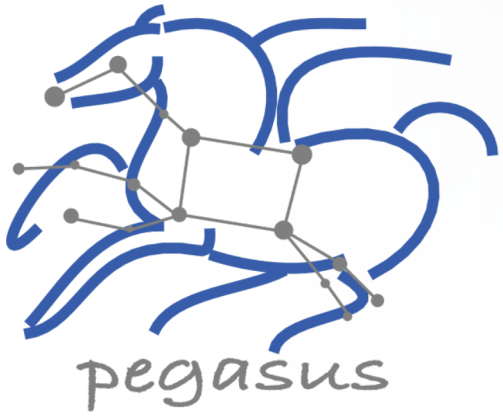



# Accelerating Scientific Workflows on HPC Platforms with In Situ Processing

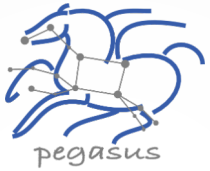

## Loïc Pottier

University of Southern California, School of Engineering
Information Sciences Institute
lpottier@isi.edu


USC Viterbi School of Engineering Information Sciences Institute

# Outline

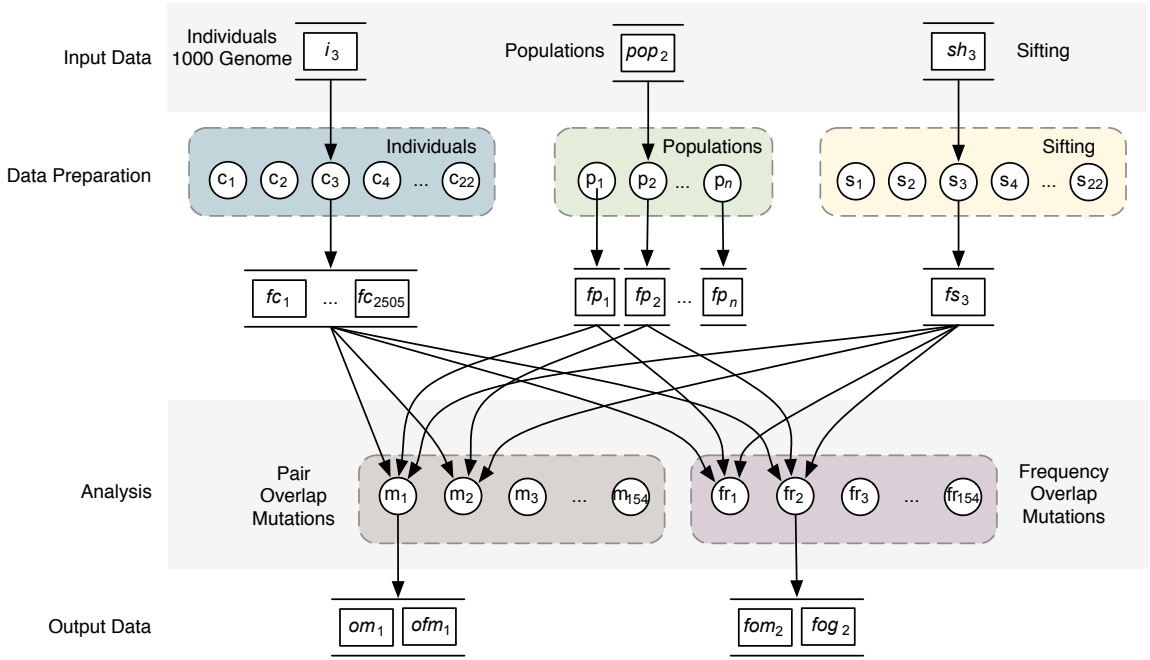
- Context
- General Approach
- Experimental Setup
- Experimental Results
- Conclusion




