Ubique: A New Model for Untangling Inter-task Data Dependence in Complex HPC Workflows

Jae-Seung Yeom*, Dong H. Ahn†, Ian Lumsden‡,
Jakob Luettgau‡, Silvina Caino-Lores‡, Michela Taufer‡

*Lawrence Livermore National Laboratory, Livermore, CA, USA
†NVIDIA Corporation
‡Department of Electrical Engineering and Computer Science, University of Tennessee, Knoxville, TN, USA

***Index Terms*—job scheduling, workflow, in situ, in transit**

***Abstract*—Exploiting task parallelism is getting increasingly difficult for diverse and complex scientific workflows running on High Performance Computing (HPC) systems. In this paper, we argue that the difficulty rises from a void in the spectrum of existing data-transfer models for resolving inter-task data dependence within a workflow and propose a novel model to fill that gap: Ubique. The Ubique model combines the best from in-transit and in situ models in order for loosely coupled producer and consumer tasks to run concurrently and to resolve their data dependencies efficiently with little or no modifications to their codes, striking a balance between transparent optimization, productivity, and performance. Our preliminary evaluation suggests that Ubique can significantly outperform the parallel file system (PFS)-based model while offering automatic data transfer and synchronization which are the features lacking in many traditional models. It also identifies the performance characteristics of its key depending subsystems, which must be understood for further broadening its benefits.**

I. Motivation and Contributions

As a growing number of scientific domains are leveraging HPC to answer some of the most challenging scientific questions, HPC is quickly becoming the third "pillar" of science, in addition to theory and experiments. However, this increasing diversity also creates a greater convergence of traditional HPC with new modeling and simulation methods as well as data analysis and data science approaches like Machine Learning (ML) and Artificial Intelligence (AI). A myriad of these methods are often combined into scientific workflows in many new and exciting ways [1]–[4], ushering in a new era of diverse and complex scientific workflows.

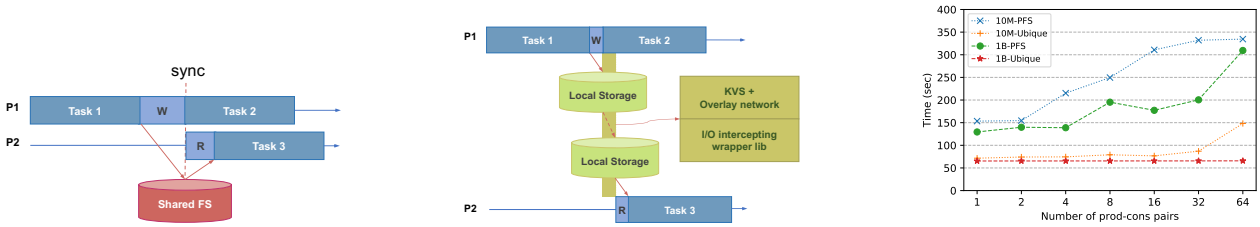
One of the defining features of these workflows is complex dependence between their tasks. In particular, how data files are produced by one task and subsequently consumed by another task represents one of the most important dependence classes for a workflow to proceed in a performant fashion. Specifically, the inter-task data dependence known as the producer-consumer pattern is commonly found in a wide range of emerging workflow initiatives including those that incorporate ML and AI [2], [5], [6].

As composite science workflows become more and more complex, researchers find the construction and optimization of a workflow increasingly challenging. In this scheme, they select preexisting composable workflow-management software components and flexibly blend them with a disparate set of domain-specific application and management software so as to create a scalable end-to-end science workflow [1]–[7].

There exists various approaches to resolve the inter-task data dependence, including those based on a PFS, a *in transit* model [2], [8]–[10] or an *in situ* model [11]–[16]. However, each of these approaches retains one or more of the following major drawbacks:

- *Lack of synchronization support at the file/data object level:* requires workflow themselves to synchronize consumer and producer tasks to handle cases like a consumer task attempting to read a file before the producer completes its writing;
- *Conflict with code change requirements:* Many emerging workflows compose reusable components with minimum or no code change requirement on the pre-existing programs, and hence extensive changes needed to implement the aforementioned synchronization mechanism are often a non-starter;
- *Poor temporal/spatial locality:* Workflows use coarse grained synchronization thereby a consumer task does not start its program execution before its dependent producer finishes its entire program execution, incurring distant temporal distance to resolve a data dependency, and each file travels a long spatial distance, missing bypass opportunities;
- *Low file metadata-operation performance:* Massive numbers of small files are often employed for emerging ML-based workflows, and hence the performance of file transfers is ultimately limited by the metadata performance of the PFS.

Hence, we propose Ubique, a novel data file transfer model to overcome the limitations of the existing models in resolving inter-task data dependence. The Ubique model combines the best from in situ and in-transit models to offer highly performant—yet non-intrusive–solutions for workflow use. Ubique is specifically defined by three properties: a) producer and consumer tasks within a workflow write/read files to/from a Ubique-managed file-paths name space; b) Ubique is solely responsible for optimizing how produced files are transferred between tasks and how the transfer operations remain transparent to the tasks; c) Ubique synchronizes file access at individual file level, removing the need for the workflow itself to implement its own synchronization mechanism.



