Scientific Applications Performance Evaluation on Burst Buffer

George S. Markomanolis, Bilel Hadri, Rooh Khurram, and Saber Feki

Supercomputing Core Laboratory, King Abdullah University of Science and Technology, Thuwal, KSA

Abstract. Parallel I/O is an integral component of modern high performance computing, especially in storing and processing very large datasets, such as the case of seismic imaging, CFD, combustion and weather modeling. The storage hierarchy includes nowadays additional layers, the latest being the usage of SSD-based storage as a Burst Buffer for I/O acceleration. We present an in-depth analysis on how to use Burst Buffer for specific cases and how the internal MPI I/O aggregators operate according to the options that the user provides during his job submission. We analyze the performance of a range of I/O intensive scientific applications, at various scales on a large installation of Lustre parallel file system compared to an SSD-based Burst Buffer. Our results show a performance improvement over Lustre when using Burst Buffer. Moreover, we show results from a data hierarchy library which indicate that the standard I/O approaches are not enough to get the expected performance from this technology. The performance gain on the total execution time of the studied applications is between 1.16 and 3 times compared to Lustre. One of the test cases achieved an impressive I/O throughput of 900 GB/s on Burst Buffer.

Keywords: DataWarp, I/O, Burst Buffer

1 Introduction

While the computational power of new supercomputers is increasing significantly, I/O throughput is not increasing with the same rate. The humorous statement by computer engineer Ken Batcher that "a supercomputer is a device for turning compute-bound problems into I/O-bound problems" is becoming more genuine, since I/O subsystems, are typically slow compared to others parts of a supercomputer. This is mainly due to the well-known performance gap that keeps outspreading between the computing components (focusing more on the speed) and the storage devices (focusing more on the capacity of storage and less on performance). Thus, any new technology that promises a boost in I/O performance is becoming popular among the HPC-related research groups. The real world engineering applications demand reduction in the total time to solution, not just on the compute time, especially in cases where I/O is a significant part of the whole execution time. Optimizing the I/O is a complex process as the

layers of data storage increases in modern supercomputers. In order to achieve better performance, it is harder but crucial to understand the signature of I/O during the execution of the application.
Burst buffer is a new I/O technology that adds a layer between the compute nodes and the standard parallel file system, for more details see [1]. In this paper, we are reporting the I/O performance of six applications on Burst Buffer technology in its Cray implementation DataWarp [2]. These applications are taken from various scientific disciplines, namely: computational fluid dynamics, combustion, climate and earth sciences. In addition to the real applications, two synthetic benchmarks are also studied. The performance of these six test cases is compared to Lustre parallel filesystem. For more information on start working with Burst Buffer and how the MPI I/O aggregators work, see [1].

2 Related Work

National Energy Research Scientific Computing Center (NERSC) have done significant work through NERSC Burst Buffer Early User Program [3]. They have shown cases where the Burst Buffer achieves better performance than Lustre. Moreover, they have identified various challenges to achieve better performance. We experienced similar challenges, and that is why in a following section we provide instructions for better utilization of the resources. In [4] the researchers evaluate the DataDirect Networks (DDN) Infinite Memory Engine (IME). For their experiments, they used IOR[5] which is designed to measure parallel file system I/O performance at both the POSIX and MPI-IO level, and the NEural Simulation Tool (NEST) applications, where they showed that IME has better I/O performance than GPFS file system. The main contribution of this paper is to report our experience on the Burst Buffer performance on a wide range of applications at large scale.

3 Burst Buffer

We used in our experiments the Shaheen II XC40 system [6], which includes richly layered data storage architecture. The primary data storage solution is a Lustre Parallel file system with a usable storage capacity of 17.2 PB delivering around 500 GB/s of I/O throughput. The Cray Sonexion 2000 installation is configured using 72 Scalable Storage Units (SSU) and 144 Object Storage Services (OSS) connected to the XC40 via 72 LNET router service nodes. ShaheenII is also composed of 268 Cray DataWarp (DW) accelerator nodes hosting a total of 536 Intel P3608 SSD cards. Each Burst Buffer node provides aggregated peak write/read bandwidth 6 GB/s and 10 GB/s respectively. The combination provides an aggregate Burst Buffer capacity of 1.56 PB to Shaheen users. This fast middle storage layer provides up to three times the performance of the Lustre parallel file system and it is connected directly to the compute nodes through Aries interconnection. The IOR benchmark was launched using all 268

DataWarp nodes and 5,628 compute nodes achieving 1.54 TB/s and 1.66 TB/s in IOR write and IOR read, respectively.

