Simple and efficient parallel programming for distributed-memory systems

Ana Moreton-Fernandez*, Arturo Gonzalez-Escribano*,[1], Diego R. Llanos*,[1]

* *Departamento de Informática, University of Valladolid, Campus Miguel Delibes, 47011 Valladolid, Spain*

ABSTRACT

Several techniques and frameworks have been proposed to automatically generate parallel programs for hybrid distributed-memory systems from high-level parallel languages or sequential codes. These techniques should take into account the combination of data communication across distributed processes, and the exploitation of shared-memory models inside each process.

Trasgo is a new programming model and compilation framework to generate parallel programs from a high-level parallel specification, based on the SPMD (Single Program Multiple Data) model. This transformation system includes several techniques to generate code that is able to compute at runtime exact coarse-grained communications for distributed message-passing processes. In this work we discuss a technique to calculate the communications needed in distributed systems between SPMD blocks, for codes with regular (affine) data accesses. It improves previous methods avoiding to take compile-time tiling decisions, such as the tile size chosen for each process. It adapts their communication, synchronization, and optimization structures to the target system, even when computing nodes have different capabilities.

We show an excerpt of a complete experimental study. Our framework can automatically produce efficient programs compared with MPI reference codes, and with codes generated with auto-parallelizing compilers.

KEYWORDS: Distributed-memory systems, Communication calculation

1 Introduction

Many automatic code generation approaches have been proposed to transform high-level parallel expressions or sequential codes to parallel programs for distributed-memory systems. These techniques avoid to the programmer to deal with issues that are key to obtain a good performance. Many of these approaches are based on new programming languages [CDIC10, EGCSY03] or automatic solutions [CG06, Bon14, YR13, KJEM12]. The most sophisticated automatic solution (in terms of data volume communicated, parametric in the

[1]E-mail: {ana,arturo,diego}@infor.uva.es

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.

Trasgo [GEL11] is a programming model and compilation framework for distributed-memory systems based on the SPMD (Single Program Multiple Data) model. It has been designed to generate parallel programs from a high-level parallel specification. The Trasgo transformation system includes several techniques to transform the code to compute at run-time exact coarse-grained communications for distributed message-passing processes.

In this work we discuss a technique of our compiler-runtime system that calculates at runtime the exact data to be received or sent for each process just before computing a SPMD block. Our runtime approach enables to use different tile sizes in the same computation at the same hierarchical level. Our experimental results show that our approach produces efficient programs compared with codes generated with auto-parallelizing compilers and MPI reference codes.

2 Trasgo and its automatic compiler-runtime tools

2.1 Challenges on distributed-memory systems

Programming for distributed memory systems has two main challenges that are key for obtaining a good performance in terms of runtime execution and memory allocation: (1) *Distributing data*: An efficient approach is one where each process allocates only the data that it computes and the data that needs to perform such computation. (2) *Communicating data among process*s: As the data and computation are distributed among the processes, it is possible that a process needs a datum that is allocated or was updated in another process. Communications between processes allow to exchange data in distributed-memory systems. However, calculating and expressing these communications is not a simple task.

2.2 Trasgo

The Trasgo model [GEL11] proposes the use of an explicitly parallel (SPMD model) but high-level and structured representation of parallel algorithms. Trasgo transforms a global address space into a partitioned address space, building the functions to compute communications across virtual processes. The code is rewritten by a back-end that generates C code with calls to the Hitmap run-time library [GETFL13]. The resulting sequential code generated for the local distributed process can be finally optimized through polyhedral tools (Pluto compiler [BHRS08]) to generate tiled and optimized parallel code for shared memory using OpenMP [MFGEL14].

2.3 Compiler-runtime tool

We have developed and implemented in Trasgo a new technique to calculate automatically at runtime the communications needed in a distributed programming model among two SPMD blocks with distributed data structures. The technique is based on a compile-time analysis that, from a set of affine expressions generates tailored functions to calculate at runtime the set of indexes accessed to read or to write in a SPMD block.

Table 1: Study 2: Performance (in seconds) for Jacobi-2d solver (N=5000, T=800), generated for distributed-memory by Trasgo, and by Pluto-distmem.

	Jacobi-2d			
	Trasgo		**Pluto**	
Machine	**Seq.**	**Comm.**	**Seq.**	**Comm.**
CETA-4	53.04	8.09	36.88	15.98
CETA-8	36.17	13.19	22.15	17.87
CETA-16	20.24	13.14	16.49	30.66
CETA-32	8.63	9.40	8.54	22.41
CETA-64	5.77	12.29	8.55	25.39

Trasgo introduces, between two SPMD blocks, a communication stage where calculates at runtime a communication pattern using the functions generated at compile time. The communication pattern contains the information needed to marshal and unmarshal the data to be received and sent. The data to be received by a local process from a process p is calculated intersecting the set of indexes written by p in the first SPMD with the set of indexes read by the local process in the second SPMD. The data to be sent is calculated by the opposite intersection (set of indexes written by the local process with the set of indexes read by p). Performing these operations for each remote process p, we obtain an exact coarse-grained communication pattern per each process where no data are communicated twice.

