Benchmarking A High Performance Computing Heterogeneous Cluster

L. Carracciuolo, D. Bottalico, D. Michelino, G. Sabella, B. Spisso

January 9, 2025

Abstract

The paper describes the results of some benchmarking tests aimed to verify and validate all the solutions implemented during the deployment of a HPC heterogeneous resource acquired by the data center of the University of Naples "Federico II" thanks to the funds of the IBiSCo (Infrastructure for Big data and Scientific COmputing) Italian National Project. The first set of benchmarks evaluates how the network interconnection technologies affect the inter- and intra-node communications of GP-GPU workloads. The second set evaluates the performance of the Lustre parallel file system to ensure an efficient environment for data-intensive applications. The tests, especially those that analyze the lower level of the middleware (micro-benchmarks), seem to confirm the ability of the resource to guarantee the expected performance.

Keywords: Benchmarking High Performance Computing Heterogeneous Computing GP-GPU InfiniBand NVLink Lustre CUDA RDMA UCX MPI

1 Introduction

In the first half of the 1990s, Thomas Sterling and Donald Becker built a cluster of networked computers, called *Beowulf* [35], as an alternative to large supercomputers. At the time, their idea of providing "*Commodity Off The Shelf (COTS)*" based systems has been a great success. This idea is still valid and can inspire the realization of HPC computing systems, whose computational power is far from that of the most powerful computers in the world, but whose architecture is already compliant to incoming exascale era systems (e.g., see The Exascale Computing Project (ECP) of U.S. Department of Energy [32]). Most likely, these systems will respond to the following description: multi-node systems, connected by high performance networks, where each node will have a high level of internal parallelism which will be also made available by technologies such as NVIDIA®and Intel®Xe GPUs.

In such context, the data center of the University of Naples "Federico II" acquired, thanks to the IBiSCo (Infrastructure for Big data and Scientific COmputing) project funds [28], a heterogeneous computational resource[2]. The use of heterogeneous features aims to ensure the best use of resources for different scenarios applications, such as distributed memory computing, GP-GPU (General-Purpose computing on Graphics Processing Units) accelerated workloads and their combinations (e.g., see [3, 4, 6, 7, 10]).

In the context of High Performance Computing, it is a common practice to evaluate performance (in terms of speedup, throughput, I/O speed, etc.) as a response to the HPC workload [17]. For this purpose, there are different suites of benchmarks, among the main ones:

• The Standard Performance Evaluation Corporation (SPEC)[31] is a consortium whose goals are to provide the industry with performance measurement tools since 1994. The development of the benchmark suites includes tools to analyze all the components of computing systems: from processors to compilers, from interconnects to run-time libraries. In the context of HPC systems can be considered: the SPECmpi for evaluating MPI-parallel performance across a wide range of cluster and SMP hardware emphasizing the performance of the type of computer processor, the number of computer processors, the communication interconnect, and the shared file system. The SPEChpc provides a set of application benchmark suites using a comprehensive measure of real-world performance offering well-selected science and engineering codes that are representative of HPC workloads.

• The HPC Challenge (HPCC) benchmark suite [15] was developed to provide a set of standardized hardware probes based on commonly occurring computational software kernels such as some parallel BLAS operations [1] as well as tools to analyze communications performance, attempting to span from high to low-level components of an HPC system.

• The CORAL Benchmarks: CORAL is a U.S. Department of Energy (DOE) project that will culminate in three ultra-high performance supercomputers at Lawrence Livermore, Oak Ridge, and Argonne national laboratories. In such context, a suite of benchmarks was developed to evaluate performances on supercomputers deployed during the project [8]. CORAL Benchmark categories represent DOE Workloads and among them should be considered: the Throughput Benchmarks representing full applications; The Skeleton Benchmarks investigating various platform characteristics including network performance, threading overheads, I/O, memory, system software, and programming models.

The benchmarks described above use one of three possible strategies: high-level, low-level, and hybrid. In the first case, the benchmarks evaluate performance by testing the application-level components; in the second case, they test low-level system functions. The strategy we use is *"hybrid"* also according to the approach described in [21, 23]: the tests evaluate the performance of the highest level components (macro benchmark tests), which can be considered tests from "the applications point of view"; down to the evaluation of the lowest level components (micro benchmark test).

[1] The BLAS (Basic Linear Algebra Subprograms) are routines that provide optimized standard building blocks for performing basic vector and matrix operations. Some vendors supply its optimized implementation of the BLAS.

