Correlations between Parallel Patterns and Multi-core Benchmarks

Vivek Kale
IWMSE workshop
May 1st, 2010
Challenges for Multi-core

Measures for Performance Evaluation of Multicore

- Benchmarks are used in high-performance scientific computing research and industry for side-by-side comparison of two or more machines.

- One scientific computation kernel in a benchmark suite is not enough for side-by-side comparison of two machines. One implementation of a benchmark code is not enough for side-by-side comparison of two machines.

- Development of benchmarks that do what other benchmarks already do may not be productive.

- With the large variety of parallel architectures, we face a major challenge: how do we provide for a fair performance comparison of such a wide variety of machines?

Defining Parallel Software Development Strategies

- Parallel Patterns are important for assessing the trade-offs between different software solutions.

- Parallel patterns must encompass the large variety of solutions parallel programmers use.

- Parallel patterns should not overlap in content, and we should have one standard pattern rather than multiple different versions of the same pattern (perhaps written by different authors).

- How do we define a standard methodology that parallel programmers and software developers can use to take advantage of parallelism of multi-cores?
Parallel Patterns Layers

Applications

Structural Patterns

Computational Patterns

Algorithmic Strategy Patterns

Implementation Strategy Patterns

Parallel Execution Strategy Pattern

Coordination

Advancing Program Counters
Improving Benchmarks through Parallel Patterns

The NAS LU benchmark is a Gauss-Seidel kernel, used for applications involving simulation of heat dissipation and fluid dynamics.

**Computational Pattern:** *structured grid*

LU can be identified directly with a structured grid computation pattern because of the border exchanges involved.

**Structural Pattern:** *iterative refinement*

LU uses the iterative refinement structural pattern because of the outer loop that it iterates over until convergence.

**Algorithmic Structure Pattern:** *geometric decomposition*

LU benchmark involves exchanges of data between neighboring cells in the grid; this requires the use of “halo” or ghost cells.

**Implementation Pattern:** *pipeline parallelism*

Processor 0 starts at the topmost row, while processor 1 starts at one timestep later, using the result of the computation of processor 0 in the first timestep. Processor 2 follows processor 1, and so on.

**Parallel Execution Strategy Pattern:** *Depends!*

NAS LU is now implemented in several different programming libraries.
Benchmark and Pattern Quality

Using Patterns to Enrich Benchmarks

**Portability:** Can parallel patterns help to design benchmarks that are more portable?

**Modularity:** Can patterns help to separate a benchmark into modules so as to understand the different parts of a system that a benchmark is testing?

Using Benchmarks to Enrich Patterns

**Scalability:** Can scalable solutions in benchmarks explain how scalability is achieved in applications?

**“Tunability”** How can parallel patterns be enriched by understanding performance tuning techniques used for benchmark codes?

**Reliable Software:** How can benchmarks capturing fault-tolerance of a system tell us how to support fault-tolerance in software?
Conclusions and Future Directions

- General Issue: Controversy over large number of parallel software solutions, and large number parallel architecture. We believe that some consensus on standards is needed for progression for multi-core.

- The use of parallel patterns can strongly benefit benchmarking of multi-core machines, and the use of benchmarks to understand real-world parallel software solutions can be useful in capturing essential parallel patterns.

- The end result will be better organization, standardization, simplicity in both parallel software design and multi-core architectures.

**Ongoing and Future Directions:**

1. More thorough application of Parallel Patterns (OPL) to more of the NAS Parallel benchmarks (we have presented LU only).

2. Consideration of other benchmarks suites: PARSEC, SPEC

3. Addition of more parallel patterns based on documentation of NAS parallel benchmarks.
Thank You!

Questions?

Feel free to contact me at: vivek@illinois.edu

