ISx in Chapel: Early Observations and Results

Brad Chamberlain, Ben Harshbarger, Lydia Duncan
Cray Inc.
Seattle WA 98164
{bradc, bharshbarg, lydia}@cray.com

Jacob Hemstad
University of Minnesota
Minneapolis, MN 55455
hemst013@umn.edu

Extended Abstract—The classic NAS Parallel Benchmarks (NPB) suite comprises a set of key computational idioms for scientific parallel computing at scale [1]. However, as time passes and system scales continue to grow, NPB is becoming increasingly dated. Recently, engineers at Intel have sought to modernize and improve upon several aspects of the Integer Sort (IS) benchmark from NPB, resulting in a new proxy application named ISx [2, 3]. ISx, like IS, is characterized by initial phases of pleasingly parallel computation that bucketize values locally, followed by an all-to-all style exchange to merge buckets across the system's compute nodes. ISx supports strong and weak scaling studies, uses uniform random key generation to guarantee load balance, and includes a verification stage.

In Autumn 2015, the Chapel team at Cray began a port of ISx as a pair-programming exercise with Jake Hemstad, a co-developer of ISx at Intel while interning there. In February 2016, we began evaluating its scalability and performance, both in isolation and relative to the SHMEM reference version. In this talk, we will report on our early experiences porting ISx to Chapel, presenting comparisons with the SHMEM version in terms of programmability and style. ISx is an interesting computation to study in Chapel due to the fact that the bucketing style of computation that it uses is very naturally suited for an explicit Single Program, Multiple Data (SPMD) style of programming rather than the global-view computations that Chapel has typically focused on in the past. That said, this talk will also consider what a global-view version of ISx would look like and discuss some of the tradeoffs that each approach entails.

We will also present performance results of ISx in Chapel as compared to the SHMEM reference version. In doing so, we'll look at the performance differences between various ways of expressing the computation in Chapel, including a version that is SPMD across locales and multithreaded across cores vs. a purer SPMD version across cores, which more closely mimics the SHMEM version. We'll also quantify the effect of key performance optimizations for the Chapel version, such as the use of bulk data transfers to support assignment between distributed array slices in one Remote Direct Memory Access (RDMA) put/get operation. Finally, we'll characterize the importance and effect of locality optimizations and cues, particularly as they apply to the use of atomic variables for multithreaded coordination in the fine-grain versions.

Bibliography

[1] D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow, "The NAS Parallel Benchmarks 2.0," NASA Technical Report NAS-95-020, NASA Ames Research Center, Moffett Field, CA, 94035-1000, December 1995. https://www.nas.nasa.gov/publications/npb.html

[2] Ulf Hanebutte, and Jacob Hemstad, "ISx: A Scalable Integer Sort for Co-design in the Exascale Era," *The 9th International Conference on Partitioned Global Address Space Programming Models (PGAS 2015)*, September 2015.

[3] https://github.com/ParRes/ISx