Systematic Benchmarking/Profiling Approaches to Improve I/O Performance of GloSea

Hassan Asghar, Soo-hyuck Choe, Jin-Wook Choi and Eun-Sung Jung
Department of Software and Communications Engineering
Hongik University
Sejong, South Korea
{haxxanasghar,shchoe,jwc9507}@g.hongik.ac.kr, ejung@hongik.ac.kr

***Abstract*—Scientific applications are journeying towards the exascale, producing a sheer amount of data that needs analysis along with a demand of significant I/O workload facing severe I/O performance bottlenecks. Many researchers have done lots of work on the computation aspect of scientific applications. On the other hand, in the case of I/O performance, this is not true. This paper aims to provide the benchmarking approaches for I/O performance improvement of climate prediction software GloSea encompassing transfer buffer size to read parallelism, dataset chunking and data compression. To conclude, we have also proposed optimization opportunities specific to GloSea using PnetCDF. Adopting these approaches not only empowers GloSea in terms of I/O performance improvement, but the strategies presented in this paper also brings more robustness and a stable communication acceleration.**

***Index Terms*—io optimization, glosea, benchmarking, darshan, climate prediction model**

I. Introduction

High performance computing (HPC) systems are progressing from petascale to exascale in coming years and will produce a prodigious amount of data, bestowing new I/O performance and management challenges.

There is tremendous growth in processor's computation power because of high-speed development in the semiconductor field in the last two decades. However, this pace has not been observed in the case of storage and I/O systems.

The imbalance between these two technologies leads to poor I/O performance and even put more pressure on I/O if simulations observations are from scientific applications like Climate [1], Fusion [2], Cosmology [3], Light Sources [4] and Astronomy [5] that produces gigabytes of data per second.

Many researchers have proposed multiple I/O optimization techniques to improve the I/O performance e.g., List I/O [6], DataType I/O [7], and Collective I/O [8]. I/O optimization is a very time consuming and error-prone job specifically for those applications where I/O behaviour is complex. I/O performance improvement is application dependent and varies from one application to another. One notable point is that, while optimization a researcher should take into account both things: the application itself and the system where the application is running.

Large scale climate modelling applications use High Performance Computing (HPC) facilities to speed up the solutions of very complex systems having a sheer amount of dataset along with frequent read and write operations. And if this simulation runs on multiple compute nodes spanning hundreds, then it becomes a potential bottleneck. In our case, we have a climate prediction application named GloSea (Global Seasonal Forecasting System) a Global Climate Model (GCM) created by the Met Office, the UK Meteorological Agency. GloSea executes I/O in a serial format using the netCDF API interface. In addition, the model consists of four main parts: the preprocessing process to prepare the input field required for model execution, the model execution process to perform actual prediction through the integration process of the numerical model, the ensemble probability prediction process to compensate for the uncertainty of the model, Finally, there is a validation process that evaluates the predictability and accuracy of the model using observations. The structure of GloSea model runs are given in Fig. 1

In this paper, we discussed approaches that would improve the I/O performance of GloSea: a) Transfer Buffer Size: It is related to the granularity level of the I/O and then executed on every request. b) Read Parallelism: For parallel reading, how many parallel threads are being used during the running. c) Preprocessing Parallelism: Preprocessing parallelism deals with data related operations e.g., decoding and data mutations. d) Dataset Chunking: Here, the dataset would split into fixed-sized data chunked and each data operation is performed on that particular chunk of the file system. e) Data Compression: There are multiple data compression libraries like GZip and LZF that supports the variable compression levels in HDF5and netCDF.

The rest of the paper is organized as follows: Sect. II reviews the related work. Sect. III presents the benchmarking and profiling. Then, Sect. IV discusses the experimental evaluation. After that, Sect. V describes the discussion. At last, the conclusion is highlighted in Sect. VI.