Our work tests and analyzes all the IBiSCo cluster components. The first set of benchmarks evaluates how the network interconnection technologies affect the inter- and intra-node communications of GP-GPU workloads. The second set evaluates the performance of the Lustre parallel file system to ensure efficient access to data storage which is a critical issue for data-intensive applications. In Section 2 we describe the cluster architecture and its middleware layer which implements all the necessary software tools for communication and data storage services. Section 3 shows tests carried out to validate what is described in the previous section. In section 4 we discuss positive aspects, observed deficiencies, and suggestions on how to improve the obtained results. The conclusion (Section 5) will summarize the contents of the work.

2 The Architecture of the Hybrid High Performance Computing Cluster

The architecture of this cluster is depicted as a set of multiple layers (Figure 1). The highest layer of the architecture consists of the application layer. The lowest one consists of the hardware resources, which comprises 32 computing nodes and 4 storage nodes. In particular, it provides 1) 128 NVIDIA Volta GPUs and about 1600 physical cores (from Intel Gen 2 Xeon Gold CPUs) distributed on 32 nodes whose connections are based on InfiniBand [18] and NVLink2 [12] technologies; 2) 320 TB distributed on 4 storage nodes connected to the computing nodes by an InfiniBand network. The top one is the application layer which is exposed to users. The efficient use of cluster technologies is made possible by a software layer interposed between the lowest and the highest levels, namely the middleware, which is based on a combination of the following technologies:

1. OpenFabrics Enterprise Distribution (OFED) [26] for drivers and libraries needed by the Mellanox InfiniBand network cards.
2. CUDA Toolkit [25] for drivers, libraries and, development environments, enables NVIDA GP-GPU.
3. *"MPI-CUDA aware"* [22] implementation of OpenMPI [27] through the UCX open-source framework [29].
4. Lustre [33] - a distributed, parallel and open source file system - provides high performance access to storage resources.

Bandwidth and latency in message exchange among processes is one of the issues preventing the full exploitation of GP-GPU potential. In this regard, NVIDIA introduced CUDA Inter-Process Copy (IPC) [19] and GPUDirect Remote Direct Memory Access (RDMA) [14] technologies for intra- and inter-node GPU process communications to make this solution available for InfiniBand-based clusters. To optimize inter-node GPU-to-GPU communications for small messages, NVIDIA offers NVIDIA gdrcopy [30]. To combine these technologies

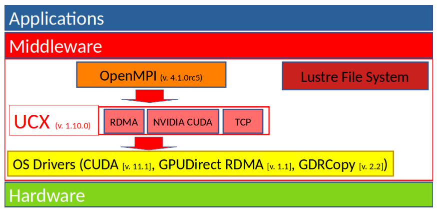


Figure 1: The Layered Cluster Architecture

with communication libraries (i.e., OpenMPI), we used the UCX open-source framework. UCX is a communication framework optimized for modern, high-bandwidth, low-latency networks. It exposes a set of abstract communication primitives that automatically choose the best available hardware resources. Supported technologies include RDMA (both InfiniBand and RoCE), TCP, GPU, shared memory, and atomic network operations.

As previously stated, a key aspect of high-performance computing is the efficient delivery of data to and from the computing nodes. The implementation adopted in the IBiSCo cluster is based on Lustre, a high-performance, parallel, and distributed file system. High-performance is guaranteed by Lustre flexibility in supporting multiple storage technologies, from the common ones based on Ethernet and TCP/IP to those with high-speed and low latency such as InfiniBand, RDMA and RoCE. Storage nodes host the OSTs[2] for the two Lustre exposed file systems, one for user home directories and one for jobs scratch area. In particular, the home file system is characterized by large disk space needs and fault tolerance, therefore it is made up of RAID-5 SAS HDD array. On the other hand, the scratch area needs fast disk access times and no redundancy requirement, hence it is hosted on SATA SSD disks.

3 Cluster Benchmarking

We have created a set of micro- and macro-benchmarks to study communication and access to resources. As for the communication tests: the micro-benchmarks have highlighted some limitations, mainly because they work with either very small or very large problems. For both intra- and inter-node communication, although peak performance is achieved (50 and 10 GB / s respectively), the tests show sudden increases. Growth should be "softened" by decreasing intermediate peaks. The discontinuity is due to several factors: first of all to the technologies used, such as the GDRCopy, which works with small message sizes. The growth of the message size leads to an automatic deactivation of the technology. This feature is essential. Depending on the type of application that uses the resources, it may be more appropriate to use one configuration of the benchmarking algorithm than another. For this reason, the choice of a