4 Experiments

In this section, we study six applications/benchmarks. The scope of this work is to evaluate their performance on Bust Buffer and identify the potential performance improvements.

4.1 Fluent

Widely used commercial code, Fluent [7], is used in this study. The provided binary was not compiled from the vendor with the appropriate MPICH version for Burst Buffer. Moreover, we used serial I/O during these tests. A large test case commonly known as F1 Race Car Test Case [8] is used. The computational domain is discretized by using 140 million cell mesh. These simulations are typically done for calculating external aerodynamics properties of race cars, with the objective of drag reduction. In order to understand the turbulent features emanating from the wake of the car, solution files are written, after every few time steps, to the disk for generating pictures and movies for post simulation analysis. Each solution dump generates 21GB of output data. In this study, the IO speed for Lustre and DataWarp is investigated. 20% improvement in I/O is observed on SSD for writing the file, as compared to Lustre. Three tests are done for checking repeatability. No significant variation is observed in the repeated tests. In the next study, the effect of specifying more SSD nodes is observed. 23% faster I/O is observed when 2 SSD nodes are specified, as compared to 1 SSD node, using more than 2 DW nodes, did not provide significant performance improvement.

4.2 Weather Research and Forecasting model (WRF)

Weather Research and Forecasting Model (WRF) [9] is one of the most used models in Earth Sciences related fields. WRF code consumes significant amount of core-hours on many supercomputers. The purpose of this study is to evaluate the I/O of a large simulation and identify if it can benefit from DataWarp. For this study, we use a benchmark with high-resolution domain - the Alaska domain with 1km resolution [10]. We use WRF v3.7.1 with Cray compiler, MPICH v7.4.2, 256 compute nodes, 4 MPI processes per node and 8 OpenMP threads. WRF can be configured to save the simulation data in history and restart files. The former includes simulation data for specific time steps and the later can be used to restart the simulation. We configure WRF for intensive I/O, so we save the history and restart files every 30 simulation minutes. The size for history and restart files is around 81 GB and 361 GB, respectively. We stage-in around 110 GB of data, and the stage-out phase after one hour of simulation yields almost 1 TB of data and this happens asynchronously, before and after the execution

of the model, without reserving compute resources. For the case that we save the data to one single file, we use PnetCDF. One significant problem with the measurement of I/O performance on WRF is the calculation of the I/O bandwidth through the reported time from the output file. However, sometimes it was quite difficult to understand why the performance is not efficient. We followed different approaches to investigate the situation. The first one, and more direct, is to activate the debug option on WRF to report all the routines that are called, to break down the I/O time, where we observed thousand smaller I/O calls that could decrease the performance.

Although the latest Cray Environment CNL 6.X is not available to us, which will fix some performance issues, we tried to increase the collective buffering to investigate any improvements and use the CrayPAT instrumentation tool to examine the I/O on DataWarp. For the sake of simplicity, some of the commands/outputs include only the information that is required. We use 64 MPI I/O aggregators for the restart files, and change the size of collective buffering to 32 MB, while the default is 8 MB.

In Table 1 we present the performance data from CrayPAT regarding the I/O performance while writing the restart file with default collective buffering 8 MB. The average achieved I/O bandwidth is 1.4 GB/s per MPI process, and each of them writes almost 4.4 GB, and totally we have 46,960 I/O calls of 8 MB each. However, the collective buffering is changed to 32 MB and the average I/O bandwidth is 1.98 GB/s with totally 12,028 I/O calls. Thus, the size of collective buffer is also important depending on the application, and in this case the I/O bandwidth was improved by 41.4%.