II. Related Work

The scientific application running on High Performance Computing (HPC) facility is usually consumed most of the time in three stages: a) communication, b) compute and c) read/write into the persistent storage. There are a lot of tools and techniques available for communication and computation

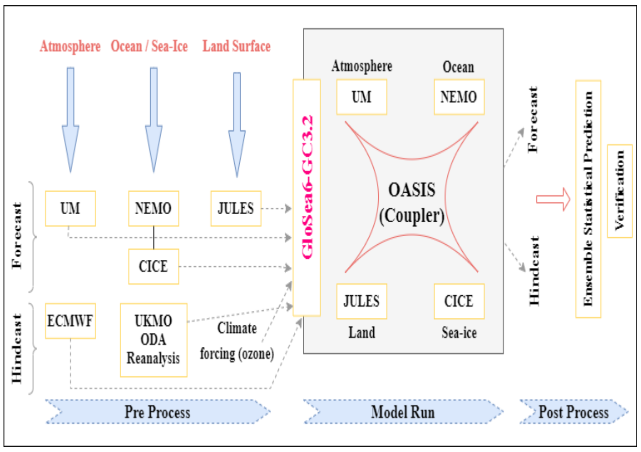


Fig. 1: Structure of GloSea6 Model Runs.

because of the maximum time spent over these two stages but, I/O is lagging and becomes a potential bottleneck. Thus, it is necessary to first profile/characterize the I/O before reaching a solid solution. There are many approaches and techniques suggested to accelerate the I/O in HPC.

Many Scientists proposed I/O optimization approaches using compression methods. For example, a lossy compression technique named Orthogonal Matching Pursuit (OMP) has been proposed for data-intensive applications to improve the bottleneck from storage in terms of bandwidth utilization acceleration. Reconstruction of compressed data is a bit costly as it utilizes the time and memory for I/O gain [9].

On the other hand, replication base strategies have also been presented for parallel I/O optimization. A pattern-directed and replication-based approach has been discussed named PDLA having two parts as H-PDLA for HDD storage and S-PDLA for SSD storage. In applications having low I/O concurrency H-PDLA first identify the access patterns and then generate a reorganized replica of each pattern. Moreover, for applications having a high I/O access rate, S-PDLA only replicates the most critical portion of the access pattern resulting in terms of reading/write bandwidth acceleration [10].

III. Benchmarking and Profiling

In this section, we have discuss the profiling and benchmarking tool that we will use in our experiments throughout this work.

A. Profiling Tool

We will use Darshan [11] for our profiling tool together with MADbench2 to generate the logs for benchmarking/profiling. Darshan has been used widely in industry and academia for I/O profiling as there is no need for source code modification and could be easily injected into the application of interest at the link phase of MPI compile script e.g., *mpicc*. In addition, Darshan adopts a portable mechanism to intercept the I/O routines. Dive deeper, darshan consists of two main components *darshan-runtime* and *darshan-util*. *Darshan-runtime* is used for instrumenting the MPI application of interest and creating the I/O logs. Once I/O characterization logs are created successfully, *darshan-util* comes into the picture to analyze the log files generated in the previous step. The resemblance of *darshan-runtime* is like a library (*libdarshan* over here) that actually intercept the I/O calls and inject the desired functions into the system libraries. *Darshan-core* (key component) and *darshan-common* (for module developer) are the main components of *darshan-runtime*. How this whole process is carried out is given in Fig 2. One point is notable over here that Darshan doesn't capture the whole I/O traces rather adopts cumulative time information.

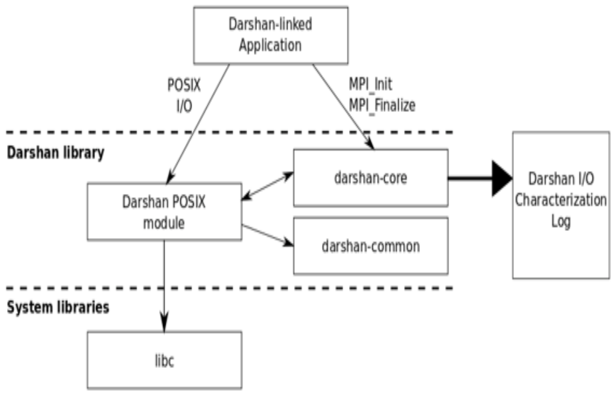


Fig. 2: Workflow of darshan.

B. Benchmarking Tool

MADbench2 is a leading benchmarking tool to test the I/O performance of scientific applications in a massively parallel architecture [12]. In addition, MADbench2 is procured from a cosmology application that inspects the Cosmic Microwave Background dataset. MADbench runs in dual mode as 1) Regular mode, where full code execute. 2) I/O mode, where calculation/communication is altered into some busywork. Diving deeper, there are multiple component functions e.g., S, D, W, and C that run in a single or multi-gang manner aiming to make the data more dense during W phase of computation. MADbench2 parameters are explained in the Table I.

