HDF5 as a Vehicle for In Transit Data Movement

Junmin Gu, Burlen Loring, Kesheng Wu, E. Wes Bethel
Lawrence Berkeley National Laboratory
{jgu,bloring,kwu,ewbethel}@lbl.gov

ABSTRACT

For *in transit* processing, one of the fundamental challenges is the efficient movement of data from producers to consumers. Exploiting the flexibility offered by the SENSEI generic *in situ* framework, we have developed a number of different *in transit* data transport mechanisms. In this work, we focus on the transport mechanism that leverages the HDF5 parallel I/O library, and investigate the performance characteristics of this transport mechanism. For *in transit* use cases at scale on HPC platforms, one might expect that an *in transit* data transport mechanism that uses faster layers of the storage hierarchy, such as DRAM memory, would always outperform a transport that uses slower layers of the storage hierarchy, such as an NVRAM-based persistent storage presented as a distributed file system. However, our test results show that the performance of the transport using NVRAM is competitive with the transport that uses socket-based data movement across varying levels of producer and consumer concurrency.

CCS CONCEPTS

• **Software and its engineering → Massively parallel systems**; • **Theory of computation** → *Parallel computing models*; • **Computing methodologies** → *Massively parallel algorithms*; *Massively parallel and high-performance simulations*;

KEYWORDS

SENSEI, *in situ* analysis, *in situ* visualization

ACM Reference Format:
Junmin Gu, Burlen Loring, Kesheng Wu, E. Wes Bethel. 2019. HDF5 as a Vehicle for In Transit Data Movement. In *ISAV: In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV'19), November 18, 2019, Denver, CO, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3364228.3364237

1 INTRODUCTION

In an *in transit* processing scenario, one of the central challenges is moving data efficiently from producers to consumers. Currently, the most successful *in transit* mechanisms rely on memory to memory data transfers [6, 8], which are efficient but limit the data size to the size of the memory. In this work, we design and implement an *in transit* transport mechanism that utilizes the file systems through the HDF5 parallel I/O library [5] and NVRAM storage. This transport mechanism is accessible to applications through the SENSEI generic *in situ* interface [2] as one of several potential *in transit* transport mechanisms that can be selected through an XML-based configuration file.

This HDF5 based transport mechanism would make *in transit* process available to tasks that require more space than the available memory. We expect this file-based option to take more time because disks and similar permanent storage media are slower than the main memory. What is unexpected is that, when using the NVRAM option, the file-based transport mechanism often completes analysis use cases in less time than a popular socket-based transport mechanism. This suggests that NVRAM can be effectively used for *in transit* processing.

The contribution of this work is twofold. First, we describe the design and implementation of a new HDF5-based *in transit* data transport mechanism that is accessible via the SENSEI API. Second, our performance measurement results yield some non-obvious insights: namely that using NVRAM-based storage as the basis for *in transit* data movement can outperform a well-known socket-based, memory-only implementation in many configurations on at-scale use scenarios on HPC platforms.

2 PREVIOUS WORK

Many scientific and engineering applications consist of data producers and data consumers, where consumers ingest and process data from the producers. In scientific computing applications, producers are typically simulations, and consumers may be other simulations, or visualization, or analysis tools. In a *post hoc* use case, all data from a producer goes to persistent storage, such as disk files, where they are loaded at a later time for subsequent processing. To minimize I/O, the *in situ* processing paradigm avoids writing data to storage by having consumers operate on the producers' data at the time it is computed or generated, with both consumers and producers executing on the same processors/cores. In *in transit* processing, this idea is broadened to have producers running on one set of MPI ranks, and consumers running on a different set of ranks. In this configuration, data must be moved from producers to consumers. Therefore, moving data efficiently is the key challenge.