[2]The Lustre Object Storage Targets (OST) are the block devices on which data is distributed

benchmark that keeps pace with current technologies is indispensable for the evaluation of the cluster. As for the macro-benchmarks, the real limitation is given by Linpack: all communications to and from GPU devices are obtained using the PCI Channel and not NVLink (since the implementation of these benchmarks does not provide for GPU-to-GPU), this implies that the cluster resources are not fully exploited. Summing up, benchmark results are provided which should be useful for:

1. filling the lack of deep understanding on how modern GP-GPU can be connected and the actual impact of "state-of-the-art" hardware/software technologies on multi-GPU application performance;

2. evaluating the usage of parallel file systems in applications with intensive parallel data access.

3.1 Communication and computation

3.1.1 Micro-benchmark tests

We evaluate the basic characteristics of the four GPU interconnections focusing on both MPI Peer-to-Peer (P2P) and MPI Collective (CL) GPU-TO-GPU communication patterns. Both for intra- and inter-node P2P, we pay special attention to assessing the communication technologies in terms of latency and bandwidth on message size. Eventually, we evaluate the latency of the collective communication patterns on both intra- and inter-node scenarios. The tool used for measuring latency and bandwidth is the CUDA-aware version of MPI OSU Micro-Benchmarks [5] which evaluates latency and bandwidth of P2P tests as follows:

Latency Test: the latency tests are performed in a ping-pong fashion, by using blocking versions of the MPI functions (`MPI_Send` and `MPI_Recv`). The sender sends a message with certain data size and waits for a reply. The recipient receives the message and returns a response with the same data size. Many iterations of this test are performed and average one-way latency numbers are obtained[3].

Bandwidth Test: Non-blocking versions of the MPI functions (`MPI_Isend` and `MPI_Irecv`) are used in this case. The sender sends a fixed number of consecutive messages to the recipient and waits for its reply. The recipient sends the reply only after all these messages are received. This process is repeated for several iterations and the bandwidth is calculated based on elapsed time (until the sender receives the reply from the recipient) and the number of bytes sent by the sender. The goal of this bandwidth test is to determine the maximum sustained data rate which can be achieved at thethe network level.

[3]We used the default number of iterations that the benchmark provides: 1000 iterations for small messages and 100 iterations for large messages.

Conversely, the latency of collective communications is measured via the following procedure: fixing a message size, many calls of `MPI_BCast`, `MPI_Gather`, `MPI_Reduce` (with `MPI_SUM` operation type) functions are carried out to compute time spent in a single call. All those time values are averaged to compute the latency number of the **Broadcast**, **Gather**, and **Reduce** tests respectively for each considered message size.

All the tests described above are carried out to evaluate the performance of both intra- and inter-node communications of the cluster where different combinations of RDMA, IPC, and gdrcopy are used as summarized in the description of Figure 2. Plots of trends (as a function of message size) for P2P intra- and inter-node communications are respectively reported in Figure 2-(a) and 2-(b). As a term of comparison, Figure 2-(c) shows the behavior of P2P Host-to-Host communications. In all latency plots, we show, as an error bar, the value of σ where σ^2 is the *Sample Variance* [4] of the measured latency times used for each mean computation: just in very few cases the variance appears significant).

In Figure 3-(a) and 3-(b) are respectively reported plots (as a function of message size) for collective intra- and inter-node communications where different combinations of RDMA, IPC, and gdrcopy are used. During the tests, in the case of intra-node collective communications, all the tasks are spawned on a single node. Conversely, when inter-node collective communication is considered one task is spawned on a single node. Tests are performed with different task numbers P. Lines in the plots representing tests executed on $P = 2, 3, 4$ are marked respectively with ■, ◆ and ▼ symbols.

All plots use a logarithmic scale with base 2 and 10 respectively for the x and y coordinate axis. From Figures 2 and 3, we can state the following:

- Significant differences can be found between the performance of intra- and inter-node P2P communications. The intra-node communication seems to reach the maximum bandwidth performance of $50GB/s$, guaranteed by the NVLink technology, already with medium-sized messages. The same behavior cannot be witnessed during inter-node communication since the performance (about $10GB/s$) is comparable to the peak performance of the InfiniBand technology achieved only transmitting large-sized messages.

- The use of gdrcopy technology (see blue and green lines of all the plots in Figure 2-(a) and 2-(b)) significantly improves the performance of P2P communications with small messages. A combination of gdrcopy and GPUDirect RDMA technologies seems to be the best choice to improve performance in all the tested configurations: it is more noticeable in P2P inter-node communications (see green lines of all the plots in Figure 2-(a) and 2-(b)) although the best performance for large messages is obtained using the GPUDirect RDMA without gdrcopy (see green lines of all the plots in Figure 2-(b)).

