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

https://pegasus.isi.edu
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

Users interact with Pegasus API to submit jobs. Pegasus WMS manages the workflow, executing jobs on either Decaf or PMC managed clusters. File I/Os are managed by Pegasus, and MPI Communications are handled by Decaf.

HPC clusters

Commodity clusters

Pegasus job

Sub-workflow job

→ File I/Os managed by Pegasus

--- MPI Communications managed by Decaf

https://pegasus.isi.edu
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 ($x \in \{1, 2, 4, 8, 16\}$)
- 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 ($x \in \{1, 2, 4, 8, 16\}$).
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

<table>
<thead>
<tr>
<th>Job</th>
<th>Execution Time (s)</th>
<th>Fraction (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ind</td>
<td>11,431</td>
<td>81.85</td>
</tr>
<tr>
<td>Frequency</td>
<td>1,492</td>
<td>10.68</td>
</tr>
<tr>
<td>Ind_Merge</td>
<td>500</td>
<td>3.58</td>
</tr>
<tr>
<td>Mutation_Overlap</td>
<td>468</td>
<td>3.35</td>
</tr>
<tr>
<td>Stage_Out</td>
<td>34</td>
<td>0.24</td>
</tr>
<tr>
<td>Stage_In</td>
<td>21</td>
<td>0.15</td>
</tr>
<tr>
<td>Auxiliary(^1)</td>
<td>16</td>
<td>0.11</td>
</tr>
<tr>
<td>Sifting</td>
<td>6</td>
<td>0.05</td>
</tr>
<tr>
<td><strong>Total(^2)</strong></td>
<td>(≈ 3.9h) 13,967</td>
<td>100</td>
</tr>
</tbody>
</table>

\(^1\) 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**

<table>
<thead>
<tr>
<th># of <strong>Ind</strong> (<strong>k</strong>)</th>
<th>Input per <strong>Ind</strong> (lines)</th>
<th>Output size per <strong>Ind</strong> (MB)</th>
<th>Peak Mem. per <strong>Ind</strong> (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>125,000</td>
<td>92.66 (±1.88e-04)</td>
<td>6.11 (±1.76e-05)</td>
</tr>
<tr>
<td>5</td>
<td>50,000</td>
<td>39.43 (±2.28e-04)</td>
<td>3.95 (±7.94e-06)</td>
</tr>
<tr>
<td>10</td>
<td>25,000</td>
<td>21.19 (±9.90e-04)</td>
<td>3.25 (±7.94e-06)</td>
</tr>
<tr>
<td>16</td>
<td>15,625</td>
<td>10.33 (±1.41e-04)</td>
<td>2.93 (±1.49e-04)</td>
</tr>
</tbody>
</table>

**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

- Individuals
- Individuals_merge
- Frequency
- Sifting
- Mutation_overlap

File I/Os

MPI Communications

Sub-workflow

k

\( P \)
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
Synthetic I/O

(a) Normalized makespan over VANILLA without sleep

(b) Normalized makespan over VANILLA with sleep (2 seconds per GB)
Genome – Strong scaling

Figure 6: Normalized makespan and speedup of Genome with 1 chromosome.
Genome – Weak scaling

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

**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)