One of the earliest works on *in transit* infrastructure is CUMULVS from 2006 [6]. It is designed to couple two different codes and move data between them. Recently, a new set of *in transit* systems have been developed. Here, we briefly describe two of them: libIS and ADIOS. The libIS library [9] is a lightweight vehicle for moving data between producers and consumers on HPC platforms. It uses either MPI- or socket-based communication for communication and memory-memory data movement. The ADIOS system [8] provides the ability to do parallel I/O to disk files, as well as the ability to leverage alternative transport layers that enable its use for *in transit* data movement between third-party producers and consumers. These transport layers include implementations like

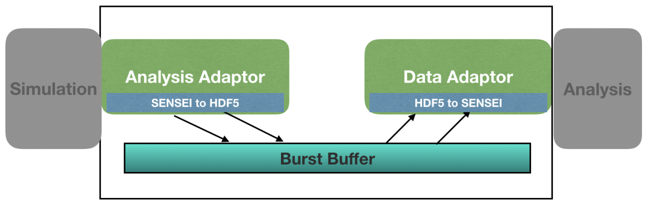

Figure 1: The HDF5 transport layout.

FlexPath [4], which does socket-based communication and memory-memory data movement between producers and consumers.

Given the variety of *in transit* systems, there is an effort to unify them under a common API known as SENSEI [2]. Current implementation of the SENSEI library could make use of many of the well-known transport mechanisms [1]. Our work was initially motivated by a need to expand SENSEI transport options to allow large data sets to be used for analyses. Though it is possible to write out data through ADIOS to disks, the current ADIOS adaptor is only able to make the data available to consumers after the file is closed. Therefore, we believe it is worthwhile to explore another popular I/O library known as HDF5 [5]. The HDF5 adaptor in the SENSEI framework has been designed with *in transit* in mind, so that a data producer writes out data at the end of every time step, and a data consumer can access data as soon as each time step is finished. In later performance studies, we will be comparing ADIOS and HDF5 as two different data transport mechanisms, where ADIOS (version 1.13) represents socket-based *in transit* data transport mechanism and HDF5 (version 1.11) represents the NVRAM option.

3 DESIGN AND IMPLEMENTATION

There are multiple components comprising our HDF5-based *in transit* data transport mechanism. One is the SENSEI-HDF5 component that performs the mappings to/from HDF5 and the underlying SENSEI data model. Another is the NVRAM-based persistent storage on the HPC platform, which is the underlying platform used for our implementation and performance studies.

Fig. 1 shows a high-level sketch of the design of the SENSEI-HDF5 transport mechanism. Following the design patterns used by other transport mechanisms in SENSEI, the HDF5 transport mechanism consists of two related adaptors known as the Analysis Adaptor and the Data Adaptor. The Analysis Adaptor implements the SENSEI interface for outputting data from data producers, and the Data Adaptor implements the input interface for consumers to ingest data. SENSEI is using VTK as the data model. Therefore, in the HDF5 transport, the Analysis Adaptor receives VTK data, and stores to HDF5 format, while the Data Adaptor reads from HDF5 file and returns VTK data for SENSEI.

The SENSEI-HDF5 transport mechanism and other SENSEI transports follow the same metaphor of producers writing data and consumers reading data. The difference is that SENSEI-HDF5 reads and writes data to file systems. Internally, HDF5 reads and writes data using efficient parallel mechanisms such as MPI-IO, to complete the I/O operations. On large HPC systems, the target file systems typically consist of a large number of disks and can achieve terabyte(TB) per second I/O speed. However, this is still much less than the aggregate read and write speed to the memory on the same HPC system.

Recently, the emergence of NVRAM technology has provided a storage medium that is faster than disk but slower than DRAM memory. A number of users have demonstrated that NVRAM can be effectively used for HPC applications [3]. This work explores the use of NVRAM for temporarily storing data files for *in transit* use cases. More specifically, we are using the Burst Buffer on the NERSC Cori system, which has a total of approximately 1.7 TB/s of peak I/O performance with 28M IOPs, and about 1.8PB of storage. This Burst Buffer system is presented to the users through Cray's own DataWarp software as a parallel file system based on XFS [3]. Thus the Burst Buffer can be easily used through the file-based transport mechanism SENSEI-HDF5.

