Tuning Parallel I/O on Blue Waters for Writing 10 Trillion Particles

Surendra Byna*, Robert Sisneros†, Kalyana Chadalavada†, and Quincey Koziol‡

*Lawrence Berkeley National Laboratory, USA. Email: sbyna@lbl.gov
†National Center for Supercomputing Applications, USA. Email: {sisneros,kalyan}@illinois.edu
‡The HDF Group. Email: koziol@hdfgroup.org

***Abstract*—Large-scale simulations running on hundreds of thousands of processors produce hundreds of terabytes of data that need to be written to files for analysis. One such application is VPIC code that simulates plasma behavior such as magnetic reconnection and turbulence in solar weather. The number of particles VPIC simulates is in the range of trillions and the size of data files to store is in the range of hundreds of terabytes. To test and optimize parallel I/O performance at this scale on Blue Waters, we used the I/O kernel extracted from a VPIC magnetic reconnection simulation. Blue Waters is a supercomputer at National Center for Supercomputing Applications (NCSA) that contains Cray XE6 and XK7 nodes with Lustre parallel file systems. In this paper, we will present optimizations used in tuning the VPIC-IO kernel to write a 5TB file with 5120 MPI processes and a 290TB file with 300,000 MPI processes.**

I. Introduction

Large-scale simulations running on hundreds of thousands of processors produce hundreds of terabytes of data that need to be written to files for future analysis. One such application is VPIC code that simulates plasma behavior such as magnetic reconnection and turbulence in solar weather. The number of particles VPIC simulates is in the range of trillions and the size of data files to store is in the range of hundreds of terabytes. To test and optimize parallel I/O performance at this scale on Blue Waters, we used the I/O kernel extracted from a VPIC magnetic reconnection simulation. Blue Waters is a supercomputer at National Center for Supercomputing Applications (NCSA) that contains Cray XE6 and XK7 nodes with Lustre parallel file systems. In this paper, we will present optimizations used in tuning the VPIC-IO kernel to write a 5TB file with 5120 MPI processes and a ≈290TB file with 300,000 MPI processes.

HDF5 is a versatile data model that can represent a number of complex data objects and a wide variety of metadata. HDF5 provides a software library that can run on a variety of computing systems ranging from laptops to massively parallel systems. The HDF5 file format is portable and comes with a high-level application programming interface for several high-level programming languages. Parallel I/O is a special feature of the HDF5 library that offers various I/O optimizations for parallel computers. HDF5 implements parallel I/O by exploiting features of MPI, including collective communication and I/O, along with internal algorithmic optimizations that enable high-performance application I/O. When an application requests a collective I/O operation, HDF5 generates two MPI datatypes, one that describes the application memory region and another that describes the region in the HDF5 file to access. The versatility and flexibility of the HDF5 library attracted numerous applications in reading and writing scientific data.

Vector Particle-in-Cell (VPIC) is a highly optimized code for simulating plasma physics phenomenon. In this paper, we use the VPIC-IO kernel, that mimics the data fields of a magnetic reconnection simulation [6]. The number of particles written by the kernel scales as the number of processes increase. As the number of processes grow to hundreds of thousands, writing to the same file can be become a performance bottleneck. We show that proper distribution of write load among processes and among I/O servers of the parallel file system is beneficial. We also test a new feature of HDF5, called multi-dataset writes, where multiple HDF5 datasets can be written to the file without a collective call between each dataset write operation.

VPIC-IO kernel uses HDF5 to write eight properties of each particle. The size of properties for each particle is 32 bytes and in our configuration each MPI process operates on 32 milion particles. HDF5 is an I/O library that provides flexible hierarchical object representation of scientific data. The parallel I/O component of HDF5 offers various optimizations to make best use of file system bandwidth on large-scale systems. These flexibility and optimization features have attracted numerous HPC applications to write and to read their data using HDF5. We have designed experiments to catalog VPICs performance and scalability on Blue Waters Lustre filesystem. HDF5 and Lustre provide several tunable parameters. We are therefore interested in analyzing the effects of these parameters on performance as well as one another toward determining upon which optimal performance and scalability are most dependent. Additionally, we test newly introduced multi-dataset write feature of HDF5. We apply the results of our analyses to the writing of over 10 trillion particles of VPIC (≈290TB) to a single shared file.