# Context

# Scientific Workflows

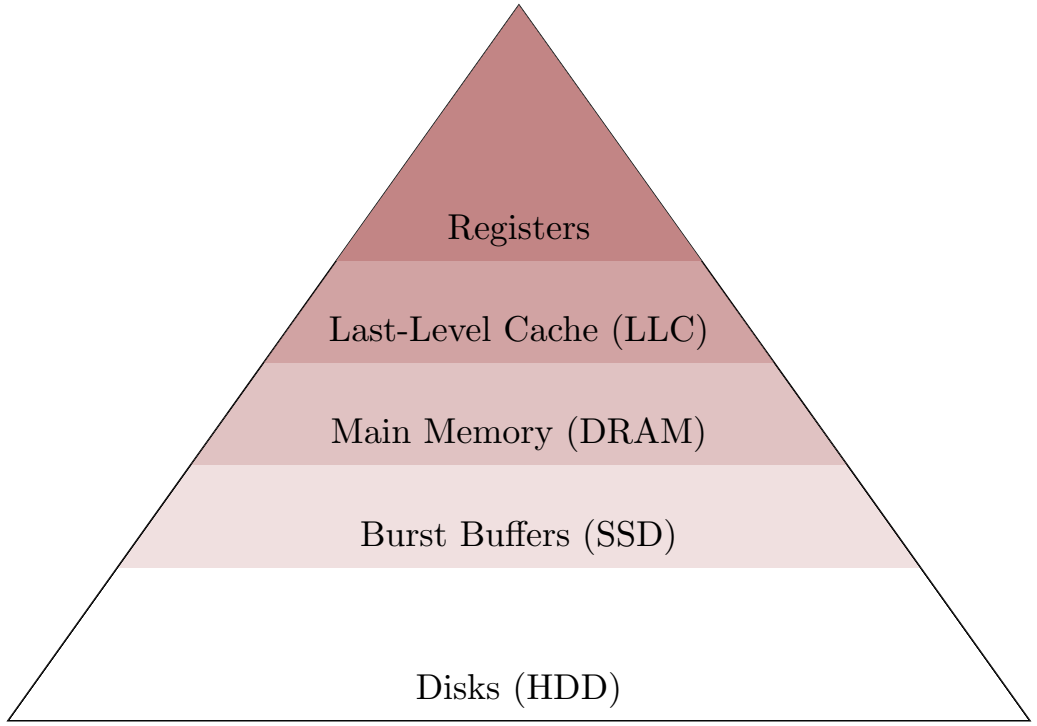
- Workflows represented as **directed acyclic graphs** (DAG)
- Workflow management systems (WMS) execute workflows on distributed resources
- Vast majority of WMS uses **files** to communicate between jobs




- Set of jobs executed in a given order based on their data dependencies

# Why in situ matters

- From post-processing to iterative processing
- Popular in molecular dynamics for example
- Simulations send data every k iterations to some analysis kernels

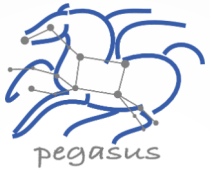



- In situ helps overcome I/O bottlenecks (slow filesystems etc)

# Goals

- WMSs use file-based I/Os with loosely-coupled jobs (HTC model)
- In situ frameworks often rely on in-memory computing
- Minimize code modifications of existing workflows

How to integrate in situ technology with traditional WMS?
⇒ **job clustering**




# General Approach

# Pegasus and Decaf

## Pegasus

- Workflow management system, runs static DAGs
- Well-established WMS (started in 2001)
- Relies on HTCondor for its execution back-end
- Pegasus relies on files to synchronize jobs

