Concurrency and Parallelism in ML

John Reppy

University of Chicago

MacQueen Fest — May 12, 2012

Personal history

- ML on Unix (Cardelli ML)
- ML + Amber $\Longrightarrow$ Pegasus ML
- Standard ML of New Jersey (Version 0.15 on tape)
- Pegasus ML + SML/NJ $\Longrightarrow$ Concurrent ML
- $\Longrightarrow$ Ph.D.!!!
- $\Longrightarrow$ Department 11261 at Bell Labs

What makes parallelism and concurrency hard?

The sequential core matters!

- Combination of shared mutable state and concurrency leads to data races and non-determinism.
- Adding synchronization to avoid these problems leads to deadlock.
- Shared memory does not scale well to NUMA and Distributed Memory architectures.
- Scaling is hard.

Claim: traditional imperative programming languages are a bad fit for concurrent and parallel programming.

Alternatives

- Java, C#. *etc.*
- Haskell
- X10

Standard ML

Claim: what we want is a strict, statically typed, functional language.
I.e., Standard ML

- Strict CBV semantics
- Type system distinguishes between mutable and non-mutable values.
- Programming style is value-oriented.

Challenges

SML does not come without challenges.

- Polymorphism
- Higher-order functions
- Garbage collection
- Exceptions

The Manticore Project

- The Manticore project is our effort to address the programming needs of commodity applications running on multicore SMP systems
- No shared memory
- Preserve determinism where possible
- Declarative mechanisms for fine-grain parallelism

The Manticore Project *(continued ...)*

Our initial language is called Parallel ML (PML).

- Sequential core language based on subset of SML: strict with no mutable storage.
- A variety of lightweight implicitly-threaded constructs for fine-grain parallelism.
- Explicitly-threaded parallelism based on CML: message passing with first-class synchronization.
- Prototype implementation with good scaling on 48-way parallel hardware for a range of applications.

Implicit threading

PML provides several light-weight syntactic forms for introducing parallel computation.

- Parallel tuples provide a basic fork-join parallel computation.
- Nested Data-parallel arrays provide fine-grain data-parallel computations over sequences.
- Parallel bindings provide data-flow parallelism with cancellation of unused subcomputations.
- Parallel cases provide non-deterministic speculative parallelism.

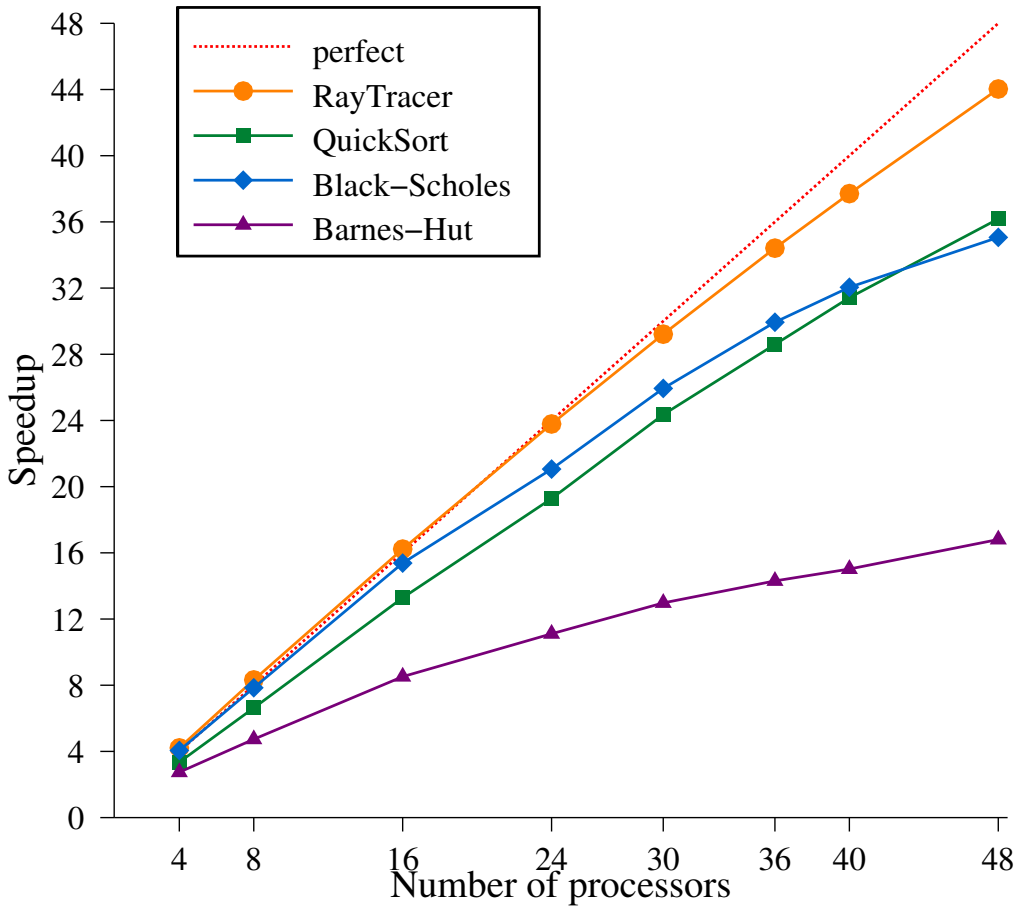
These forms are annotations that mark a computation as a good candidate for parallel execution, but the details are left to the implementation.