TABLE I: MADbench2 parameters with explanation

Parameter	Explanation
NPRO	Number of processes
NPIX	Pseudo-data size
NBIN	Pseudo-data size
NGANG	Gang-level parallelism
SBLOCKSIZE	Block size matrices
FBLOCKSIZE	File block size – all IO will start at a file-offset
RMOD	Simultaneous reading

C. XIOS Server

XIOS stands for XML IO Server and is used in data production scenarios for climate modelling applications e.g., GloSea. In XIOS Server [13], different models are coupled together using OASIS Coupler, where simulations are long-running, generating a lot of data (PB of data). XIOS resolved the following challenges: a) Flexibility in I/O management and definition of data, b) I/O performance acceleration challenge and c) post-treatment with data produced. The functionalities of the XIOS Server are shown in Fig 3.

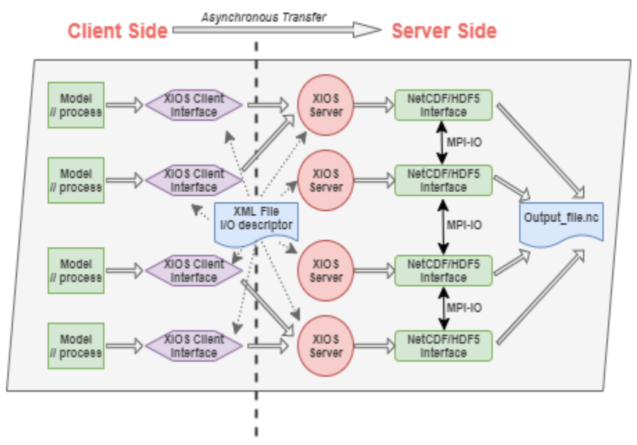


Fig. 3: Functionalities of XIOS Server.

D. *Parameter Set and Performance Metrics*

MADbench2 offers many parameters to see the application behaviour/pattern and later will be used for performance optimization. We have chosen the following parameters that are directly related to I/O performance.

- File Block Size: This parameter used as all I/O will start at a file-offset that is an integer multiple of FBLOCK-SIZE.
- Read Mod: It is used for simultaneous reading when set to 1 means one processor will read at once.
- Write Mod: Likewise read, write parallelism will be used for simultaneous write operations.

Remaining parameters are for setting the pseudo-code data related settings and gangs level parallelism. Besides aforementioned parameters, there are some other environment variables, out of them two variables are related with I/O given below.

- I/O Mode: Here, we can set the I/O mode before running the experiments as SYNC or ASYNC.
- File Type: File type defines that it should be a UNIQUE FILE or SHARED.

Besides the above, there are some other configurations specific to the application e.g., GloSea

- File Access Pattern: In most scientific applications, files are accessed utilizing an independent I/O i.e., POSIX or by exploiting the collective I/O interfaces i.e., MPI-IO. Independent I/O might be classified into further two scenarios as 1) where multiple processes are reading from multiple files and 2) where multiple processes are reading from a shared file. These two scenarios are highly dependent on the application as to how a dataset is being generated and divided for processing.

E. *Performance Metrics*

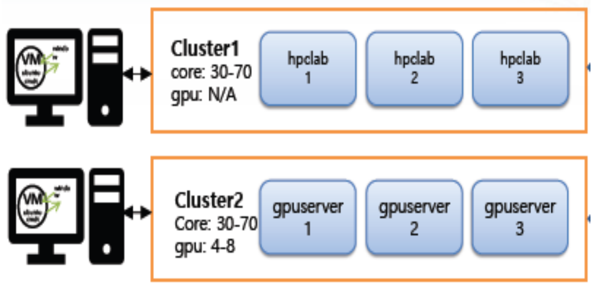
In order to see the I/O behavior and pattern there must be some performance metrics to gauge the performance of the application. I/O performance are usually calculated by using the two performance metrics: a) *Total Time Spent in I/O (sec)* and b) *I/O Bandwidth (mb/s)* We will use the I/O Bandwidth (mb/s) and Total Time Spent in I/O (sec) performance metrics for our analysis. These details can be captured using Darshan logs as it is the only tool widely used for I/O related information fetching. Besides Darshan, MADbench2 provides more in-depth details as it divides the I/O bandwidth in Read Bandwidth and Write Bandwidth and a mechanism to calculate it using the output logs of MADbench2 but when it comes to actual application like GloSea the infomration can be get only using Darshan.

