Multi-core Programming - Introduction

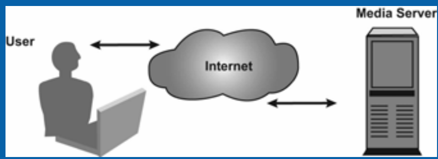
Based on slides from Intel Software College
and
Multi-Core Programming –
increasing performance through software multi-threading
by Shameem Akhter and Jason Roberts,

- *"We will go from putting Hyper-Threading Technology in our products to bringing dual core capability in our mainstream client microprocessors over time. For the software developers out there, you need to assume that threading is pervasive."*

Paul Otellini
Chief Executive Officer
Intel Developer Forum, Fall 2003

Concurrency – in everyday use

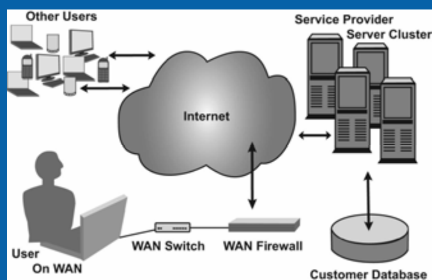
- User watching streaming video on a laptop in hotel room
- Simplistic user view – just like watching broadcast TV



- Reality
  - PC must download streaming video data, decompress/decode it, display it on the screen, must also handle streaming audio and send to soundcard
  - OS may be doing system tasks
  - Server must receive the broadcast, encode/compress it in near real-time, send it to possibly thousands of users.

Reality of Streaming Video

- Requires managing many independent subsystems in parallel
  - Job may be decomposed into tasks that handle different parts
  - Concurrency is what permits efficient use of system resources to maximize performance
  - Concurrency – abstraction for implementing naturally parallel applications



Concurrency in sequential systems !!

- Streaming Video
  - While waiting to receive a frame, decode previous frame
- FTP server
  - Create a task (thread) for each user that connects
  - Much simpler and easier to maintain

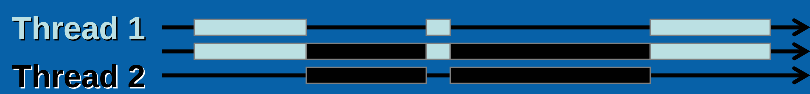
Concurrency v Parallelism

- Parallel
  - Multiple jobs (threads) are running simultaneously on different hardware resources or processing elements (PEs)
  - Each can execute and make progress at the same time
  - Each PE can execute an instruction from a different thread simultaneously
- Concurrency
  - We often say multiple threads or processors are running on the same PE or CPU at the same time
  - But this means that the execution of the threads are interleaved in time
  - A single PE is only executing an instruction from a single thread at any particular time
- To have parallelism concurrency must use multiple hardware resources

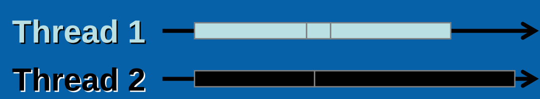
Concurrency vs. Parallelism

– Concurrency: two or more threads are <u>in progress</u> at the same time:

•



– Parallelism: two or more threads are executing at the same time



– Multiple cores needed

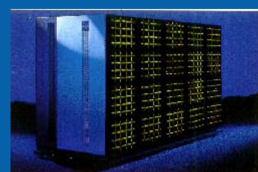
Multiprocessing v Multitasking

- **Multiprocessing** is the use of two or more central processing units (CPUs) within a single computer system
- **Multitasking** is the apparent simultaneous performance of two or more tasks by a computer's CPU

Bleeding Edge of Computer Architecture

In the 1980's, it was a Vector SMP.

Custom components throughout



In the 1990's, it was a massively parallel computer.

COTS CPUs, everything else custom

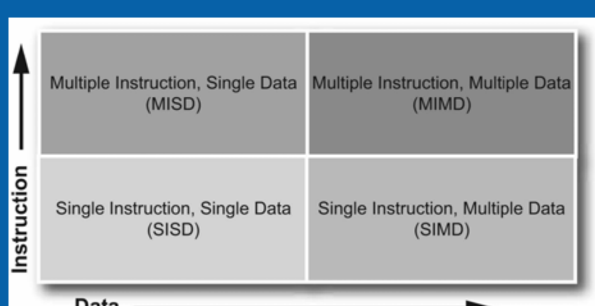
… mid to late 1990's, clusters.