[4]The following formula is used to compute the *Sample Variance* σ^2 of a set of n values $\{x_i\}_{i=1,\dots,n}$ whose mean value is $\bar{x}$: $\sigma^2 = \frac{\sum_{i=1}^{n-1}(x_i - \bar{x})^2}{n-1}$.

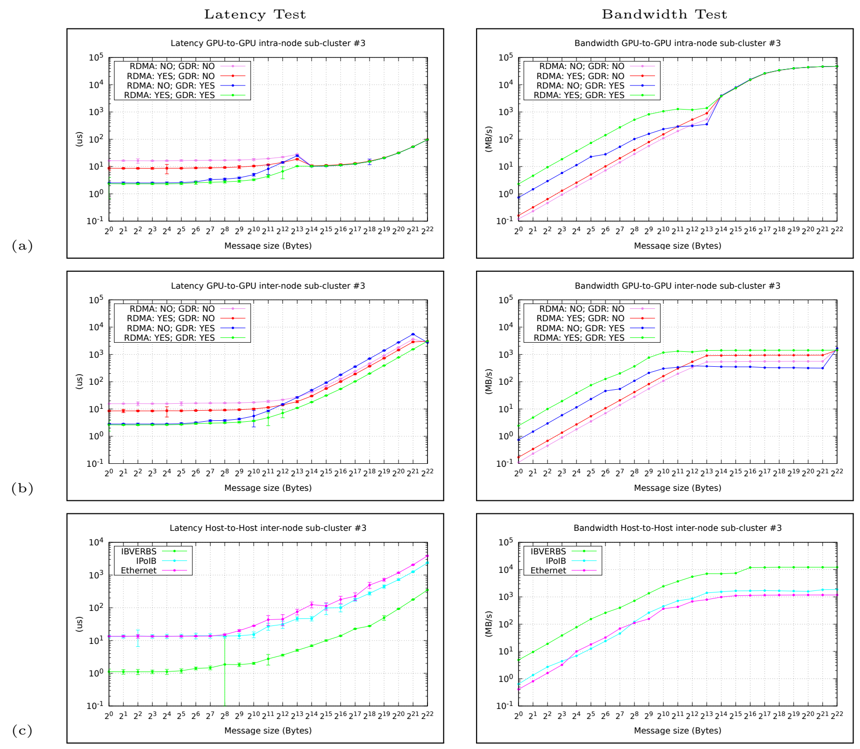


Figure 2: Communication and computation micro-benchmarks results. Latency and bandwidth of P2P GPU-TO-GPU intra-node (a) and inter-node (b) communication and of Host-to-Host (c) communication on the considered sub-clusters

- All the configurations tested show equivalent performance when P2P intra-node communication uses large messages (see Figure 2-(a)).
- The sustainable performance values for GPU-TO-GPU inter-node communications seem to be, in most cases, about a tenth of the value measured for Host-to-Host communications, which reach the InfiniBand peak performance (see Figure 2-(c)).
- No particularly significant changes can be observed in the Collective Reduce test if different combinations of RDMA, IPC, and gdrcopy are used. These differences seem more noticeable in inter-node communications (see Figure 3-(b))
- In the other Collective Tests certain differences, can only be found for small message sizes when different combinations of RDMA, IPC, and gdrcopy are used.