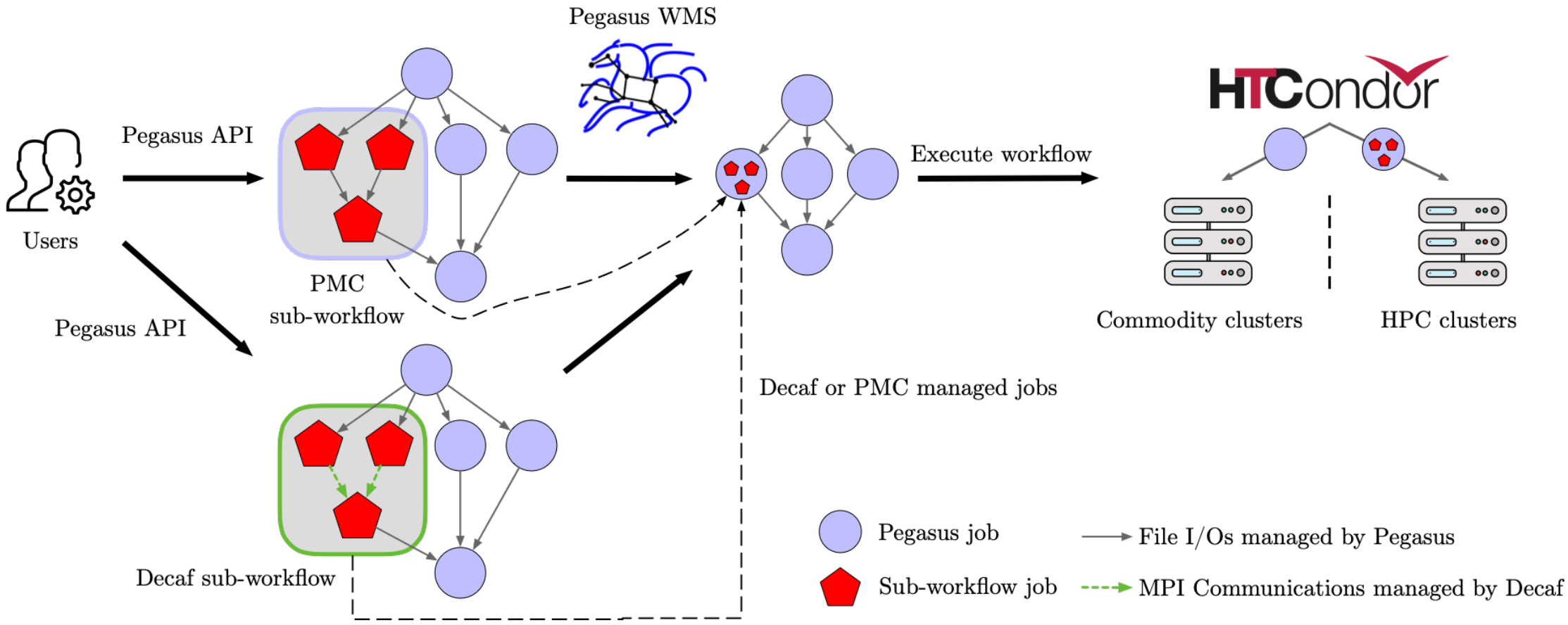
## Decaf

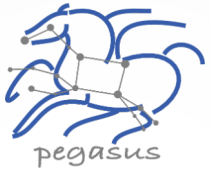
- Middleware for building and executing in situ workflows (from ANL)
- Producer/consumer model using MPI communicators (point to point)
- Multiple-program-multiple-data (MPMD) model

# Job Clustering

- Users express their computations using the **Pegasus API**
- Job clustering:
  - Cluster jobs with the **pegasus-mpi-cluster** (PMC) engine
  - Users simply **annotate the jobs** that have to be clustered together
- Pegasus **automatically infers** the correct Decaf/PMC representation and creates the appropriate workflow representation

# Integration






# Experimental Setup

# Platform

- **Cori** at National Energy Research Scientific Computing Center (NERSC)
- Cray XC40 with:
  - 2 Intel Xeon E5-2698 v3 (16 cores each)
  - 128 GB of DRAM
  - Cray Aries interconnection network
- **Only** CPU nodes in this work
- Pegasus submits to **Slurm**

# SyntheticIO

- Each job **reads/writes** a file of x GB (x ∈ {1, 2, 4, 8, 16})
- Each job **sleeps** for 2 seconds per GB written
- We cluster **all jobs together** (one cluster)

