2021 IEEE International Conference on Cluster Computing (CLUSTER) | 978-1-7281-9666-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/Cluster48925.2021.00099

A Scalability Study of Data Exchange in HPC Multi-component Workflows

Jie Yin†, Atsushi Hori†, Balazs Gerofi‡, Yutaka Ishikawa†
†*National Institute of Informatics, JAPAN*
‡*RIKEN Center for Computational Science, JAPAN*
yinj@nii.ac.jp, ahori@nii.ac.jp, bgerofi@riken.jp, yutaka.ishikawa@nii.ac.jp

***Abstract*—Multi-component workflows play a significant role in High-Performance Computing and Big Data applications. They usually contain multiple, independently developed components that execute side-by-side to perform sophisticated computation and data exchange through file I/O over parallel file system. However, file I/O can become an impediment in such systems and cause undesirable performance degradation due to its relatively low speed (compared to the interconnect fabric), which is unacceptable especially for applications with strict time constraints. The Data Transfer Framework (DTF) is an I/O arbitration layer working with the PnetCDF I/O library aiming at eliminating the bottleneck by transparently redirecting file I/O operations through the parallel file system to message passing via the high-speed interconnect between coupled components. Scalable and high-speed data transfer between components can be thus easily achieved with minimal development effort by using DTF. However, previous work provides insufficient scalability evaluation of the framework. In order to comprehensively evaluate the scalability of an I/O middleware like DTF and highlight its major advantages, we develop an I/O benchmark for multi-component workflows. Using the benchmark we conduct large-scale scalability evaluation using up to 32,768 compute nodes on supercomputer Fugaku and 2,048 compute nodes on Oakforest-PACS by comparing direct data transfer to file I/O performed on Lustre file system and Fugaku's Lightweight Layered IO-Accelerator (LLIO). We provide insights into DTF's scalability and performance enhancements with the intention to impact future I/O middleware and inter-component data exchange design in multi-component workflows.**

***Index Terms*—Multi-component Workflow, Parallel I/O, Inter-component Data Transfer**

I. Introduction

The rapid growth of the computational speed of supercomputers creates new possibilities to solve more complex scientific problems within a reasonable time limit. It also brings opportunities for developers to pursue better performance and solutions for their High-Performance Computing (HPC) applications with different computational models. Multi-component workflow is one of the prevalent scientific computation models in High-Performance Computing, which tightly couples independently developed components by executing side-by-side and exchanging a gigantic amount of computational data between them for the succeeding computation. Big Data Assimilation is a representative application deployed widely in many real-world systems indispensable to modern society, such as in weather forecasting systems [1]. File I/O using the parallel file system has been the traditional approach for data exchange between components because each component of such workflow is usually developed by independent development team and File I/O is the easiest way to implement. I/O libraries are common approaches to generate data files and store massive data sets in an organized and portable way. Some HPC application-oriented optimizations are usually introduced into such libraries to improve I/O performance [2], [3]. For example, Parallel netCDF (PnetCDF) [3] is one of the widely used I/O libraries which provides parallel support for accessing data stored in NetCDF format using MPI-IO [4]. However, in some multi-component applications with time constraints, massive data transfer between components through file I/O prevents the progress of achieving the real-timeliness due to its relatively slow processing speed. Coupling toolkits [5], [6] and highly optimized I/O libraries [7], [8] are the two proposed solutions for this bottleneck. Several of the proposed solutions requires a great effort to be deployed in the existent multi-component HPC applications because dedicated APIs are required. There exists a pressing need for an easy-to-use approach to diminish data exchange time for multi-component applications, which motivated researchers to design I/O middlewares such as the Data Transfer Framework (DTF) [9] for meeting the expectations stated above.

DTF is an I/O arbitration framework aiming at reducing data exchange time between coupled components through high performance interconnect by message passing instead of file I/O. Our implementation of DTF works with the Parallel netCDF (PnetCDF) I/O library and avoid from modifying application code to the maximum extent possible. Section II gives a detailed description of DTF and its mechanism.

