CS290B: Java-Centric Cluster & Concurrent Computing
John-Olav Storvold, Marianne Melhoos, Johannes Lier, Julie Rekdal
UCSB, Spring 2015

Cilk: An Efficient Multithreaded Runtime System

Introduction
Cilk is a runtime system with a work-stealing scheduler that is efficient in both theory and practice. Cilk is built on the C programming language and is designed for multithreaded programming. Threads can spawn children and successor threads which in total makes the Cilk model a directed acyclic graph (DAG). The threads cannot block so in order to receive a return value from a child thread a successor thread must be spawned to receive the value when it is produced.

The Cilk programming environment and implementation
A thread in Cilk contains a pointer to a *closure* which is passed as an argument. The closure data structure contains pointer to the C function for a thread, a slot for each of the specified arguments and a join counter that keeps track of number of arguments passed. A closure is ready is all arguments is present or waiting if some are missing. As mentioned earlier, the Cilk procedures do not return values in the traditional way. The programmer has to code the wanted procedure as two threads. The first one spawns the child procedure, and the second is a successor which receives the value as an argument from the child procedure.

The Cilk work-stealing scheduler
Cilk uses a work-stealing technique for scheduling. When a processor runs out of work it steals the shallowest ready thread-task of another random processor. The work from the shallowest level of the ready queue is chosen because the amount of work is larger, which leads to fewer steals which often lower the communication cost of the program. Another reason is that the work they steal usually make progress along the critical path if processors are idle.

Performance of Cilk application
The efficiency is close to 1 for programs with moderately long threads. This is because the Clk overhead is small. If average parallelism is large the speedup (T_1/T_p) is almost perfect. If the average parallelism is on the other hand small, the speedup is much smaller.

Conclusion
Cilk is about abstractions like work and critical path, that characterizes the algorithm performances and are not depended on the machine configuration. Cilk guarantees performance as a function of these two abstractions. Despite this, the capabilities are limited and the explicit continuationpassing style can be burdensome for the programmers. Cilk is good for dynamic, asynchronous MIMD computations, but not for traditional parallel applications. To improve Cilk they try among other things to implement “DAG-consistent” shared memory without costly communication or hardware support. It is important not to destroy the guaranteed performance of the scheduler.