { "name":"wrfoutdir/*", "committed" : "on_close", "mode" : "update"} ]
   },
   { "name" : "v-script",
     "input_stream" : ["wrfoutdir/*”]}
  ],
}
```

EuroHPC Joint Undertaking

Weather Forecast Visualization Workflow on HPC4AI



http://meteo.uniparthenope.it
12Z21Nov2023 (d01/wrf5)
Sea level pressure, wind barbs, hourly precipitations and cloudness
Total Cloud Cover [0-1]
Total Precipitation [mm/1h]
Total Accumulated Snow [cm/1h]
Weather Forecast (parallel application)
WRF
Visualization script (sequential Python script)
v-script
CAPIO_DIR
File System

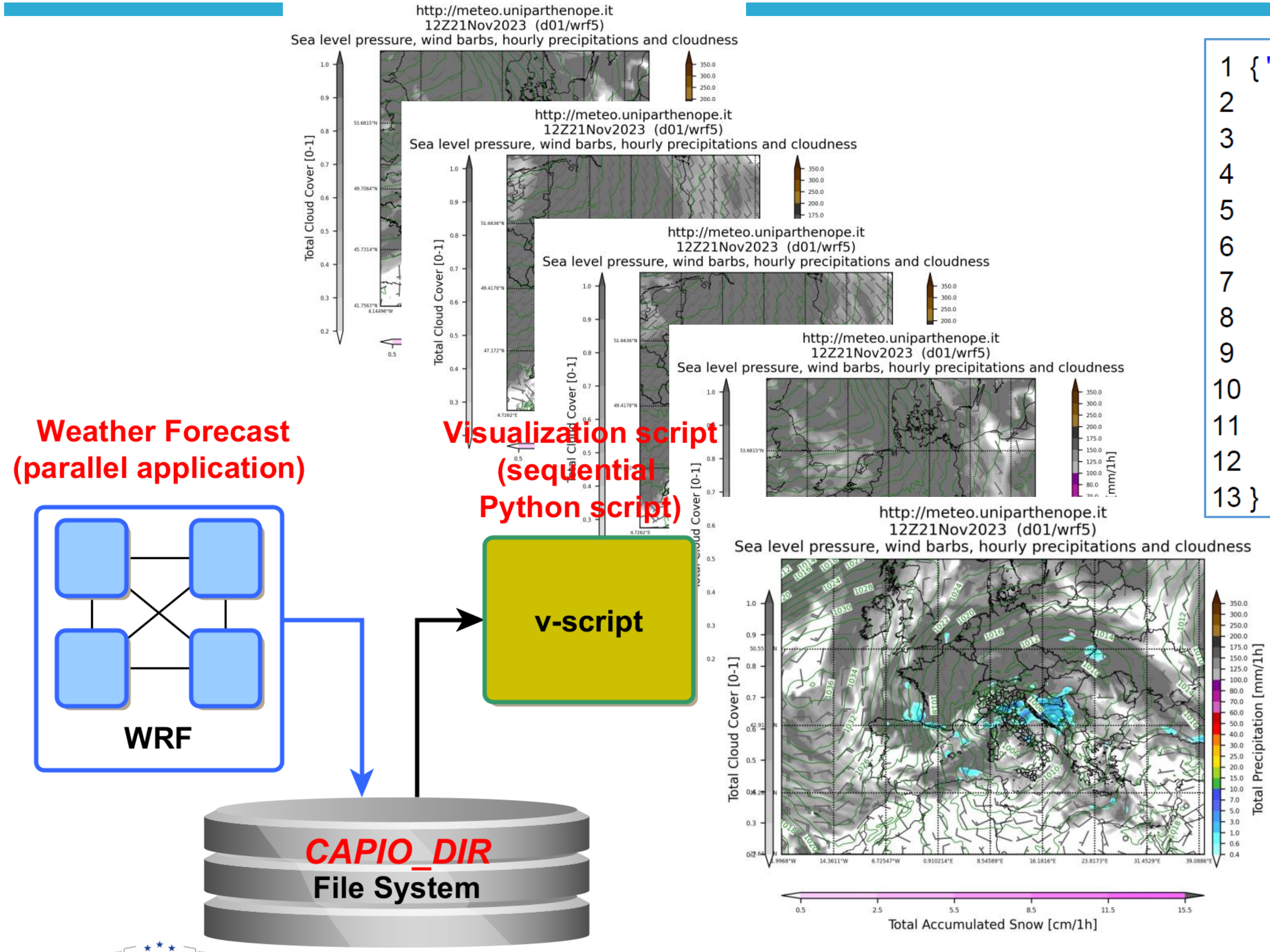
```
{ "name" : "WRF_WORKFLOW",
  "IO_Graph" : [
   { "name" : "WRF",
     "output_stream": [ "wrfoutdir", "wrfoutdir/*" ],
     "streaming": [
       { "name" : "wrfoutdir”, "type":"d",
                        "committed":"on_termination", "mode" : "no_update"},
       { "name":"wrfoutdir/*", "committed" : "on_close", "mode" : "update"} ]
   },
   { "name" : "v-script",
     "input_stream" : ["wrfoutdir/*”]}
  ],
}
```

CAPIO config-file (JSON)

Weather Forecast Visualization Workflow on HPC4AI

Weather Forecast (parallel application)