Write MBytes	Write Rate (MB/s)	Writes	Bytes/Call	Filename/PE
369,720	1,437	46,690	8,303,272	Total - wrfrst_d01
4,448	1,371	560	8,328,689	pe.584
4,456	1,390	562	8,313,976	pe.528

Table 1. Measuring WRF I/O with CrayPAT for the restart file with default collective buffer size

The CrayPAT results are not in agreement with WRF output because CrayPAT measures the I/O per MPI aggregator, but WRF reports the total time of the collective I/O, including the data communication.

From the CrayPAT analysis, Lustre achieved a peak performance of 65 GB/s while a peak of 82 GB/s was reached using Burst Buffer. According to the WRF output report timings, we have the results of Fig. 1 where the I/O write performance of Burst Buffer increases while we increase the number of DataWarp nodes and we always use 144 OSTs for Lustre I/O. The DataWarp performance gets closer to the Lustre, but not better than it. However, reading the input file on DataWarp is more efficient than Lustre and the total execution time is shorter. If we compare the total execution time which is presented in Fig. 1 on right side, then the Burst Buffer is faster than Lustre for almost all the cases,

with maximum improvement of 16%. In WRF there is significant computation duration which is not influenced from DataWarp.

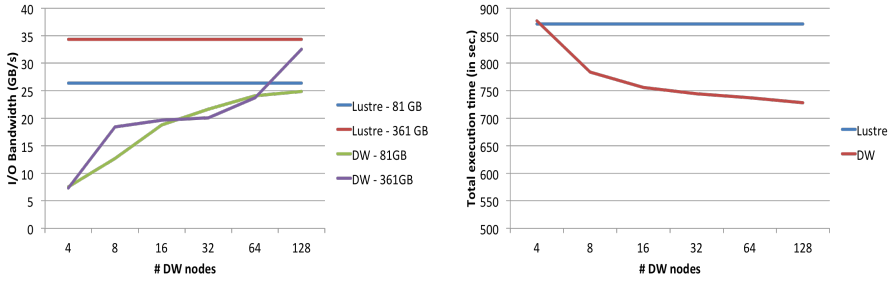

Fig. 1. DataWarp vs Lustre on WRF I/O performance

4.3 NGA

This case study is benchmarking a code on a turbulence partially premixed flames at high Reynolds number. The NGA code [11] can perform large-eddy simulation (LES) and direct numerical simulation (DNS). The benchmark used is this study is the Bunsen flame at a Reynolds number Re = 11,200 using 2.8 Billion grid points requiring around 25 runs of 24hours using 1024 nodes with 32,768 cores. Typically, in production mode, the checkpointing is performed several times daily, generating each time a solution file of approximately 560 GB. Fig. 2 shows the results using different DataWarp nodes, Burst Buffer demonstrated with up to 3.75 times performance improvement with 32 nodes when compared to Lustre. Overall, this achieves up to 25% decrease of the complete simulation if regular checkpointing are performed. According to our experience, DataWarp requires a significant amount of data to stress the SSDs, more DW nodes we use, more data we need for the cases that the I/O is not optimized for such technologies.

4.4 ATLIB

Natural migration [12] computes an image according to the equation

$$Image(x) = \sum_{s}^{N} \sum_{r}^{N} [G(s,x,t) * G(r,x,t)] \cdot G_1(s,r,t),$$

where s and r, respectively, denote seismic data source and receiver coordinates, and x denotes image coordinates. The Green's functions $G(s,x,t)$, and $G(r,x,t)$, and the scattered data $G_1(s,r,t)$ are precomputed and stored in a single file with more than 86GB of size. Within the file, there are N=5297 Greens functions; each is a 3D volume in time and 2D space. The implementation of natural migration

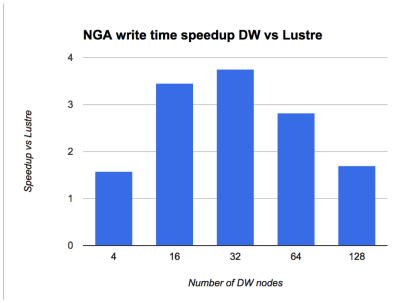

Fig. 2. Write time speedup on Burst Buffer compared to Lustre using different DataWarp Nodes

is a hybrid MPI/OpenMP code distributed such that each MPI process runs on a socket with 16 OMP threads. The algorithm does N^2 data accesses to the Green's functions for computing the image. The time convolution ($*$) and dot-product ($\cdot$) operations in the equation above are computationally cheap compared to the IO cost for retrieving the Green's function off disks. Therefore, the performance of the natural migration algorithm is I/O bound. Optimization of such algorithm is necessary considering that the equation above is applied repeatedly for different geophysical parameters.

