

Julia Lipman

CONTACT INFORMATION

Computer Science and Engineering
University of Michigan
2260 Hayward
Ann Arbor, MI 48109 USA

Voice: (734) 272-3624
E-mail: jlipman@eecs.umich.edu

RESEARCH INTERESTS

Parallel computing, discrete mathematics

EDUCATION

University of Michigan, Ann Arbor, Michigan USA

Ph.D., Computer Science

- Thesis Topic: “Performance Analysis of Local Synchronization”
- Advisor: Quentin Stout

Massachusetts Institute of Technology, Cambridge, Massachusetts USA

B.S., Mathematics

PUBLICATIONS

Local Synchronization with Batching (with Quentin F. Stout), submitted.

A Performance Analysis of Local Synchronization (with Quentin F. Stout), Proceedings 18th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), 2006.

HONORS AND AWARDS

EECS Outstanding Graduate Student Instructor, Honorable Mention, 2007

Theory division nominee for the University of Michigan Computer Science and Engineering Graduate Honors Competition, 2005

Rackham Merit Fellowship, 2001

ACADEMIC EXPERIENCE

University of Michigan, Ann Arbor, MI USA

Lecturer

September, 2007-present

Instructor of a programming and data structures course in C++ with 240 students.

University of Chicago Young Scholars Program, Chicago, IL USA

Instructor

July 2005

Designed and taught intensive summer course on the theory of computation for mathematically talented eleventh- and twelfth-graders.

University of Michigan, Ann Arbor, MI USA

Graduate Student Instructor

January, 2004-April, 2007

Teaching discussion sections for junior-level theory of computation course and sophomore-level discrete mathematics course.

WORK EXPERIENCE

Boston.com, Boston, MA USA

Technology Journalist

April, 2001- January, 2002

Wrote weekly technology column for *The Boston Globe's* website.

RESEARCH
SUMMARY

The research I have done under the the direction of Quentin Stout concerns synchronization on parallel machines, or, more generally, among the agents of any other distributed system. We find bounds on the expected time to complete independent, identically distributed tasks under global and local synchronization. Many parallel programs use global, or barrier, synchronization, which can cause delays due to the variability in task times even for tasks that should take approximately the same amount of time. Local synchronization, in which processors synchronize with only a subset of other processors, would be adequate in many of these cases. My research explores the question of how much global synchronization slows down a system as compared to various types of local synchronization.

When task times are geometrically distributed — specifically, each task is modeled as flipping a fair coin until heads appears — each processor is expected to take 2 time units to complete each task if there is no synchronization at all. Under global synchronization among n processors, the expected time grows as $\log n$. However, under local synchronization on a directed cycle of n processors, we have shown that the expected time approaches an exact limit of $2 + \sqrt{2}$ time units as n goes to infinity.

We have also shown that it is not true in general that the expected time is bounded by a constant for graphs of bounded degree. In particular, the expected time to complete some power-law-distributed tasks under synchronization on a grid of two or more dimensions can grow as $\Omega(n^{1/\alpha})$ for some positive α .

However, with normally distributed task times, the expected time is bounded by a constant as n increases for local synchronization on any graph with degree bounded by a constant, while global synchronization causes the expected time to increase as $\sqrt{\log n}$. We have shown an extreme-value-theory proof of these bounds.

We have also started to look at a number of different synchronization models, such as randomized, in which each processor synchronizes with a randomly chosen subset of its neighbors, and first-neighbor, in which each processor waits only for the first k of its neighbors to finish a given task, for some fixed k . We have shown that if $k = O(\sqrt{n})$ and the task time distribution has all finite moments, then this time is also bounded above by a constant as the number of processors increases.