Visualization script (sequential Python script)

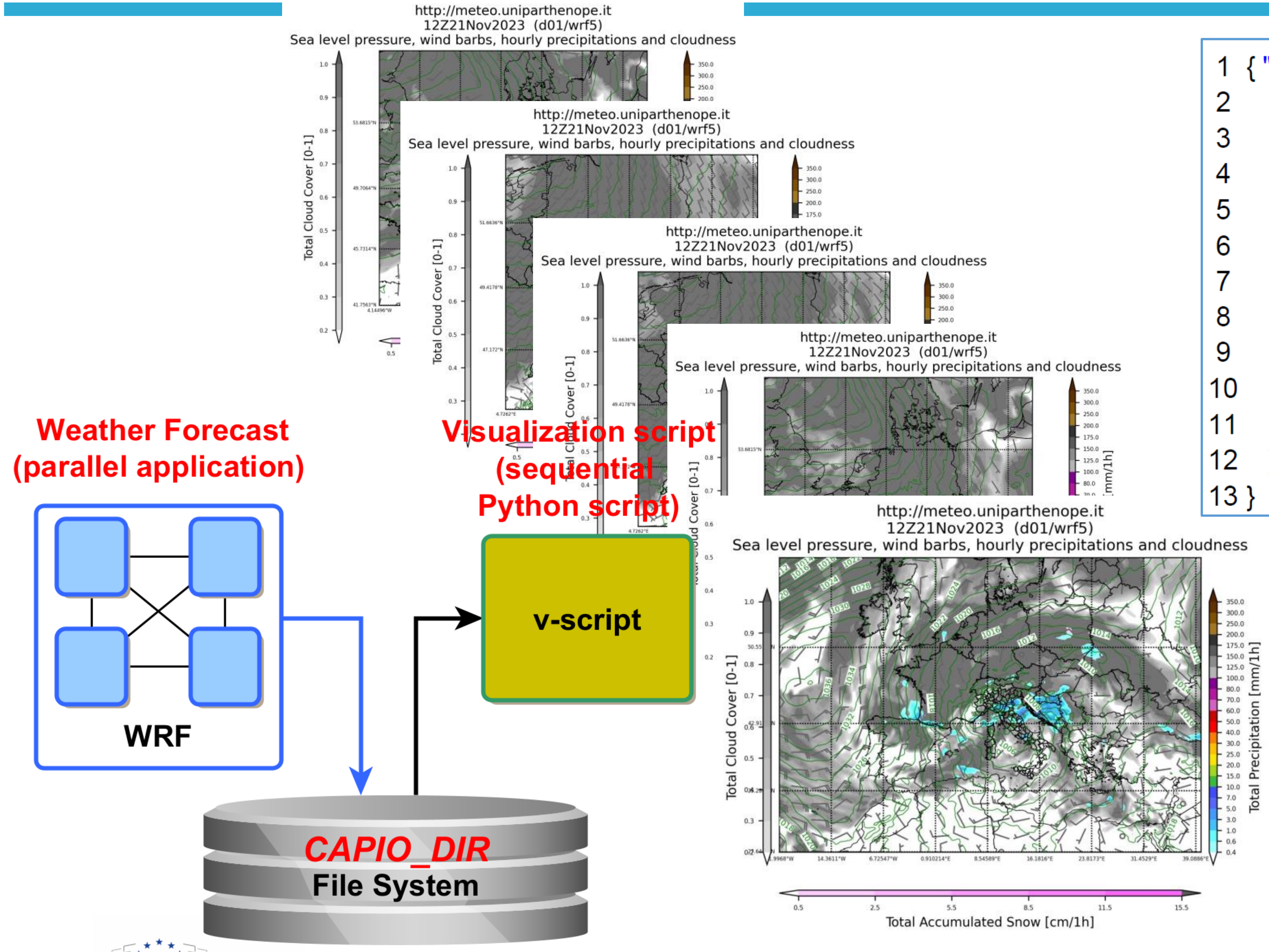


WRF
v-script
CAPIO_DIR
File System
http://meteo.uniparthenope.it
12Z21Nov2023 (d01/wrf5)
Sea level pressure, wind barbs, hourly precipitations and cloudness
Total Cloud Cover [0-1]
Total Precipitation [mm/1h]
Total Accumulated Snow [cm/1h]

CAPIO config-file (JSON)

```
{ "name" : "WRF_WORKFLOW",
  "IO_Graph" : [
   { "name" : "WRF",
     "output_stream": [ "wrfoutdir", "wrfoutdir/*" ],
     "streaming": [
       { "name" : "wrfoutdir", "type":"d",
                   "committed":"on_termination", "mode" : "no_update"},
       { "name":"wrfoutdir/*", "committed" : "on_close", "mode" : "update"} ]
   },
   { "name" : "v-script",
     "input_stream" : ["wrfoutdir/*"]}
  ],
}
```

Weather Forecast Visualization Workflow on HPC4AI



CAPIO config-file (JSON)

```
{ "name" : "WRF_WORKFLOW",
  "IO_Graph" : [
   { "name" : "WRF",
     "output_stream": [ "wrfoutdir", "wrfoutdir/*" ],
     "streaming": [
       { "name" : "wrfoutdir", "type":"d",
                   "committed":"on_termination", "mode" : "no_update"},
       { "name":"wrfoutdir/*", "committed" : "on_close", "mode" : "update"} ]
