FIDES

ITDM
your technology partner

Extending a scientific workflow engine with streaming I/O capabilities: DAGonStar and CAPIO

DagOn CAPIO
Directed Acyclic Graph on Cross-Application Programmable I/O

Simone Perrotta, Ciro Giuseppe De Vita, Gennaro Mellone, Marco Edoardo Santimaria, Giuseppe Salvi, Marco Lapegna, Massimo Torquati, Angelo Ciaramella

Introduction

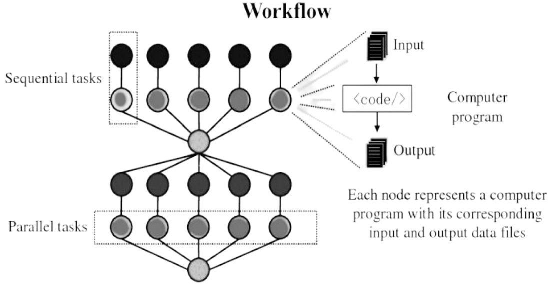
1.1 Introducing data-intensive workflows

Data-intensive workflows:

- Involves complex **sequences** of computational **tasks**;
- Requires **resilient** systems for effective data flow and processing.

Challenges with Traditional Workflow Engines:

- Significant **limitations** with **real-time data streams**;
- Struggles with **in-memory** data management;
- Increasing data complexity and scale exacerbate these issues.



Introducing WFEs and DAGonStar

Workflow Engines (WFEs):

- Designed to manage complex scientific workflows;
- Example: **DAGonStar**.

Functionality:

- Use directed acyclic graphs (**DAGs**) to ensure correct data flow;
- Enable parallel **task execution**.

Limitations:

- Performance can be limited by reliance on traditional disk-based storage.

Introducing CAPIO

CAPIO:

- Innovative **in-memory file storage system**.

Purpose:

- Overcomes limitations of **disk-based storage** in high-performance computing.

Benefits:

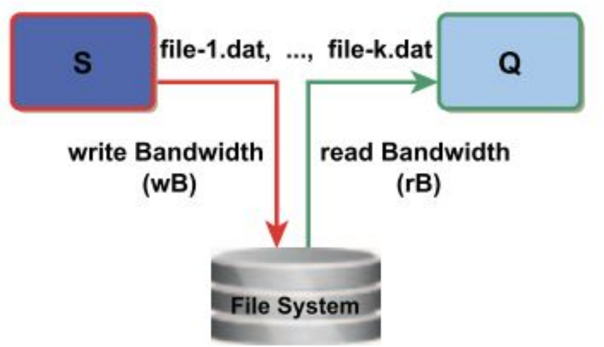
- Faster **data access**;
- Reduced **latency** crucial for **real-time processing**.

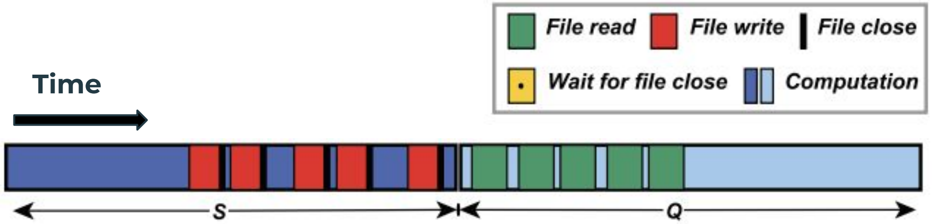
Architecture:

- Supports **concurrent access**;
- Facilitates **parallel processing**;
- Ideal for managing high-speed **data streams** in modern scientific workflows.

Why integrate DAGonStar with CAPIO?

- **Workflow Description**: DAGonStar uses the workflow://schema to describe workflows as dataflows. This means that by analyzing the data flow processed and managed by the various tasks, we can perform I/O overlap to save a significant percentage of the total execution time.

