xBGAS: An Address Space Extension for Scalable High Performance Computing

Xi Wang
Texas Tech University
Lubbock, Texas, 79409-3104
xi.wang@ttu.edu

John D. Leidel
Tactical Computing Laboratories
Muenster, Texas. 76252
jleidel@tactcomplabs.com

Brody Williams
Texas Tech University
Lubbock, Texas, 79409-3104
Brody.Williams@ttu.edu

Alan Ehret
Boston University
Boston, MA, 02215
ehretaj@bu.edu

Miguel Mark
Boston University
Boston, MA, 02215
mmark9@bu.edu

Michel Kinsy
Boston University
Boston, MA, 02215
mkinsy@bu.edu

Yong Chen
Texas Tech University
Lubbock, Texas, 79409-3104
yong.chen@ttu.edu

***Abstract*—Given the recent reemergence of extended memory interconnection technologies such as GenZ [1], CCIX [2] and OpenCAPI [3], architects in high performance computing (HPC) and high performance analytics (HPA) now seek to exploit these interconnection methodologies for use in extended or partitioned addressing models across device and system architectural domains. However, we have yet to see a commercial architecture with native extended addressing capabilities in the ISA garner wide adoption.**

Therefore, the Extended Base Global Address Space (xBGAS) extension is introduced to provide extended addressing capabilities for HPC beyond the core 64-bit addressing modes inherent in the base RISC-V RV64I ISA. The introduced extended memory models can be potentially applied to numerous domains such as HPC-PGAS, HPC-FLAT, MMAP-IO, CloudBSP, etc [4]. In this work, we focus on the xBGAS extension for use with the HPC-PGAS model.

The goal of the xBGAS in this work is to provide non-blocking, one-sided communication at the machine level. The xBGAS extension leverages standard memory operations to perform remote memory accesses and thus bypasses the kernel and avoids unnecessary blocking calls. Therefore, the performance of remote memory accesses is enhanced. Furthermore, implementations of the PGAS model, such as those based on MPI, RDMA, SHMEM, etc., typically require use of complicated multi-layer runtime libraries to realize remote communication. However, the machine-level communication support provided by xBGAS bypasses many of these redundant layers and dramatically reduces the overhead of function calls in runtime libraries while simultaneously simplifying the programming models.

Driven by the aforementioned target, we extend the base RV64 registers with 32 extended xBGAS registers (e0∼e31) as shown in Figure 1. The extended registers are utilized to store the upper 64-bits of a target address. In addition to the registers themselves, we also extend the ISA of RV64I to provide instructions that manipulate these extended registers. As demonstrated in Figure 2, we decompose the *eld* instruction as an example. *eld* is an extended instruction that loads 64B data into a specified destination register. The address of the data access consists of two segments, the lower 64-bit address (0∼63) held in x21, and the upper 64-bit address (64∼127) stored in e21. The combined 128-bit address will be used to load the data.

We introduce the overall architecture of the proposed xBGAS extension in Figure 3. Inside each node, a new component we designate as the *object look-aside buffer* (OLB) is included to hold the address mapping of the upper 64-bit addresses. Each