Previous work [9] has introduced preliminary performance evaluation comparing data exchange using DTF to file I/O. In-depth behavior analysis of DTF has been conducted using the S3D-IO benchmark [10], and its performance for multi-component application has been evaluated using a real-time severe weather prediction system named SCALE-LETKF [1]. SCALE-LETKF is a real-time severe weather prediction application combines two independently developed components SCALE and LETKF, which conduct ensemble-based weather simulation and data assimilation using real-world observation data and weather simulation results, respectively. In order to fully utilize the observation data which is delivered by weather radar every 30 seconds and perform real-time weather prediction, every prediction cycle of SCALE-LETKF including its massive computation and inter-component data exchange must

978-1-7281-9666-4/21/$31.00 ©2021 IEEE
DOI 10.1109/Cluster48925.2021.00099

be finished within the strict time constraints. SCALE-LETKF is scheduled to be deployed at the Tokyo 2020 Olympics to assist real-time weather forecasting during the event. Preliminary evaluation results using real-world observation data set suggested that DTF can help SCALE-LETKF achieving its real-timeliness.

However, there are some limitations in the previous scalability assessment. Previous work used real-time application to test the scalability of DTF, which is a mixture of intensive computation and massive data transfer [9]. Alternating computation and I/O operations hinder the accurate evaluation of inter-component data transfer scalability. Furthermore, previous evaluation used only one experimental data set for the scalability assessment. Parameters in real-world data sets are often based on practical observations, which makes it difficult to flexibly adjust data exchange behavior and gain a thorough insight into the scalability of inter-component data transfer under different circumstances.

To the best of our knowledge, there is no benchmark for evaluating inter-component data exchange for multi-component workflows, although such complex workflow plays an important role in HPC scientific computing. In light of the limitations listed above, we believe that proposing a benchmark for evaluating the scalability and performance of different data exchange approaches for multi-component workflows would be beneficial for future development of such HPC applications. We summarize the contributions of this paper as follows:

1) We propose an I/O benchmark for multi-component workflows with adjustable data size and adapt the benchmark to the I/O model of SCALE-LETKF, a cutting-edge weather forecasting system.
2) We present large-scale scalability measurements taken on two supercomputers, using up to 32,768 compute nodes of Fugaku and 2,048 nodes of Oakforest-PACS (OFP). We compare DTF to Lustre file system and to Fugaku's hierarchical storage system utilizing compute node SSDs called the Lightweight Layered IO-Accelerator (LLIO) [11].
3) We provide scalability prospect for a real-world use-case of the SCALE-LETKF weather forecasting workflow on Fugaku using the configuration to be deployed at the Tokyo 2020 Olympics. The results present the data exchange scalability for possible future development on a bigger scale.

The rest of this paper is organized as follows. Section II includes a description of DTF and its simple usage. An overview of the proposed benchmark is given in Section III and scalability results comparing DTF to file I/O on supercomputer Fugaku and OFP are provided in Section IV. Section V surveys related work and Section VI discusses our current and potential future efforts for the benchmark development. The final section concludes the paper.

II. Data Transfer Framework (DTF)

DTF [9] is an I/O arbitration middleware devoted to improve data exchange performance and efficiency between coupled components in a multi-component application. The principle of DTF is to silently redirect the file I/O based inter-component data exchange to message passing with the least effort while avoiding modification of original application code to the maximum extent possible. In this section, we provide a brief overview of DTF, describe how it works under the hood, and highlight its advantages.

A. DTF Overview

DTF works with I/O operations for files in the Network Common Data Form (NetCDF) format [12]. As shown in Figure 1, DTF is integrated with the parallelized implementation of the NetCDF format named Parallel netCDF (PnetCDF) which is built on top of MPI-IO [4], to allow multiple processes performing I/O to the same NetCDF file in parallel as its name suggests. PnetCDF is proven to produce an outstanding file I/O performance compared to other I/O libraries like serial netCDF and parallel HDF5 in terms of scalability and bandwidth [3]. According to the previous work, it only requires about 50 lines of code modification in PnetCDF for integrating with DTF [9].

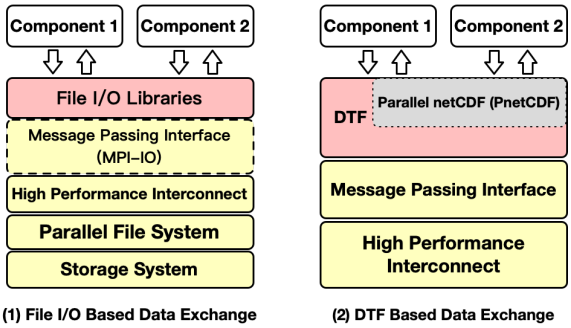


Fig. 1: An overview of traditional file I/O based inter-component data exchange and the Data Transfer Framework (DTF) approach.