An equal amount of work is given to each MPI process in the scalability aspect of this performance analysis. Our experiments show that the performance of the code gets worse when the number of MPI processes/Lustre clients is increased. This performance degradation, however, dramatically improves by increasing the Lustre stripe count. As an example, the execution time on 100 nodes is around 452 seconds versus 930 seconds while using 250 nodes with a stripe count of 4. By increasing the stripe count to 10, the execution time on 250 nodes is significantly reduced to 505 seconds. Fig. 3 on the left shows the same kind of performance analysis but using DataWarp Burst Buffer. The number of DataWarp nodes is the equivalent of stripe count in Lustre. Similarly, we notice that the performance on DataWarp improves while increasing the number of DataWarp nodes up to 40 nodes. However, increasing it to 100 just hits the performance significantly with any of the node counts used in this study. This could be explained by the difference of the underlying file system putting together the pool of allocated SSDs and its limited scalability in comparison to Lustre parallel file system in its current version. Nonetheless, DataWarp Burst Buffer technology provides good performance benefits of up to 34% for small to medium size runs (50, 100 and 250 nodes) as shown in Fig. 3, right side. Performance degradation is noticed on higher node counts (500 and 1000), and gets worse and worse as the number of I/O clients is increased. While we scale to 2649 compute nodes (to read all the sources with one execution) and 144 I/O clients, we could achieve similar performance to 50 compute nodes, which is quite efficient.

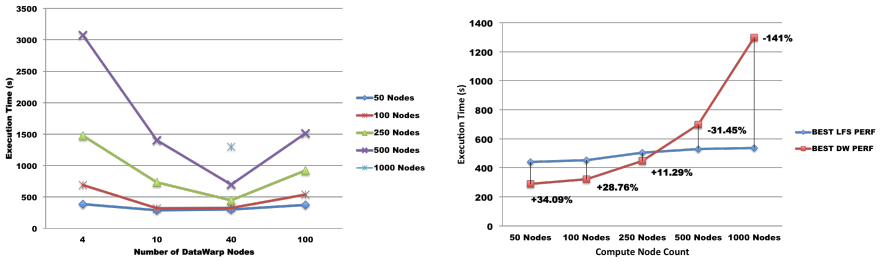

Fig. 3. Execution time per iteration of ATLIB on Burst Buffer and comparison with Lustre

4.5 NAS Parallel Benchmarks Block Triagonal I/O

The NAS Parallel Benchmarks (NPB) [13] are constituted by a set of applications/benchmarks with a scope to evaluate the performance of parallel supercomputers. In this study, we use the NAS Parallel Benchmarks Block-Tridiagonal (BT) I/O [14] to evaluate the I/O bandwidth on the available DataWarp installation. BT I/O presents a block-tridiagonal partitioning pattern on a three-dimensional array across a square number of processes. Each process handles multiple Cartesian subsets of the entire data set, and they increase with the square root of the number of processes participating in the computation. Multiple global arrays are consecutively written to a shared file by appending one after another. The number of global arrays can be adjusted, more information is provided in [15] For our experiments, we compile NPB v3.3.1 with Cray-MPICH v7.4.2, Cray compiler v8.5.2, and Parallel-NetCDF 1.7.0. We are interested in studying the I/O performance of various file formats and one of them is Parallel NetCDF (PnetCDF). Thus we chose an implementation of BT I/O which employs this format [16]. For the experiments that take place on Lustre, we always use the maximum amount of OSTs, which is 144. In Fig. 4, we compare results from Lustre and DataWarp. We have two categories of experiments for Lustre; one is with 128 compute nodes (using 32 cores per node) because this is also the maximum number of compute nodes that are used for the DataWarp experiments. The second one is with 512 compute nodes because we can achieve the peak performance on Lustre, and increasing the number of nodes does not provide any gain of the performance. For DataWarp we achieve maximum performance with 128 compute nodes and while we scale, we increase the problem size. We start with 400GB of PnetCDF file with 2 DataWarp nodes. After numerous experiments we found a sweet spot at 51TB i.e. 256 DataWarp nodes. From the results in Fig. 4, left side, we observe that DataWarp is at least three times faster than Lustre, achieving close to 90 GB/s, which corresponds almost to the ratio of IOR performance between DataWarp and Lustre. We achieved the maximum performance by applying 8 MPI aggregators per DataWarp node.