While HDF5 offers various optimizations and features, choosing the right combinations is necessary to obtain good I/O performance. In this paper, we present an application case study running VPIC [5] on Blue Waters at the National Center for Supercomputing Applications (NCSA). Since HDF5 opti-

mizations vary based on the underlying parallel file system and the data access patterns of applications, our study must leverage many relevant optimization strategies, including those specific to Blue Waters.

Overall, the contributions of this paper are:

- Studying the scalability of writing up to 10 trillion particles of VPIC, that amounts to ≈290 TB data file
- Testing the newly introduced multi-dataset write feature of HDF5

The reminder of the paper is organized as follows: We describe our target hardware, Blue Waters, as well as our target application, VPIC, in Section II. We then discuss related work in Section III. We analyze the parallel I/O performance VPIC in Section IV and conclude in Section V.

II. Background

A. Blue Waters Systems Overview

Blue Waters is a Cray XE6-XK7 supercomputing system managed by the National Center for Supercomputing Applications for the National Science Foundation. The system has a peak performance of 13.34 PF, aggregate IO throughput in excess of 1 TB/s, 26 PB online disk capacity and nearly 200 PB of nearline storage. Blue Waters contains two types of compute nodes: XE6 and XK7. There are 22,640 XE6 nodes and 4,224 XK7 nodes. Each XE6 node has two 16 core AMD 6276 CPUs, 64 GB of main memory. Each XK7 node has one 16 core AMD 6276 CPU, 32 GB of main memory and one Nvidia Kepler K20X graphics processing unit (GPU) with 6 GB of GDDR5 on-board memory.

The compute and file system nodes are interconnected using the Cray Gemini high speed interconnection network. Two nodes share a single Gemini ASIC (Application-Specific Integrated Circuit), which contains two network interface controllers (NICs) and a YARC-2 router. The network is organized in a 24 X 24 X 24 3D torus topology.

Blue Waters provides users with three distinct file systems: home file system for user home areas, project file system for group level file sharing and collaboration, and a scratch file system for applications. All are implemented using the Cray Sonexion Lustre storage technology and each has its own metadata and object storage servers. The scratch file system is the fastest of the three and provides approximately 980 GB/s peak throughput. The scratch file system consists of 1440 Object Storage Targets (OST) spread among 360 Object Storage Servers (OSS). Each pair of OSS manage eight OSTs in an active-active failover high availability configuration.

Using the Cray Lustre Network Driver (LND), compute nodes mount Lustre file systems over the Gemini network. File system operations from the clients are routed to the appropriate Lustre device using the Cray XIO nodes that implement the LNET routing services and handle packet routing between different networks. There are 482 LNET routers for the scratch file system. Currently, the compute nodes use Lustre version 1.8.6.

On Blue Waters, LNET routers are organized into groups of four and each group is connected to three OSSs. A set of four LNET routers are configured as the primary routers for OSTs on these OSS units. A second group of LNETs act as a backup ensuring alternate routes to any OST in the event of failures.

B. The VPIC Application

VPIC [5] is a highly optimized and scalable particle-in-cell code to simulate plasma physics phenomenon, such as collisionless magnetic reconnection [7]. Magnetic reconnection is an important mechanism that releases energy explosively as field lines break and reconnect in plasmas, such as when the Earth's magnetosphere reacts to solar eruptions. VPIC is able to simulate trillions of particles using hundreds of thousands of CPU cores. Each particle has a total of eight properties including their location, id, and momentum in three dimensions. Since the simulation takes extensive amount of computation, we have separated the I/O kernel of VPIC and developed VPIC-IO benchmark [6]. VPIC-IO benchmark (also referred as VPIC-IO kernel) writes particle data from each MPI process to a shared file using the H5Part library [8] and HDF5 file format. Each particle is represented by six floating point and two integer values. Each of these properties is structured as a HDF5 dataset. In our experiments, particle count was kept constant at 32 million per rank.