(a) PFS (b) Ubique (c) makespan performance

Fig. 1. Comparison of producer-consumer benchmark runs

Specifically, we make the following contributions:

- The design points of Ubique in relation to other data-transfer models that are in use for inter-task data dependence;
- We define transparent file-level synchronization techniques and an evaluation of transparency for a representative HPC language at the Application Binary Interface (ABI) level;
- We perform an empirical exploration of the performance and scalability.

A. Transparency and intrusiveness

Ubique delivers its benefits without requiring users to modify their code. It currently provides a complete transparent interface for C language. The C library contains the wrapper functions of `open()/close()` and `fopen()/fclose()`, and intercepts C file I/O calls via dynamic linking and symbol replacement. This is possible because C runtimes provide ABI-level guarantees to allow for transparently intercepting our target POSIX file I/O calls:

```
LD_PRELOAD=libdyad_sync.so:${LD_PRELOAD} ./app
```

B. Performance evaluation

Figure 1 compares the performance with PFS and with Ubique. With the former, producer-consumer pairs share data files via Lustre. With the latter, files are written to and read from tmpfs. For the files of size 1B and 10M, Ubique outperforms the alternative by a factor of 3 on average and 4.7 maximum. We set the interval between successive readings or writings to 1 sec while transferring 64 files by each producer-consumer pair, mimicking the computation load of both producers and consumers.

II. Conclusion

In this paper, we propose Ubique, a new data file transfer model to enable modern HPC workflows to expose and exploit task-level parallelism. For this purpose, it transparently resolves data dependence between workflow tasks via fine-grained synchronization and direct data file transfers over network while avoiding persistent storage I/O bottlenecks. Our initial evaluation shows that Ubique outperforms the PFS-based model for small to medium size data files by a factor of 3 on average. Our study significantly lights our path to advancing our proposed model for broader HPC workflow use.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-ABS-837680

References

[1] D. H. Ahn, N. Bass *et al.*, "Flux: Overcoming scheduling challenges for exascale workflows," *Future Generation Computer Systems*, vol. 110, pp. 202–213, 2020.

[2] H. Bhatia, F. D. Natale *et al.*, "Generalizable coordination of large multiscale workflow: Challenges and learnings at scale," in *ACM/IEEE Conf. on Supercomputing*, 2021.

[3] C.-C. Yang, G. Domeniconi *et al.*, "Design of ai-enhanced drug lead optimization workflow for hpc and cloud," in *IEEE Int. Conf. on Big Data*, Dec. 2020, pp. 5861–5863.

[4] A. J. Minnich, K. McLoughlin *et al.*, "AMPL: A Data-Driven Modeling Pipeline for Drug Discovery," *J. of Chemical Information and Modeling*, vol. 60, no. 4, pp. 1955–1968, 2020.

[5] J. L. Peterson et al., "Enabling machine learning-ready HPC ensembles with Merlin," *Future Generation Computer Systems*, vol. 131, pp. 255–268, June 2022.

[6] S. A. Jacobs, T. Moon *et al.*, "Enabling rapid covid-19 small molecule drug design through scalable deep learning of generative models," *The Int. J. of High Performance Computing Applications*, May 2021.

[7] Rob Farber, "Workflow Technologies Impact SC20 Gordon Bell COVID-19 Award Winner and Two of the Three Finalists," https://www.exascaleproject.org/workflow-technologies-impact-sc20-gordon-bell-covid-19-award-winner-and-two-of-the-three-finalists, 2020.

[8] H. Abbasi, M. Wolf *et al.*, "DataStager: Scalable Data Staging Services for Petascale Applications," in *ACM Int. Symp. on High Performance Distributed Computing*, 2009, pp. 39–48.

[9] C. Docan, M. Parashar, and S. Klasky, "Dataspaces: An interaction and coordination framework for coupled simulation workflows," in *ACM Intl. Symp. on High Performance Distributed Computing*, 2010.

[10] C. Docan, M. Parashar *et al.*, "Moving the Code to the Data - Dynamic Code Deployment Using ActiveSpaces," in *IEEE Int. Parallel and Distributed Processing Symp.*, 2011.

[11] T. Tu, H. Yu *et al.*, "From Mesh Generation to Scientific Visualization: An End-to-End Approach to Parallel Supercomputing," in *ACM/IEEE Conf. on Supercomputing*, 2006.

[12] H. Yu, T. Tu *et al.*, "Remote Runtime Steering of Integrated Terascale Simulation and Visualization," in *ACM/IEEE Conf. on Supercomputing*, 2006.

[13] H. Yu, C. Wang *et al.*, "In situ visualization for large-scale combustion simulations," *IEEE Computer Graphics and Applications*, vol. 30, no. 3, pp. 45–57, 2010.

[14] N. Fabian, K. Moreland *et al.*, "The paraview coprocessing library: A scalable, general purpose in situ visualization library," in *IEEE Symp. on Large Data Analysis and Visualization*, 2011.

[15] B. Whitlock, J. M. Favre, and J. S. Meredith, "Parallel in Situ Coupling of Simulation with a Fully Featured Visualization System," in *Eurographics Conf. on Parallel Graphics and Visualization*, 2011.

[16] S. Lakshminarasimhan, J. Jenkins *et al.*, "ISABELA-QA: Query-Driven Analytics with ISABELA-Compressed Extreme-Scale Scientific Data," in *ACM/IEEE Conf. on Supercomputing*, 2011.