However, it is important to know if the I/O bandwidth is efficient as compared to the optimal performance. From IOR results we know that the maximum I/O bandwidth per DataWarp nodes for write, is around 5.7 GB/s. Thus we can

extrapolate and calculate the I/O bandwidth efficiency to obtain the results of Fig. 4, right side. The I/O efficiency starts a bit over 50% with 2 DataWarp nodes and falls under 10% with 256 DataWarp nodes which mean that the I/O bandwidth is not scalable in this case. Thus, with this benchmark, DataWarp is three times better than Lustre, but the I/O pattern does not utilize optimally the DataWarp. As we use 128 compute node, and we have 8 MPI I/O aggregators per DataWarp node, with 16 DataWarp nodes, we have totally 128 MPI I/O aggregators, which means one per compute node. Increasing further the DataWarp nodes, we have more than one MPI I/O aggregator per node. Thus, there is some network contention and the performance is decreased.

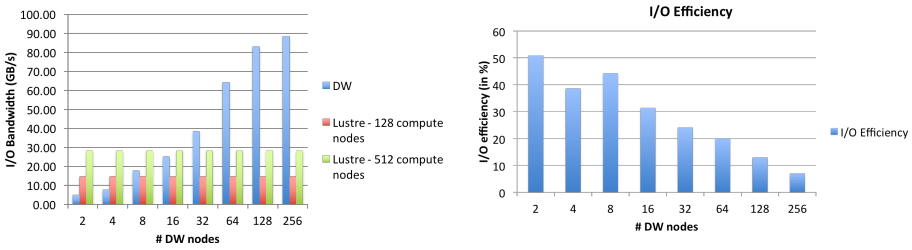

Fig. 4. NAS BT I/O on DW

From Darshan data, Fig. 5, it is clear that while we increase the size of the output file from 400 GB (left side) to 3.2 TB (right side), there are more write calls and so more seek operations, while more than 85% of the total execution time, is I/O. This hurts both filesystems, but DW is constituted by SSDs drives which are faster as also the increase of the MPI I/O aggregators can improve the performance according to [1].

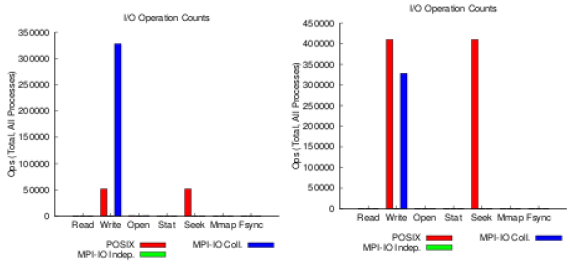

Fig. 5. Comparison of DataWarp and Lustre for NAS BT IO

4.6 Parallel IDX Benchmark

The IDX format provides efficient, cache oblivious, and progressive access to large-scale scientific data by storing the data in a hierarchical Z (HZ) order [17].