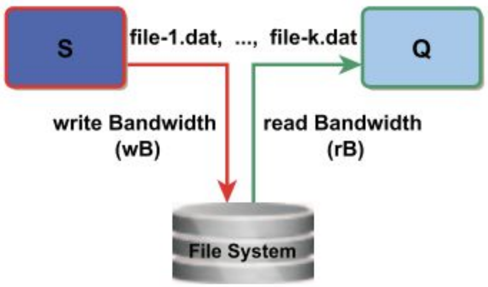


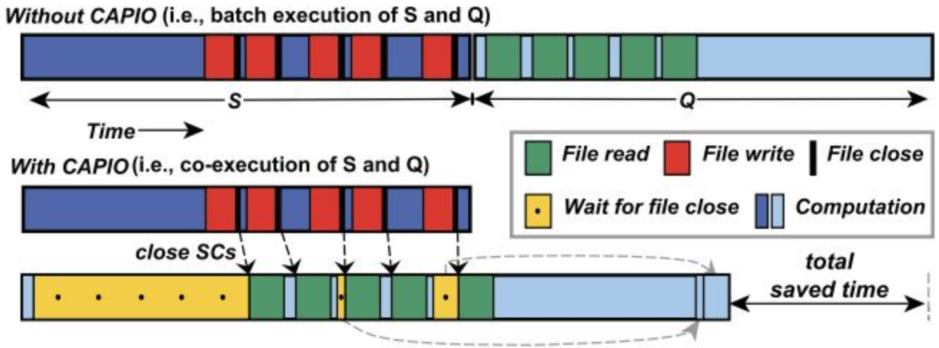


Normal batch execution without I/O overlap

Timeline of integration

- **Current State**: DAGonStar with workflow://schema, no I/O overlap.
- **Integration Goal**: Combine DAGonStar's robust workflow description with CAPIO's efficient streaming I/O.
- **Expected Outcome**: Achieve simultaneous computation and I/O for improved performance and efficiency.





Our objective

Integration of CAPIO with DAGonStar:

- Creates a **hybrid system**.

Combination:

- Efficient task orchestration (**DAGonStar**);
- High-speed, low-latency data handling (**CAPIO**).

Paper Details:

- Design and implementation;
- Highlighting how CAPIO's streaming I/O capabilities enhance **DAGonStar's performance**.

Primary Objective:

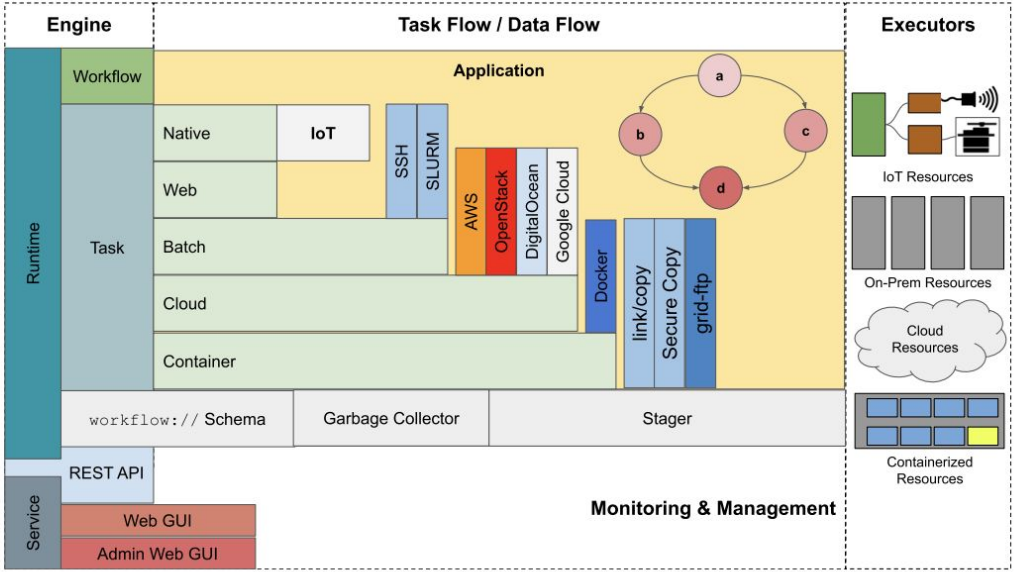
- Demonstrate significant **performance improvement**;
- Particularly for real-time data processing in scientific workflows.

