Lecture 2: Distributed and Parallel Systems

CS178: Programming Parallel and Distributed Systems

January 29, 2001
Steven P. Reiss

I. Motivation for D&P Systems

A. Comes from applications

1. Complex applications require complex machines

a) Scientific applications require peta-flop +

b) Solving NP-complete problems

c) Analyzing massive amounts of data

2. Some applications are inherently distributed

a) Web-based applications (web stores)

b) Email, news, electronic conferencing, ...

c) Physician billing

d) Multiplayer games

3. Some applications are easier to build in components

a) Programming environment

(1) Separate debugger, compiler, editor, ...

b) Web store

(1) Inventory system (what is in stock; descriptions; ...)

(2) Order management system (transactions for orders, keeping track of orders, shipping, etc.)

(3) Billing system (credit card authorization, sending out bills, tracking POs, ...)

(4) Personal preferences

(5) Actual customer interface

B. Comes from hardware limitations

1. Physical limits to a single CPU

a) Speed of light and access to data

b) Energy consumption and heat dissipation

c) Switching time for a single transistor

II. What are the Problems

A. Data-oriented problems

1. Most of the problems involving these applications are data-oriented

2. How to get data to the processors

a) Organizing the problem

b) Organizing the input

c) Organizing the computation

3. How to combine resultant data from the processors

4. Data costs often make computation virtually free

B. Control-oriented problems

1. Coordination

a) Ensuring different parts of the system are in sync

b) Making sure that different parts don't trample on each other

2. Synchronization

a) How to get different processors/processes in sync

b) Techniques for sharing data and computation

c) Techniques for sharing control and processors

C. Algorithmic problems

1. What is different

a) Emphasis on data (input/output complexity)

b) Planning on things being done independently or in parallel

2. Techniques that are used

D. Program understanding

1. Parallel and distributed systems are nondeterministic

a) Can't tell a priori what order things will occur in

b) Hard to comprehend what things might be possible

2. **Difficult to conceive what is happening when hundreds/thousands things go on at once**

3. **Behavior can be non-intuitive**

    a) Adding processors/threads slows down the system

4. **Difficult to understand what is happening and why**

    a) Tools are fairly primitive and don't always convey the right information

III.What are the Solutions

A. Data-oriented problems (algorithmic solutions)

1. **Iterative Parallelism**

    a) Some programs are naturally parallel

    b) Matrix multiplication

2. **Recursive Parallelism**

    a) Others can be made so

    b) Adaptive quadrature

3. **Others require more work**

    a) Lots in parallel, but some interaction as well

    b) Various techniques can be used here

4. **Inherently distributed**

    a) Client-server applications

    b) Web-based applications

    c) Still need to understand what goes where

B. Data-oriented solutions

1. **Inline parallelism**

    a) Pipelined CPUs and architectures

    b) Vector machines

    c) SIMD machines -- single control unit, multiple data paths

2. **Multiple threads and shared memory**

    a) Relatively easy to program

    b) Common data available to all

    c) Need to worry about synchronization

d) Scales to hundreds of processors, but no more

3. Network shared memory

a) Techniques for doing shared memory across multiple machines

b) DSM vs DSO

4. Message passing among processes

a) Need message primitives (send/receive)

b) Synchronous vs. asynchronous send/receive

c) Polling vs interrupt

d) Combine with threads

5. Remote Procedure Call

a) Rather than sending messages, make procedure calls to that execute in another process

b) Good for client/server computation

c) This is part of the .Net framework

d) Calls can be synchronous or asynchronous

6. Remote Objects

a) This can be extended to having objects exist in one process and making method calls on them there

b) Other processes just have handles to these objects

c) Techniques for duplication, sharing can be added

C. Control-Oriented Solutions

1. Synchronization constructs

a) Semaphores, mutexes, condition variables, read-write locks

b) All roughly equivalent

c) All provide means for controlling the interaction among processors

2. Language features

a) Monitors -- modula 2, modula 3, Java

b) Rendezvous -- ada

c) Message-based -- NIL

D. Understanding Solutions

1. **Multithreaded debuggers**
2. **Multiple processor profiling**
3. **Processor utilization and status recording over time**
4. **Better tools**

IV. Sample Problem

A. This is best understood by looking at sample problems

1. **This is what we will be doing throughout the course**
2. **Solutions are always easiest to understand in the concrete**

B. Finding collocations

1. **Web base (20G+ of web pages)**
2. **Want to find what strings constitute phrases**
   - a) f(A B) compared to f(A) and f(B) statistically
   - b) Need to get counts for all words and sequences
   - c) Up to length 7.

C. Simple approach

1. **Keep hash table of words, phrases in memory**
2. **Go through source, increment counts for each word and phrase as a word is read**
3. **Access counts at the end to determine which phrases are valid ones.**
4. **Problems**
   - a) Can’t fit any of the hash tables in memory
   - b) Lots of file I/O processing needs to be done

D. How might we make this run faster

1. **Algorithmic approaches**
   - a) Iterative/recursive parallelism
   - b) Message passing approaches
2. **How to organize the data**
   - a) Initial data organization

b) Key is the intermediate data

3. **How to merge results and use them**

V. Next Time

A. Read Chapter 2 of Andrews

B. We will cover mainly hardware for multithreaded programming

C. Start covering software solutions