vGASNet: A PGAS Communication Library Supporting Out-of-Core Processing

(Extended Abstract)

Ryo Matsumiya
Tokyo Institute of Technology
matsumiya.r.aa@m.titech.ac.jp

Toshio Endo
Tokyo Institute of Technology
endo@is.titech.ac.jp

1 INTRODUCTION

Partitioned Global Address Space (PGAS) is a programming model which can ease developing parallel applications. There are many programming runtimes adopting PGAS [11, 13, 14, 16]. Recently, out-of-core processing is required in many fields. Unfortunately, few PGAS frameworks support out-of-core processing.

ComEx-PM is a PGAS communication library which supports out-of-core processing [12]. Like TSUBAME-KFC/DL and Baecon, they assumed each computation node had its own SSD [3, 6]. Their library uses SSDs as main memories, and the DRAMs as cache. However, ComEx-PM caused significant performance degradation with out-of-core processing due to a large mount of SSD accesses.

We developed vGASNet, a novel PGAS communication library supporting out-of-core processing. To reduce the number of SSD accesses, vGASNet adopts a cache mechanism called cooperative caching [4]. Cooperative caching uses not only local DRAMs but also remote DRAMs.

We present cache mechanism of vGASNet. We also show performance evaluation of vGASNet. Thanks to cooperative caching, vGASNet gained maximum 15.3x throughput on 32 nodes. Our contributions are followed:

- We developed vGASNet, a PGAS communication library which provides PGAS framework with out-of-core supporting.
- We proposed a cooperative caching mechanism for vGASNet.
- We evaluated vGASNet on 32 nodes of TSUBAME-KFC/DL.

2 DESIGN AND IMPLEMENTATION

This section shows the design of vGASNet. The interface of vGASNet is based on GASNet [9]. vGASNet (and GASNet) consists of two kinds of functions, core API and extended API. Core API accommodates essential communication functions such as `gasnet_init()`. Core API also accommodates message passing interfaces based on Active Message [15]. On the other hand, extended API provides high-level operations for remote memory access (RMA) and collective communication.

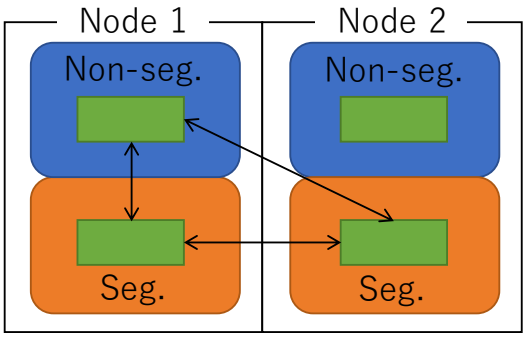

Figure 1: Segmented and non-segmented memory region of GASNet

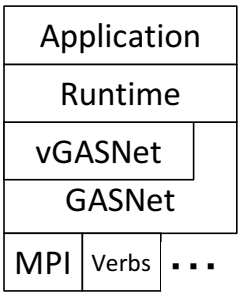

Figure 2: Software stack of vGASNet

Our focus is RMA functions such as `gasnet_put()` and `gasnet_get()`. During its initialization, GASNet allocates segmented memory region. As shown in fig.1, segmented memory region which one rank allocates can be accessed by other ranks via the RMA operations. Each node can access data (green boxes) in its non-segmented memory (blue boxes) and in all segmented memory (orange boxes). Although GASNet allocates the memory region on DRAM, vGASNet allocates on SSD attached to the computation node. In other words, vGASNet allows each node to access both local and remote SSDs via RMA operations.

The software stack of vGASNet is shown as fig.2. Generally, applications use GASNet and vGASNet via a PGAS runtime. vGASNet uses GASNet to communicate using interconnect networks such as Inifiniband and Omni-path. Some functions of vGASNet such as Active Message are implemented to use that of GASNet directly.

HPCAsia '18, January 28–31, Tokyo, Japan

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

3 CACHE MECHANISM

Even SSD-optimized protocol such as NVMe, access throughput of SSD is much slower than one of DRAM. For performance improvement, vGASNet uses DRAMs as SSDs' caches. The segmented memory region is divided into fixed-size pages. Each cache is assigned to one of the pages.

Using remote DRAMs as a cache, the amount of accesses to the SSDs can be reduced. Moreover, interconnect networks such as Infiniband FDR and Omni-Path are faster than SSDs. This means caches on the other nodes can improve the performance of vGASNet.

Therefore, vGASNet is introduced cooperative caching, which uses not only local cache but also remote caches. When a node is to access a segment memory region, the node tries to refer the cache lines assigned to the memory region. If the cache line is alive, the node use it. Otherwise, the node tries to refer the same cache lines in other nodes.

