Projects and Open Problems

Using Cilk

- Parallelize an application from your own research domain.

Improving Cilk

- Design and implement a provably good parallel `malloc()` function.
- Experiment with linguistic extensions to Cilk to support static threading, file I/O, etc.
- Implement a parallel garbage collector for Cilk.

NonDeterminator

- Make the Nondeterminator data-race detector run in parallel.
- Design an efficient Nondeterminator-2 that works with locks. The Nondeterminator presented in class does not work when locks are introduced. The Nondeterminator-2 does work with locks, but the performance guarantees are dramatically weaker.
- Design a more efficient Nondeterminator based on a different least-common-ancestor algorithm? The SP-bags algorithm is strongly based on an least-common-ancestor algorithm of Tarjan, which introduces the inverse Ackerman overhead. The simplest optimal LCA algorithm is by Bender and Farach-Colton (The LCA Problem Revisited, LATIN 2000), and also appears on the class web page. It may not be useful however, because the nondeterminacy races should be found online. Can this or other LCA algorithms also be modified to work in the same online manner?

Algorithms Projects

- Design an efficient parallel max-flow algorithm, simplex algorithm, sparse-matrix algorithm, etc.
- Devise a provably good multithreaded sorting algorithm which performs well in parallel.
- Design a scalable cache-consistency protocol that is provably good.
- Transactional Memory?
- Race detection in transactional memory.

Theory Projects

- Improve the cache-fault bounds for work-stealing.
- Resolve the gap between the lower and upper bounds for the VLSI area of an n-vertex planar graph.
- Transform programs written for synchronous processors so that they run efficiently on asynchronous processors. Aumann and Rabin (Clock Construction in Fully Asynchronous Parallel Systems and PRAM Simulation. TCS(1&2): 3-30 (1994)) develop some nice tools for this problem. Aumann, Kedem, Palem, Rabin (Highly Efficient Asynchronous Execution of Large-Grained Parallel Programs.FOCS 1993) improve the algorithm for large-grained parallel programs.

  Can you design a methadology for running multithreaded parallel programs on fully asynchronous systems? While there have been some generalizations (e.g., nondeterministic parallel programs), no one has considered this problem previously. The closest paper is by Bender and Rabin (Online Scheduling of Parallel Programs on Heterogeneous Systems with Applications to Cilk. " Theory of Computing Systems, 2002.) However, their algorithm sacrifices fault-tolerance for the sake of efficiency in the common case and is designed for a different setting.