Outline

- Introduction
  - what is parallel computing?
  - why you should care?
- Administrative stuff
  - course organization
  - workload and grading
- Inevitability of parallel computing
  - application demands
  - technology and architectural trends
  - economics
- Convergence of parallel architectures
  - shared address space, message passing, data parallel, data flow, systolic
  - a generic parallel architecture

[ Culler/Singh/Gupta: Chapter 1 ]

9/5/2001

What is Parallel Computing?

“A collection of processing elements that can communicate and cooperate to solve large problems fast”

Almasi/Gottlieb

- “Communicate and cooperate”
  - node and interconnect architecture
  - problem partitioning and orchestration
- “Large problems fast”
  - programming model
  - match of model/architecture
- Focus of this course:
  - parallel programming models
  - parallel architectures
  - interaction between models and architectures

9/5/2001

Why Study Parallel Computing?

- Inevitability of parallel computing
  - fueled by application demand for performance
    - scientific: weather forecasting, pharmaceutical design, genomics
    - commercial: OLTP, decision support, data mining
    - scalable web servers
  - enabled by technology and architecture trends
    - limits to sequential CPU, memory, storage performance
      - parallelism is an effective way of utilizing growing transistor budgets
    - low incremental cost of supporting parallelism
- Convergence of parallel computer organizations
  - driven by technology constraints and economies of scale
    - laptops and supercomputers share the same building block
    - growing consensus on fundamental principles and design tradeoffs
    - naming, ordering, replication, communication

- Parallelism has become central and mainstream

9/5/2001
Course Outcomes

- Parallel architectures
  Q: which are the dominant organizations?
  A: small-scale shared memory (SMPs), large-scale distributed memory, and large-scale shared memory

- Programming models
  Q: how to program these architectures?
  A: data parallel, message passing, shared memory, and hybrid models

- Programming for performance
  Q: how are programming models mapped to the underlying architecture, and how can this mapping be exploited for performance?

- Target audience
  - parallel systems researchers and developers
  - computer architects
  - application developers/computational scientists

Course Syllabus

Lecture 1: Introduction
Lectures 2-3: Parallel programs and programming models
  - data parallel, message passing, shared memory
Lectures 4-7: Parallel architectures
  - small scale shared-memory
  - large-scale distributed memory
  - large-scale shared memory
Lectures 8-11: Programming for performance
  - understanding mapping between programming models and architectures
Lecture 12: Future directions

- Assumed background
  - computer architecture: instruction execution, system organization, caches
  - operating systems: processes, virtual memory, scheduling

Course Resources

- Text books (recommended)

- Supercomputing Accounts
  - National Center for Supercomputing Applications (NCSA)
    - SGI Cray Origin 2000
      - 128 nodes, large-scale distributed and shared memory
  - New York University
    - 10-12 node Beowulf PC cluster
    - more details next week

Workload and Grading

- Classes and assigned readings
- Five homeworks: 60%
  - HW1: systems issues
  - HW2: SMP architectures, and thread programming
  - HW3: distributed memory architectures, and message-passing
  - HW4: shared memory architectures, and shared-memory programming
  - HW5: cluster architectures, and put/get programming
- Course project: 40%
  - should be done in groups of 2-3
  - three components
    - proposal (2 pages) (due October 10)
    - 1st part of report (+5 pages) (due November 28)
    - final report (+15 pages) (due December 12)
  - guidelines and sample topics in handout
Outline

- Introduction
  - what is parallel computing?
  - why you should care?
- Administrivia
  - course organization
  - workload and grading

Inevitability of parallel computing
- application demands
- technology and architectural trends
- economics
- Convergence of parallel architectures
  - shared address space, message passing, data parallel, data flow, systolic
  - a generic parallel architecture

Inevitability of Parallel Computing

Three driving forces (both justify and influence parallel computing)
- Application demand for performance
  - scientific, commercial, scalable servers
- Technology and architecture trends
  - thread-level parallelism permits better use of on-chip transistors
- Economics
  - low incremental cost of supporting parallelism

