Data Lifecycles for Optimizing Data Movement

Hyungro Lee, Jesun Firoz, Nathan R. Tallent
Pacific Northwest National Lab
{hyungro.lee,jesun.firoz,tallent}@pnnl.gov

Meng Tang, Anthony Kougkas, Xian-He Sun
Illinois Institute of Technology
mtang11@hawk.iit.edu, {akougkas,sun}@iit.edu

Introduction. Scientific exploration is increasingly dependent on the convergence of scientific modeling, data analytics, and machine learning. The result is data-intensive workflows that are composed of multiple stages of computation and communication between distributed and heterogeneous computing resources. Data movement through storage systems is frequently the most significant bottleneck, which is compounded by increasingly large data volumes and rates. To identify opportunities for optimizing data movement, we are developing novel workflow telemetry that highlights data objects' dynamic flow, reuse, lifetime, and locality. Our objective is to enable modeling and reasoning about task-data locality, especially compared to default placement and data exchange, and the scheduling of anticipatory data movement that selects what data should be staged in memory and when.

Related Work. The previous work on improving I/O performance for scientific workflows has focused on the hardware aspect of data movement such as memory, storage, or network and has overlooked logical data information to expose. For example, techniques for NVRAM-optimized filesystems [1] and burst buffers [2] have shown I/O streaming and separation, but study on data life cycles is scarce [3] to manage bottlenecks.

Approach. To gain insight about the data access pattern, our approach involves construction of a "producer-consumer" graph as a first step to capture the data life cycle. Vertices in this graph may denote either tasks or data objects. For example, the simulation tasks in DeepDriveMD [4] can be considered as "producers" that generate data objects and the aggregator task reads these data as "consumers". While gathering telemetry, we record the access frequencies of the data blocks (at the byte level). An edge between a pair of vertices depicts the relationship between either a producer task and the data object generated, or a consumer task and the data object read. The access frequency of each block then can be placed as a weight on the edges to denote the importance of the link. We leverage a remote I/O infrastructure, TAZeR [5], to intercept and record POSIX I/O calls. Alternatively, as our next objective, we are currently pursuing leveraging HDF5 file semantic information to track accesses in an intelligent way. In doing so, we want to infer whether a *subset* of data is accessed rather than the whole dataset in the downstream tasks and enable only bringing in (cache) the relevant part of the data accessed, rather than the whole dataset. We anticipate that by co-locating data with computation based on the access and by exploiting the deep memory hierarchy to turn the network transfers to memory transfers as much as possible, the penalty associated with staging will be reduced by transforming the network-bound workload to the memory-bound workload.

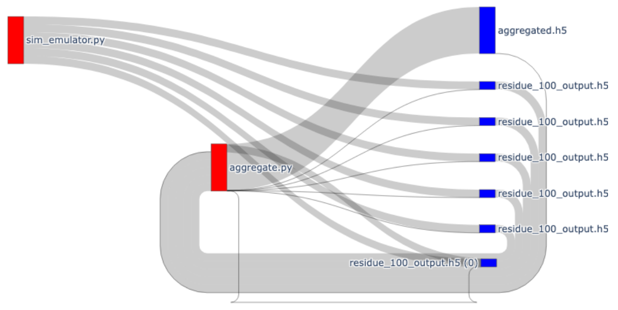

Fig. 1. Data Flow example between Molecular Dynamics Simulation and data preprocessing for ML Training (Concatenation) in DeepDriveMD. Here, nodes represent tasks (red color) and data objects (blue color). The thickness of the links represents data volumes (block access frequency) (grey color).

Results. We collected telemetry from an exemplary workflow, DeepDriveMD, and created a Sankey diagram (Figure 1) to demonstrate the data life cycle. Here, captured data flow between "a producer" task (simulation.py) and "a consumer" task (aggregator.py) reveals actual data movement between the simulations and the preprocessing steps for ML training.

Ongoing/Future work. Our current workflow for capturing data life cycle with TaZeR tracks byte-level information with storage interfaces such as POSIX, disregarding any opportunity for leveraging semantic information found in HDF5. To extract the data's logical names and address (e.g., logical index), we are currently working with the Hermes [6] I/O buffering middleware that interfaces with HDF5's high-level data abstractions. We anticipate that the results will both inform users and identify optimization opportunities to maximize data reuse and locality and minimize data movement.

References

[1] P. Fernando, A. Gavrilovska, S. Kannan, and G. Eisenhauer, "Nvstream: Accelerating hpc workflows with nvram-based transport for streaming objects," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, 2018, pp. 231–242.

[2] D. Koo, J. Lee, J. Liu, E.-K. Byun, J.-H. Kwak, G. K. Lockwood, S. Hwang, K. Antypas, K. Wu, and H. Eom, "An empirical study of i/o separation for burst buffers in hpc systems," *Journal of Parallel and Distributed Computing*, vol. 148, pp. 96–108, 2021.

[3] F. Chowdhury, F. Di Natale, A. Moody, K. Mohror, and W. Yu, "Dfman: A graph-based optimization of dataflow scheduling on high-performance computing systems," in *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2022, pp. 368–378.

[4] H. Lee, M. Turilli, S. Jha, D. Bhowmik, H. Ma, and A. Ramanathan, "Deepdrivemd: Deep-learning driven adaptive molecular simulations for protein folding," in *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*. IEEE, 2019, pp. 12–19.

[5] J. Suetterlein, R. D. Friese, N. R. Tallent, and M. Schram, "TAZeR: Hiding the cost of remote I/O in distributed scientific workflows," in *Proc. of the 2019 IEEE Intl. Conf. on Big Data*. IEEE Computer Society, December 2019, pp. 383–394.

[6] A. Kougkas, H. Devarajan, and X.-H. Sun, "Hermes: a heterogeneous-aware multi-tiered distributed i/o buffering system," in *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*, 2018, pp. 219–230.