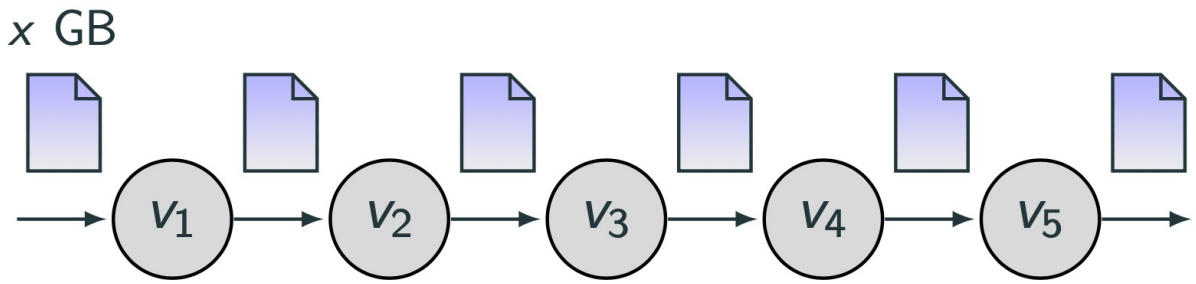



**Figure 3:** SyntheticIO with 5 jobs, each job reads/writes a file of $x$ GB ($x \in \{1, 2, 4, 8, 16\}$).

# Genome

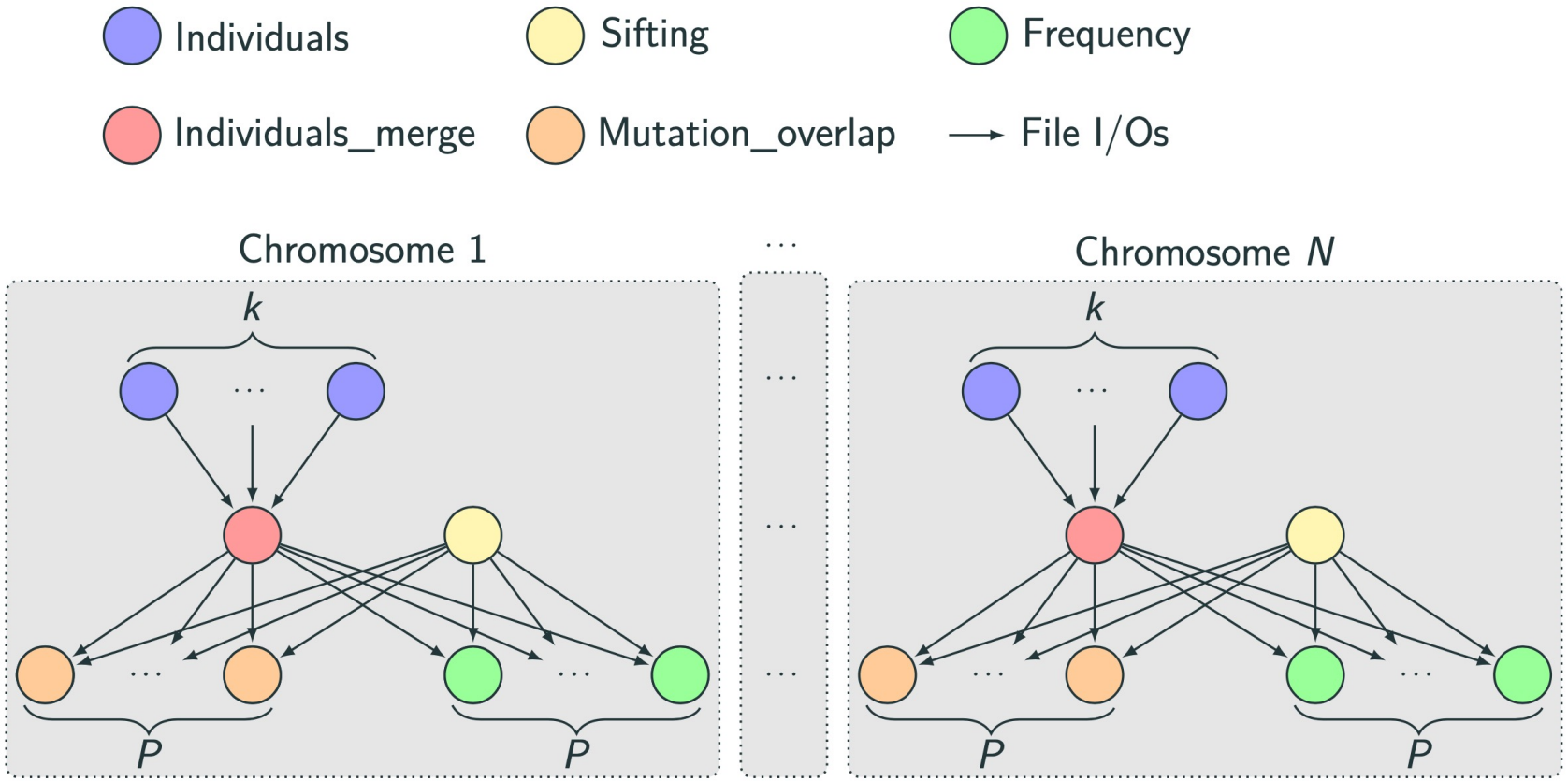



**Figure 4:** GENOME workflow with $N$ chromosomes and $k$ *Ind* jobs per chromosome and $P$ super populations ($N = 1$ and $P = 7$ in this study).

