Domain Specific Language Support for Exascale

1 Introduction

High performance computing (HPC) systems have experienced significant disruption, the result of technological trends that influence the strategy for implementing applications that can utilize the hardware effectively. The driving force is the need to curb power consumption. As a result, we are seeing exponential increases in parallelism within a single processor, reduced memory capacity per processor core and complicated memory hierarchies. Since supercomputers comprise an interconnected collection of servers or "nodes," the impact of disruption is a daunting array of programming challenges needed to deliver high performance, causing low level details to impose a hardship on the application focused programmer.

Large scale HPC applications run on distributed memory computers and employ message passing to move data across distributed memory, generally via the de-facto standard interface, MPI. While MPIs bulk-synchronous execution model works reasonably well for problems with a uniform structure, many cutting-edge applications use irregular representations in space and time in order to reduce memory and computation costs, posing a hardship for the programmer.

In HPC, performance often benefits from heuristic knowledge about the application. Since such knowledge cannot be discovered by traditional program development tools such as translators, optimizations are traditionally the responsibility by the programmer. More recently, *domain specific translation* has emerged as a means of incorporating heuristic (domain science) knowledge into the translator. While such translators aren't general purpose, they significantly improve programmer productivity. Research during the performance period resulted in 2 such translators–Bamboo and Toucan–that were able to overlap communication with computation automatically in legacy MPI applications, thereby improving application performance.

2 Domain Specification Translation of legacy MPI

As noted above, HPC relies on MPI to move data through distributed memory. An important optimization for MPI applications is to overlap communication with computation. However, MPI applications typically do not infer overlap, and must be written to manage it explicitly, imposing high software overheads that impede its use in practice. An alternative is to employ a non-bulk synchronous, data-driven execution model (e.g. dataflow) that can overlap communication and computation automatically. The remaining issue is how to avoid high programming overheads needed to manually recode legacy applications. Fortunately, there is a connection between message passing and data driven execution: communication between matched pairs of message sends and receives in a running MPI programming corresponds to data motion in a data flow graph.

The project developed two translators based on this observation that can infer overlap of computation and communication in MPI applications. Bamboo [1, 2, 3] (with former student Tan Nguyen, Phd 2014, now at LBNL) and Toucan [4] (with current student Sergio Martin, at UCSD) infer the data flow graph from calls sites of MPI sends and receives, and convert an MPI application to run as a data-driven program that masks communication automatically. The work is unique in that it uses domain-specific knowledge obtained from MPI call sites and relies on over decomposition of processes into multiple tasks, rather than explicit heterogeneous parallel control flow via MPI processes and OpenMP. The translation process comprises two phases: (1) add programmer annotation (2) pass the code through the translator.

Bamboos optimizations have been shown to reduce communication delays significantly in stencil methods and dense linear algebra at scale. In the case of dense linear algebra, Bamboo achieved the performance benefit of lookahead by restructuring the much simpler, but lower-performing, non-lookahead version of the well-known HPL benchmark [5], replacing the complicated synchronization used in the lookahead algorithm with an external dynamic scheduler invoked by code automatically generated by Bamboo.

Though Bamboo was able to hide communication, it employed static inlining to schedule communication, which cannot support recursion, and also lead to significant code expansion. The Toucan translator employed dynamic scheduling in lieu of inlining, rewriting the MPI calls to invoke Toucan's run time system. Code expansion is modest, so Toucan could handle full scale production codes, for example the Cart3D[1] aerospace design simulator developed for NASA. This code has a large user community, and as a result, Toucan is an enabling technology for modernizing production MPI applications.

3 Conclusions

Results with the Bamboo and Toucan translators showed that automated code restructuring produces performant code–that is competitive with hand coding–while avoiding the complexity of hand coding. Moreover, these translators demonstrate the power of domain specific translation: to overcome the limitations of conventional compilers and language constructs. The translators treated MPI constructs as if they were a primitive language constricts in an embedding language, in particular, C and C++. Thus, it should be possible to use domain specific translation to optimize applications that use other commonly used application development libraries besides MPI.

References

[1] T. Nguyen, P. Cicotti, E. Bylaska, D. Quinlan, and S. B. Baden, "Bamboo: Translating mpi applications to a latency-tolerant, data-driven form," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 39:1–39:11. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389050

[2] ——, "Bamboo - preliminary scaling results on multiple hybrid nodes of knights corner and sandy bridge processors," in *Third International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing, WOLFHPC '13*, 2013.

[3] ——, "Bamboo – translating mpi applications to a latency-tolerant, data-driven form," in *Proceedings of the 2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1109/SC.2012.23

[4] S. M. Martin, M. J. Berger, and S. B. Baden, "Toucan - a translator for communication tolerant mpi applications," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2017, pp. 998–1007.

[5] T. Nguyen and S. B. Baden, "Lu factorization: Towards hiding communication overheads with a lookahead-free algorithm," in *2015 IEEE International Conference on Cluster Computing*, Sept 2015, pp. 394–397.

[1]https://www.nas.nasa.gov/publications/software/docs/cart3d