A key reason for us to work with the SENSEI framework is that it provides an easy way to switch among the available transport mechanisms. In SENSEI-instrumented codes, this switching is accomplished by modifying parameters in an XML configuration file. For example, for the ADIOS transport using FlexPath, the configuration looks like this:

```
<sensei>
  <analysis type="adios1" filename="./test"
    method="FLEXPATH" enabled="1 "/>
</sensei>
```

To apply the HDF5 transport, change the parameter values of type and method:

```
<sensei>
  <analysis type="hdf5" filename="/burst/buffer/file"
    method="stream" enabled="1" />
</sensei>
```

4 TEST SETUP

The main objective for our study is to measure the performance of two *in transit* data transport systems. One uses a socket-based, memory-memory data transfer and the other uses temporary files to transfer the data. Both these options are implemented under the SENSEI framework, with the socket-based option using ADIOS (v1.13) and FlexPath, while the file-based option used HDF5 (v1.11).

The data producer used for this study is a SENSEI miniapp known as the oscillator [1]. It first generates arrays of particle data on meshes, then delivers data using a given transport mechanism. The consumer uses the assigned transport mechanism to read the data and then perform its computation. In tests for this paper, the consumers all compute a one-dimensional histogram.

The computational platform we use for this study is the Supercomputer Cori at NERSC. It is a Cray XC40. Cori contains two different kinds of nodes: 2,388 Intel Xeon "Haswell" processor nodes and 9,688 Intel Xeon Phi "Knight's Landing" (KNL) nodes. Cori also features 288 Burst Buffer nodes presented as a shared parallel file system through the DataWarp software. For this test, we have created a 10TB reservation consisting of 271 Burst Buffer Nodes. The granularity of DataWarp is 20GB.

Another relevant feature on Cori is that Cray provides its own version of the MPI library that bypasses many layers of the IP software stack, while applications use socket for data communication would not have the same accesses to the core communication fabric. This potentially could lead to different performances for the software using MPI and MPI-IO library as in the case of using HDF5

M ranks	nodes	Sent total	Sent rank
1024	32	1 TB	1 GB
2048	64	2 TB	1 GB
4096	128	4 TB	1 GB
8192	256	8 TB	1 GB

M:N Ratio	Received
64:1	64x
32:1	32x
16:1	16x
8:1	8x

Table 1: Run configurations. M and N are for simulation and analysis ranks, respectively. The left table shows the simulation configurations while the right shows all the M:N ratios used to set up Analysis configurations. Data received by a consumer rank is a multiple of the data sent from a producer rank, determined by the M:N ratio.

to read and write files, and those software using socket for data communication as in the case of ADIOS FLEXPATH.

For each test configuration, the simulation job and the analysis job are launched back to back. The time to solution is measured from the beginning of the first task (simulation) to the completion of the last task (analysis). In addition, the memory footprint through out the job duration is also measured.

Let M and N be the number of ranks used for simulation (data producer) and analysis (data consumer) respectively. We plan to keep the simulation work on each rank constant, i.e. weak scaling. A default data partition is applied and simulation data is evenly distributed to analysis processors. Therefore, the elapsed time primarily varies depending on the M:N ratio. We use four different such ratios for each M. The data size produced from simulation varies from 1TB to 8TB.

The details of the configurations used are shown in Table 1. We use M and N to represent simulation and analysis ranks respectively. Note that each M:N ratio is applied to every simulation configuration. With four different simulation ranks and four different M:N ratios, a total of 16 combinations will be evaluated.

All the data used for this paper is collected on the KNL nodes from the Cori system at NERSC. Data produced are sent through sockets, staged on Burst Buffer, or staged on Cori's Lustre System, which is configured to use 128 stripe count and 32MB per stripe.