   },
   { "name" : "v-script",
     "input_stream" : ["wrfoutdir/*"]}
  ],
}
```

BeeGFS File System
4 x 24 MPI processes

	w/o CAPIO	w/ CAPIO
Total time	2538s	2099s
First file available after	761s	81s

- 2-domains, 25 hours of simulation
- WRF uses 96 MPI Fortran processes

Summary

- ❑ We presented CAPIO, an I/O middleware developed within the **ADMIRE project**
  - ➢ Supports the POSIX standard and targets all workflows whose I/O back-end uses POSIX I/O SCs
  - ➢ Shifts I/O coordination toward a declarative approach through a new, I/O-tailored coordination language based on the JSON syntax.
  - ➢ Avoids touching the existing codebase, while still providing performance improvements
- ❑ Additional effort is needed to
  - ➢ Enhance the expressiveness of the configuration file by enriching features (more hints)
  - ➢ Introduce multi-back-end support (e.g., leveraging ADIOS2 as communication transport, using multiple communication protocols, …)
  - ➢ Test CAPIO with more application workflows
  - ➢ Integrate CAPIO in existing WMSs (e.g., StreamFlow, Dagonstar, …)

StreamFlow Towards Cloud-HPC Continuum

Dagonstar

References and Project Team

- ❑ Git: https://github.com/High-Performance-IO/capio
- ❑ Paper: A.R. Martinelli, M. Torquati, I. Colonnelli, B. Cantalupo, M. Aldinucci "**CAPIO: a Middleware for Transparent I/O Streaming in Data-Intensive Workflows**", In IEEE 30th International Conference on High Performance Computing, Data, and Analytics (HiPC '23), Goa, India, 2023 (To Appear)

CAPIO
Cross Application Programmable I/O

Project Team

Alberto R. Martinelli

Iacopo Colonnelli

Marco E. Santimaria

Barbara Cantalupo

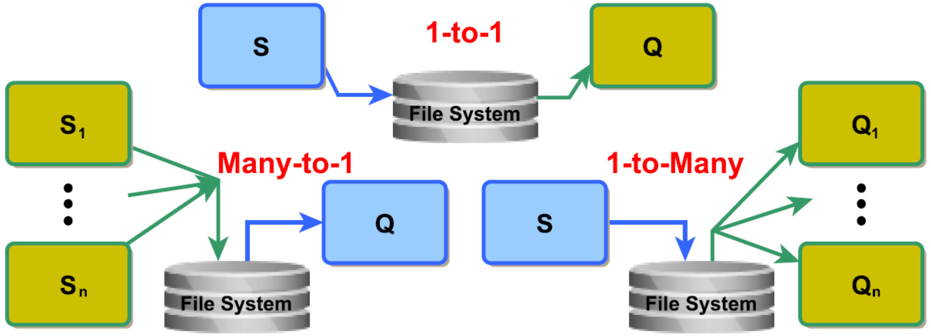
Marco Aldinucci

Massimo Torquati

Thank you, any questions

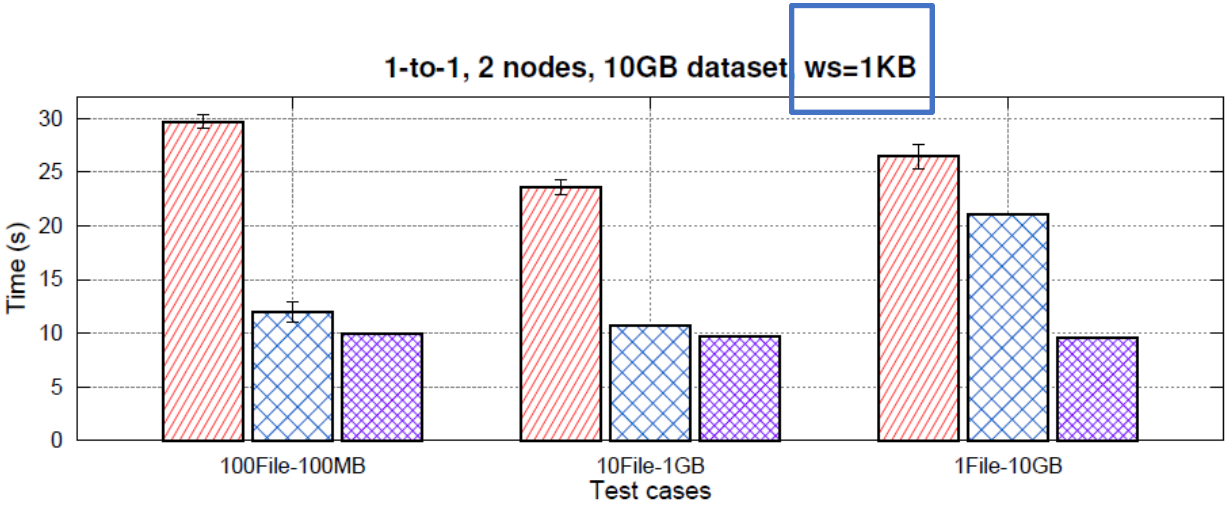
Massimo Torquati (CINI - University of Pisa)
massimo.torquati@unipi.it

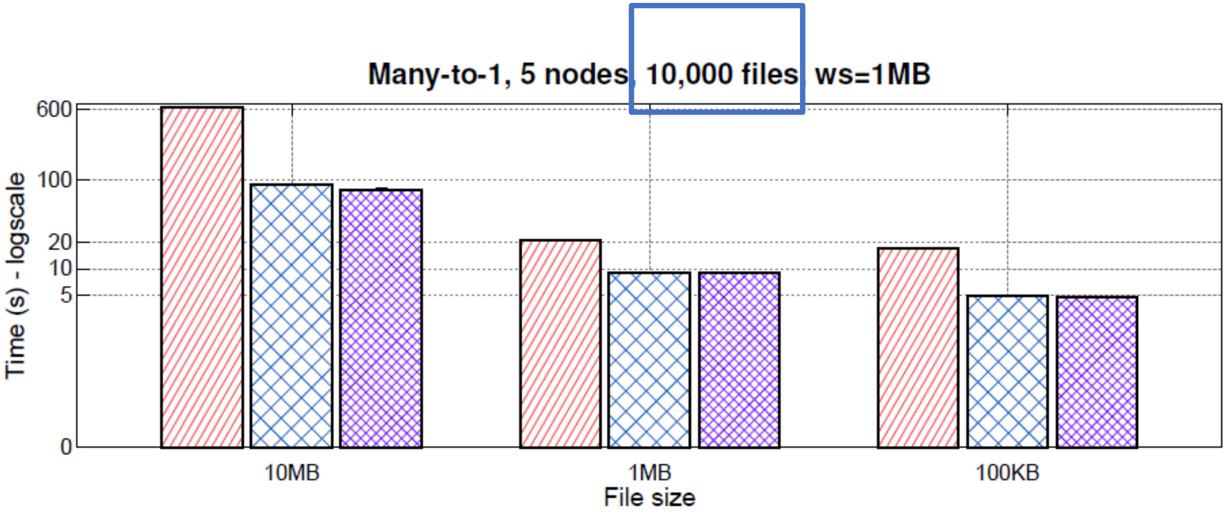
Benchmarks on Galileo100



CAPIO config-file (JSON)

```
{ "name" : "benchmarks",
  "IO_Graph" : [
    {"name" : "S",   "output_stream": [ { "files": ["file*.dat"] } ],
     "streaming": [ { "name" : "file*.dat", "committed":"on_close", "mode":"MODE"}] },
    {"name" : "Q",   "input_stream" : ["file*.dat"] }
  ]
}
```


1-to-1, 2 nodes, 10GB dataset ws=1KB


Many-to-1, 5 nodes 10,000 files ws=1MB