3 Experimental study

In this section we show an excerpt of an experimental study where we compare a compile-time state-of-the-art tool that also generates communication code, Pluto [DRRB13], with our runtime proposal. We have performed the experimentation in a homogeneous distributed-memory system called *CETA*. The cluster nodes are connected by Infiniband technology, and they have two Intel Nehalem-based Xeon 5520 CPUs at 2.27 GHz, with 4 cores each using *mpich3* v3.1.3 as MPI implementation.

Table 1 shows the performance results for a Jacobi-2d solver. The results indicate that the code generated by Pluto is more efficient for a low number of processes, but does not scale as well as Trasgo. The code transformations performed by Pluto include skewing the time outer-most loop to parallelize the complete affine-nest-loop. It derives in a lot of re-utilization and memory hierarchies exploitation inside the processes. On the other hand, Trasgo codes have a hierarchical approach in which only spatial parallelism is exploited at the distributed-level, like in classical manual message-passing approaches. This derives in coarse communications, and less opportunities to exploit the shared-memory level due to extra synchronizations. However, as the number of processes grows, Pluto reveals its more clumsy communications calculations, while the granularity of the Trasgo communications decreases, and its reduced cost for communications becomes much more relevant.

4 Conclusion

We have designed and developed a framework to help to the programmer to deal with the main challenges on distributed-memory systems, data partition and communication.

The new technique implemented in our framework calculates exact coarse-grained communications in terms of data volume (no data is communicated twice). Moreover, it applies tiling technique after the data structures have been parted. This enables to use different tile sizes in the same computation, an important feature to achieve a good performance on distributed systems with nodes of different capabilities. The Trasgo framework is available at `http://trasgo.infor.uva.es`.

Acknowledgements

This research has been partially supported by MICINN (Spain) and ERDF program of the European Union: HomProg-HetSys project (TIN2014-58876-P), and COST Program Action IC1305: Network for Sustainable Ultrascale Computing (NESUS). by the computing facilities of Extremadura Research Centre for Advanced Technologies (CETA-CIEMAT), funded by the European Regional Development Fund (ERDF). CETA-CIEMAT belongs to CIEMAT and the Government of Spain.

References

[BHRS08] Uday Bondhugula, Albert Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral program optimization system. In *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, June 2008.

[Bon14] U. Bondhugula. Compiling affine loop nests for distributed-memory parallel architectures. In *Proc. SC'2014*, Denver, CO, USA, 2014. ACM.

[CDIC10] B.L. Chamberlain, S.J. Deitz, D. Iten, and S-E. Choi. User-defined distributions and layouts in Chapel: Philosophy and framework. In *2nd USENIX Workshop on Hot Topics in Parallelism*, June 2010.

[CG06] Michael Claßen and Martin 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.

[DRRB13] Roshan Dathathri, Chandan Reddy, Thejas Ramashekar, and Uday Bondhugula. Generating efficient data movement code for heterogeneous architectures with distributed-memory. In *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on*, pages 375–386. IEEE, 2013.

[EGCSY03] T. El-Ghazawi, W. Carlson, T. Sterling, and K. Yelick. *UPC : distributed shared-memory programming*. Wiley-Interscience, 2003.

[GEL11] A. Gonzalez-Escribano and D.R. Llanos. Trasgo: A nested-parallel programming system. *The Journal of Supercomputing*, 58(2):226–234, 2011.

[GETFL13] A. Gonzalez-Escribano, Y. Torres, J. Fresno, and D.R. Llanos. An extensible system for multilevel automatic data partition and mapping. *IEEE TPDS*, 25(5):1145–1154, 2013. (doi:10.1109/TPDS.2013.83).

[KJEM12] Okwan Kwon, Fahed Jubair, Rudolf Eigenmann, and Samuel Midkiff. A hybrid approach of OpenMP for clusters. In *ACM SIGPLAN Notices*, volume 47, pages 75–84. ACM, 2012.

[MFGEL14] A. Moreton-Fernandez, A. Gonzalez-Escribano, and D.R. Llanos. Exploiting distributed and shared memory hierarchies with Hitmap. In *Proc. HPCS'2014*, pages 278–286, Bologna (Italy), 2014.

[YR13] Tomofumi Yuki and Sanjay Rajopadhye. Parametrically Tiled Distributed Memory Parallelization of Polyhedral Programs. Technical Report CS13-105, Colorado State University, June 2013.