Execution of coupled components are started by using the colon notation of *mpiexec*, also known as Multiple Program Multiple Data (MPMD) execution method. Most of the MPI functions are replaced by using the PMPI interface as if each component has its own `MPI_COMM_WORLD`. During the initialization stage by calling `dtf_init()` in user applications, DTF sets up a connection session between the two components by establishing an MPI inter-communicator. All the succeeding inter-components communications happen through this inter-communicator. While PnetCDF interface being invoked during I/O sessions in each component, DTF redirects the execution flow to its embedded function calls and performs inter-component message passing instead of the original I/O processing. Its internal operations are categorized into the following two parts.

1) I/O Request Handling: Several processes in the writer component will be entrusted with being *request matchers* when an I/O session starts, which are responsible for collecting all the posted I/O requests from other processes in both components and performing request matching. The number of matchers is configurable through environment variable.

Whenever a PnetCDF function from the *get* or *put* families is invoked to access a NetCDF variable, DTF will redirect this call to its internal function and initialize an I/O request object, which contains metadata about the request (i.e., start coordinate, shape, request type and address of user buffer). Request matching is performed by matchers based on I/O requests when the user signals DTF to start transferring data between the coupled components by invoking a DTF function `dtf_transfer()`. Once two requests are matched, matchers will notify the matched writer process to send the requested data blocks to its reader. DTF is well aware that two components may perform asymmetric I/O operations, which happens when the reader processes never read some data blocks that are written by the writer. Request matching perfectly solves the situation by only transferring data blocks that are explicitly requested by the reader.

2) I/O Replaying: I/O replaying is an optimization solution inspired by the I/O pattern of some multi-component applications. These applications usually perform multiple iterations of identical data exchange. When I/O replaying is enabled, DTF records the history of request matching for the first iteration, and repeats the same data exchange behavior for the rest of the iterations. Request matching process is skipped for the subsequent cycles so that DTF can start transferring data immediately. This feature can be easily enabled for applications by presetting the value of option *replay_io* in the configuration file.

B. Simple Use of DTF

DTF needs three preparation steps to be done before multi-component applications can switch from file I/O to high-speed data exchange.

- **Write a configuration file.** User is expected to briefly describe the I/O dependencies between the two components and specify the mode of data exchange for each listed file pattern in key-value pairs. Other optional configurations are also available for users to flexibly adjust DTF's behavior and fully boost data transfer between components, such as replaying I/O and buffering user data.
- **Insert DTF interfaces.** Three intuitive DTF functions should be inserted into the code of both components for initializing the library, instructing DTF to start exchanging data, and finalizing the library.
- **Build with DTF based PnetCDF library.** Lastly, user should recompile each component with the provided PnetCDF library integrated with DTF. DTF is ready-to-use once this step is completed.

III. Benchmark Overview

In this work, we propose an I/O benchmark for multi-component workflows and adapt the benchmark to an ensemble-based data exchange model. In an ensemble-based data exchange model, processes of each component are grouped into multiple ensembles and performing independent inter-component data exchange by each ensemble. We configure its behavior to simulate that of the severe weather prediction application SCALE-LETKF. This section presents an brief overview of the benchmark.

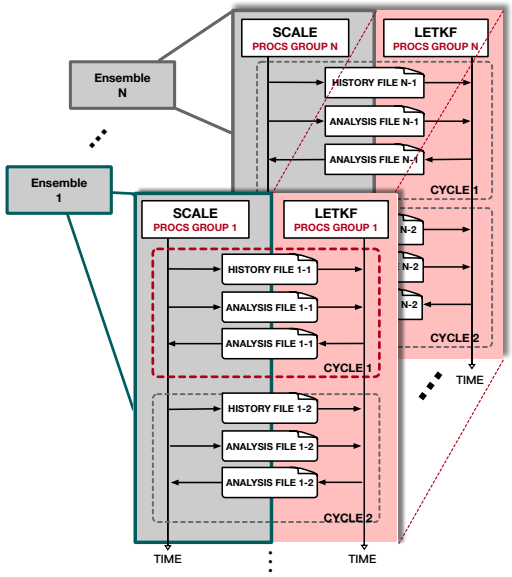


Fig. 2: An overview of SCALE-LETKF's inter-component data exchange for each cycle in each ensemble.

