Introduction

The Web and Java have stolen some of the limelight over the past few years, but the role of parallel processing in the drive to ultimate performance remains unquestioned. National programs like Petaflops and ASCI attest continuing support for research and development in basic parallel processing technologies. While the argument will persist over what proportion of applications can benefit economically from parallel execution, we remain convinced that lack of good system software and tools is the main technical reason for the limited deployment of large scale parallel processing systems.

Sponsored by ARPA, PCRC (Parallel Compiler Runtime Consortium) is an effort to establish a common base for parallel compiler development. This common base is a *runtime library*, which is supposed to be available to the compilers. It is linked to the generated "node program".

The basic idea is not new. Almost every compiler targetting a parallel architecture is supported by some sort of runtime, which resolves issues impossible to handle at compile-time. Those runtimes are usually designed at a low level in the hope of squeezing out the maximum performance. According to this mindset, low-level `send()` and `receive()` operations are encouraged in the node program. A consequence of this approach is that an *operational* parallel compiler is very difficult to construct—the compiler has too many details to take care of, and often ends up resorting to naive strategies. The result is that performance is much lower than it should be—promised global optimization by thin runtimes is hard to exploit *[what does this mean?]*. Our philosophy in PCRC is to build a relatively high level runtime library. We believe this approach will greatly facilitate compiler construction and deliver comparable or better performance for typical applications. An important aspect of PCRC runtime is that it is supposed to be a common base for different compilers. This demands a flexible design, and an implementation that adopts sound engineering practices.

PCRC is a joint effort among 10 organizations. This paper reports the work carried out at Syracuse University, in NPAC. In particular, section 2 introduces the architecture, functionality, and implementation of core PCRC runtime library (which will shortly be made public domain). Three groups of functions: data management, data movement, and computational routines, are described. Section 3 introduces an HPF compiler that uses this runtime. This serves as an exemplar for the use of the runtime. It also describes an infrastructure that has been developed in parallel with compiler construction. The algorithm for communication detection and the approaches employed for procedure interface processing and memory management are illustrated through segments of a node program generated by the compiler. Section 4 is a performance study. We compared performance of our compiler with top commercial HPF compilers on a Laplace solver and some test programs. The data show that our compiler produces higher performance and more consistent code than the commercial ones. Section 4 is a brief overview of related work, and planned future work.