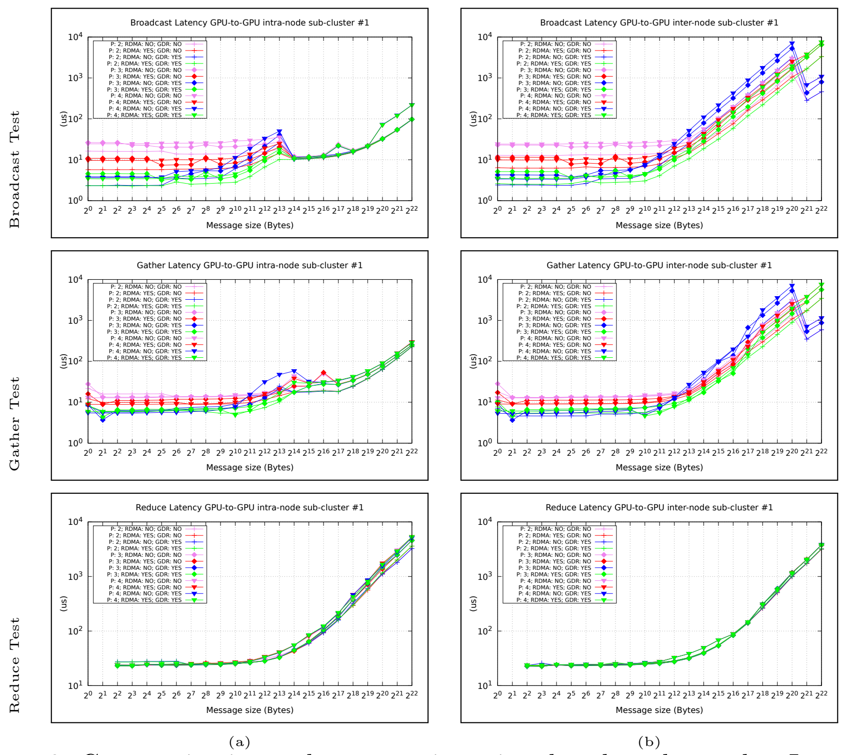


Figure 3: Communication and computation micro-benchmarks results. Latency of GPU-to-GPU collective communications on the cluster: intra-node (a) and inter-node (b) communications

3.1.2 Macro-benchmark tests

To evaluate how the implemented multi-GPU heterogeneous computational resource responds to a typical parallel workload from Scientific Computing, the CUDA-Aware version of the High Performance Linpack (HPL) Benchmark is used. The HPL benchmark [1] is a software package that solves a (random) dense linear system in double precision arithmetic on distributed-memory architectures. The HPL package provides a timing program to quantify the time it took to compute it. The best performance evaluation, in terms of thethe number of floating operations per second, is currently used to compile the list of the most powerful computers in the world [34]. The CUDA-Aware HPL benchmark [11] uses CUDA libraries to accelerate the HPL benchmark on heterogeneous clusters, where both CPUs and GPUs are used with minor or no modifications to the source code of HPL. A host library intercepts the calls to BLAS *DGEMM* and *DTRSM* procedures and executes them simultaneously on both GPUs and CPU cores. However, the benchmark has a limit: all communications to and from GPU devices are performed using the PCI channel.

In Figure 4 we show the results of the CUDA-Aware HPL benchmark executed on some nodes of the IBiSCo cluster: the number of total MPI tasks is $4P$ where P is the number of involved nodes. The tests are performed using different values for the problem dimension N. The graphs show:

$T(P,N)$: The execution time of the benchmark as a function of the number P of nodes for some values of N;

$S(P,N)$: The Speed-Up of the execution as a function of the number P of nodes for some values of N. So, $S(P,N) = \frac{T(1,N)}{T(P,N)}$;

$SP(P,N)$: The Sustained Performance (expressed in GigaFLOPS) is obtained during the execution as a function of the problem dimension N for some values of P. It represents the number of Floating Point operations executable by an algorithm in a time range;

$SPF(P,N)$: The fraction of Peak Performance is obtained during the execution as a function of the problem dimension N for some values of P. So, $SPF(P,N) = \frac{SP(P,N)}{PP(P)}$ where $PP(P)$ is the Peak Performance of P nodes when for each node all four GPU devices are considered[5].

From the plots in Figure 4 we can observe:

- the super linear speedup which is most remarkable for large problems. We think this is due to the increased time spent on CPU-GPU communications mainly as a consequence of a saturated PCI channel (indeed that all the four GPUs of a node are involved in computations);
- the very low scalability of the benchmark as the number of parallel tasks increase;
- the very small fraction of the Peak Performance scored during executions: if we consider very large problems we get just under 10% of max computational power which can be guaranteed by the computational resources.

3.2 Communication and data storage

3.2.1 Micro-benchmark tests

We evaluate the basic characteristics of the implemented Lustre file systems using the IOzone File system Benchmark [20], which generates and measures the time to complete a set of file operations as read, write, re-read, re-write. In Figure 5 we show the throughput performance for the same above-mentioned operations both with and without the `SYNC` IOZone option[6]. The plots show single stream performance as a *"Heat Map"* of file size and request size for two Lustre-based file systems which are an aggregation of SAS HDDs and SATA SSDs respectively both available on storage nodes. In the same plots, we show,

[5]Let $PP(P) = (4NCores_{GPU}Clock_{GPU} + NCores_{CPU}Clock_{CPU})\,P$.

[6]When this option is activated, IOZone will open the files with the `O_SYNC` flag. This forces all writes to the file to go completely to disk before returning to the benchmark.