As shown in Figure 2, the component SCALE is composed of multiple process groups as ensembles, in which each ensemble performs a series of iterative data exchange with processes of the other component LETKF. The data exchange process of each iteration within each ensemble consists of three sessions with two type of files involved, which are named as *history* and *analysis* files respectively. During each iteration when file I/O based data exchange is configured, SCALE will output its simulation results into a new *history* file and a *analysis* file for data assimilation performed by LETKF, and LETKF will overwrite its assimilation results to the generated *analysis* file for SCALE as input data for the simulation of the next iteration. Besides performing inter-component data exchange in an ensemble based data exchange model, difference in amount of I/O performed by the two components is another major characteristic of its data exchange model. SCALE outputs computation data to 89 NetCDF variables of *history* file and 143 variables of *analysis* file, while LETKF only reads data from 20 NetCDF variables of *history* file and 11 variables of *analysis* file. The asymmetry between components incurs a large amount of execution time being wasted on the undesirable I/O operations when file I/O is configured for data

exchange. Fortunately, DTF smartly diminishes this problem according to its request matching design as introduced in Section II-A1, which saves plenty of time from exchanging redundant data.

There are five parameters determining the amount of data of each process that will be transferred between components. These five parameters for each process are `IMAX`, `JMAX`, `KMAX`, `LKMAX` and `UKMAX`, which represents, respectively, the number of grid points in longitudinal direction, latitudinal direction and vertical direction of atmospheric, land and urban model in real-world geographic perspective, and the length of multi-dimensional data arrays in the application. Additionally, the overall amount of data exchange between components can be adjusted by designating the number of ensembles and processes through command line options.

IV. Evaluation

In this section, we present the scalability of inter-component data exchange conducted on two supercomputers, Fugaku and OFP, between DTF and PnetCDF file I/O. Fugaku has its compute nodes connected via the newly designed 6D mesh/torus interconnect called Tofu Interconnect D (TofuD) [13] and equipped with a three-layered storage system of which the first and the second layers are used for the evaluation experiments. The second layer of the storage system is a Lustre-based global file system named Fujitsu FEFS [14], [15], and the first layer is called Lightweight Layered IO-Accelerator (LLIO), which is designed as a dedicated file system for optimizing file I/O of HPC applications. LLIO consists of three areas, which are node temporary area, shared temporary area and 2nd-layer cache area [14]. In this work, all the file I/O operations performed using LLIO are located on the shared temporary area. OFP is another Fujitsu built supercomputer which is managed by The University of Tsukuba and The University of Tokyo [16]. All the compute nodes are interconnected by Intel's Omni-Path network and able to access a Lustre-based global parallel file system.

We configured the benchmark with DTF following the steps introduced in Section II. Average *data transfer time* for each iteration is measured in the following experiments to discuss the potential of DTF for assisting SCALE-LETKF in achieving its real-timeliness and evaluate its data exchange scalability. In terms of DTF, *Data transfer time* represents the overall elapsed time from I/O requests posted until all the requested data delivered during three data exchange sessions of each iteration. From the perspective of file I/O, *data transfer time* means the total amount of time consumed in writing and reading data during three sessions of each iteration. Because this benchmark only contains data exchange operations, the reported time exactly reflects the speed and efficiency of data exchange between components without interference from irrelevant operations. We also enabled I/O replaying optimization because every iteration of SCALE-LETKF has exactly the same I/O behavior.

