Advanced Multicore Programming

Dr Andrew Moss

[1]BTH Karlskrona

September 27, 2011

The Optimisation Process

Optimisation is solving a particular kind of puzzle

- Techniques (e.g splitting, tiling etc) are legal moves
- Performance Measure is the score
- Game is to use fewest moves (i.e work) to get the best score

Problems occur when we make illegal moves

- Finding the highest score is easy
- Not breaking the target code is harder

Correctness vs Performance

Correctness dominates performance

- There is no tradeoff
- We don't care how quickly broken code doesn't work
- It is essential to test code constantly while optimising

Programmer time is expensive

- Making a programmer run tests repeatedly is hard
- Attention span...
- Automating testing is better

Diagnostic Techniques

Complex code require some inguenuity to check

- Can't rely on exhaustive testcases
- Can't rely on manually checking in a debugger

Must invent ways to reduce complex questions into easier ones

- Can we reduce the dimensionality of the problem?
  - e.g. restrict particles to a plane within the space
- Can we extract more information from the system?
  - e.g. put a movable camera into a simulation to overcome occlusion
  - e.g. project shadows onto a plane to overcome perspective
- Can we reduce the size of the problem?
  - e.g. lower the number of particles to see their motion better
- Can we use specific test scenarios for testing?

Feedback Directed Optimisation

If it helps, think of this as a game

1. Pick a move, either
   - Optimisation technique to apply
   - New parameter to try
2. Verify the code
3. If it works, profile it
4. New high score?
   - Save this version

Best Results

The more techniques you know

- The more things to try
- Hopefully the better the score

The more techniques you know

- The more things to try (geometric increase in parameterisations)
- Smaller chance of finding the best one

There is a certain art to knowing which is good and bad. . .

Where to focus effort

Trying different parameters is mechanical

- Don’t do repetetive things — script them

Writing different optimisations is tricky

- Needs human input

Software Engineering teaches people to write generic code

- Reuses prevents much of the rewriting
- Generic code is slow
- Writing slow code is a bad way to increase performance

Standard tricks

Code is just text

- Can be used in more than one program
- Allows a test-harness to be used around code for verification
- Different program can be used to profile the target code

What can go wrong?

- Executing code in a different context can produce different behaviour
- How do you know that the code behaves the same in both cases?

Similarities

Diagnostic techniques work because...

- Chain of reasoning
- If I simplify part X then part Y is preserved
- If I need to check part Y then this is sound

Profiling code out of context...

- Should follow a similar chain
- If I replace part X then part Y is preserved
- If it is part Y being profiled this is sound...

Actual line of reasoning is up to you