The HZ order is calculated for each data sample using the spatial coordinates of that sample. In order to study large datasets a parallel version of IDX, called PIDX [18], was developed. To achieve high scalability with PIDX, the developers of the framework did implement the total I/O procedure in three phases. Initially, we have the restructuring; blocks of data are created to optimize the layout for I/O. In the continuation, we have the in-core reorganization of data in a read-friendly format following by the data aggregation to optimize disk access. During data restructuring, there is high utilization of the network between the participated processes, and only a subset of them have the required data to participate in next phase. Then, HZ encoding is applied locally on all processes of phase 1. Afterward, the data aggregation occurs, and the data are written to many IDX files. The data aggregation is constituted by steps, in which the first one, data are gathered to aggregators using one-sided MPI communication, and then each aggregator writes its IDX file. This method combines an aggregation strategy that the final phase does not create contention because it creates multiple files. In this work, we use a PIDX tutorial that the developers include in the distribution and they call it *checkpoint_simple*. It reproduces the I/O in the case that we integrate PIDX in a real application. In order to produce an average I/O workload per MPI process, that could correspond to a real application, we declare in our experiment that each MPI process handles 64MB of data, so all together the 32 cores, are saving 2GB of data in a file. Moreover, the 64 MB per process are constituted by 32 variables. In Fig. 6 we present the results using 144 OSTs on Lustre and 16 to 256 DataWarp nodes.

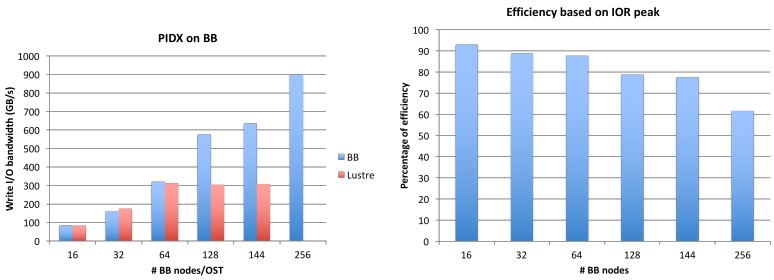

Fig. 6. Comparison of DataWarp and Lustre for PIDX

Moreover, we use 256 compute nodes for 16 DataWarp nodes, till 1024 compute nodes for 256 DataWarp nodes. For 256 compute nodes, we save 512 GB files, and the global domain is 20x8 x 2048 x 512, while for 1024 compute nodes, we save 2TB files, and the global domain is 2048 x 2048 x 2048. The requested size of I/O per MPI process remains 64 MB for all the experiments. Although, till 64 DataWarp nodes, the Burst Buffer and Lustre have similar write I/O performance, for more DataWarp nodes, Burst Buffer is scaling while Lustre does not. More accurate for 256 DataWarp nodes, Burst Buffer, achieves 900 GB/s which is three times faster than Lustre's peak, 300 GB/s. All the experiments

took place in non dedicated mode and peak performance can be influenced. The right part of Fig. 6 shows the PIDX I/O efficiency based on IOR peak results. Till 144 DataWarp nodes, the I/O efficiency is above 75%, and it drops a bit more than 60% for 256 DataWarp nodes. As we scale on the system, and with regard to the phase of PIDX that utilizes the network, if the system is busy the results can vary because of the Aries network on XC-40. Thus, we believe that some experiments could be better with dedicated mode and newer compute node Linux which will be available on this system in a few months. From the results we understand that PIDX is an efficient I/O library and it is quite scalable. Thus, it is evident that we need to adapt our I/O approach.

5 Conclusions and Future Work

With the continuous and fast growth of the computational power, the I/O in scientific applications becomes a more significant performance bottleneck. We have evaluated the impact of different file system configurations including the newly added I/O layer of Burst Buffer with various scientific applications. A performance improvement is noted in all case and a correlation between the best balance between the number of compute and DataWarp nodes was identified in some of these cases.

The modified NAS BT benchmark with PnetCDF obtained 3 times better performance than Lustre. The serialized I/O in the Fluent software is also evaluated, and a performance gain of 20% with Burst Buffer was observed. The NGA application had a significant I/O performance improvement, which led to 25% total execution time improvement. ATLIB was studied with various combinations of DataWarp and compute nodes and its performance improved by 34%. PIDX achieved 900 GB/s, which is a significant result since this a benchmark could provide a faster I/O solution for real applications by using its API. For future improvements, we consider modifying the open source codes to integrate PIDX library or any other efficient I/O library for DataWarp, such as LibHIO [19]. Moreover, we foresee the need for an auto-tuning tool for optimizing I/O operations on DataWap to alleviate the burden of tuning so many parameters from end users.

Acknowledgment

For computer time, this research used the resources of the Supercomputing Core Laboratory at King Abdullah University of Science & Technology (KAUST) in Thuwal, Saudi Arabia.