Current situation
- Today’s microprocessors have multiprocessor support
- Multiprocessor servers and workstations
  - IBM, Sun, SGI, HP/Compaq/Digital
- Tomorrow’s microprocessors will be multiprocessors

Application Demands: Scientific Computing

- Large parallel machines a mainstay in many industries
  - Petroleum
    - reservoir analysis
  - Automotive
    - crash simulation, drag analysis, combustion efficiency
  - Aeronautics
    - airflow analysis, engine efficiency, structural mechanics, electromagnetism
  - Computer-aided design
  - Pharmaceuticals
    - molecular modeling
  - Genome analysis
    - (Celera has largest parallel machine outside National labs)
  - Visualization
    - entertainment (animation and special effects: Toy Story, Star Wars, …)
      - architecture (walk-throughs and rendering)
  - Financial modeling
    - yield and derivative analysis

Parallel processing required for (physically) meaningful results
Application Demand and Hardware Capability

- Example: Speech and image processing

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GIPS</td>
<td>1,000 Words Continuous Speech Recognition</td>
<td>ISDN-CD Stereo Receiver</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 GIPS</td>
<td>5,000 Words Continuous Speech Recognition</td>
<td>HDTV Receiver</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 MIPS</td>
<td>Sub-Band Speech Coding</td>
<td>Speaker Verification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 MIPS</td>
<td>200 Words Isolated Speech Recognition</td>
<td>CELP Speech Coding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 MIPS</td>
<td>Telephone Number Recognition</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Application demands fuel hardware advances, and vice-versa

Application Demands: Commercial Computing

- Widespread use of parallelism
  - however, two main differences
    - on a smaller scale (number of processors)
    - loosely coupled
  - computational power determines business scale
- Parallelism benefits several applications
  - online-transaction processing, decision support,
  - data mining, data warehousing
  - web and application servers
    - Google runs on an 8000-node Linux cluster! (as of April 2001)
- Transaction Processing Performance Council (TPC) benchmarks
  - TPC-C order entry, TPC-D decision support
  - explicit scaling criteria: enterprise size scales with system size
  - throughput metric: transactions per minute (tpm)

Summary of Application Trends

- Scientific/Engineering computing
  - transition to parallel computing has occurred
  - application demand drives hardware capabilities, and vice-versa
- Commercial computing
  - widespread use of moderate-size, loosely-coupled systems
- General-purpose (sequential) computing
  - multiprocessor servers and workstations for throughput
  - increasing use of multithreaded programs
- Solid application demand exists and will increase
  - for both small- and large-scale parallel systems

TPC-C Results for March 1996

- Parallelism is pervasive
- Small to moderate scale parallelism very important
Technology Trends

- Exponential growth in commodity uniprocessor performance
  - transistor count: 2x every 3 years, DRAM size: 4x every 3 years

  ➤ Natural building block for multiprocessors is now also the fastest!

Technology Trends: A Closer Look

- Basic advance is decreasing VLSI feature size ($\lambda$)
  - clock rate improves as $\lambda$, transistor count as $\lambda^2$ (or faster)

  ➤ Parallelism likely to contribute more to performance improvements

Technology Trends: Similar Story for Storage

- Divergence between memory capacity and speed more pronounced
  - capacity increased by 1000x from 1980-95, speed only 2x
  - larger memories are slower, while processors get faster
    - need to transfer more data in parallel
    - need deeper cache hierarchies

- How to organize caches?
  - parallelism increases effective size of each level of hierarchy, without increasing access time

- Parallelism within memory systems too
  - new designs fetch many bits within memory chip; follow with fast pipelined transfer across narrower interface

Architectural Trends

- Architecture translates technology into performance
  - tradeoff: parallelism versus locality
    - currently: 1/3 compute, 1/3 cache, 1/3 off-chip connect