Challenges revisited

SML does not come without challenges.

- Polymorphism — whole program monomorphism using MLton's front end
- Higher-order functions — advanced CFA techniques
- Garbage collection — DGL split-heap GC and parallel global GC
- Exceptions — reduce use of arithmetic exceptions

PML performance



Speedup over sequential PML.

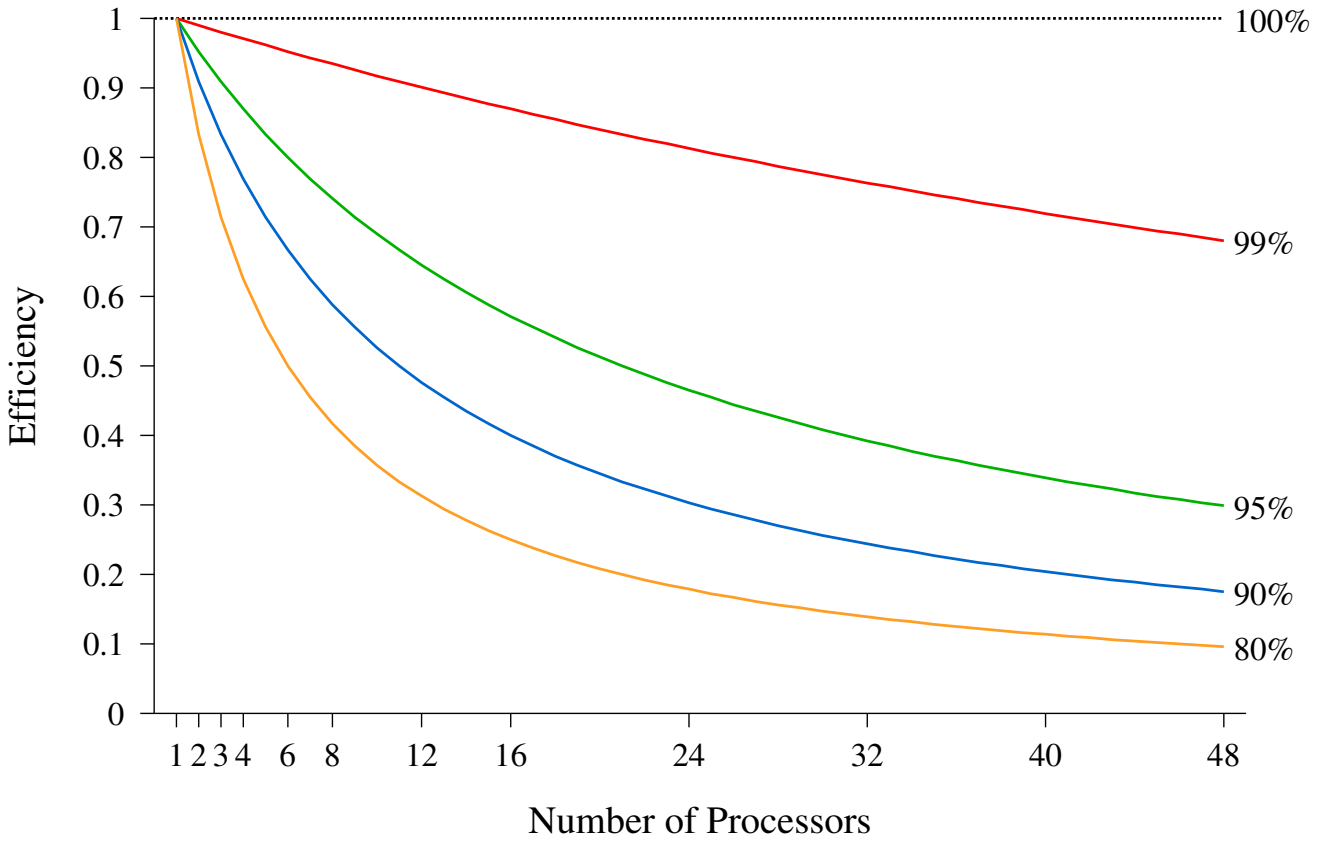
The need for shared mutable state

- Mutable storage is a very powerful communication mechanism: essentially a broadcast mechanism supported by the memory hardware.
- Sequential algorithms and data-structures gain significant (asymptotic) performance benefits from shared memory (*e.g.*, union-find with path compression).
- Some algorithms seem hard/impossible to parallelize without shared state (e.g., mesh refinement).
- But shared memory makes parallel programming hard, so we want to be cautious in adding it to PML.

