2021 IEEE International Conference on Cluster Computing (CLUSTER) | 978-1-7281-9666-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/Cluster48925.2021.00082

Toward a Comprehensive Benchmark Suite for Evaluating GASPI in HPC Environments

Sarah Neuwirth
Institute of Computer Science
Goethe-University Frankfurt, Germany
s.neuwirth@em.uni-frankfurt.de

***Abstract*—With the forthcoming age of exascale computing, the efficient support of different programming models has become a crucial performance factor for high-performance computing systems. When adopting novel programming paradigms, the performance assessment through standardized and comparable benchmarks plays an essential role in both the effective use of the heterogeneous system hardware and the application performance tuning. Alternatives to MPI such as the partitioned global address space (PGAS) model have become increasingly popular. One such PGAS API is the Global Address Space Programming Interface (GASPI). This paper introduces the GASPI Benchmark Suite (GBS), which combines a comprehensive, standardized set of microbenchmarks with application kernels. The microbenchmarks target common GASPI communication patterns, including one-sided, collective, passive, and global atomics, while the application kernels stress communication schemes commonly found in real HPC applications. The effectiveness of GBS is demonstrated by evaluating the GASPI communication performance for the two networking communication standards Ethernet and InfiniBand.**

***Index Terms*—PGAS, Programming Models, Performance Characterization, GASPI, Benchmark Suite, HPC**

I. Introduction and Motivation

For many years, the *Message Passing Interface* (MPI) has been considered the de-facto standard in the high-performance computing (HPC) community when it comes to developing parallel applications. One alternative to MPI is the *Global Address Space Programming Interface* (GASPI) [1], [2], which provides researchers and scientific application developers with one-sided RDMA-driven communication based on the *Partitioned Global Address Space* (PGAS) [3] parallel programming model. GASPI aims to initiate a paradigm shift from bulk-synchronous two-sided message exchange towards an asynchronous data-flow and execution model and also supports the segmentation of the memory in multiple parts, as shown in Figure 1. These memory segments can be defined differently on every node, thus GASPI does not enforce a symmetric memory model unlike for example OpenSHMEM [4].

When adopting novel programming interfaces such as GASPI, the performance assessment through standardized and comparable benchmarks plays an essential role in both the effective use of the heterogeneous system hardware and the application performance tuning. Comprehensive benchmark suites can be used to evaluate the potential performance for various communication patterns and different HPC compute environments. PGAS benchmark examples are the *OpenSHMEM Benchmark Suite* (OBS) [4], the *UPC Operations Microbenchmarking Suite* (UOMS) [5], and the OpenSHEM, UPC and UPC++ microbenchmarks included in the *OSU Micro-Benchmarks* (OMB) [6] suite. However to the best of the author's knowledge, their implementations and outputs all differ and do not follow a standardized approach, which makes it difficult to compare the performance of different PGAS-based programming languages and language extensions.

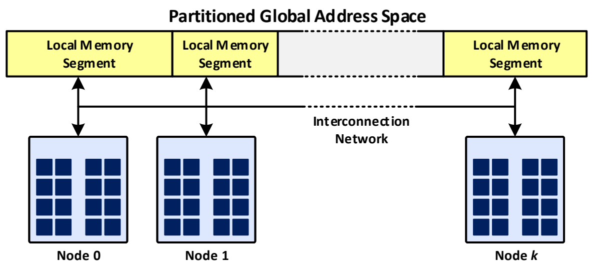


Fig. 1: Segmented partitioned global address space in GASPI.

In this work, the *GASPI Benchmark Suite* (GBS) is introduced, which provides a collection of microbenchmarks and application kernels for the GASPI API. Its design relies on a comprehensive approach, which will provide a basis for the development of a standardized PGAS benchmark methodology. GBS offers a useful mean of measuring performance and assessing the effects of new features as well as implementation strategies. The suite can also be used to assess the performance benefits of GASPI for different application codes.

II. GASPI Benchmark Suite

In the absence of real GASPI application code, microbenchmarks and application kernels are the only assessments available to compare and analyze the different features and available implementations. The *GASPI Benchmark Suite* (GBS) is one such collection to evaluate the GASPI standard and the GPI-2 [1] library, the first GASPI implementation.

A. Microbenchmarks

The GBS microbenchmarks are designed to perform a minimal test for each GASPI communication primitive. A total of 27 benchmarks have been implemented, including: *one- and two-sided PingPong*, *uni- and bidirectional PUT/GET*, *allreduce*, and global atomics such *fetch-and-add* and *compare-and-swap*. Each benchmark invokes a global barrier before

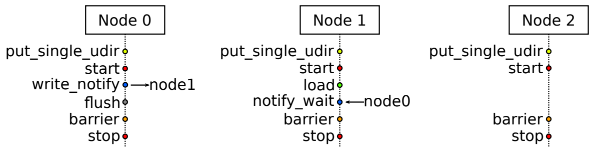


Fig. 2: Visualization of the true one-sided PUT benchmark.

the time measurement is launched. Another barrier is passed before the measurement is stopped. This design closely follows the IMB-MPI component of the IMB [7] as well as the point-to-point MPI and PGAS benchmarks included in the OMB. All of the tests report latencies (min/max/average), bandwidth when a function does data transfer, and the message rate. Since the intention is to strictly test the performance of GASPI rather than a correctness test, no data checks are performed.

GBS also features a set of *true one-sided exchange benchmarks*. The objective is to determine if a one-sided RMA operation really only involves the initiating side and does not affect the application itself. Figure 2 depicts the full sequence diagram of the true one-sided PUT benchmark.