COTS components everywhere

Flynn's Taxonomy of Parallel Computers 1972

Classify by two dimensions

- Instruction streams
- Data streams



Flynn's Taxonomy of Parallel Computers 1972

- SISD – single instruction, single data
  - Traditional sequential computers
  - Instructions executed in serial manner
- MISD – multiple instruction, single data
  - More a theoretical model

Flynn's Taxonomy of Parallel Computers 1972

- SIMD - single instruction, multiple data
  - same instruction applied to data on each of many processors
  - particularly useful for signal processing, image processing, multimedia
  - original array/vector machines
  - Almost all computers today have SIMD capabilities, e.g. MMX, SSE, SSE2, SSE3, AltiVec on PowerPC
  - provide capability to process multiple data streams in a single clock

Flynn's Taxonomy of Parallel Computers 1972

- MIMD multiple instruction, multiple data
  - execute different instruction on different data
  - Most common parallel platform today
  - Multi-core computers

Expanded Taxonomy of Parallel Architectures

- Arranged by tightness of coupling i.e. latency
- Systolic – special hardware implementation of algorithms, signal processors, FPGA
- Vector – pipelining of arithmetic operations (ALU) and memory bank accesses (Cray)
- SIMD (Associative) – Single Instruction Multiple Data, same instruction applied to data on each of many processors (CM-1, MPP, Staran, Aspro, Wavetracer)
- Dataflow – fine grained asynchronous flow control depending on data precedence constraints
- PIM (processor-in-memory) – combine memory and ALU on one circuit die. Gives high memory bandwidth and low latency

Expanded Taxonomy of Parallel Architectures

- MIMD (Multiple Instruction Multiple Data) – execute different instruction on different data
- MPP (Massively Parallel Processors)
  - Distributed memory (Intel Paragon)
  - Shared Memory w/o coherent caches (BBN Butterfly, T3E)
  - CC-NUMA [cache coherent non-uniform memory archicture] (HP Exemplar, SGI Origin 2000)
- Clusters – ensemble of commodity components connected by an interconnection network within a single administrative domain and usually in one room
- (Geographically) Distributed Systems – exploit available cycles (Grid, DSI, Entropia, SETI@home)

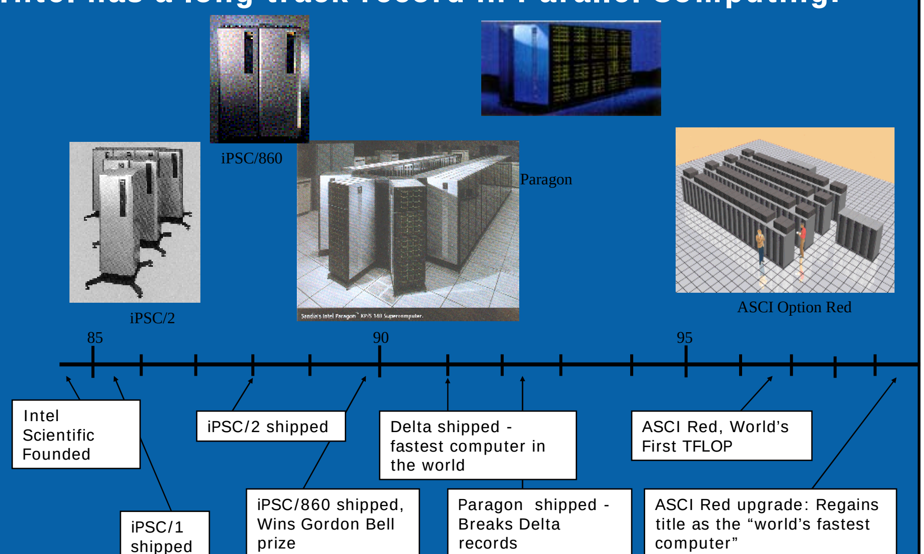
From Cray to Beowulf

- Vector computers
- Parallel Computers
  - shared memory, bus based (SGI Origin 2000)
  - distributed memory, interconnection network based (IBM SP2)
- Network of Workstations (Sun, HP, IBM, DEC) - possibly shared use
  - NOW (Berkeley), COW (Wisconsin)
- PC (Beowulf) Cluster – originally dedicated use
  - Beowulf (CESDIS, Goddard Flight Center, 1994)
  - Possibly SMP nodes

