A Runtime Analysis For Communication Calculation on Distributed-Memory Systems

Ana Moreton-Fernandez
Departamento de Informática, Edif. Tec. de la Información, Universidad de Valladolid, Valladolid, Spain
ana@infor.uva.es

Arturo Gonzalez-Escribano
Departamento de Informática, Edif. Tec. de la Información, Universidad de Valladolid, Valladolid, Spain
arturo@infor.uva.es

Diego R. Llanos
Departamento de Informática, Edif. Tec. de la Información, Universidad de Valladolid, Valladolid, Spain
diego@infor.uva.es

Abstract

Parallel machines are becoming more heterogeneous, mixing devices with different capabilities in the context of hybrid clusters, with hierarchical shared- and distributed-memory levels. Programming for these heterogeneous systems is challenging and error-prone. Many automatic code generation approaches have been proposed to transform high-level parallel programs or sequential codes to low-level parallel programs for hybrid clusters with distributed memory. Many of these distributed-memory approaches are based on compile-time automatic solutions [1–3]. These techniques abstract many issues related to the execution platform, while still deliver good performance. However, they generate a generic code that cannot take into account some specific details about the final machines where the application would be executed. As example, the most sophisticated code generators for distributed memory (in terms of data volume communicated, and parametric in the number of processes and problem sizes) still needs to fix a single tile size at compile time, even if the distributed system has nodes with different capabilities.

On the other hand, compiler scalability is a well known problem. Currently, according to [4], the costly integer programming algorithms used to generate optimized codes make their use impractical in real scientific applications for the common users.

We propose to move to runtime part of the compile-time analysis needed to generate the communication code for distributed-memory systems. Communication stages on distributed-memory systems have a significant impact on performance, thus the reduction of the communication times is key for improving performance in terms of runtime execution. We have developed a technique that uses a hierarchical tiling array library to represent and manage rectangular index spaces at runtime. The data to be received and/or sent by a local process to another one is calculated by intersecting the set of indexes read or written by a process with the set of indexes written or read by the local process.

The main advantage of our technique is that it automatically computes, at runtime, the exact coarse-grained communications to be used by a distributed message-passing model. By *exact communications* we mean: a) only needed data are communicated; (b) each needed data element is communicated to a target process only once (redundant communication is avoided due to the runtime analysis); and c) no control information exchange is needed. Additionally, the communications performed are *coarse-grained*. Communication calculation is done once for the whole index space mapped to a process at runtime, independently of the number or the sizes of tiles generated inside the process afterwards. This also allows the use, in the same computation, of different tile sizes for different processes at the same hierarchical level, a useful feature for distributed-memory systems that include machines with different architectures [5].

Our experimental results for several cases of study indicate that, despite our runtime calculation, we obtain similar results to optimized codes directly written with MPI and we outperform the results of other communication code generators.

References

[1] U. Bondhugula. Compiling affine loop nests for distributed-memory parallel architectures. In *2013 SC-International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 1–12. IEEE, 2013.

[2] M. Claßen and M. Griebl. Automatic code generation for distributed memory architectures in the polytope model. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 7–pp. IEEE, 2006.

[3] O. Kwon, F. Jubair, R. Eigenmann, and S. Midkiff. A hybrid approach of openmp for clusters. In *Proceedings of the 17th ACM SIGPLAN symposium on Principles and Practice of Parallel Programming (PPoPP)*, 2012.

[4] S. Mehta and P.-C. Yew. Improving compiler scalability: optimizing large programs at small price. In *Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, 2015.

[5] S. Mehta, G. Beeraka, and P.-C. Yew. Tile size selection revisited. *ACM Transactions on Architecture and Code Optimization (TACO)*, 10(4):35, 2013.