References

1. Markomanolis, G.S.: Getting started with the burst buffer, https://doi.org/10.6084/m9.figshare.4871738

2. Cray: XC-40, datawarp applications I/O accelerator http://www.cray.com/sites/default/files/resources/CrayXC40-DataWarp.pdf.
3. Bhimji, W., Bard, D., Romanus, M., Paul, D., Ovsyannikov, A., Friesen, B., Bryson, M., Correa, J., Lockwood, G.K., Tsulaia, V., Byna, S., Farrell, S., Gursoy, D., Daley, C., Beckner, V., Straalen, B.V., Trebotich, D., Tull, C., Weber, G., Wright, N.J., Antypas, K., Prabhat: Accelerating science with the Nersc burst buffer early user program. Cray User Group (2016)
4. Schenck, W., El Sayed, S., Foszczynski, M., Homberg, W., Pleiter, D. In: Early Evaluation of the "Infinite Memory Engine" Burst Buffer Solution. Springer International Publishing, Cham (2016) 604–615
5. IOR: test $http://www.csm.ornl.gov/essc/io/IOR - 2.10.1.ornl.13/USER\_GUIDE$.
6. Hadri, B., Kortas, S., Feki, S., Khurram, R.: Overview of the KAUST's Cray X40 system – Shaheen II. Cray User Group (2015)
7. ANSYS: Ansys® academic research, release 17.0, fluent, ansys, inc.
8. Ansys: External flow over a formula-1 race car (2016)
9. Skamarock, W.C., Klemp, J.B., Dudhia, J., Gill, D.O., Barker, D.M., Duda, M.G., Wang, X.Y.H.W., Powers, J.G.: A description of the advanced research wrf version 3. NCAR Tech. Note NCAR/TN-475+STR (2008)
10. Morton, D., Nudson, O., Stephenson, C.: Benchmarking and evaluation of the weather research and forecasting (WRF) model on the Cray XT5 (2009)
11. Desjardins, O., Blanquart, G., Balarac, G., Pitsch, H.: High order conservative finite difference scheme for variable density low mach number turbulent flows. J. Comput. Phys. **227**(15) (July 2008) 7125–7159
12. AlTheyab, A., Lin, F.C., Schuster, G.T.: Imaging near-surface heterogeneities by natural migration of backscattered surface waves. Geophysical Journal International **204** (February 2016) 1332–1341
13. Bailey, D.H., Barszcz, E., Barton, J.T., Browning, D.S., Carter, R.L., Dagum, L., Fatoohi, R.A., Frederickson, P.O., Lasinski, T.A., Schreiber, R.S., Simon, H.D., Venkatakrishnan, V., Weeratunga, S.K.: The NAS parallel benchmarks-summary and preliminary results. In: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing. Supercomputing '91, New York, NY, USA, ACM (1991) 158–165
14. Wong, P., d.W.: R.F.V: NAS parallel benchmarks I/O version 2.4 (2003)
15. keng Liao, W.: Design and evaluation of MPI file domain partitioning methods under extent-based file locking protocol. IEEE Trans. Parallel Distrib. Syst. **22**(2) (2011) 260–272
16. University, N.: Benchmarking MPI-I/O with PnetCDF on NAS parallel benchmark BT, http://cucis.ece.northwestern.edu/projects/pnetcdf/#benchmarks (2013) http://cucis.ece.northwestern.edu/projects/PnetCDF/#benchmarks.
17. Pascucci, V., Frank, R.J.: Global static indexing for real-time exploration of very large regular grids. In: Proceedings of the 2001 ACM/IEEE Conference on Supercomputing. SC '01, New York, NY, USA, ACM (2001) 2–2
18. Kumar, S., Vishwanath, V., Carns, P.H., Summa, B., Scorzelli, G., Pascucci, V., Ross, R.B., Chen, J., Kolla, H., Grout, R.W.: PIDX: Efficient parallel I/O for multi-resolution multi-dimensional scientific datasets. In: CLUSTER. (2011)
19. Security, L.A.N.: Libhio, a library intended for writing data to hierarchical data store systems (2016) https://github.com/hpc/libhio.