- Four generations: tube, transistor, IC, VLSI

- Greatest trend in VLSI is an increase in the exploited parallelism
  - up to 1985: bit level parallelism (4-bit -> 8 bit -> 16-bit -> 32-bit)
    - great inflection point when 32-bit micro and cache fit on a chip
    - gains are slowing: 64-bit adoption under way, 128-bit is far away
  - 1985 - 1995: instruction level parallelism
    - pipelining, RISC (simple instruction sets + compiler advances)
    - superscalar execution: on-chip caches and functional units
    - greater sophistication: out of order execution, speculation, predication
  - next step: thread level parallelism?
Phases in VLSI Generation

Limits of Instruction-level Parallelism (ILP)

- Reported speedups for superscalar processors: 1.37 - 17+
  - function of application domain (numerical versus non-numerical)
  - capabilities of processor modeled

Recent research shows need to look across threads

Architectural Trends: System Design

- Micro on a chip makes it natural to connect many to shared memory
  - bus-based multiprocessors dominate server and enterprise market

Economics

- Commodity microprocessors not only fast but CHEAP
  - development cost ($5-100M) amortized over volumes of millions
  - building block offers significant cost-performance benefits

- Current situation: Multiprocessors are being pushed by software vendors as well as hardware vendors
  - standardization by Intel makes small, bus-based SMPs commodity
  - soon multiprocessing on the desktop

Parallel computing is becoming increasingly mainstream!

- Example: How economics affects platforms for scientific computing
  - large scale multiprocessors replace vector supercomputers
  - consequence: a supercomputer and a laptop share the same building block
LINPACK Parallel Performance

![Graph showing LINPACK performance over time](image)

Summary: Why Parallel Computing?

- Increasingly attractive
  - economics, technology, architecture, application demand
- Increasingly central and mainstream
- Parallelism exploited at many levels
  - instruction-level parallelism
  - multiprocessor servers
  - large-scale multiprocessors (“MPPs”)
- Focus of this class: multiprocessor level of parallelism
  - wide range of architectures make sense (cost, performance, scalability)
- Same story from memory system perspective
  - increase bandwidth, reduce average latency with many local memories

Outline

- Introduction
  - what is parallel computing?
  - why you should care?
- Administrivia
  - course organization
  - workload and grading
- Inevitability of parallel computing
  - application demands
  - technology and architectural trends
  - economics
  - Convergence of parallel architectures
    - shared address space, message passing, data parallel, data flow, systolic
    - a generic parallel architecture

{Culler/Singh/Gupta: Chapter 1}

Then and Now

- Historically, parallel architectures tied to programming models
  - divergent architectures, with no predictable pattern of growth
- Today: Consensus on a layered communication architecture
  - node architecture + support for communication and cooperation
Communication abstraction: primitives for implementing the model
- supported directly by HW, by the OS, or by user-level software
- today: convergence in organizations
  - HW/SW interface is relatively flat
  - compiler/software plays important bridging role

• Programming models specify communication and synchronization
  - multiprogramming: no communication/synchronization
  - shared address space: bulletin board
  - message passing: like letters or phone calls, explicit point to point
  - data parallel: more regimented, global actions on data

• Rationale
  - historically: machines tailored to programming models
    - architecture = prog. model + comm. abstraction + machine organization
  - now: separation of programming models and architectures
  - tracing the evolution helps identify core concepts, understand convergence

• Five architectures of interest
  - dominant: shared address space, message passing, data parallel
  - others: dataflow, systolic arrays

• For each, let us examine
  - programming model
  - motivation
  - intended applications
  - contributions to convergence
Shared Address Space Architectures

- Programming model
  - process: virtual address space plus one or more threads of control
  - portions of address spaces of processes are shared
  - writes to shared address visible to all threads (in other processes as well)

Virtual address spaces for a collection of processes communicating via shared addresses

Machine physical address space

Load

Store

Common physical addresses

P0

P1

P2

Pn

Private portion of address space

Shared portion of address space

Shared Address Space Architectures (contd.)

