How to Program a Parallel Computer?

Martti Penttonen

University of Kuopio, Department of Computer Science, P.O.Box 1627, 70211
Kuopio, Finland, email penttonen@cs.uku.fi

Abstract. The aim of this survey is to show that efficient programmer-friendly parallel computation is possible, under certain general assumptions. However, a lot of work remains to be done before parallel computation is everyday practice.

The main reason limiting the efficiency of the sequential computer is the so called *von Neumann bottleneck*. As all data is processed in a single processor, it becomes a bottleneck of execution. The speed of processors has grown tremendously — now we have processors executing more than 10^9 instructions per second. Also the *bandwidth* (i.e. the number of bits per second that can be moved between two points) of data communication has grown enormously — we can speak about transfer rates of terabits per second. However, increased processor speed and bandwidth are not enough. As the speed of light is constant, a datum farther than 15 cm cannot be fetched within a clock cycle. Also the speed of memories has not grown as fast as the speed of processors. Thus getting a datum from memory takes the time of tens of instructions. This is the *latency* problem.

The von Neumann bottleneck is removed by using more than one processor. Ideally, by p processors we should be able to do p times more than one processor, assuming that processors can work independently. However, if we should speedup a single computation, getting p-fold efficiency is challenging. By the theory of parallel algorithms we know that many common problems have algorithms that consist of a large number of parallel threads, and thus they can utilize a large number of processors. Highly parallel algorithms also solve the latency of memory access, using the *slackness* principle.

In this talk, we shall first show, how a lot of parallel threads can be found in problems that seem sequential at first sight. Then we shall show, how slackness principle is applied to hide the latency. Finally, we shall consider computer architectures that have enough bandwidth to allow efficient execution of parallel algorithms.