HDF5 recently implemented a new feature called multi-dataset write operations. Traditionally, HDF5 required a collective operation after finishing each HDF5 dataset, causing all processes to block after writing a dataset. In case of VPIC-IO, where eight properties of particles are written into eight HDF5 datasets, the MPI processes were synchronizing eight times. With multi-dataset write operation, the processes only synchronize once at the end of writing all the datasets. When testing the performance of multi-dataset operations, we used a modified implementation of the VPIC-IO kernel with multi-dataset write calls.

III. Related Work

Various optimization strategies have been proposed to tune parallel I/O performance of applications or I/O kernels. Howison et al. [9] also perform manual tuning of various benchmarks that select parameters for HDF5 (chunk size), MPI-IO (collective buffer size and the number of aggregator nodes) and Lustre parameters (stripe size and stripe count) on Hopper supercomputer at NERSC. Yu et al. [10] characterize, tune, and optimize parallel I/O performance on Lustre file system of Jaguar, a Cray XT supercomputer, at Oak Ridge National Laboratory (ORNL). The authors tuned data sieving buffer size, I/O aggregator buffer size, and the number of I/O aggregator processes. In our study, in addition to applying the strategies proposed by existing research, we took advantage of two new HDF5 features.

Recent I/O auto-tuning efforts [4], [3], which include some of the authors of this paper, traverse the vast optimization parameter space and select optimal parameters the achieve the best I/O. The performance advantage shown in this paper

adds few more optimizations to the search space, namely: appending hyperslabs and multi-dataset operations.

IV. Results

In evaluating VPIC on Blue Waters system, we used default Cray Compilation Environment (CCE, v.8.2.2), Cray MPI (v6.2.0), and Parallel HDF5 1.8.11 for compiling the VPIC-IO kernel. We ran the VPIC-IO experiments on a non-dedicated system. To factor out noise from other jobs on the system, we ran the experiments multiple times and discarded the outliers. A custom Lustre client, v.1.8.6 with backported bug fixes and patches is used on the Blue Waters compute nodes. Though the Blue Waters scratch file system contains 1440 OSTs, the current Lustre deployment allows a single file to be striped across a maximum of 160 OSTs. Our theoretical peak is iconsequently reduced to approximately 107 GB/s.

A. Tuning baseline performance

We used an instance of VPIC-IO kernel using 5120 MPI processes as the baseline for tuning I/O performance. At this scale, VPIC-IO writes a 5 TB file to disk. We ran this test multiple times to determine optimal file striping parameters and runtime environment configuration. Initial experiments consistently delivered a throughput of $\sim$25 GB/s. We instrumented the code to measure performance of file open, write, and close operations individually. We identified that the file close operation was consuming an abnormally high amount of time. *H5Fclose* function performs file size verification operations that either extend or truncate the file to match the allocated size. These operations initiate several expensive metadata operations and negatively impact the file close time [6]. We patched the HDF5 v1.8.11 to disable the file verification step in *H5Fclose* and compiled with GCC v4.8.2. This resulted in significantly reduced file close time.

VPIC-IO kernel calls H5Part write function resulting in a transfer size of 128 MB per variable. We set the HDF5 alignment to 128 MB using the H5Part API. The Blue Waters runtime configuration turns on the Cray MPI-IO Collective Buffering algorithm by default. This optimization helps improve I/O performance in specific cases of large unaligned writes and small writes to the same OSTs [1]. Collective Buffering is a two step process that involves data movement among nodes and consolidates I/O requests among aggregators. A single variable write from 5120 node amounts to 655 GB, which is prohibitively large for collective buffering to yield any benefit. To minimize the data movement, we turn off collective buffering. With this tuning parameters, we observed an I/O performance of 43.78 GB/s in writing 5 TB particle data by VPIC-IO kernel.

