

Final Review

ICS432 - Fall 2008 Concurrent and High-Performance Programming

Henri Casanova (henric@hawaii.edu)

What Questions on the Final?

- Three categories of questions
 1. Some questions going back to the midterm
 2. *Quite a few* “what is this?”, “explain that” questions about concepts
 - in lecture notes starting with the “Transaction Memory” notes
 3. Some actual exercises about concepts/techniques
 - in lecture notes starting with the “Transaction Memory” notes
- Let’s review possibilities for the above three categories

Midterm-like Questions

(Just tell me if you want to receive midterm solutions)

- The “what do threads share?” question!
- One “look at code and see what’s wrong questions”
 - Pthread or Java
- One “write a simple piece of code” questions
 - Pseudo-code or Java
- If you’ve done the homework assignments, and were able to do similar questions on the midterm, these should be no problem whatsoever

“Concept” Questions

Questions like: “what is XXX?”, “explain how XXX works”, “what is the motivation for XXX?”, “what is a drawback of XXX?”, etc

- Transaction Memory
- Pipelining
- Vector instructions
- Multi-threaded architectures (Hyperthreading, etc.)
- Moore’s Law
- Shared/Private caches
- Cache coherence
- Ways to time code
- Speedup/Efficiency
- The memory bottleneck, locality, row-major/column-major
- Amdahl’s law
- Profiler
- Types of shared-memory programs
- Load-balancing
- Scheduling notions, particularly in OpenMP
- Code optimization
- Parallel computing (methods, Top500)
- Large-scale volunteer computing (SETI@home)
- Peer-to-peer systems

Sample “Concept” Questions

- Why do people think that transaction memory is a good thing?
- Give two examples of where pipelining may occur when a program runs on a computer
- What’s the idea behind vector instructions?
- Why is Hyperthreading a good idea?
- Is Moore’s Law still true?
- Are L1 caches typically shared among cores?
- What is “memory bus snooping” used for?
- What is the definition of parallel efficiency?
- What is the technological solution to slow main memory?
- What are the two kinds of locality?
- State Amdahl’s law
- Why is a profiler useful?
- Give an example of a program that would require load balancing among threads
- What is loop unrolling?
- What type of supercomputers are most common today in the Top500 list?
- What about the SETI application make it amenable to volunteer computing?
- How is content found in unstructured p2p systems?

- Parallel Speedup/Eff
- The Memory hierarchy
 - Matrix multiplication
 - Counting cache misses
- OpenMP Scheduling
- Program Optimization

Concurrency and Performance

- The metric for parallel/concurrent performance: the **parallel speedup**
speedup = (sequential exec time) / (parallel exec time)
- Typically one doesn't know how to parallelize the entire program
- There is only a fraction, say f , that can be parallelize
 - Sometimes it's $(1-f)$ that can be parallelize, depending on people/exercise
- When reasoning about the parallel speedup, one typically assumes *perfect* parallelization
 - If the parallelizable portion of the code takes time T , the the parallelized same portion takes time T/p on p cores.
- The metric used to see how well the processors are utilized is the **parallel efficiency**
efficiency = (speedup) / (number of cores)

Amdahl's Law

- Consider a program that runs in time T on 1 core
- Consider that one knows how parallelize a fraction f of that time
- Then, the execution on p cores is:
$$(1-f) T + f T / p$$
- Therefore, the speedup is:
$$T / [(1-f) T + f T / p]$$

$$= 1 / [(1-f) + f / p]$$
 (an upper bound is $1/f$)
- And the parallel efficiency is:
$$1 / [(1-f) p + f]$$

How to Remember Amdahl's Law?

- speedup = $1 / (1 - f + f/p)$
 - 1: sequential time
 - $-f$: I remove the non-parallelizable portion of the sequential time
 - $+f/p$: I replace it by its (perfectly) parallelized version

Sample Exercises

- Consider a sequential program. I know how to parallelize 70% of it. I have a 32-core machine. What is the speedup I can hope to achieve?
- What fraction of a program should be parallelizable so that it can run at >50% efficiency on 10 cores?
- I want to run a program at 80% efficiency, knowing that I can parallelize 90% of it. What is the maximum number of cores I can use? And what is the speedup?

Programming for the Memory Hierarchy

- Row-major vs. Column-major
 - When accessing elements row-wise for a 2-D array stored in row-major, one incurs a cache miss every (cache line size) / (element size) memory access
 - When accessing elements column-wise for a 2-D array stored in column-major, one incurs a cache miss for every memory access
- Loop ordering for the matrix-multiplication
 - Three kinds of accesses
 - constant, sequential, strided
 - great, ok, bad
 - see lecture slides about the optimal loop ordering for matrix multiplication
 - should we look at them now?

Sample Exercise

- Consider the following code:

```
int A[100][100];
for (i=0; i<100; i++)
  for (j=0; j<100; j++)
    a[i][j] = 1;
```
- How many cache misses if the cache line size is 64 bytes?

Scheduling

- You should know the three OpenMP scheduling modes of course
- If all iterations of a loop take the same time, do we go faster with `schedule(static)` or `schedule(dynamic)`?
- Why may one prefer `schedule(guided)` to `schedule(dynamic)`?
- Why does OpenMP let us specify a chunk size?

That's it

- The final is on Thursday 12/18 at **2:15PM**
- You should take a calculator for the exam
 - for speedup computations
- any other questions?