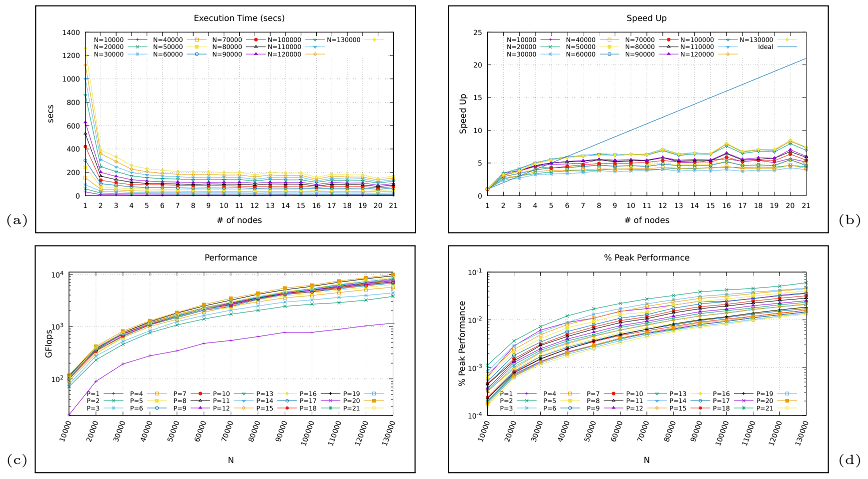


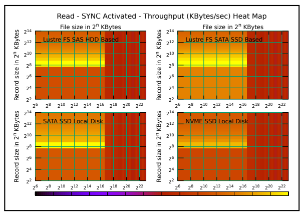
Figure 4: Communication and computation macro-benchmarks results: The CUDA-Aware HPL benchmark Execution Time $T(P)$ (a), Speed-Up $S(P)$ (b), the Sustained Performance $SP(P,N)$ (c) and the fraction of Peak Performance $SPF(P,N)$ (d).

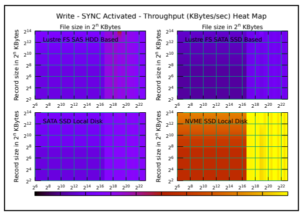
as a term of comparison, the results of the same test performed using two XFS file systems configured on different types of local disks (SATA SSD and PCIe NVMe SSD) available on computing nodes. All plots use a logarithmic scale with base 2 for the x and y coordinate axes. From such plots, the following statements can be argued:

- on read operations, all the tested file systems show comparable performance and suffer from large file size;
- the Lustre file system seems to be especially performing on write operations when file size increases. This is more noticeable if the option `SYNC` is activated;
- on write operations, the performance of Lustre file systems seems to be comparable (in terms of order of magnitude) with results obtained on slow local disks (especially if the option `SYNC` is disabled);

3.2.2 Macro-benchmark tests

We use a benchmark based on the Block-Tridiagonal (BT) problem of the NAS Parallel Benchmarks (NPB)[24], which is employed to test the I/O capabilities of high-performance computing systems, especially parallel systems. As improvements were made to parallel systems, the speed with which computed results are being written to and read from files still represents a bottleneck in practical applications. The benchmark, named BT-IO, is based on the MPI I/O Application Programmer Interface [9] which is part of the MPI. In Figure 6 we report the results of th BT-IO benchmark in its *"simple"* configuration where data, scattered in memory across the processors, are written to the same


(a)


(b)

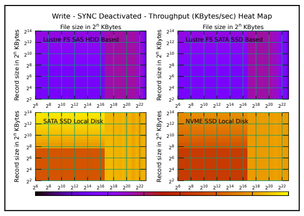
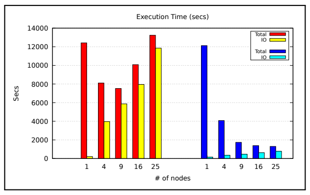
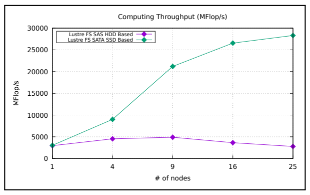

(c)

Figure 5: Communication and storage micro-benchmarks results: IOZone throughput performance (in KB/s) for read (a) and write operations with (b) and without (c) the SYNC options. For better readability, we preferred to use a different color map in each plot.