A similar performance evaluation for ADIOS has been reported by Kress et al [7], where the in-memory transport mechanism was thoroughly examined. In this study, we contrast the in-memory mechanism with the option of using NVRAM.

5 PERFORMANCE MEASUREMENTS

To understand performance, we have collected wall-clock times and memory footprints for every rank of each simulation and analysis jobs.

Fig. 2 has two pictures. On the left side, the picture shows the total elapsed time of three different transport options using socket, burst buffer, and disk (labeled as Lustre). In this case, the data producer and consumer ratio is fixed as 16:1. We can see that the time required by the disk-based option (labeled Lustre) grows with the number of simulation cores, and the time to solution is noticeably longer with this disk-based option as expected. Therefore, we will not evaluate this option further.

The picture on the right side of Fig. 2 shows only two data transport options: socket and burst buffer, with more test configurations. We expect the socket-based option to use less time than the burst buffer option. However, the actual performance measurements show that most often this is not the case, which is a surprise. There are potentially many different reasons for this surprise. For example, both ADIOS (used for the socket communication) and HDF5 (for file I/O operations) are complex software packages with many parameters that could be adjusted. We might have just happen to use a unusual combination of the parameters. Next, we provide some additional performance measurements to see if we could pinpoint the root cause of the surprising observation.

Fig. 3 shows the memory footprint of one rank from the oscillator (on the right) and the consumer (on the left), where the option with ADIOS FLEXPATH is shown in blue and the option with HDF5 is shown in red. On the simulation side (writer), the socket-based option appears to be accumulating some number of time steps, which causes the memory usage to increase. This increasing memory usage is likely the result of a setting that allows ADIOS unlimited amount of memory. Since the actual amount of memory used is still relatively small, we don't expect this setting to cause performance issues.

On the right side of Fig. 3, we see that the analysis programs use the same amount of memory regardless of the transport option used. As expected, when more producers send their data to a single consumer, more memory the consumer needs and more time is need to complete the analysis operations, even though the difference in analysis time is relatively small.

From Fig. 3, we see again that the socket option requires more time on this specific rank, which appears as the blue lines extending more to the right than the red lines. To further investigate this time difference, we show a more detailed progress chart for two different configurations on rank 0 in Fig. 4, where the configuration shown on the left side has eight times more data send to each analysis program than that on the right side. In this case, we see that the analysis program took much longer on the left side, which delays the overall completion time, while on the right side, data produced by each time step can be consumed relatively quickly, and the overall completion time is noticeably shorter.

6 CONCLUSION AND FUTURE WORK:

We designed and implemented an *in transit* mechanism in the SENSEI framework using the parallel I/O library named HDF5. It allows us to access both burst buffer and disk based file systems. We designed this file-based transport option for analysis tasks that require more data than what could be held in memory and expect it to be slower than the cases where the data fits in memory. However, our tests show that in many test cases, by placing HDF5 files on burst buffers, we were able to complete the *in transit*() analysis tasks faster than a popular socket-based in-memory transport option.

Many factors, such as the socket communication may be slower than Cray MPI on the test machine, could explain observed performance measurements. We attempted to dive into some details, but clearly we have not fully explored the parameters that could affect the observed performance. For future work, we plan to explore lower level details of the transport mechanisms to better understand the performance observed. We also plan to explore other streaming mechanisms to improve *in transit* performance.