- Motivation: Programming convenience
  - location transparency
    - communication is implicitly initiated by loads and stores
  - similar programming model to time-sharing on uniprocessors

- Communication hardware also natural extension of uniprocessor
  - addition of processors similar to memory modules, I/O controllers

Evolution: Four Organizations

- Mainframes
  - motivated by multiprogramming
  - extends crossbar for memory modules and I/O
    - initially, limited by processor cost
    - later, by cost of crossbar
  - high incremental cost
  - e.g., IBM S/390

- Minicomputers (SMPs)
  - motivated by multiprogramming, TP
  - all components on a shared bus
    - latency larger than for uniprocessor
    - bus is bandwidth bottleneck
  - caching is key: coherence problem
  - low incremental cost

Example of an SMP: Intel Pentium Pro Quad

- All coherence and multiprocessing glue in processor module
- Highly integrated, targeted at high volume
- Low latency and bandwidth
Evolution: Four Organizations (contd.)

- Dance Hall
  - problem: interconnect cost (crossbar), or bandwidth (bus)
  - solution: scalable interconnection network
    - bandwidth scalable
    - however, larger access latencies
    - caching is key: coherence problem
  - e.g., NYU Ultracomputer

- Distributed Memory (NUMA)
  - message transactions across a general-purpose network
    - e.g. read-request, read-response
  - caching of non-local data is key
  - coherence costs
  - e.g., Cray T3E, Origin 2000

Example of a NUMA: Cray T3E

- Scales up to 1024 processors, 480MB/s links
- Nonlocal references accessed using communication requests
  - generated automatically by the memory controller
  - no hardware coherence mechanism (unlike SGI Origin)

Message Passing Architectures

- Programming model
  - directly access only private address space (local memory), communicate via explicit messages (send/receive)
  - in simplest form, achieves pair-wise synchronization
    - model is removed from basic hardware operations
    - library or OS intervention for copying, buffer management, protection

Message Passing Architectures

- Complete computer as building block, including I/O
  - communication via explicit I/O operations

- High-level block diagram similar to distributed-memory shared address space machines
  - but communication integrated at I/O level, needn’t be into memory system
  - like networks of workstations (clusters), but tighter integration
  - easier to build than scalable shared address space machines
Example of a Message Passing Machine: IBM SP-2

- Made out of essentially complete RS6000 workstations
- Network interface integrated in I/O bus (bw limited by I/O bus)

Evolution of Message-Passing Machines

- Early machines: FIFO on each link
  - hw close to programming model
  - synchronous operations
  - replaced by DMA
    - enables non-blocking ops
    - buffered by system at destination
- Now: diminishing role of topology
  - topology important for store-and-forward routing
  - introduction of pipelined (cut-through) routing made it less so
    - Virtual cut through, wormhole routing
    - cost is in node-network interface

Toward Architectural Convergence

- Evolution and role of software have blurred boundary between programming models
  - send/recv on shared address space (SAS) architectures: buffers
  - global address space on message passing (MP) architectures
    - hashing and page-based (or finer-grained) shared virtual memory
- Convergence in hardware organizations as well
  - tighter NI integration even for MP
  - hardware SAS passes messages at lower level
  - even clusters of workstations/SMPs are parallel systems
    - emergence of fast system area networks (SAN)

Data Parallel Systems

- Programming model
  - operations performed in parallel on each element of data structure
  - logically single thread of control, performs sequential or parallel steps
    - conceptually, a processor associated with each data element
- Architectural model
  - array of many simple cheap processors, each with little memory
  - a control processor issues instructions
  - specialized and general communication, cheap global synchronization
- Original motivations
  - matches simple differential equation solvers
  - centralize high cost of instruction fetch/sequencing
Applications of Data Parallelism