Design and architecture

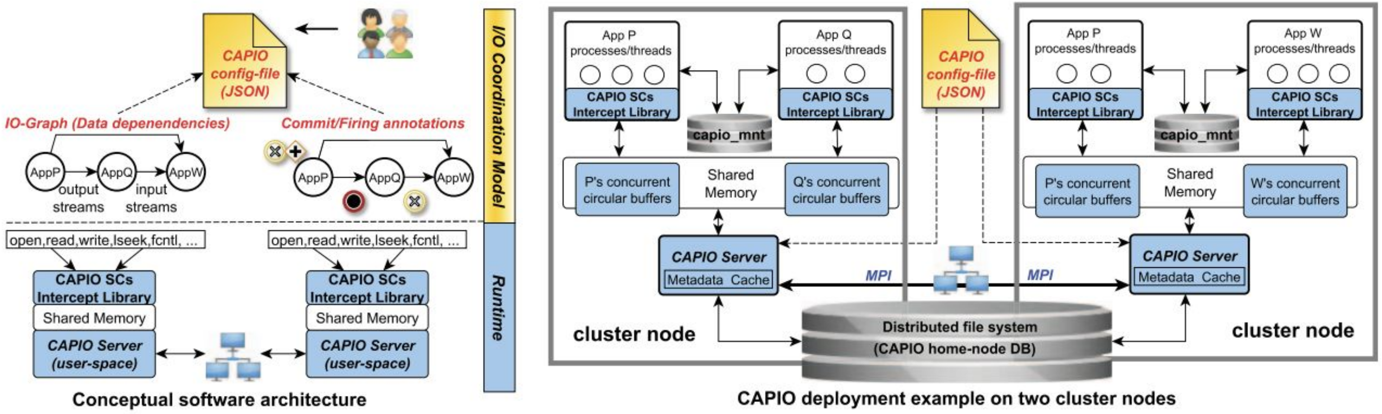
DAGonStar’s architecture

Principal components:

- **Runtime**;
- **Service**;
- **Workflow:// Schema**;
- **Garbage collector**;
- **Stager**;

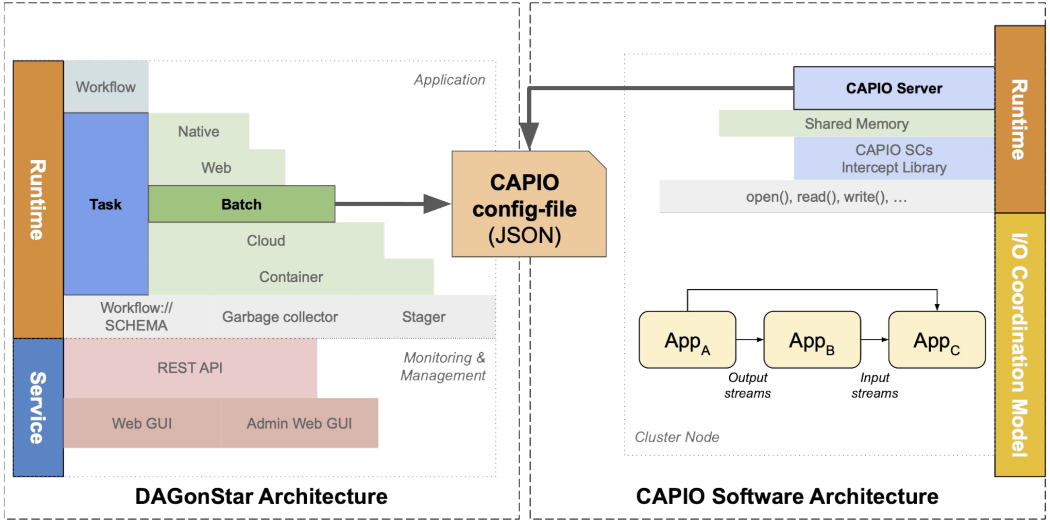


CAPIO's architecture



- The CAPIO **server**, which will run on each node belonging to the cluster. A **JSON** configuration file must be passed to this during execution, which indicates how and where the **streaming** must be carried out, and will generally be produced by users or software;
- The CAPIO **system call intercept library**, a library that allows the CAPIO server to stream by intercepting essential posix calls regarding file management.