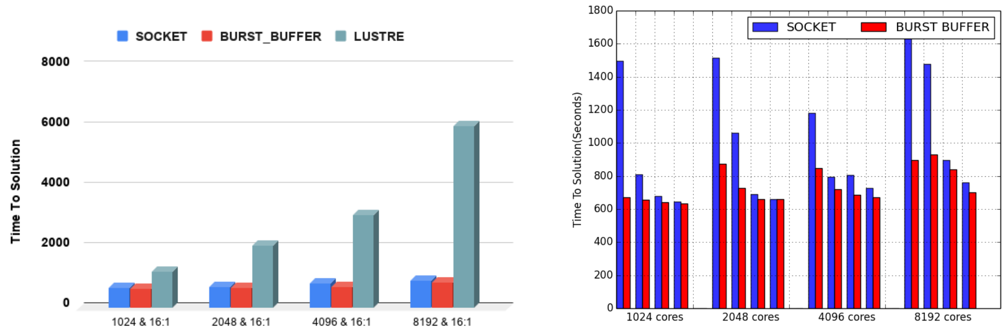

Figure 2: LEFT: Time to solution for socket-based (ADIOS-FLEXPATH), Burst Buffer (HDF5), and disk(HDF5) approaches. RIGHT: Time to solution for socket-bsed(ADIOS-FLEXPATH) and Burst Buffer (HDF5) approaches. Configurations are stated in Table 1.

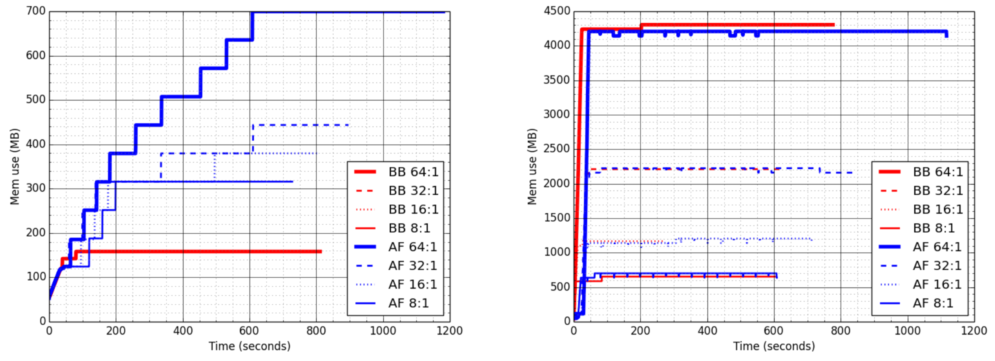

Figure 3: Memory usage as a function of run time for 4096 write ranks. Simulation(writer) on the left, and analysis(reader) on the right. Red lines denote runs using BB(Burst Buffers) and blue lines denote runs with AF(ADIOS-FLEXPATH) staging method.

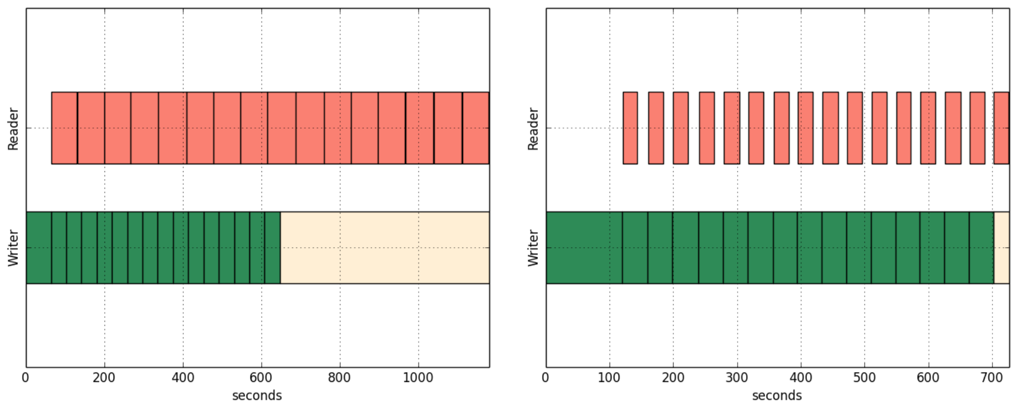

Figure 4: Processing time on rank 0 for the simulation and analysis using ADIOS FLEXPATH transport, with 4096 ranks of writers. LEFT: M:N ratio is 64:1. RIGHT: M:N ratio is 8:1.