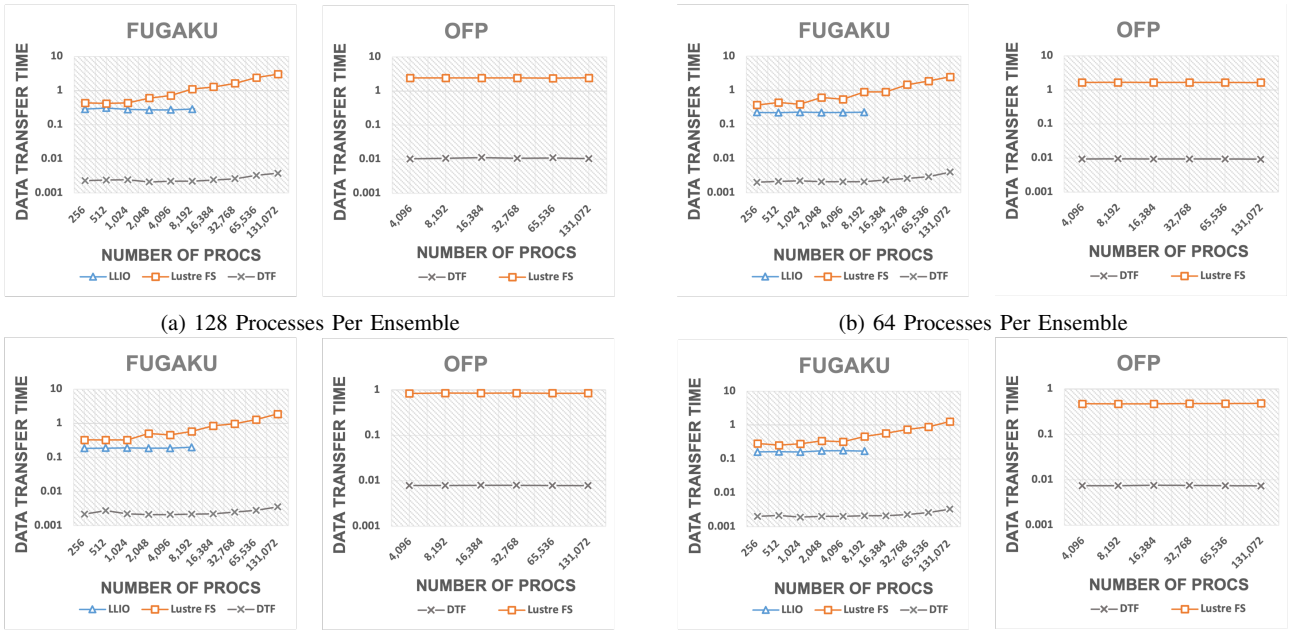
A. Scalability Comparison Between Fugaku and OFP

Data size parameters introduced for scalability comparison between Fugaku and OFP are based on the latest configuration of SCALE-LETKF real-world experiment performed on OFP [17], in which `IMAX`, `JMAX`, `KMAX`, `LKMAX` and `UKMAX` are set into 8, 8, 60, 7 and 5 respectively. In order to fully utilize the limited compute resources of OFP, 64 processes are spawned on each node and up to 131,072 processes are generated when all the available 2,048 nodes are used. We choose to spawn 4 processes per node on Fugaku and scale the utilized number of nodes up to 32,768, while total number of processes stay the same as OFP. On the other hand, only up to 2,048 nodes are used to spawn 8,192 processes in the experiments conducted using LLIO on Fugaku due to system limitations. We designed four test cases, 128 processes per ensemble, 64 processes per ensemble, 32 processes per ensemble and 16 processes per ensemble to present the scalability changes along with the increasing number of ensembles while total amount of exchanged data remains the same as shown in Table I. The maximum number of ensembles in the experiments is 4,096 when 16 processes per ensemble is configured and 131,072 processes are spawned for the two components.

Scalability comparison using proposed test cases are presented in Figure 3. It is clear that time spent on data exchange is significantly reduced by DTF compared to file I/O on both supercomputer platforms. On Fugaku, LLIO obviously improved file I/O performance and kept a steady data transfer time, while performance of file I/O using Lustre file system deteriorates along with the increase of the number of processes. Based on the collected time data, file I/O becomes almost 7x slower when total number of processes reach the maximum compared to the initial case, which is also worse than the file I/O data transfer time of its OFP counterpart. Data transfer time of DTF is at a steady rate when the number of spawned processes is lower than 16,384 and slightly increases when the number becomes larger in each case. On OFP system, both file I/O and DTF show good scalability along with the increase of the number of processes even the data exchange is slower than Fugaku.

B. Scalability Prospect for Real-world Experiments

In the latest real-world severe weather prediction experiments of SCALE-LETKF for Tokyo 2020 Olympics conducted on Fugaku, high resolution real-world data based on 500-meter land surface mesh has been used for severe weather prediction. The number of ensembles is limited to the largest possible number when the timeliness requirement can be satisfied, which is configured into 1,000 ensembles in this real-world experiment. In this section, we presented the scalability evaluation of inter-component data exchange of SCALE-LETKF using the benchmark introduced above in the same parameters as the real-world experiments, which are 32, 64, 45, 7 and 5 for `IMAX`, `JMAX`, `KMAX`, `LKMAX` and `UKMAX` respectively. The maximum number of ensembles in our evaluation measures reached 4,096. As shown in Table II, the overall amount of data exchanged between components per iteration is 25