Our architecture



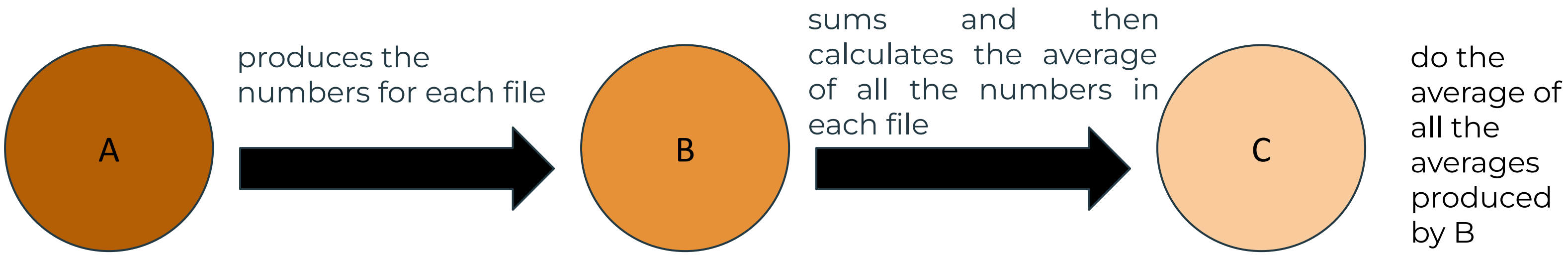
- DAGonStar batch tasks generate the **JSON** file based on the **dependencies** between **tasks**, identified thanks to the **workflow:// Schema**;
- This JSON file is used by the CAPIO server for **configuration**;
- Tasks A and B make up a pipeline in which A produces files and B reads them;
- Posix calls made on these output files will be **intercepted** by the CAPIO server, allowing it to process this data in RAM.

Case studies

Introducing the pipeline

The presented case studies all focus on the use of a **pipeline**, which includes:

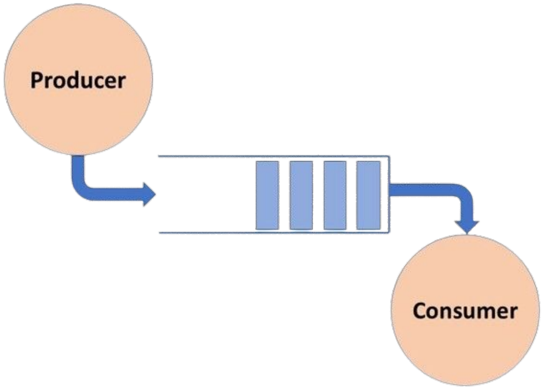
- **Producer** A: which **generates** numbers by inserting them into files;
- **Consumer** B: which **reads** these files, **sums** all the numbers within each file, calculates the **average**, and saves it in another file.
- In our scenario, there is also another component of the pipeline, **C**: which **opens** all the files produced by B, and computes the **average of all the individual averages**.



Pipeline implementation

The implementation of the pipeline was carried out following these **steps**:

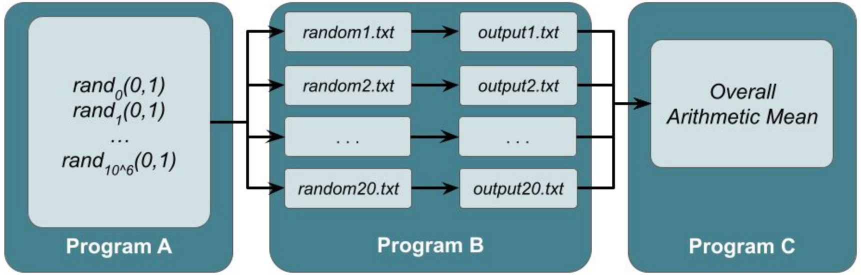
- Implementation of the **pipeline** composed of two C programs, namely **A** and **B** in **CAPIO**;
- Identify the points in DAGonStar to modify for integration purposes and apply these improvements;
- Create two tasks that make up the pipeline in **DAGonStar** plus another task that saves the results **permanently**;
- Run the pipeline workflow and collect **timing results** for comparison.



