Trasgo 2.0: Code generation for parallel distributed- and shared-memory hierarchical systems

Ana Moreton-Fernandez, Arturo Gonzalez-Escribano, and Diego R. Llanos

Departamento de Informática, Universidad de Valladolid
ana.moreton@alumnos.uva.es,{arturo,diego}@infor.uva.es

1 Extended Abstract

Current multicomputers are typically built as interconnected clusters of shared-memory multicore computers. A common programming approach for these clusters is to simply use a message-passing paradigm, launching as many processes as cores available. Nevertheless, to better exploit the scalability of these clusters and highly-parallel multicore systems, it is needed to efficiently use their distributed- and shared-memory hierarchies. This implies to combine different programming paradigms and tools at different levels of the program design.

Programming in this kind of environment is challenging. Many successful parallel programming models and tools have been proposed for specific environments. However, the application programmer still faces many important decisions not related with the parallel algorithms, but with implementation issues that are key for obtaining efficient programs. For example, decisions about partition and locality vs. synchronization/communication costs; grain selection and tiling; proper parallelization strategies for each grain level; or mapping, layout, and scheduling details. Moreover, many of these decisions may change for different machine details or structure, or even with data sizes.

This paper presents an automatic code generation system for mixed distributed- and shared-memory parallel multicomputers. We present an extension of the Trasgo programming model. This extended model supports a wider range of parallel structures and applications where coordination is expressed at an abstract level. Transparent modular objects are invoked to guide the partition and mapping of both data and processes, across the whole system. We present a technique that, for affine expressions, compute exact aggregated communications at the distributed level. It uses intersection of remote and local footprints in terms of the mapping policies selected. Moreover, Trasgo 2.0 integrates polyhedral analysis tools to obtain optimizations inside each shared-memory parallel node at the shared level. This approach allows to automatically generate multilevel parallel programs that adapt their communication and synchronization structures to the target machine. Our experimental results for both, shared- and distributed-memory environments, show how this approach can automatically produce efficient codes when compared with manually-optimized codes using MPI or OpenMP models.