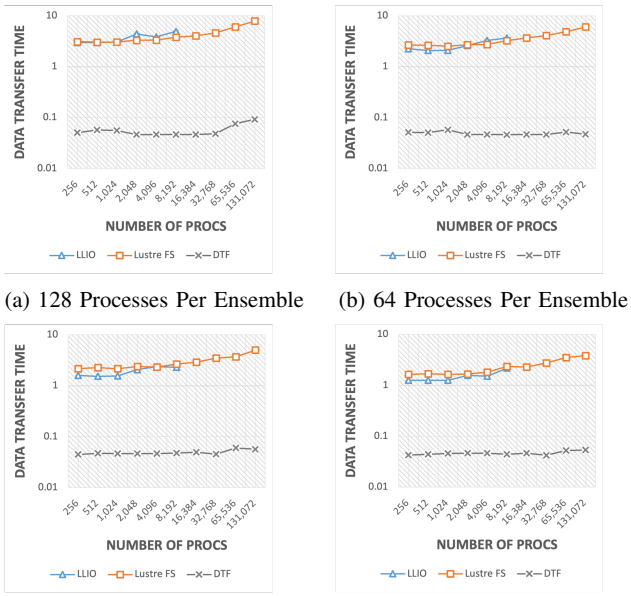
(a) 128 Processes Per Ensemble

(b) 64 Processes Per Ensemble

(c) 32 Processes Per Ensemble

(d) 16 Processes Per Ensemble

Fig. 3: Scalability comparison between Supercomputer Fugaku and (OFP). The x-axis of each figure represents the total number of processes spawned on each platform and the y-axis means the average data transfer time (second) of each iteration in logarithmic scale.



(a) 128 Processes Per Ensemble

(b) 64 Processes Per Ensemble

(c) 32 Processes Per Ensemble

(d) 16 Processes Per Ensemble

Fig. 4: Scalability evaluation on Fugaku Supercomputer using 500-m land-surface mesh parameters. The x-axis of each figure represents the total number processes spawned on Fugaku and the y-axis means the average data transfer time (second) of each iteration in logarithmic scale.

times larger than the experimental configuration introduced in Section IV-A.

Same as the test cases proposed in the previous section, data exchange scalability of DTF, Lustre file system based file I/O and LLIO for growing number of ensembles and processes are measured on Fugaku. Based on the experiment results shown in Figure 4, current LLIO system lost its advantage when the size of generated files becomes larger. Especially in the first test case when each ensemble is composed of 128 processes, the amount of data exchanged between components within each ensemble is multiple times larger than other test cases. An significant growth in data transfer time using LLIO can be observed along with the increasing number of processes. This is because the larger data size cannot be fit into the SSD of LLIO. It also can be seen from the results that data transfer time for Lustre file system based file I/O also grows at a steady rate. According to the collected time data, data transfer time becomes almost 2.5x larger when 131,072 processes are spawned on the system compared to the initial case. On the other hand, data transfer performed by DTF stays flat and shows better scalability than file I/O in each test case as expected. These evaluation results provide a scalability prospect by simulating inter-component data exchange for future real-world experiments with larger number of forecasting ensembles for higher-accuracy weather prediction.

V. Related Work

Inter-component data exchange in multi-component workflows has been discussed in many research works. Researchers

TABLE I: Total amount of data exchanged by File I/O and DTF per cycle (GiB) in each test case of Figure 3.

# PROCS	DTF	FILE
256	0.06	0.15
512	0.12	0.30
1,024	0.24	0.60
2,048	0.48	1.20
4,096	0.97	2.41
8,192	1.94	4.81
16,384	3.88	9.63
32,768	7.75	19.25
65,536	15.50	38.50
131,072	31.00	77.00

TABLE II: Total amount of data exchanged by File I/O and DTF per cycle (GiB) in each test case of Figure 4.

# PROCS	DTF	FILE
256	1.50	3.65
512	3.00	7.31
1,024	6.00	14.62
2,048	12.01	29.23
4,096	24.02	58.47
8,192	48.03	116.94
16,384	96.06	233.88
32,768	192.13	467.75
65,536	384.25	935.50
131,072	768.50	1871.00