- Each PE contains an employee record with his/her salary
  
  ```
  if salary > 100K then
      salary = salary *1.05
  else
      salary = salary *1.10
  
  logically, the whole operation is a single step
  - some processors enabled for arithmetic operation, others disabled
  ```

- Other examples
  - finite differences, linear algebra
  - document searching, graphics, image processing

- Popular architecture in the late 1980s and early 1990s:
  - Thinking Machines CM-1, CM-2 (and CM-5)
  - Maspar MP-1 and MP-2

Evolution and Convergence

- Rigid control structure popular in the 1960s
  - cost savings of centralized sequencer high
  - Flynn taxonomy: SIMD (SISD: uniprocessor, MIMD: multiprocessor)

- Replaced by vectors in mid-70s
  - more flexible w.r.t. memory layout and easier to manage

- Revived in mid-80s
  - when only 32-bit datapath slices would fit on chip

- Other reasons for demise
  - simple, regular applications have good locality, can do well anyway
  - hardwiring data parallelism limits applications

- Lasting contributions
  - programming model converges with SPMD (single program multiple data)
  - need for fast global synchronization, structured global address space

Dataflow Architectures

- Represent computation as a graph of essential dependences
  - logical processor at each node, activated by availability of operands
    - message (tokens) carrying tag of next instruction sent to next processor
    - tag compared with others in matching store; match fires execution

Evolution and Convergence

- Characteristics
  - ability to name operations, synchronization, dynamic scheduling

- Problems
  - operations have locality: useful to group together
  - complexity of matching store and memory units
  - too much parallelism (?)

- Converged to use conventional processors and memory
  - differences
    - support for large, dynamic set of threads to map to processors
    - integration of communication and thread generation

- Eventually, separation of programming model from hardware

- Lasting contributions:
  - tightly integrated communication and fine-grained synchronization
  - remains useful concept for software (compilers etc.)
Systolic Architectures

- Replace single processor with array of regular processing elements
  - orchestrate data flow for high throughput with less memory access

- Differences from other organizations
  - pipelining: nonlinear array structure, multidirectional data flow, each PE may have (small) local instruction and data memory
  - SIMD: each PE may do something different

- Original motivations
  - VLSI enables inexpensive special-purpose chips
  - represent algorithms directly by chips connected in regular pattern

Example realization: iWARP
- use quite general processors: variety of algorithms on same hardware
  - but dedicated interconnect channels: register-to-register data transfer

Specialized, and ran into same problems as SIMD
- general purpose systems work well for same algorithms (locality etc.)
- current day manifestation: Embedded digital signal processors (DSPs)

Systolic Architectures (contd.)

\[ y(i) = w_1 \times x(i) + w_2 \times x(i + 1) + w_3 \times x(i + 2) + w_4 \times x(i + 3) \]

Convergence: Generic Parallel Architecture

- A generic modern multiprocessor

Node: processor(s), memory system, plus communication assist
- network interface and communication controller

Scalable network

Convergence allows lots of innovation, now within framework
- integration of assist with node, what operations, how efficiently...

Lecture Summary

- Parallel computing
  A collection of processing elements that can communicate and cooperate to solve large problems fast

- Parallel computing has become central and mainstream
  - application demands
  - technology and architectural trends
  - economics

- Convergence in parallel architectures
  - initially: close coupling of programming model and architecture
    - shared address space, message passing, data parallel, data flow, systolic
  - now: separation and identification of dominant models/architectures
    - programming models: data parallel, message passing, shared memory
    - architectures: small-scale shared memory, large-scale distributed memory, large-scale shared memory
Next Lecture

- Parallel programs
  - common issues
    - decomposition, assignment, orchestration, mapping
  - case studies
    - Ocean, Barnes, Ray Tracing, Data Mining
  - interplay between components of a parallel system
    - architecture, OS and compilers, programming models, applications

Readings
- Culler/Singh/Gupta: Chapter 2
- Andrews: Chapter 1