B. Performance of 10 trillion particle run

Based on the performance of the baseline configuration, we devised a larger run configuration as follows: 298,048 ranks, 32 million particles per MPI process resulting in more than 10 trillion particles and 291 TB on disk file size. Tuning parameters from the baseline run were applied to this job. However, compute node to OST ratio would be 18628:160 compared to 320:160 for baseline run resulting in an OST over-subscription of 116:1 compared to 2:1. This was determined to be an unrealistic configuration to achieve best performance. To improve the OST to compute node ratio, we doubled the number of ranks per node to 32 using 9,314 nodes resulting in an OST over-subscription of 58:1. In order to further minimize OST contention, we increased the stripe size of Lustre to 1 GB. With these tuning parameters, we observed a performance of 51.81 GB/s. In Fig. 1, we show the performance of VPIC-IO as the number of processes increase. The I/O performance increased as we use more number of processes because of reduced contention from other jobs when using higher portion of the Blue Waters system for running VPIC-IO.

C. Performance of HDF Muti-Dataset operations

HDF Multi-dataset operations support single collective operation on multiple datasets. To evaluate the performance of this feature, we modified VPIC-IO to use the multi-dataset write calls. Job configuration was set to match the baseline configuration described above. Initial experiments indicate that write performance benefits to some extent.

To further explore opportunities for improving throughput, we considered reducing OST contention. In the default MPI rank placement scheme, SMP-style, ranks are assigned sequentially starting from core 0 on node 0 within the allocated nodes. This rank layout along with a stripe count of 160 results in each node communicating with 16 OSTs and each OST is written to by 32 processes on 32 different nodes. Minimizing the number of nodes communicating with an OST can improve performance. Collective buffering will aggregate I/O with one or more writers per OST but is inefficient due to the large data transfer size of VPIC-IO. Cray MPI provides a rank reordering mechanism to allow users to place ranks in a custom configuration [2]. Using the rank reordering method, ranks that would write to the same OST can be placed on two or four consecutive nodes. This rank placement guarantees at any given time a single OST is accessed by two or four nodes, respectively, and that the interconnect path from the writer to the OST is the same for all writers. With this approach, the spirit of collective buffering is retained without excessive data movement among nodes for aggregation. We used a stripe size of 1 GB, which yielded the best performance for the 10 trillion particle run, for the multi-dataset write experiment. As shown in Table I, the multi-dataset write achieves $approx$56 GB/s performance, $approx$1.3X improvement over the single-dataset write performance for storing 5TB data.

D. I/O Performance Limitations

The best performance observed during these experiments represents 52.5% of the theoretical peak of 160 OSTs. While this is 2x the performance of the default configuration (25 GB/s), there is still room for improvement. Large-scale runs using a single shared file will also need Lustre wide striping

Run	Code	Nodes	Cores	File Size	Stripe Size	Write Time	Total HDF Time	Throughput
Max Throughput	Multiple Dataset	320	5,120	5 TB	1 GB	91.08 s	167.78 s	56.21 GB/s
Code Comparison	Single Dataset	320	5,120	5 TB	128 MB	116.94 s	117.37 s	43.78 GB/s
Hero Run	Single Dataset	9,314	298,048	291 TB	1 GB	5,763.14 s	5,779.89 s	51.81 GB/s

TABLE I
COMPARISON OF VPIC-IO KERNEL PARAMETERS AND OBSERVED I/O THROUGHPUT.