have been looking for an optimal way to transfer massive computation data between components instead of using file I/O, which is relatively easier to implement but has lower data transfer rate and efficiency. Coupling toolkits are designed as one of the approaches in some research works, such as Model Coupling Toolkit (MCT) in [5], Decaf proposed in [18] and OASIS3 coupler introduced in [19] and [20] for Earth System Model. Besides data transferring, these coupling toolkits usually provide additional data processing operations during coupling sessions such as data interpolation. There are some I/O libraries proposed in research works as another data transfer strategy such as DART [8] and ADIOS [7]. The performance advantages of proposed I/O library compared to some traditional file I/O libraries such as HDF-5 and independent MPI-IO are shown in the evaluation section of [7]. Unfortunately, deploying these I/O libraries into other multi-component workflows requires huge efforts to modify the existent I/O implementation of the application because dedicated APIs must be used. Approach proposed in [21] uses dedicated nodes for data transferring between the computation component and analysis tool, and Flexpath proposed in [22] provides a type-based publish/subscribe framework for coupling computation component with analytic component based on ADIOS library. I/O Arbitration Framework (FARB) introduced in [23] proposed a similar solution to DTF, which is designed for multi-component workflows based on NetCDF I/O library. It expects a file-per-process I/O pattern and transfers data between coupled components in a process-to-process manner. Due to its characteristics of design and I/O pattern, the whole file will be transferred between components even when the reader only needs partial file. Results of small-scale and intermediate-scale performance evaluations using SCALE-LETKF application scaling up to 4,800 compute nodes are presented in the paper. Large-scale systematic scalability and performance evaluation of inter-component data exchange can be barely found in the research works using the proposed approaches, which became one of the motivations to design such benchmark frameworks introduced in this paper for straightforwardly demonstrating their advantages compared to the traditional approaches on modern supercomputer platforms.

VI. Discussion and Future Work

While our proposed benchmark is currently limited to the I/O behavior of SCALE-LETKF, it lays the ground work for a more general framework that can address a wider set of workflow applications. It is desirable to explore scalability for a diverse set of multi-component workflows utilizing various I/O patterns. Therefore, we put our current efforts in generalizing the benchmark and providing an benchmark framework for evaluating inter-component data exchange of different I/O models.

We are also interested in the performance comparison between DTF and other optimized data transfer approaches such as the methods introduced in Section V. By using an I/O benchmark isolated from massive computation operations, it will be much easier to put different approaches into practice for further comparison under varied data exchange scenarios.

Currently DTF only supports PnetCDF file I/O interfaces. There are other advanced libraries optimized for high-performance I/O such as HDF5 [2] and NetCDF-4 [24]. By integrating support for multiple I/O libraries into DTF the benchmark provides possibility for comprehensive studies of I/O middleware performance and scalability. Such support is left for future work.

VII. Conclusion

In this work, several limitations of inter-component data exchange scalability evaluation conducted in previous work are addressed. In order to break through the limitation and conduct systematic scalability evaluation between different inter-component data exchange approaches, we proposed an multi-component I/O benchmark as introduced in Section IV and adapted it to the data exchange model of a real-time weather forecasting application SCALE-LETKF. By using the proposed benchmark, we are able to conduct in-depth evaluation of data transfer in multi-component workflows such as SCALE-LETKF and gain the insight into the scalability of prospective bigger scale data exchange in future development. According to the evaluation results collected on two supercomputers, Fugaku and OFP, DTF is confirmed to have significant scalability advantage over file I/O. We also provided a series of data exchange scalability measures of SCALE-LETKF using the configurations of severe weather prediction for Tokyo 2020 Olympics. Based on the analysis and results we have at the current stage, we conclude that multi-component applications can easily achieve high performance and scalable data exchange by deploying I/O middlewares such as DTF through minimal code changes.

VIII. Acknowledgement

This work has been partially funded by Japan Science and Technology Agency (JST) AIP Acceleration Research Grant Number JPMJCR19U2.

REFERENCES

[1] T. Miyoshi, G. Y. Lien, S. Satoh, T. Ushio, K. Bessho, H. Tomita, S. Nishizawa, R. Yoshida, S. A. Adachi, J. Liao *et al.*, ""big data assimilation" toward post-petascale severe weather prediction: An overview and progress," *Proceedings of the IEEE*, vol. 104, no. 11, pp. 2155–2179, 2016.

[2] M. Howison, "Tuning hdf5 for lustre file systems," 2010.

[3] J. Li, W. k. Liao, A. Choudhary, R. Ross, R. Thakur, W. Gropp, R. Latham, A. Siegel, B. Gallagher, and M. Zingale, "Parallel netcdf: A high-performance scientific i/o interface," in *SC'03: Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. IEEE, 2003.

[4] S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock, "I/o performance challenges at leadership scale," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. IEEE, 2009, pp. 1–12.

[5] J. Larson, R. Jacob, and E. Ong, "The model coupling toolkit: a new fortran90 toolkit for building multiphysics parallel coupled models," *The International Journal of High Performance Computing Applications*, vol. 19, no. 3, pp. 277–292, 2005.