The cache-coherence protocol of vGASNet is called MOESIF. MOESIF protocol is combined from two practical protocols, MOESI and MOSIF.

MOESI protocol is used in 64-bits AMD multicore-processors [1]. MOESI protocol allows the dirty cache not to write-back to the SSDs when another node refers to the cache line.

MOSIF protocol is used in Intel multicore-processors [7]. Under MOSIF protocol, the node whose cache is used for cooperative caching is specified per cache line. We use a FIFO-based stack to specify the node. Like general FIFO-based data structures, an element is pushed on the top of the stack. When the stack is accessed, its top element is referred. Then, the top element is popped and re-pushed the bottom of the stack. In vGASNet, each cache line has its own stack. An element of the stack indicates the node which stores the cache. When a node stores a cache line, the element which indicates the node is pushed to the stack. When a node tries to access a cache line, the stack is accessed. The element which indicates a node is erased when the node evicts the cache.

The cache replacement policy of vGASNet is based on Least Recently Used (LRU). Although pure LRU uses only one LRU queue, our policy uses two queues. One queue is LRU queue, every cache is firstly pushed into this queue. When cache pool is filled, vGASNet selects from the bottom of the LRU queue. If the selected cache is not exclusive (i.e. its cache line is cached in another node), the cache is evicted. Otherwise, the cache is pushed to the other FIFO-based queue. And then, vGASNet selects the next bottom of the LRU queue. If the FIFO-based queue is larger than half of the cache pool, the cache at the bottom of the FIFO-based queue is evicted. An element of the FIFO-based queue is erased and re-pushed when and only when the assigned cache line is accessed.

4 PERFORMANCE EVALUATION

4.1 Environment

For performance evaluation, we use 32 nodes of TSUBAME-KFC/DL. Each node has an Intel DC S3500 SSD, 64 GB DRAM, an Intel Xeon E5-2620 v2 CPU. We used Infiniband 4X FDR as interconnection and GASNet v1.30.0. In our experiments, the cache line size is 4 MB. The cache pool size of each node is 16 GB.

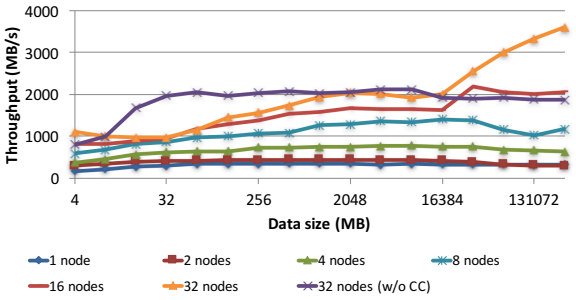

Figure 3: Sequential access throughput

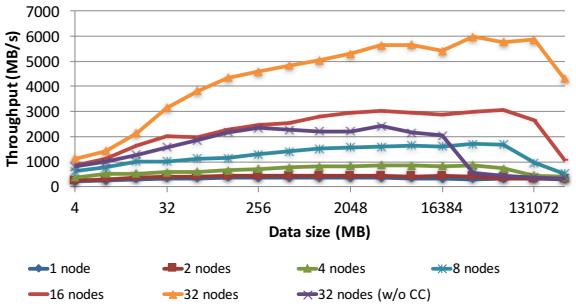

Figure 4: Random access throughput

To evaluate the efficiency of cooperative caching, we developed another implementation of vGASNet which does not use cooperative caching (indicates "w/o CC" in the graphs). This implementation uses only local caches and remote caches in the node which stores the original data in its SSD. To assume cache consistency, write-backing to the SSDs is occurred during memory barrier and false sharing.

4.2 Sequential access performance

We developed a sequential access program using vGASNet and GASNet. All nodes get one datum every 4 MB sequentially. The gotten datum is put on rank0 beforehand.

Fig.3 shows the performance of the program. When the gotten datum is larger than cache pool, vGASNet with cooperative caching mechanism achieves 1.92x faster on 32 nodes. This is because there are load imbalancing among the nodes. Without cooperative caching, the caches of head of the datum are evicted. Under our cooperative caching method, such caches are not evicted.

4.3 Random access performance

We also developed a random access program using vGASNet and GASNet. All nodes get one datum every 4 MB. The gotten datum is put on rank0 beforehand. Unlike the program ran in sec.4.2, the order of accessing block is randomized.

Fig.4 shows the result. Surprisingly, using vGASNet, random access is faster and more scalable than sequential access. The reason is that the caches are scattered across nodes. Because the access

patterns are different among the nodes, the loads for transferring caches are distributed.

5 RELATED WORK

ComEx-PM is a PGAS communication library supporting out-of-core processing [12]. However, cache mechanism of ComEx-PM depends on Linux kenrel VFS cache.