REFERENCES

[1] U. Ayachit, A. Bauer, E. P. N. Duque, G. Eisenhauer, N. Ferrier, J. Gu, K. Jansen, B. Loring, Z. Lukić, S. Menon, D. Morozov, P. O'Leary, M. Rasquin, C. P. Stone, V. Vishwanath, G. H. Weber, B. Whitlock, M. Wolf, K. Wu, and E. W. Bethel. 2016. Performance Analysis, Design Considerations, and Applications of Extreme-scale *In Situ* Infrastructures. In *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC16)*. Salt Lake City, UT, USA. https://doi.org/10.1109/SC.2016.78 LBNL-1007264.

[2] U. Ayachit, M. Whitlock, B. Wolf, B. Loring, B. Geveci, D. Lonie, and E. W. Bethel. 2016. The SENSEI Generic In Situ Interface. In *Proceedings of In Situ Infrastructures for Enabling Extreme-scale Analysis and Visualization (ISAV 2016)*. Salt Lake City, UT, USA. https://doi.org/10.1109/ISAV.2016.13 LBNL-1007263.

[3] W Bhimji, D Bard, M Romanus, D Paul, A Ovsyannikov, B Friesen, M Bryson, J Correa, G K Lockwood, V Tsulaia, S Byna, S Farrell, D Gursoy, C Daley, V Beckner, B Van Straalen, D Trebotich, C Tull, G Weber, N J Wright, K Antypas, and Prabhat. 2016. Accelerating Science with the NERSC Burst Buffer Early User Program. In *CUG 2016*. https://cug.org/proceedings/cug2016_proceedings/includes/files/pap162s2-file1.pdf

[4] J Dayal, D Bratcher, G Eisenhauer, K Schwan, M Wolf, X Zhang, H Abbasi, S Klasky, and N Podhorszki. 2014. Flexpath: Type-based publish/subscribe system for large-scale science analytics. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 246–255.

[5] M Folk, G Heber, Q Koziol, E Pourmal, and D Robinson. 2011. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases*. ACM, 36–47. Software at http://www.hdfgroup.org/HDF5/.

[6] James A. Kohl, Torsten Wilde, and David E. Bernholdt. 2006. Cumulvs: Interacting with High-Performance Scientific Simulations, for Visualization, Steering and Fault Tolerance. *The International Journal of High Performance Computing Applications* 20, 2 (2006), 255–285. https://doi.org/10.1177/1094342006064502

[7] J. Kress, M. Larsen, J. Choi, M. Kim, M. Wolf, N. Podhorszki, S. Klasky, H. Childs, and D. Pugmire. 2019. Comparing the Efficiency of In Situ Visualization Paradigms at Scale. In *High Performance Computing*, Michèle Weiland, Guido Juckeland, Carsten Trinitis, and Ponnuswamy Sadayappan (Eds.). Springer International Publishing, Cham, 99–117.

[8] Q. Liu, J. Logan, Y. Tian, H. Abbasi, N. Podhorszki, Jong Y. Choi, S. Klasky, R. Tchoua, J. Lofstead, R. Oldfield, M. Parashar, N. Samatova, K. Schwan, A. Shoshani, M. Wolf, K. Wu, and W. Yu. 2014. Hello ADIOS: the challenges and lessons of developing leadership class I/O frameworks. *Concurrency and Computation: Practice and Experience* 26, 7 (2014), 1453–1473.

[9] W Usher, S Rizzi, I Wald, J Amstutz, Jh Insley, V Vishwanath, N Ferrier, M E Papka, and V Pascucci. 2018. libIS: A Lightweight Library for Flexible in Transit Visualization. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization (ISAV '18)*. ACM, New York, NY, USA, 33–38. https://doi.org/10.1145/3281464.3281466