(a)


(b)

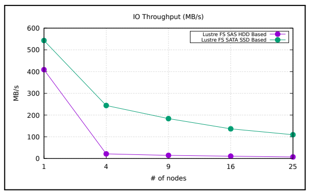

(c)

Figure 6: Communication and storage macro-benchmarks results. BT-IO results: the total time of execution versus the time spent during IO phases (a), the throughput of computing (b) and IO (c) stages expressed in MFlops/sec and MB/s respectively

file. What is considered here is the class *"E"* problem dimension. During execution, one MPI task is allocated to each node, and both the Lustre file systems described above are considered. From such plots we can argue:

- time spent during the IO stages might account for a significant portion ($> 50\%$) of total execution time when the number of parallel tasks is large;
- the write pattern used by the tests, where each processor writes the data elements it is responsible for directly into an output file, confirms the weak performance due to a very high degree of fragmentation [36]. The Lustre file system based on SSD disks better manages the such type of pattern also when the number of processors becomes large;
- IO throughput seems far from the values measured by micro-benchmarks which appear to be about a bigger order of magnitude.

4 Discussion on the results

The tests that analyze the lower level of the middleware (micro-benchmarks), seem to confirm the ability of the resource to guarantee the expected performance.

All the macro-benchmarks confirm that the goal of achieving the maximum performance of IT systems is extremely demanding. Although useful for evaluating the cluster created and highlighting the strengths of its resources, the benchmarks are also intended to bring out any issues. In this case, the problems in some of the results shown depend largely on the chosen benchmarks. In fact, for the most part, they cannot fully keep up with new technologies.

Our future work will be to find (or create) a version of macro benchmarks that can to make the most of the heterogeneity of the systems with solutions that: 1) use both the CPU and GPU present on the individual nodes, 2) exploit all the most performing communications channels available, 3) by CUDA-Aware messages passing library and innovative tools such as the Software for Linear Algebra Targeting Exascale (SLATE) library [13] or innovative approach as the HPL-AI Mixed-Precision Benchmark [16].

5 Conclusion

The paper describes the results of some benchmarking tests aimed to verify and validate all the solutions implemented during the deployment of a computing cluster within the Italian National Project IBiSCo able to satisfy the different computing needs of the project partners. All the strategies implemented have been verified and evaluated by the appropriate tools used to estimate some significant performance indexes of all the components of the system from a micro and macro point of view. From the communication between nodes with multiple GP-GPU in a distributed memory environment to the efficiency of the application during the IO phases.

Acknowledgment

This work has been funded by project code PIR01_00011 “IBISCo”, PON 2014-2020, for all three entities (INFN, UNINA, and CNR).

References

[1] A. Petitet et al.: A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers. `https://www.netlib.org/benchmark/hpl/index.html`

[2] Barone, G. B. et al.: Designing And Implementing A High-Performance Computing Heterogeneous Cluster. In: 2022 International Conference on Electrical, Computer and Energy Technologies (ICECET). pp. 1–6 (2022). https://doi.org/10.1109/ICECET55527.2022.9872709

[3] Bertero, M. et al.: MedIGrid: a medical imaging application for computational Grids. In: Proceedings International Parallel and Distributed Processing Symposium. pp. 8 pp.– (2003). https://doi.org/10.1109/IPDPS.2003.1213457

[4] Boccia, V. et al.: HADAB: Enabling Fault Tolerance in Parallel Applications Running in Distributed Environments. In: Lecture Notes in Computer Science, LNCS vol 7203 part 1. pp. 700–709 (2012). https://doi.org/10.1007/978-3-642-31464-3_71

[5] Bureddy, D. et al.: OMB-GPU: A Micro-Benchmark Suite for Evaluating MPI Libraries on GPU Clusters. In: Recent Advances in the Message Passing Interface. pp. 110–120 (2012). https://doi.org/10.1007/978-3-642-33518-1_16

[6] Carracciuolo, L. et. al: Implementation of a non-linear solver on heterogeneous architectures. Concurrency and Computation: Practice and Experience **30**(24), e4903 (2018). https://doi.org/10.1002/cpe.4903

[7] Carracciuolo, L. et al.: About the granularity portability of block-based Krylov methods in heterogeneous computing environments. Concurrency and Computation: Practice and Experience **33**(4), e6008 (2021). https://doi.org/10.1002/cpe.6008

[8] CORAL procurement benchmarks. `https://asc.llnl.gov/sites/asc/files/2020-06/CORALBenchmarksProcedure-v26.pdf`

[9] Corbett, P. et al.: Overview of the MPI-IO Parallel I/O Interface, pp. 127–146. Springer US, Boston, MA (1996). https://doi.org/10.1007/978-1-4613-1401-1_5

