Scalable Atomics for the Partitioned Global Address Space

Louis Jenkins

Abstract

Atomics are the fundamental building blocks of concurrent constructs, such as non-blocking data structures, synchronization techniques such as Read-Copy-Update, and even concurrent-safe memory reclamation algorithms. The Partitioned Global Address Space (PGAS) is meant to bridge the gap between distributed and shared memory in a way that accessing memory can be performed in a way that is transparent with regards to where the memory is actually allocated. In this way, it makes distributed computing all the easier but it does not come without a cost, as pointers to global memory, such as UPC's 'global_pointer' or Chapel's 'wide_ptr_t', which contains not only the memory address but also the rank of the compute node that the memory is located on, are larger than what most architectures can support for atomic operations. A solution to this problem would not only make porting shared-memory algorithms to distributed memory possible, but also enable the creation of new algorithms for high-performance computing.

Project Details

While this problem exists in UPC, I choose Chapel(Chamberlain, Callahan, and Zima 2007) as my primary focus as it is the most susceptible to accepting feedback and contributions. In Chapel, there are two types of pointers: pointers to memory that is proven to be local, or 'narrow pointers', which are 32-bit or 64-bit integers representing memory address locations, and pointers to memory that may not be local, or rather may be hosted on another compute node, called 'wide pointers'. Atomics on sizes smaller than 64-bits, or 8-bytes, can be handled with processor atomics if they are local, but will vary in one of two ways depending on the configuration of the cluster. If Chapel's *communication layer*, which is the abstraction layer responsible for handling all Remote Direct Memory Access (RDMA) PUT and GET operations, is GASNet, it will be constrained to using remote processor atomics, which essentially is a Remote Procedure Call (RPC) that will use computational resources of the destination compute node to perform the atomic operation; if the communication layer is uGNI, it will make use of Cray's Atomic Memory Operators (AMO) to perform a specialized RDMA

that is handled directly by the Gemini/Aries Network Interface Controller (NIC). The difference between remote processor atomics and network atomics can be shown below as at least an order of magnitude difference at 32 compute nodes on a Cray XC40 dual-socket Broadwell compute nodes, 22 cores per socket.

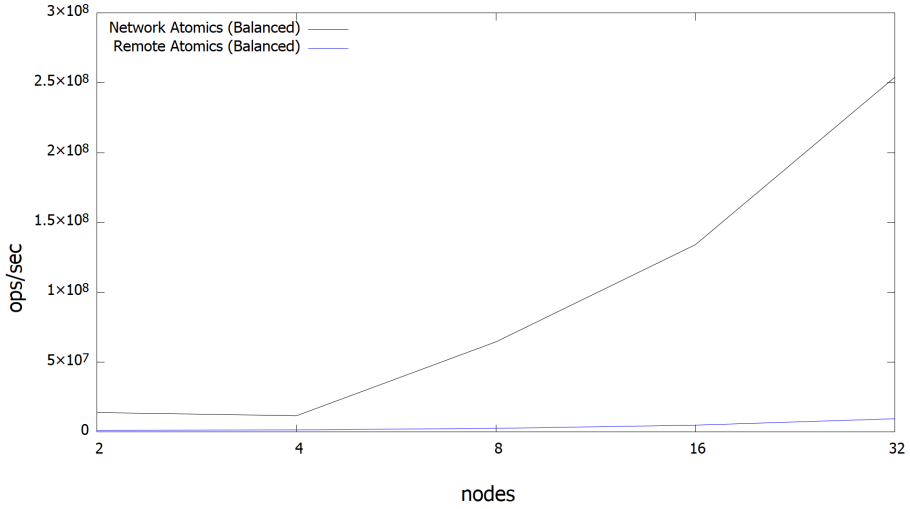

Figure 1: Atomic operations to random indices in block distributed array.