# Execution time breakdown for Genome

| Job | Execution Time (s) | Fraction (%) |
|---|---|---|
| *Ind* | 11,431 | 81.85 |
| *Frequency* | 1,492 | 10.68 |
| $Ind_{Merge}$ | 500 | 3.58 |
| *Mutation_Overlap* | 468 | 3.35 |
| *Stage_Out* | 34 | 0.24 |
| *Stage_In* | 21 | 0.15 |
| *Auxiliary*[1] | 16 | 0.11 |
| *Sifting* | 6 | 0.05 |
| **Total**[2] | (≈ 3.9h) 13,967 | 100 |

[1] Internal jobs managed by Pegasus.

[2] Total execution time is not the makespan of the workflow, it is simply the sum of all job execution times.

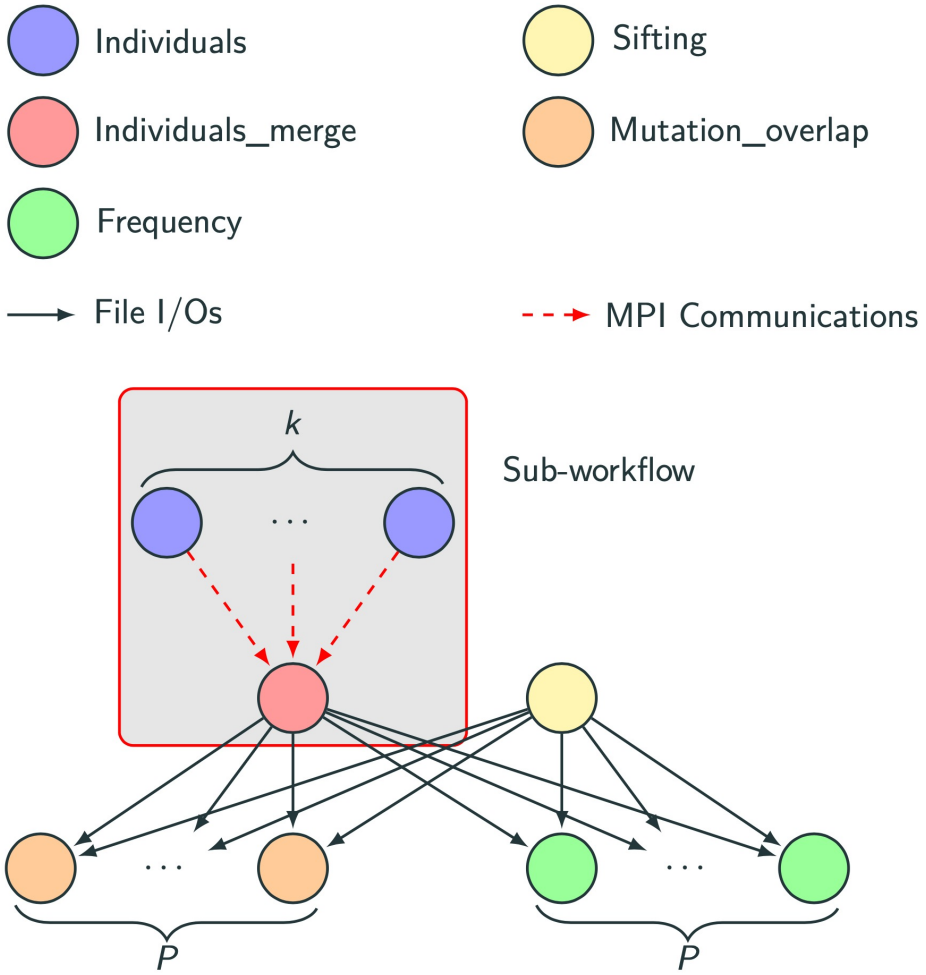
**Table 1:** Execution time breakdown within GENOME with $k = 10$ *individuals* jobs and 1 chromosome. This instance has been executed using Cori at NERSC.

# I/O characteristics for *Inds* jobs

| # of *Ind* ($k$) | Input per *Ind* (lines) | Output size per *Ind* (MB) | Peak Mem. per *Ind* (GB) |
|---|---|---|---|
| 2 | 125, 000 | 92.66 ($\pm$1.88e-04) | 6.11 ($\pm$1.76e-05) |
| 5 | 50, 000 | 39.43 ($\pm$2.28e-04) | 3.95 ($\pm$7.94e-06) |
| 10 | 25, 000 | 21.19 ($\pm$9.90e-04) | 3.25 ($\pm$7.94e-06) |
| 16 | 15, 625 | 10.33 ($\pm$1.41e-04) | 2.93 ($\pm$1.49e-04) |

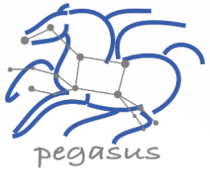
**Table 2:** I/O characteristics of *Ind* jobs in GENOME (2, 504 files/job). Each value is the result of 3 trials.