The design challenge

- How do we add shared memory while preserving PML's declarative programming model for fine-grain parallelism?
- Some races are okay in an implicitly threaded setting.
- Deadlock is not okay in an implicitly threaded setting.

Limits on parallel performance: Amdahl's Law



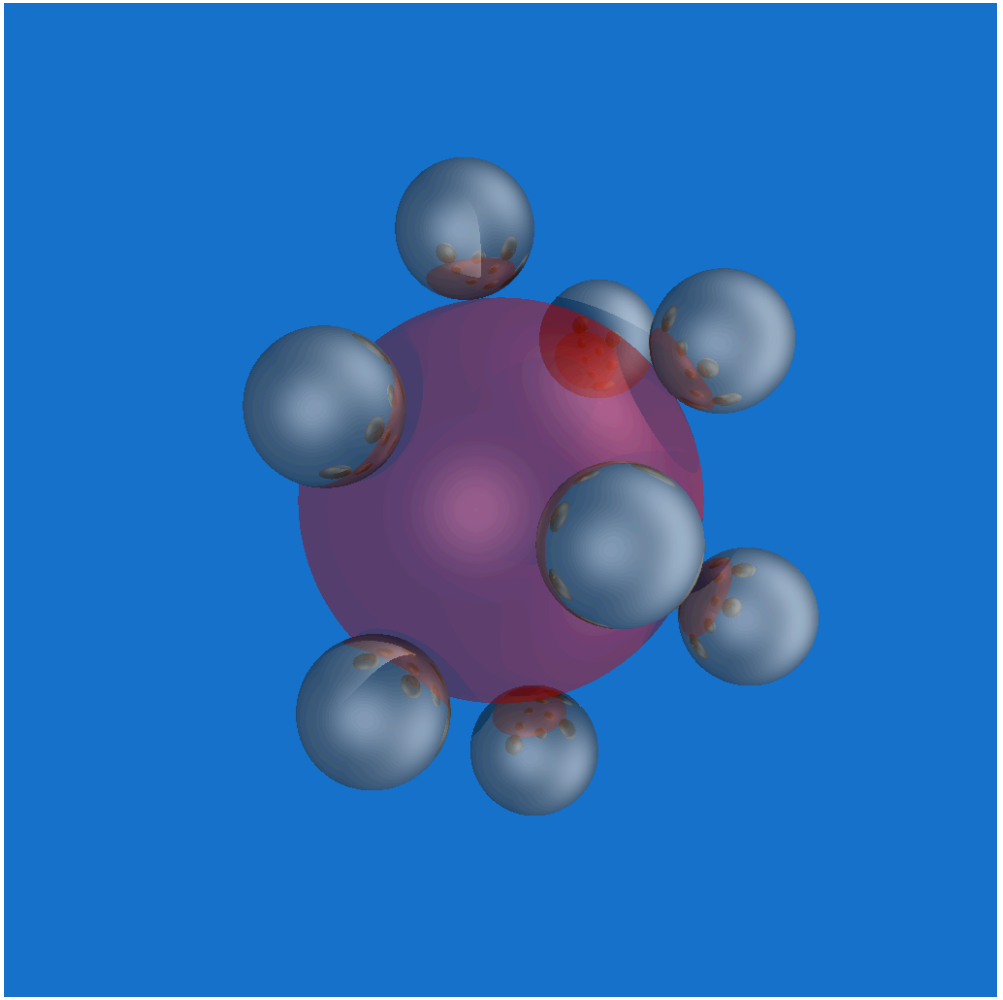
Speculation

- Amdahl's Law tells us that as the number of cores increases, execution time will be dominated by sequential code.
- Speculation is an important tool for introducing parallelism in otherwise sequential code.
- PML supports both deterministic and nondeterministic speculation.
- For many applications, we can relax determinism and still get a correct answer.

Credits

- Matthew Fluet (RIT)
- Claudio Russo (MSR Cambridge)
- Sven Auhagen, Lars Bergstrom, Mike Rainey, Adam Shaw, and Yingqi Xiao (U. of Chicago Graduate Students)
- Carsen Berger, Stephen Rosen, and Nora Sandler (U. of Chicago Undergraduates)
- Chelsea Bingiel, Nic Ford, Korie Klein, Joshua Knox, Jordan Lewis, and Damon Wang (Past U. of Chicago Undergraduates)
- National Science Foundation

Questions?



http://manticore.cs.uchicago.edu