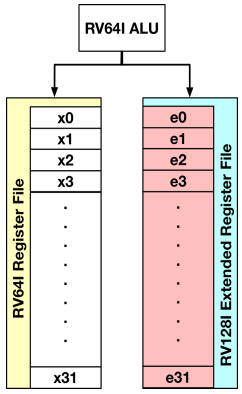

Figure 1. Register Extension of xBGAS

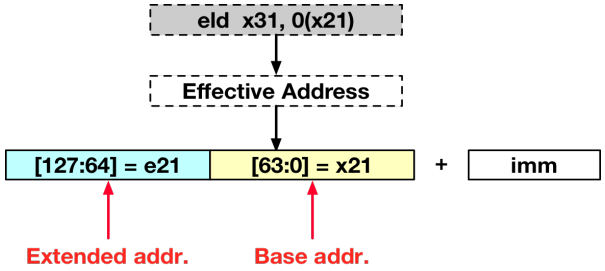

Figure 2. Integer Load/Store Instruction

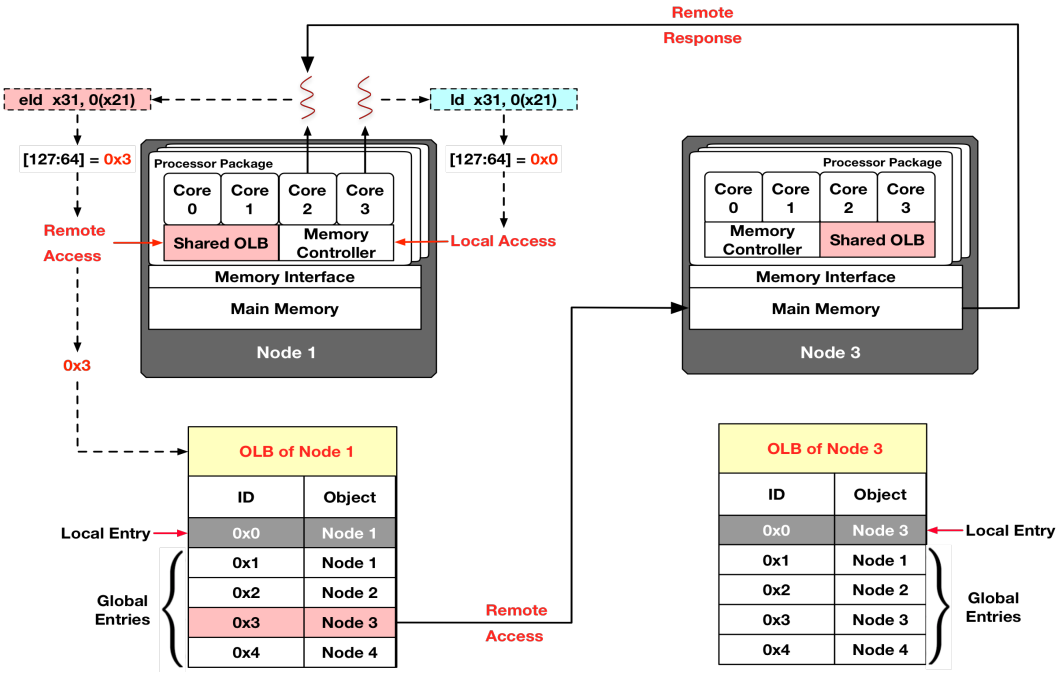

Figure 3. The architecture of the xBGAS

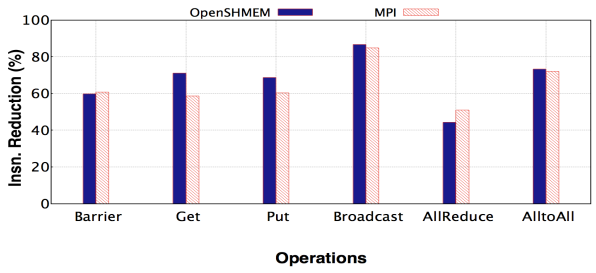

Figure 4. xBGAS Inst. Reductions by OP

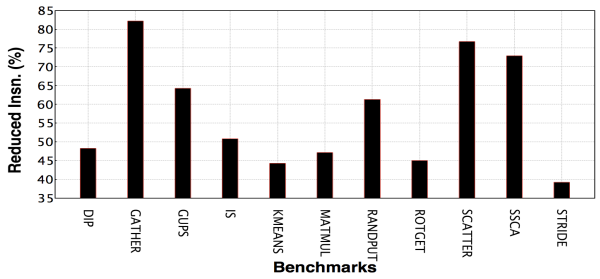

Figure 5. Overall Instruction Reductions

OLB consists of two columns. The first column contains the index of nodes and the second column stores corresponding objects. The upper 64-bit addresses are utilized to visit the OLB during remote memory accesses. It is also noticeable that index 0x0 always correlates to the local entry of each node. This signifies that if the upper 64-bit segment of a given address is zero, then the corresponding memory access is a local memory access. As shown in Figure 3, there are two threads running in node 1. The standard load operation is routed to the memory controller since the upper 64-bit address is 0. In contrast, the extended load instruction (*eld*) is forwarded to the OLB. Since the upper 64-bit address is 0x3, node 3 will be accessed remotely by node 1.

We demonstrate the potential performance impacts of the xBGAS extension on the HPC-PGAS model by examining the number of instructions generated by the xBGAS runtime library in comparison to conventional paradigms. We compare against OpenSHMEM [5], a prime example of the HPC-PGAS model, as well as the ubiquitous MPI send/receive model. OpenMPI 4.0.1, built upon UCX, is utilized for the implementation of both models. Figure 4 illustrates the percentage of instructions eliminated through the use of xBGAS for a single occurrence of common collective operations. As shown, utilization of xBGAS decreases generated instruction counts for OpenSHMEM and MPI by an average of 64.66% and 60.25%, respectively, across each tested operation. In order to better quantify the effect of xBGAS on more substantial workloads, we also record the total number of instructions generated by remote communication calls during execution of a varied set of kernels and benchmarks. We directly compare these results to those of our OpenSHMEM implementation in Figure 5. Here, xBGAS exhibits an average instruction count reduction of 57.46% across the tested workloads. These results showcase the significant promise xBGAS holds for the future of HPC-PGAS.

References

[1] GenZ Consortium. GenZ Core Specification. http://genzconsortium.org/specifications/draft-core-specification-july-2017/, 2017. Online; accessed July 2017.

[2] CCIX Consortium. https://www.ccixconsortium.com/, 2017. Online; accessed October 2017.

[3] OpenCAPI Consortium. http://opencapi.org/, 2017. Online; accessed October 2017.

[4] Farzad Fatollahi-Fard Kurt Keville John D. Leidel, David Donofrio. xBGAS: A Bridge Proposal for RV128 and HPC. Technical report, 7th RISC-V Workshop, 2017.

[5] Stephen W. Poole, Oscar Hernandez, Jeffery A. Kuehn, Galen M. Shipman, Anthony Curtis, and Karl Feind. *OpenSHMEM - Toward a Unified RMA Model*, pages 1379–1391. Springer US, Boston, MA, 2011.