IV. Experimental Evaluation

In this section, we will discuss the experimental setup running MADbench2 together with Darshan.

A. *Testbed Description*

We will run our benchmark tool "MADbench2" with Darshan together for recording I/O related details. We have three different variations in hardware and the detail of each facility is given below in detail as a) *Cluster1*: Consist upon three nodes as HPC1, HPC2 HPC3 each having 30-70 cores. b) *Cluster2*: Consist upon three nodes as GPUServer1, GPUServer2 GPUServer3 each having 30-70 cores with 3-4 GPU. Organization and specifications of cluster are given in Fig 4.



(a) Cluster1 and cluster2 specification.

(b) Cluster1

(c) Cluster2

Fig. 4: Testbed specification and organization.

Finally, c) *Supercomputer*: Our Cray XC system is based on the Massively Parallel Processor (MPP) architecture equipped with Intel Haswell processors and consists of three 8 cages cabinets with a peak theoretical performance of 447.3 TFlop/s. In addition, the memory capacity of the compute node portion of the entire system is 56.0TB and using Cray Sonexion data storage system that provides more than 3 PiB of usable storage and more than 50GB/sec of I/O bandwidth, including a total of two login nodes and twelve post-processing nodes of the same architecture as the compute nodes.

B. Methodology

The whole procedure of benchmarking/profiling for I/O performance improvement is defined in the below Fig. 5.

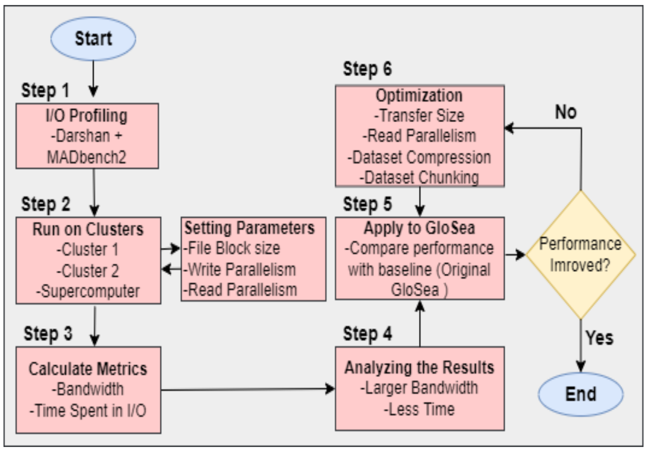


Fig. 5: Flowchart for I/O performance improvement.

First of all, we will run MADbench2 along with Darshan for profiling on different hardware variations like in the case we have categorized into Cluster1, Cluster2 and Supercomputer. Meanwhile, when we configure the benchmark, we apply different parameters setting e.g., File Block Size, Write Mod and Read Mod etc. After a successful run, we have now the Darshan log file and we can calculate our performance evaluation metrics like I/O Bandwidth (mbps) and Time Spent in I/O (sec). After calculating the performance metrics, we are now able to to analyze that at which parameter setting we have larger bandwidth and less I/O time. After analysis, the best setting (at which Block Size, and simultaneous Read Size we are getting lower time and greater bandwidth). Mirror these settings to the original application like Glosea. Now we have run our original application along with the optimized setting, now it's time to compare this application in terms of performance (bandwidth and I/O) with the baseline. If performance improved compared with baseline simply ends the process and we have achieved the threshold like (10% or so) otherwise go to again the optimization phase and apply more optimization techniques like compression or read and write threads increasing.

V. Disscussion

A. Approaches

Here we discuss approaches that would improve the I/O performance GloSea: a) *Transfer Buffer Size*: It is related to the granularity level of the I/O and then executed on every request. b) *Read Parallelism*: For parallel reading, how many parallel threads are being used during the running. c) *Preprocessing Parallelism*: Preprocessing parallelism deals with data related operations e.g., decoding and data mutations. d) *Dataset Chunking*: Here, the dataset would split into fixed-sized data chunked and each data operation is performed on that particular chunk of the file system. e) *Data Compression*: There are multiple data compression libraries like GZip and LZF that supports the variable compression levels in HDF5 and netCDF.

There are some optimization opportunities specific to GloSea perspective that could enhance the I/O performance gain. As GloSea serially execute I/O exploiting netCDF API interface. Many of organization uses the netCDF file format in the scientific community especially in climate modelling applications to store large datasets. We can see in the following figure how netCDF operate serially in Fig. 6

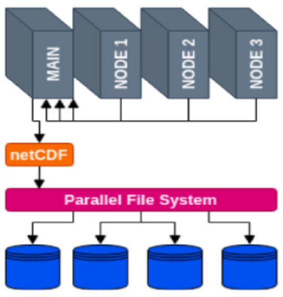


Fig. 6: Serial processing of netCDF.

