Evaluating Next Generation PGAS Languages for Computational Chemistry

Daniel Chavarría-Miranda (presenting author),
Sriram Krishnamoorthy,
Joseph Manzano,
Abhinav Vishnu
High-Performance Computing,
Pacific Northwest National Laboratory
{ daniel.chavarria, sriram, joseph.manzano, abhinav.vishnu }@pnnl.gov

Computational Chemistry algorithms and applications have aggressively used PGAS technologies to handle the challenges behind realizing high performance, scalable implementations. In particular, the use of Global Arrays (GA) to support distributed, global view, multi-dimensional dense arrays has been critical to be able to design and implement scalable versions of complex quantum chemistry algorithms inside NWChem. The challenges behind these algorithms broadly fall into two main categories: handling block-sparse data access patterns and handling load imbalance induced by those patterns.
In contrast to many physics applications, these chemistry algorithms cannot easily exploit domain decomposition and spatially constrained interactions to enhance locality and reduce the need for inter-process communication. Locality can be exploited only after appropriate blocks of data have been fetched from the global space. However, the locations of those blocks of data in the global space is input- and data-dependent and does not easily match common array data distributions onto the participating processes.

In order to evaluate the productivity and performance impact of next-generation PGAS languages on computational chemistry algorithms, we have implemented a kernel from the Self-Consistent Field (SCF) computational chemistry algorithm in Chapel and X10. This kernel is used to build the Hartree-Fock matrix, which is the basis for higher-order methods. The selected kernel computes the contribution of two-electron interactions to the Fock build. The original SCF code uses Global Arrays to store the principal data structures: the Schwarz matrix, the density matrix and the Fock matrix. All of these matrices are distributed equally amongst the running processes using a two-dimensional distribution, where each process stores a single contiguous block. The two-electron contribution involves a computationally sparse n^4 calculation over an n^2 data space. The computation is organized as a set of n^4 tasks that need to be enumerated and evaluated. Most of those tasks do not add any significant contribution to the Fock matrix. In fact, only a small percentage of them (< 1 % for larger inputs) do.

We discuss our experience implementing this algorithm in both next-generation PGAS languages. We compare and contrast the commonalities and differences between the two implementations, as well as compare their performance on single and multiple nodes of an Infiniband cluster. We also compare the performance against a C++ implementation using Global Arrays and discuss the advantages and challenges of language-based vs. library-based programming models.