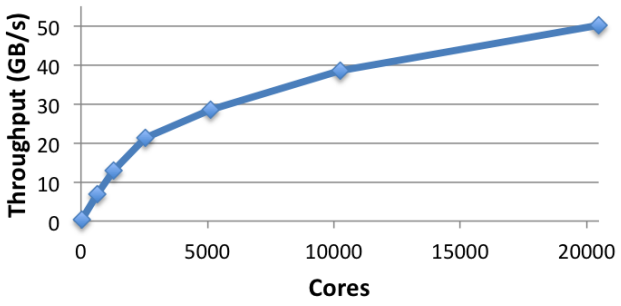

Fig. 1. As the number of cores writing to a single shared file increases, observed I/O throughput increases accordingly.

support so the compute node to OST ratio can be maintained at reasonable levels. Experiments with newer versions of Lustre client and tuning client side parameters like $max\_rpcs\_in\_flight$ have shown promising improvement in per-node throughput. It is preferable to use one OST per OSS to maximize performance. However, due to limitation in current system configuration, the default OST assignment provided by the MDS server was used. Future work will focus on the impact of changes in the Blue Waters Lustre configuration and the resulting impact on I/O throughput of this workload.

V. CONCLUSIONS

Tuning parallel I/O for large-scale simulations and analysis functions is often challenging. In this study, we tuned I/O performance of an application running using newly developed features of parallel HDF5.

We used an instance of VPIC-IO kernel using 5120 MPI processes as the baseline for tuning I/O performance. At this scale, our VPIC I/O kernel writes a 5 TB file to disk to 160 OSTs. With a reasonable initial configuration we consistently achieved only 25% of the theoretical peak for these write operations. We found the system default collective buffering algorithm to directly conflict with our careful efforts to align I/O operations. Simply correcting this allowed us to nearly double our throughput. Additional configuration updates allowed us to not only carry this increase to our 10 trillion-particle run, but also improve upon it. This is a significant result as the 298,048 ranks for that test are still writing to only 160 OSTs.

There is therefore more scope for I/O throughput improvement that requires configuration changes to the parallel file systems. In our future work, we will consider those changes and tune the I/O performance further. Our eventual goal is to generalize these optimizations and select them automatically based on the I/O patterns of the applications.

ACKNOWLEDGMENT

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. This work is supported by the Director, Office of Science, Office of Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

[1] Getting Started on MPI I/O. http://docs.cray.com/books/S-2490-40/. S249040.
[2] Message Passing Toolkit (MPT) 7.0 Man Pages. http://docs.cray.com/cgi-bin/craydoc.cgi?mode=Show;f=man/xe_mptm/70/cat3/intro_mpi.3.html.
[3] B. Behzad, S. Byna, S. M. Wild, M. Prabhat, and M. Snir. Improving Parallel I/O Autotuning with Performance Modeling. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, 2014.
[4] B. Behzad, L. Huong Vu Thanh, J. Huchette, S. Byna, et al. Taming Parallel I/O Complexity with Auto-Tuning. SC '13, 2013.
[5] K. J. Bowers, B. J. Albright, L. Yin, B. Bergen, and T. J. T. Kwan. Ultrahigh performance three-dimensional electromagnetic relativistic kinetic plasma simulation. *Physics of Plasmas*, 15(5):7, 2008.
[6] S. Byna, A. Uselton, Prabhat, D. Knaak, and H. He. Trillion Particles, 120,000 cores, and 350 TBs: Lessons Learned from a Hero I/O Run on Hopper. In *Proceedings of 2013 Cray User Group*, May 2013.
[7] W. Daughton, J. D. Scudder, and H. Karimabadi. Fully kinetic simulations of undriven magnetic reconnection with open boundary conditions. *Physics of Plasmas*, 13, 2006.
[8] M. Howison, A. Adelmann, et al. H5hut: A High-Performance I/O Library for Particle-Based Simulations. In *IASDS 2010*, Heraklion, Crete, Greece, Sept. 2010. LBNL-4021E.
[9] M. Howison, Q. Koziol, et al. Tuning HDF5 for Lustre File Systems. In *IASDS 2010*, Sep 2010.
[10] W. Yu, J. Vetter, and H. Oral. Performance characterization and optimization of parallel I/O on the Cray XT. In *International Parallel and Distributed Processing Symposium, 2008 (IPDPS 2008)*, April 2008.