# Job clustering for Genome



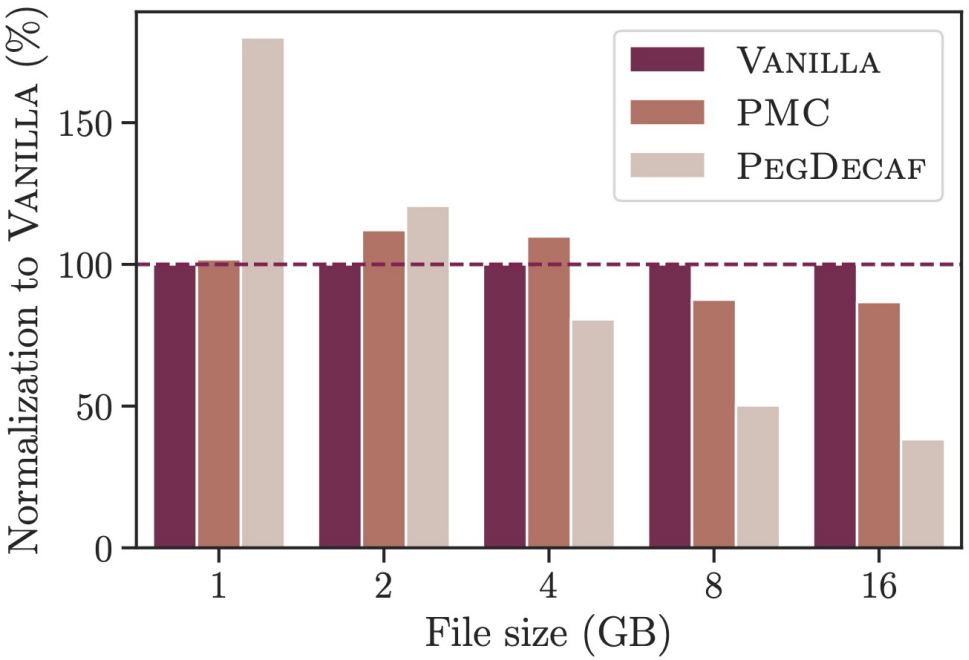
# Execution scenarios

- **Vanilla**: Baseline scenario, no job clustering and file-based communications
- **PMC**: Leverages Pegasus-MPI-Cluster (PMC) to execute portion of the workflow (sub-workflow)
- **PegDecaf**: Leverages Decaf (MPI communications) to execute portion of the workflow

