Program Optimization for Multi-core Architectures

Sanjeev K Aggarwal (ska@iitk.ac.in)
M Chaudhuri (mainak@iitk.ac.in)
R Moona (moona@iitk.ac.in)

Department of Computer Science and Engineering,
IIT Kanpur - 208016
India

Acknowledgement: This course has been developed with support from Intel Semiconductors (US) Limited
Motivation for doing this course

• Processors and computer systems are becoming more and more powerful
• Applications are becoming more and more demanding
• Important applications:
  – Engineering simulations (FM, CFD, structures)
  – Biology (Genetics, cell structure, molecular biology)
  – Particle and nuclear physics, high energy physics, astrophysics, weather prediction, drug design, tomography, molecular dynamics …
  – Entertainment industry
Motivation …

• These application require too much compute power
• Compute power is available!
• However, software systems (algorithms, compilers, libraries, debuggers) AND programmers (mainly trained in sequential programming) are unable to exploit the compute power
Motivation …

- Solving these complex problems and programming these architectures require different methodology
- Better algorithms, compilers, libraries, profilers, application tuners, debuggers etc. have to be designed
- Programmers have to be trained to use these tools
High Performance Systems

• Power of high performance systems come from
  – Hardware technology: faster machines, more cache, low latencies between devices
  – Multilevel architectural parallelism
    • Pipelines: out of order execution
    • Vector: handles arrays with a single instruction
    • Parallel: lot of processors each capable of executing independent instruction stream
    • VLIW: handles many instructions in a single cycle
    • Clusters: large number of systems on a very fast network
  – Grid: cooperation of a large number of systems
Processor design problems

- Processor frequency and power consumption seem to be scaling in lockstep
- How can machines stay on historic performance curve without burning down the system!?
- Moore’s Law is still applicable
- Physics and chemistry at the nano-scale:
  - Materials
  - Transistor leakage current, quantum effect coming into play
- Wire lengths have to be reduced
Processor design problems …

• For 200 MHz frequency steps two steps back on frequency cuts power consumption by \(\sim 40\%\)

• Same thermal envelope for dual core running at \(n-2\) as the single processor running at \(n\)
Sources and types of Parallelism

• Structured: identical tasks on different data sets
• Unstructured: different data streams and different instructions
• Algorithm level: appropriate algorithms and data structures
• Programming:
  – specify parallelism in parallel languages
  – write sequential code and use compilers
  – use course grain parallelism: independent modules
  – use medium grain parallelism: loop level
  – use fine grain parallelism: basic block or statement
• Expressing parallelism in programs
  – no good languages
  – most applications are not multi-threaded
  – Writing multi-threaded code increases software costs
  – programmers are unable to exploit whatever little is available
What will we learn in the course?

- Processor architectures with focus on memory hierarchy, instruction level parallelism and multi-core architectures
- Program analysis techniques for redundancy removal and optimization for high performance architectures
- Concurrency and operating systems issues in using these architectures
- Programming techniques for exploiting parallelism (use of message passing libraries)
- Tools for code analysis and optimization (Intel compilers, profilers and application tuning tools)
What will we learn …

• Understand paradigms for programming high end machines, compilers and runtime systems
  – Applications requirements
  – Shared-memory programming
  – Optimistic and pessimistic parallelization
  – Memory hierarchy optimization

• Focus on software problem for **multicore processors**
What do we expect to achieve by the end of the course?

• Faculty who can teach this course and conduct research in this area

• Students who can design, develop, understand, modify/enhance, and maintain complex applications which run on high performance architectures (in addition to doing research!)

• A set of slides, notes, projects and laboratory exercises which can be used for teaching this course in future both at IITK and at other universities
Organization of the course

• Approximately 40 lectures of one hour each (both by faculty and students)

• One term paper/project to be done individually. It is important to start early. (30% credit)

• Some small programming assignments for laboratory (20% credit)

• One mid semester examination (20% credit)

• One end semester examination (30% credit)

• Every one is expected to participate in the discussions in the class
Ethical Issues

• Copying material from internet and other sources
  – DO NOT use “cut and paste” technology to prepare your reports
  – DO NOT copy assignments
  – You can borrow ideas (after giving due credit) but not the text/programs
  – Look at the word Plagiarism in www.dictionary.com
    “a piece of writing that has been copied from someone else and is presented as being your own work”
Ethical Issue …

– Look at the word Plagiarism in http://en.wikipedia.org/wiki/Plagiarism

“Plagiarism is a form of cheating, and within academia is seen as academic dishonesty. It is a matter of deceit: fooling a reader into believing that certain written material is original when it is not. Plagiarism is a serious and punishable academic offense, when the goal is to obtain some sort of personal academic credit or personal recognition”
Background required

• Knowledge of basic compiler, operating systems and computer organization

• Focus on back-end of the compiler and concurrency issues in operating systems

• Computer Organization
  – Computer Organization and Design by Patterson and Hennessy

• Compiler reference: Dragon book
  – Compilers: Principles, Tools and Techniques by Aho, Sethi and Ullman

• Operating Systems reference
  – Operating Systems Concepts by Silberschatz and Galvin
References

- No specific text book!!

- Material has been collected from various sources like books, research papers, position papers etc.

- We will make the material available as we go along

- Our slides and class notes will be useful material
Some useful references

• **ACKNOWLEDGEMENT**: Some of the figures have been taken from the Wolfe's book on High Performance Compilers for Parallel Computing.

• Material on OpenMP are the tutorials given by Tim Mattson (Intel) and Rudolf Eigenmann (Purdue University) at Super Computing 2001.

• **Computer Architecture**
Some useful References …

• **Redundancy removal**

• **High Performance Compilers, Data Dependence Analysis**

• **Operating Systems**
  – Coulouris, Dollimore and Kindberg Distributed Systems Concept and Design, Addison-Wesley.
  – Silberschatz, Galvin, Operating Systems Principles, Addison-Wesley