Automatic Parallelization in the Manycore Era

Diego R. Llanos, Arturo Gonzalez-Escribano
Dpto. Informática, Universidad de Valladolid, Spain
{diego,arturo}@infor.uva.es

Extended abstract

During the last decade, advances both in manycore processors design and parallel programming models have led to a rise of parallel development of regular applications. One successful example is the ubiquous use of GPUs for parallel computing. Irregular applications, on the other side, are still difficult to parallelize by hand, because they require a deep knowledge of the problem itself, the parallel programming model being used, and the underlying architecture. Having a system that automatically parallelize them would be highly desirable. For an increasing number of applications, the use of advanced compile-time techniques such as the polyhedral model allows to obtain parallel versions automatically. However, many of them still defy automatic parallelization at compile time, due to their complex control path, and to the lack of runtime information that would be needed at compile time to ensure correctness. This is why a mechanism to automatically develop a parallel version of any sequential program is still a desirable goal.

Thread-level speculation (TLS) is a well-known technique that allows to automatically extract parallelism of sequential applications, mostly at loop level. Many software-based TLS proposals have been published so far, requiring different degrees of human intervention. Our contribution to the field is ATLaS, a software-based TLS runtime library to safely execute in parallel any loop, including those that may suffer from dependence violations. ATLaS is intended to be used in the context of OpenMP programs, by offering a non-standard speculative clause to let the user mark the variables whose use may lead to a dependence violation, letting the runtime library to guarantee correctness in any case. In fact, ATLaS can be used in fully-automatic mode, just by labeling all variables of the loop as speculative, at the cost of a potential performance loss.

The purpose of this contribution is to discuss the role of software-based TLS solutions in the following years. From the software side, automatic parallelization techniques such as those based on the polyhedral model extracts parallelism from an increasing number of applications. The question here is whether this reduces the need from speculative runtime techniques. From the hardware side, the advent of manycore systems with dozens or hundreds of processors makes classic TLS techniques to have diminished returns. To deal with this scenario, an update of TLS runtime architectures may be desirable.