HHRT and Papyrus are runtimes to enable MPI programs to support out-of-core processing [5, 8]. Unlike vGASNet, their communication interfaces are not RMA-based.

Cooperative caching is originally proposed for a distributed file system [4]. Batsakis and Burns proposed write-enabled cooperative caching mechanism [2]. We introduced a write-enabled cooperative caching mechanism into a PGAS communication library.

6 CONCLUSION AND FUTURE WORK

We developed vGASNet, which enables PGAS runtimes to support out-of-core processing. vGASNet uses node-local SSDs as main memory and DRAMs as caches. The cache mechanism of vGASNet adopts cooperative caching technology, which uses remote caches. Thanks to this technology, vGASNet gains maximum 15.3x performance.

We plan to employ vGASNet in actual PGAS frameworks, and to measure application performances. Recently, GASNet-EX, a successor of GASNet, is being developed [10]. We also plan to make vGASNet compatible with GASNet-EX.

ACKNOWLEDGEMENT

This work is supported by JST-CREST.

REFERENCES

[1] Advanced Micro Devices. AMD64 Technology AMD64 Architecture Programmer ' s Manual Volume 2: System Programming. http://developer.amd.com/wordpress/media/2012/10/24593_APM_v21.pdf.

[2] Batsakis, A., and Burns, R. NFS-CD: Write-Enabled Cooperative Caching in NFS. *IEEE Transactions on Parallel and Distributed Systems 19*, 3 (2007), 323–333.

[3] Brook, R. G., Heinecke, A., Costa, A. B., Peltz, P., Betro, V. C., Baer, T., Bader, M., and Dubey, P. Beacon: Exploring the Deployment and Application of Intel Xeon Phi Coprocessors for Scientific Computing. *Computing in Science & Engineering 17*, 2 (2015), 65–72.

[4] Dahlin, M. D., Wang, R. Y., Anderson, T. E., and Patterson, D. A. Cooperative Caching: Using Remote Client Memory to Improve File System Performance. In *Proceedings of the 1st USENIX Conference on Operating System Design and Implementation (OSDI '94)* (1994).

[5] Endo, T. Realizing Out-of-Core Stencil Computations using Multi-Tier Memory Hierarchy on GPGPU Clusters. In *Proceedings of the IEEE International Conference on Cluster Computing 2016 (CLUSTER '16)* (2016).

[6] Endo, T., Nukada, A., and Matsuoka, S. TSUBAME-KFC: a Modern Liquid Submersion Cooling Prototype towards Exascale Becoming the Greenest Supercomputer in the World. In *Proceedings of the 2014 IEEE International Conference on Parallel and Distributed Systems (ICPADS '14)* (2014).

[7] Kanter, D. The Common System Interface: Intel ' s Future Interconnect. *Real World Tech*, 5 (2012).

[8] Kim, J., Sajjapongse, K., Lee, S., and Vetter, J. S. Design and Implementation of Papyrus: Parallel Aggregate Persistent Storage. In *Proceedings of the 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS '17)* (2017).

[9] Lawrence Berkeley National Laboratory. GASNet Communication System. https://gasnet.lbl.gov.

[10] Lawrence Berkeley National Laboratory. GASNet-EX Collaboration. https://sites.google.com/a/lbl.gov/gasnet-ex-collaboration/.

[11] Lee, J., and Sato, M. Implementation and Performance Evaluation of XcalableMP: A Parallel Programming Language for Distributed Memory Systems. In *Proceedings of the International Conference on Parallel Processing Workshops (ICPPW '10)* (2010).

[12] Matsumiya, R., and Endo, T. PGAS Communication Runtime for Extreme Large Data Computation. In *Proceedings of the 2nd International Workshop on Extreme Scale Programming Models and Middleware (ESPM2 '16)* (2016).

[13] Nieplocha, J., Harrison, R. J., and Littlefield, R. J. Global Arrays: A Portable " Shared-Memory " Programming Model for Distributed Memory Computers. In *Proceedings of the IEEE/ACM Conference on Supercomputing (SC '94)* (1994).

[14] UPC Consortium. UPC Specifications, v1.2. Tech. rep., Lawrence Berkeley National Lab, 2005.

[15] von Eicken, T., Culler, D. E., Goldstein, S. C., and Schauser, K. E. Active Messages: a Mechanism for Integrated Communication and Computation. In *Proceedings of the 19th Annual International Symposium on Computer Architecture (ISCA '92)* (1992).

[16] Zheng, Y., Kamil, A., Driscoll, M. B., Shan, H., and Yelick, K. UPC++: A PGAS extension for C++. In *Proceedings of the IEEE 28th International Parallel and Distributed Processing Symposium (IPDPS '14)* (2014).