[10] D'Amore, L. et alet al..: A Scalable Space-Time Domain Decomposition Approach for Solving Large Scale Nonlinear Regularized Inverse Ill Posed Problems in 4D Variational Data Assimilation. J. Sci. Comput. **91**(2) (2022). https://doi.org/10.1007/s10915-022-01826-7

[11] Fatica, M.: Accelerating Linpack with CUDA on Heterogenous Clusters. In: 2nd Workshop on General Purpose Processing on Graphics Processing Units. p. 46–51. GPGPU-2, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1513895.1513901

[12] Foley, Denis et al.: Ultra-Performance Pascal GPU and NVLink Interconnect. IEEE Micro **37**(2), 7–17 (2017). https://doi.org/10.1109/MM.2017.37

[13] Gates, Mark et al.: SLATE: Design of a Modern Distributed and Accelerated Linear Algebra Library. In: International Conference for High Performance Computing, Networking, Storage and Analysis (2019). https://doi.org/10.1145/3295500.3356223

[14] GPUDirect RDMA - CUDA Toolkit DOC. `https://docs.nvidia.com/cuda/gpudirect-rdma/index.html`

[15] HPC Challenge Benchmark. `https://hpcchallenge.org/hpcc/`

[16] HPL-AI Mixed-Precision Benchmark. `https://hpl-mxp.org/`

[17] Ihde, Nina et al.: A Survey of Big Data, High Performance Computing, and Machine Learning Benchmarks. In: Performance Evaluation and Benchmarking. pp. 98–118 (2022). https://doi.org/10.1007/978-3-030-94437-7_7

[18] InfiniBand network standard. https://en.wikipedia.org/wiki/InfiniBand

[19] Interprocess Communication - Programming Guide :: CUDA Toolkit DOC. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#interprocess-communication

[20] IOzone Filesystem Benchmark. https://www.iozone.org/

[21] Khorassani, Kawthar Shafie et al.: Performance Evaluation of MPI Libraries on GPU-Enabled OpenPOWER Architectures: Early Experiences. In: High Performance Computing. pp. 361–378 (2019). https://doi.org//10.1007/978-3-030-34356-9_28

[22] Kraus, J.: An Introduction to CUDA-Aware MPI. https://developer.nvidia.com/blog/introduction-cuda-aware-mpi/

[23] Li, Ang et al.: Tartan: Evaluating Modern GPU Interconnect via a Multi-GPU Benchmark Suite. In: 2018 IEEE International Symposium on Workload Characterization (IISWC). pp. 191–202 (2018). https://doi.org/10.1109/IISWC.2018.8573483

[24] NAS Parallel Benchmarks. https://www.nas.nasa.gov/software/npb.html

[25] Nickolls, John et al.: Scalable Parallel Programming with CUDA. Queue **6**(2), 40–53 (Mar 2008). https://doi.org/10.1145/1365490.1365500

[26] NVIDIA Mellanox OFED DOC. https://docs.mellanox.com/display/MLNXOFEDv531001/NVIDIA+MLNX_OFED+Documentation+Rev+5.3-1.0.0.1

[27] Open MPI: Open Source High Performance Computing. https://www.open-mpi.org/

[28] Programma Operativo Nazionale Ricerca e Innovazione 2014-2020: Progetto IBiSCo. https://www.na.infn.it/fondi-esterni/pon

[29] Shamis, Pavel et al.: UCX: An Open Source Framework for HPC Network APIs and Beyond. In: IEEE 23rd Annual Symposium on High-Performance Interconnects. pp. 40–43 (2015). https://doi.org/10.1109/HOTI.2015.13

[30] Shi, Rong et al.: Designing efficient small message transfer mechanism for inter-node MPI communication on InfiniBand GPU clusters. In: 21st International Conference on High Performance Computing (HiPC). pp. 1–10 (2014). https://doi.org/10.1109/HiPC.2014.7116873

[31] Standard Performance Evaluation Corporation. https://www.spec.org/

[32] The Exascale Computing Project Website. https://www.exascaleproject.org/

[33] The Lustre file system. https://www.lustre.org/

[34] The Top 500 list Website. https://www.top500.org/

[35] Thomas Sterling et al.: BEOWULF: A Parallel Workstation For Scientific Computation. In: 24th International Conference on Parallel Processing. pp. 11–14. CRC Press (1995)

[36] Wong, Parkson et al.: NAS Parallel Benchmarks I/O Version 2.4. NAS Technical Report NAS-03-002 (01 2003)