B. Application Kernels

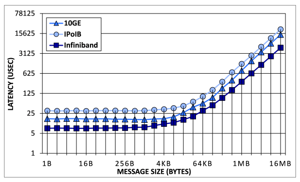
GBS is complemented by three application kernels, which adopt common communication patterns found in HPC applications. Their purpose is to evaluate the performance of programming models like GASPI for actual use cases.

The *Heat Distribution* benchmark is a stencil application code that computes the expansion of heat on a plane. The physical accuracy of the computation is less important — the emphasis of this benchmark is put on the halo exchange communication scheme, which is used to measure the message rate. The *SSCA1* benchmark [8] implements a Smith-Waterman local sequence alignment algorithm with Godah's improvements for gap scoring. Its value in benchmarking GASPI libraries is that it can be used to test strategies in managing many small messages with PUTs and GETs issued in the same inner loop. The *RandomAccess* benchmark is derived from the HPC Challenge Benchmark [9]. The Giga UPdates per Second (GUPS) are calculated by identifying the number of memory locations that can be randomly updated in one second, divided by 1 billion. The application kernels are also implemented in a multithreaded version with OpenMP.

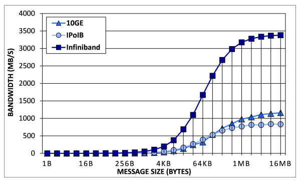
III. Performance Evaluation

GBS is evaluated on a small test bed. The test system consists of two nodes each equipped with an Intel Xeon Silver 4110 CPU, 32 GB DDR4 RAM, MPICH-3.2, and CentOS 7. GBS is evaluated with three different network configurations: 10 GBit/s Ethernet, IPoIB on InfiniBand QDR, and InfiniBand QDR. The GPI-2 library [10] implements the GASPI standard and supports native InfiniBand communication and standard socket communication via the TCP/IP stack.

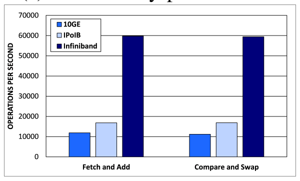
For brevity reasons, Figure 3 provides the performance results of selected benchmarks. Figures 3a, 3b and 3c present the results of the unidirectional PUT and GET benchmarks, and the performance of the global atomic operations for IPoIB, 10 GBit/s Ethernet, and InfiniBand QDR. In Figure 3d, GASPI is compared to the traditional MPI Send/Recv scheme and MPI one-sided communication using the InfiniBand QDR connection. The results validate the claim that GASPI is better suited for asynchronous communication; for large messages, GASPI yields similar performance results as MPI of approximately 3.2 GB/s of throughput performance.



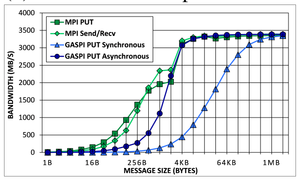
(a) PUT latency performance.



(b) GET bandwidth performance.



(c) Global atomics.



(d) PUT bandwidth comparison.

Fig. 3: Selected GBS microbenchmark results.

IV. Conclusion

This work introduces the GASPI Benchmark Suite. Through microbenchmarks and application kernels, various communication primitives and patterns can be stressed on different HPC systems. Future work will target the systematic comparison with other PGAS solutions such as OpenSHMEM and the development of a standardized benchmark design approach.

References

[1] D. Grünewald and C. Simmendinger, "The GASPI API specification and its implementation GPI 2.0," in *7th International Conference on PGAS Programming Models*, vol. 243, p. 52, 2013.
[2] GASPI Forum, "GASPI: Global Address Space Programming Interface. Specification of a PGAS API for communication. Version 17.1." https://www.gaspi.de/#specification. Online; accessed 14-July-2021.
[3] M. De Wael, S. Marr, B. De Fraine, T. Van Cutsem, and W. De Meuter, "Partitioned Global Address Space Languages," *ACM Comput. Surv.*, vol. 47, May 2015.
[4] T. Naughton, F. Aderholdt, M. Baker, S. Pophale, M. G. Venkata, and N. Imam, "Oak Ridge OpenSHMEM Benchmark Suite," in *Workshop on OpenSHMEM and Related Technologies*, pp. 202–216, Springer, 2018.
[5] D. A. Mallón, *Design of Scalable PGAS Collectives for NUMA and Manycore Systems*. PhD thesis, University of A Coruña, Spain, 2014.
[6] NOWLAB, "OSU Micro-Benchmarks (OMB)." https://mvapich.cse.ohio-state.edu/benchmarks/. Online; accessed 14-July-2021.
[7] Intel, "Intel MPI Benchmarks (IMB)." https://github.com/intel/mpi-benchmarks. Online; accessed 14-July-2021.
[8] M. Baker, A. Welch, and M. Gorentla Venkata, "Parallelizing the Smith-Waterman Algorithm Using OpenSHMEM and MPI-3 One-Sided Interfaces," in *OpenSHMEM and Related Technologies. Experiences, Implementations, and Technologies* (M. Gorentla Venkata, P. Shamis, N. Imam, and M. G. Lopez, eds.), pp. 178–191, Springer, 2015.
[9] J. Dongarra and P. Luszczek, "HPC challenge: Design, history, and implementation highlights," in *Contemporary High Performance Computing*, pp. 13–30, Chapman and Hall/CRC, 2017.
[10] Fraunhofer ITWM, "GPI-2: Global Address Programming Interface 2." https://github.com/cc-hpc-itwm/GPI-2. Online; accessed 14-July-2021.