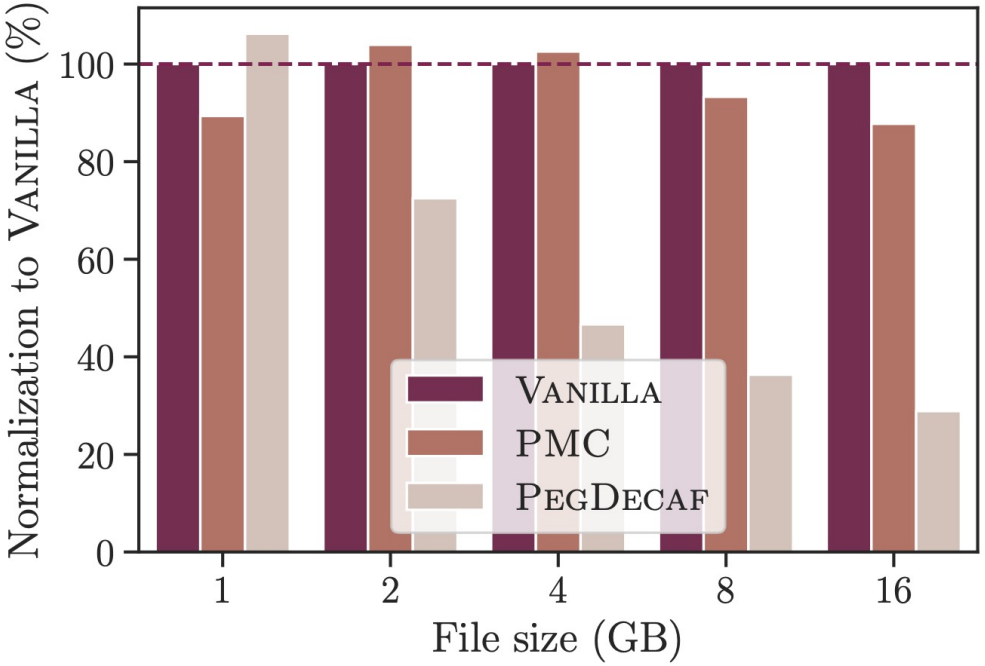



# Experimental Results

# SyntheticIO

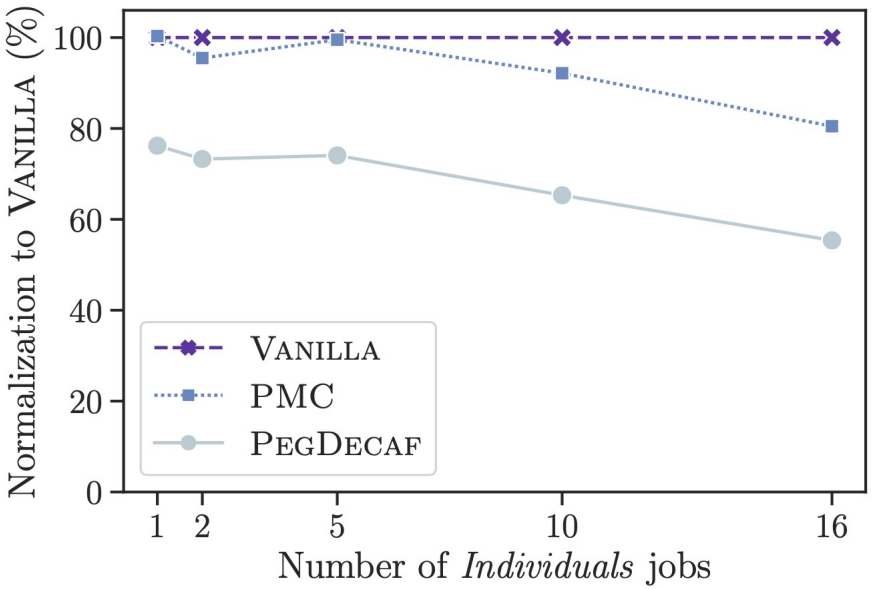



**(a)** Normalized makespan over VANILLA without sleep

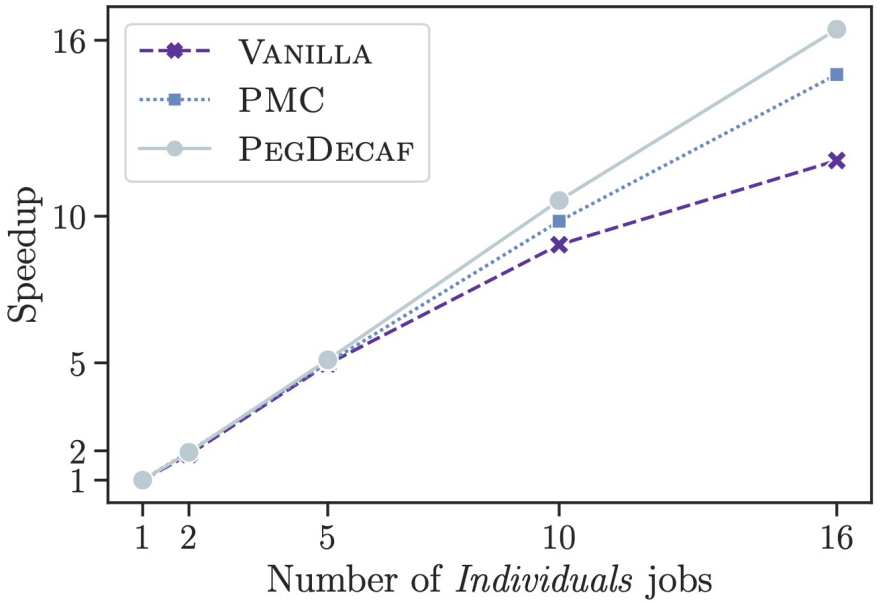



**(b)** Normalized makespan over VANILLA with sleep (2 seconds per GB)

# Genome – Strong scaling




(a) Normalized makespan over VANILLA

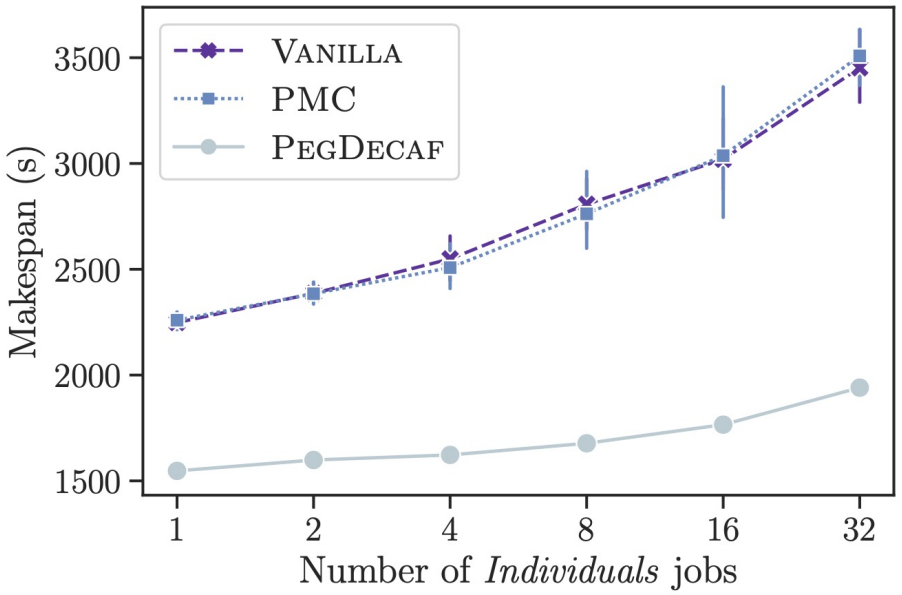



(b) Strong scaling speedup

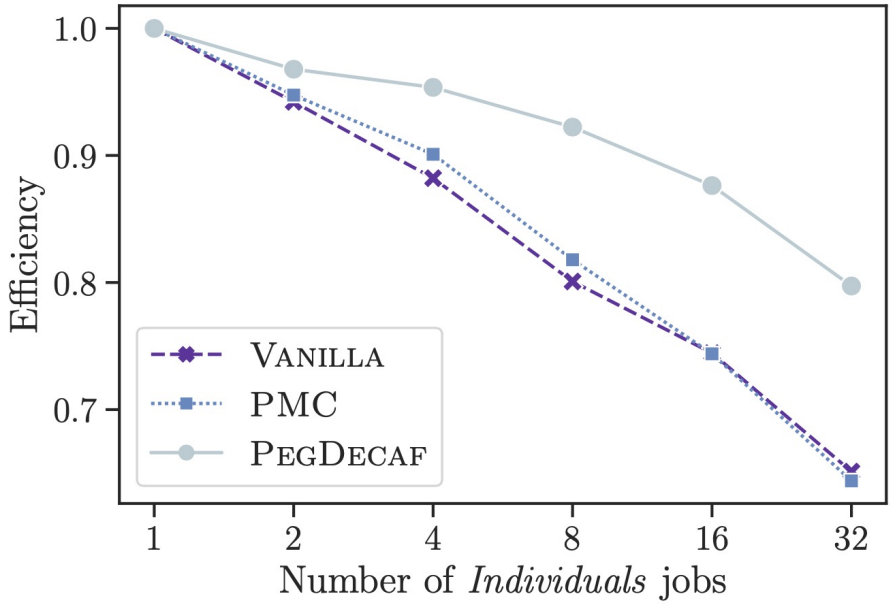
**Figure 6:** Normalized makespan and speedup of GENOME with 1 chromosome.

# Genome – Weak scaling

- Each **Ind** job runs on **one dedicated** node

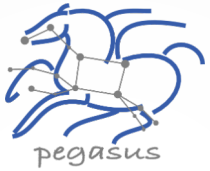



**(a)** Weak scaling makespan




**(b)** Weak scaling efficiency

**Figure 7:** Weak scaling study of Genome where each *Ind* processes 5,000 lines.




# Conclusion

# Conclusion

## Summary

- In situ communications improves the makespan of data-intensive workflows
- Larger improvements when communications can be overlapped with computations
- Job clustering improve wall time thanks to less submissions
- But ... which jobs should be clustered together?

## Future Work

- Release Decaf support in the next version of Pegasus
- Develop heuristics to determine appropriate clusters
- Extend experiments to larger workflows with in situ components (e.g., MD)