[6] A. P. Craig, R. Jacob, B. Kauffman, T. Bettge, J. Larson, E. Ong, C. Ding, and Y. He, "Cpl6: The new extensible, high performance parallel coupler for the community climate system model," *The International Journal of High Performance Computing Applications*, vol. 19, no. 3, pp. 309–327, 2005.

[7] J. Lofstead, F. Zheng, S. Klasky, and K. Schwan, "Adaptable, metadata rich io methods for portable high performance io," in *2009 IEEE International Symposium on Parallel & Distributed Processing*. IEEE, 2009, pp. 1–10.

[8] C. Docan, M. Parashar, and S. Klasky, "Enabling high-speed asynchronous data extraction and transfer using dart," *Concurrency and Computation: Practice and Experience*, vol. 22, no. 9, pp. 1181–1204, 2010.

[9] T. V. Martsinkevich, B. Gerofi, G. Y. Lien, S. Nishizawa, W. k. Liao, T. Miyoshi, H. Tomita, Y. Ishikawa, and A. Choudhary, "Dtf: An i/o arbitration framework for multi-component data processing workflows," in *International Conference on High Performance Computing*. Springer, 2018, pp. 63–80.

[10] Northwestern University, "The s3d-io benchmark," https://github.com/wkliao/S3D-IO.

[11] Fujitsu, "File system and power management enhanced for supercomputer fugaku," https://www.fujitsu.com/global/about/resources/publications/technicalreview/2020-03/article05.html.

[12] Unidata, "Network common data form (netcdf)," https://www.unidata.ucar.edu/software/netcdf, UCAR Community Programs.

[13] Y. Ajima, T. Kawashima, T. Okamoto, N. Shida, K. Hirai, T. Shimizu, S. Hiramoto, Y. Ikeda, T. Yoshikawa, K. Uchida *et al.*, "The tofu interconnect d," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 646–654.

[14] RIKEN Center for Computational Science, "Fugaku supercomputer," https://www.r-ccs.riken.jp/en/fugaku/about, RIKEN Center for Computational Science.

[15] Fujitsu, "Fefs: Scalable cluster file system," https://www.fujitsu.com/downloads/TC/sc11/fefs-sc11.pdf.

[16] Joint Center for Advanced HPC (JCAHPC), "Basic specification of oakforest-pacs," http://jcahpc.jp/eng/ofp_intro.html.

[17] T. Miyoshi, T. Honda, A. Amemiya, S. Otsuka, Y. Maejima, J. Taylor, H. Tomita, S. Nishizawa, K. Sueki, T. Yamaura *et al.*, "Big data assimilation: Real-time demonstration experiment of 30-second-update forecasting in tokyo in august 2020," in *EGU General Assembly Conference Abstracts*, 2021, pp. EGU21–6890.

[18] M. Dorier, M. Dreher, T. Peterka, J. M. Wozniak, G. Antoniu, and B. Raffin, "Lessons learned from building in situ coupling frameworks," in *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, 2015, pp. 19–24.

[19] S. Valcke, V. Balaji, A. Craig, C. DeLuca, R. Dunlap, R. Ford, R. Jacob, J. Larson, R. O'Kuinghttons, G. Riley *et al.*, "Coupling technologies for earth system modelling," *Geoscientific Model Development*, vol. 5, no. 6, pp. 1589–1596, 2012.

[20] S. Valcke, "The oasis3 coupler: A european climate modelling community software," *Geoscientific Model Development*, vol. 6, no. 2, pp. 373–388, 2013.

[21] V. Vishwanath, M. Hereld, and M. E. Papka, "Toward simulation-time data analysis and i/o acceleration on leadership-class systems," in *2011 IEEE Symposium on Large Data Analysis and Visualization*, 2011, pp. 9–14.

[22] J. Dayal, D. Bratcher, G. Eisenhauer, K. Schwan, M. Wolf, X. Zhang, H. Abbasi, S. Klasky, and N. Podhorszki, "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, 2014, pp. 246–255.

[23] J. Liao, B. Gerofi, G. Y. Lien, S. Nishizawa, T. Miyoshi, H. Tomita, and Y. Ishikawa, "Toward a general i/o arbitration framework for netcdf based big data processing," in *European Conference on Parallel Processing*. Springer, 2016, pp. 293–305.

[24] R. Rew, E. Hartnett, J. Caron *et al.*, "Netcdf-4: Software implementing an enhanced data model for the geosciences," in *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanograph, and Hydrology*, vol. 6, 2006.