The performance of the PCJ library for massively parallel computing

Marek Nowicki\textsuperscript{1}, Piotr Bała\textsuperscript{1,2}

\textsuperscript{1} Nicolaus Copernicus University, Toruń, Poland
\textsuperscript{2} ICM, University of Warsaw, Warsaw, Poland
\{faramir@mat.umk.pl, bala@icm.edu.pl\}

This paper presents a new version of the PCJ library \cite{1, 2} for Java language that helps to perform parallel and distributed calculations. The current version is able to work on the multicore systems connected with the typical interconnect such as ethernet or infiniband providing users with the uniform view across nodes.

The library implements partitioned global address space model and was inspired by languages like Co-Array Fortran, Unified Parallel C and Titanium. In contrast to listed languages, the PCJ does not extend nor modify language syntax. When developing the PCJ library, we put emphasis on compliance with Java standards. The programmer does not have to use additional libraries, which are not part of the standard Java distribution.

In the PCJ, each task has its own local memory and stores and access variables only locally. Some variables can be shared between tasks and that variables can be accessed, read and modified by other tasks. The library provides methods to perform basic operations like synchronization of tasks, get and put values in asynchronous one-sided way. Additionally the library offers methods for creating groups of tasks, broadcasting and monitoring variables.

We have used PCJ library to parallelize example application, in this case ray tracing. We have measured the performance of 3D ray tracing of the scene rendered at a resolution of NxN pixels. The PCJ version of the RayTracer test contains a naive implementation of Reduce operation that uses asynchronous get (getAsync) method. The speedup for PCJ library is competitive compare to MPICH2; the gained speedup for larger cases is even better.

The tests show that there are still areas for improvements, especially in the case of internode communication. The mechanisms for synchronizing and transmitting messages should be improved. Additionally, there are no advanced techniques for the breakdown recovery and node failure handling.

References

1. PCJ library is available from the authors upon request.
This work is licensed under a Creative Commons Attribution-NoDerivs 3.0 Unported License. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nd/3.0/