Meanwhile, if we use PnetCDF (parallel I/O library), which read and write netCDF files exploiting parallel I/O. PnetCDF is the best candidate for parallel I/O because it gives the same interface as netCDF API but uses the MPI-IO for simultaneous read and writing between the nodes. PnetCDF provides the following optimizations opportunities:

- Buffering for reads and writes.
- Allow underline MPI structure to decide the best time for read/write operations.
- Optimized accessing of the same variable many times.
- PnetCDF is built on MPI-IO so it allows hints to pass to the underlying structure.
- Hints are specific to PnetCDF and ROMIO hints as well.
- Hints are for PFS striping, Chunk Size and cache sizes.
- Along with the above, PFS specific hints like Lustre could improve the I/O performance significantly.

VI. Conclusion

In this paper, we have provided the benchmarking approaches for I/O performance improvement of climate prediction software GloSea. These include transfer buffer size, read parallelism, dataset chunking and data compression. We will run all these approaches in our weather prediction software GloSea and calculate the performance metrics: e.g. total time spent in I/O (sec) and I/O bandwidth (mb/s) using variable parameter set (file block size, read mod, write mod). In addition, we have also discussed I/O optimization opportunistic specific to GloSea using PnetCDF includes buffering for read/write, hints for PFS striping, chunk size and cache sizes. The approaches suggested over here can be extended to other climate applications too that confront homogenous I/O challenges.

Acknowledgment

This work was funded by the Korea Meteorological Administration Research and Development Program under Grant KMI-2021-01310.

References

[1] D. Bader, W. Collins et al., "Accelerated Climate Modeling for Energy (ACME) Project Strategy and Initial Implementation Plan," 2014.

[2] C.-S. Chang, S. Ku, and H. Weitzner, "Numerical Study of Neoclassical Plasma Pedestal in a Tokamak Geometry," Physics of Plasmas, vol. 11, no. 5, pp. 2649–2667, 2004.

[3] S. Habib, V. Morozov et al., "HACC: Extreme Scaling and Performance across Diverse Architectures," in SC13. ACM, 2013, p. 6.

[4] A. MacDowell, D. Parkinson et al., "X-Ray Micro- Tomography at the Advanced Light Source," in Developments in X-Ray Tomography VIII, vol. 8506. International Society for Optics and Photonics, 2012, p. 850618.

[5] "turbulence energetics in stably stratified geophysical flows: Strong and weak mixing regimes."

[6] A. Ching, A. Choudhary, K. Coloma, W. Liao, R. Ross, and W. Gropp, "Noncontiguous I/O Accesses through MPI-IO," in Proceedings of IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, 2003.

[7] A. Ching, A. Choudhary, W. Liao, R. Ross, and W. Gropp, "Efficient Structured Data Access in Parallel File Systems," in Proceedings of IEEE International Conference on Cluster Computing, 2003

[8] R. Thakur, W. Gropp, and E. Lusk, "Data Sieving and Collective I/O in ROMIO," in Proceedings of the 7th Symposium on the Frontiers of Massively Parallel Computation, 1999.

[9] H. M. Makrani, H. Sayadi, S. Manoj, S. Raftirad, and H. Homayoun, "Compressive sensing on storage data: An effective solution to alleviate I/0 bottleneck in data- intensive workloads," 2018 IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP), 2018.

[10] S. He, Y. Yin, X.-H. Sun, X. Zhang, and Z. Li, "Optimizing parallel I/O accesses through pattern-directed and layout-aware replication," IEEE Transactions on Computers, vol. 69, no. 2, pp. 212–225, 2020.

[11] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, "24/7 characterization of Petascale I/O workloads," 2009 IEEE International Conference on Cluster Computing and Workshops, 2009.

[12] "Virtual Institute for I/O," MADbench2 [Virtual Institute for I/O]. [Online]. Available: https://www.vi4io.org/tools/benchmarks/madbench2. [Accessed: 31-Oct-2021].

[13] "Welcome to Xios Wiki Page," XIOS. [Online]. Available: https://forge.ipsl.jussieu.fr/ioserver. [Accessed: 16-Nov-2021].