In the case of wide pointers, which are 'widened' to contain not only the memory address but the rank of the compute node the memory is hosted on, are generally considered to be 128-bits wide, twice the size of the native word on all but the most esoteric machines. Unfortunately, even Cray's NIC does not support 128-bit operations, and so the obvious way would be to use RPC and perform an `CMPXCHG16B` instruction, which is available on modern Intel and AMD processors, such as the (Intel) Broadwell compute nodes. While this may work, as shown in Figure 1, remote-execution is not a scalable solution and will easily lead to a bottleneck in any application. A prototype that I have designed and implemented over a year ago("GlobalAtomicObject: chapel-lang/chapel issue #6663" 2017) that makes use of 64-bit integer descriptors that are indices into a cyclic-distributed array has defeated the naive approach, even under intense time constraints. Also compared was an optimization that is safe for up to 2^16 - 1 compute nodes, which makes use of pointer compression, which takes advantage of the fact that only the lowest 48 bits of the virtual address space is ever used, leaving the upper 16 bits safe to modify, which directly compresses the size of pointers down to 64-bits. The comparison can be seen in Figure 2.

There have been multiple improvements to the prototype since then, such as making the descriptor table concurrent-safe to access while being resized via a Read-Copy-Update strategy(Jenkins 2018). This is only the beginning, as there are bound to be many other problems that can be solved on the way.

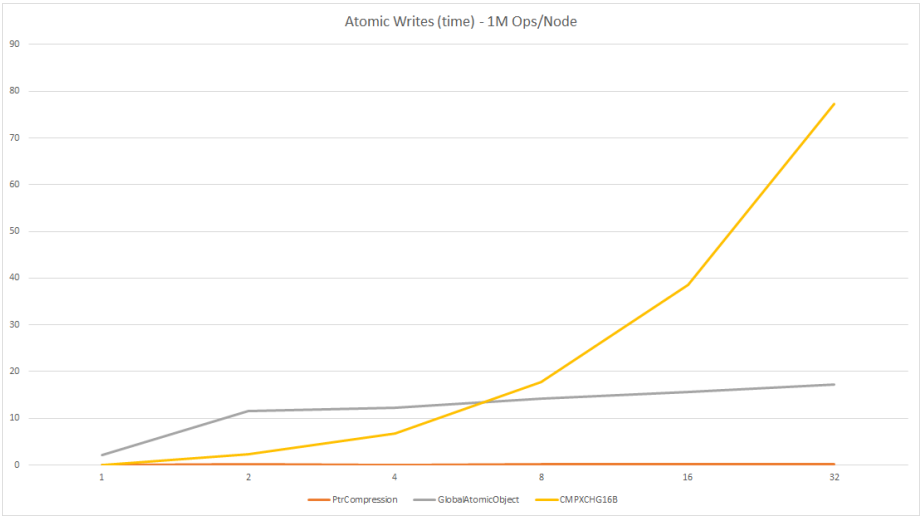

Figure 2: Atomic write operations using XCHG128B vs Prototype vs Compression

Broader Impact

Atomics are fundamental building blocks, and as such having access to these are necessary for the creation of some innovative new ideas. Having scalable atomics, whether they are narrow or wide, can lead to the production of more complex abstractions that make high-performance computing easier. As Information Technology is pushing more and more towards multi- and many-core, and soon many-node distributed computing, the problem of scalable atomics will eventually arise.

Resources

While I currently have access to a Cray XC50 supercomputer, provided by Cray's Marketing Partner Network (MPN), having exclusive access to a Cray XC machine to prevent congestion would be desirable, but continued access to Cray's MPN would be the bare-minimum. Cray XC30-AC machines are being sold for $500,000 to $3,000,000 which would suffice for independent research. Also required is funding of the independent research itself, which covers cost-of-living for Rochester, NY and an estimated $10,000 for attending conferences. Total estimated to be a lumpsum of $3,000,000 for the Cray XC30-AC and $75,000 per fiscal year.

References

Chamberlain, Bradford L, David Callahan, and Hans P Zima. 2007. "Parallel programmability and the chapel language." *The International Journal of High Performance Computing Applications* 21 (3): 291–312.

"GlobalAtomicObject: chapel-lang/chapel issue #6663." 2017. https://github.com/chapel-lang/chapel/issues/6663.

Jenkins, Louis. 2018. "RCUArray: An RCU-Like Parallel-Safe Distributed Resizable Array." In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 925–33. IEEE.