Experimented pipelines

There were various types of pipelines tested, but they all have in common the type of numbers within the files, as they are all between **0** and **1** with a decimal precision of **6** digits. The specific types of pipelines experimented with are as follows:

- 10 files with 1 million numbers per file;
- 10 files with 2 million numbers per file;
- 20 files with 1 million numbers per file;
- 20 files with 2 million numbers per file;
- 30 files with 1 million numbers per file;
- 30 files with 2 million numbers per file;
- 40 files with 1 million numbers per file;
- 40 files with 2 million numbers per file.

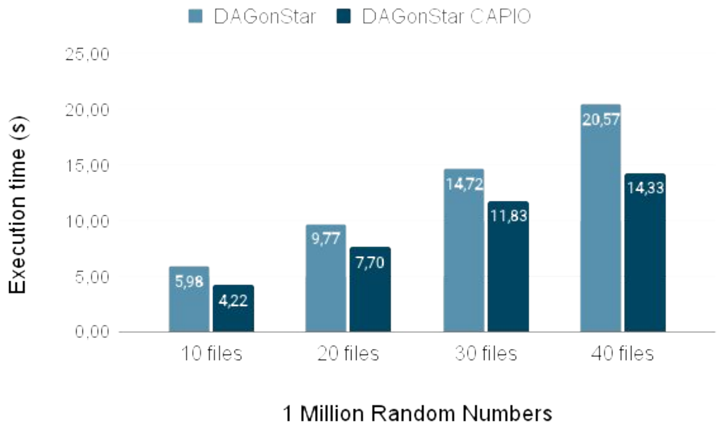


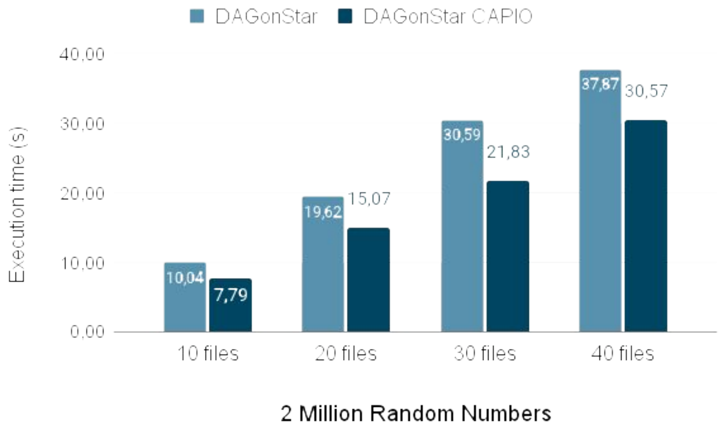
Evaluation and results

We tested the pipeline in two scenarios:

- DAGonStar running **bash scripts sequentially**;
- DAGonStar with **CAPIO integration**.

Execution times were recorded from the start of Program A to the end of Program C to compare performance gains.





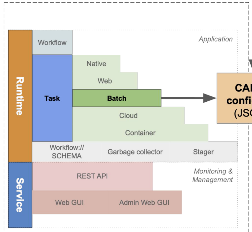
Conclusions

This work was carried out according to the following points:

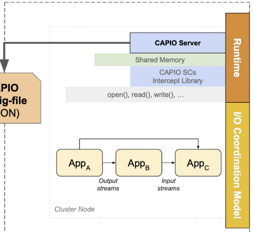
- **Exploration**: Examined workflows, WMS, and DAGonStar;
- **Study**: Analyzed CAPIO middleware;
- **Integration**: Integrated CAPIO into DAGonStar.

The results of this work have shown that:

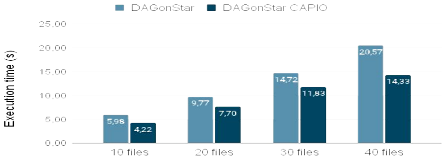
- Execution times were reduced by **20%** to **32%**;
- There are **significant benefits** of using **RAM-based file systems** in **Workflow Management Systems.**



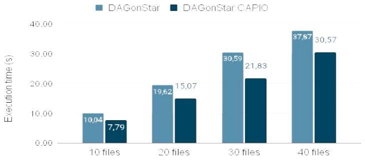
DAGonStar Architecture



CAPIO Software Architecture



1 Million Random Numbers



2 Million Random Numbers