Evolution of Parallel Machines

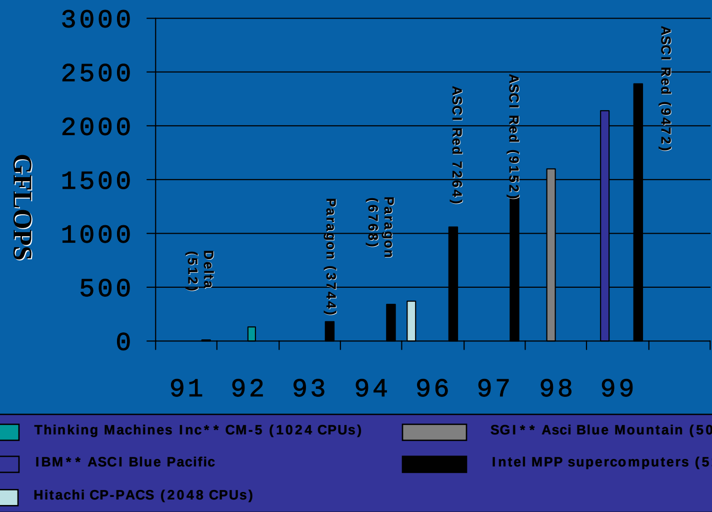
- Originally Parallel Machines had
  - custom chips (CPU), custom bus/interconnection network, custom I/O system
  - proprietary compiler or library
- More recently parallel machines have
  - custom bus/interconnection network and possibly I/O system
  - standard chips
  - standard compilers (f90) or library (MPI or OpenMP)

Intel has a long track record in Parallel Computing.

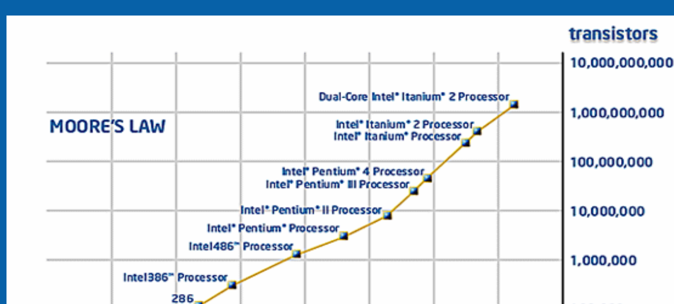


… and we were pretty good at it

We held the MP-LINPACK record* over most of the 90's



* Data from the Linpack Report, CS-89-85, April 11, 1999



- Chip complexity is not proportional to the number of transistors
- Per-transistor complexity is less in large cache arrays than in execution units
- This doesn't mean the that the performance is increasing exponential:
  e.g. PIII 500/1000: Speedup 2.3 ~ 3x Transistors ~ 2x MHz

„The complexity for minimum component costs has increased at a rate of roughly a factor of two per year ... Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer.„ Electronics Magazine 19 April 1965.

The most popular formulation: The number of transistors on integrated circuits is doubling every 12(18) months.

Mistaken Interpretation of Moore's Law

- Clock frequency will double every 18 to 24 months
  - Because clock frequency has been used metric of performance
  - For 40 years clock speed did approximately do this
  - No longer true
- Many other ways to improve performance
  - Instruction level parallelism (ILP) or dynamic out-of-order execution
    - Reorder instructions to eliminate pipeline stalls
    - Increase number of insts executed in single clock cycle
    - Hardware level parallelism invisible to programmer
  - Increase number of physical processors (Multiprocessor systems)
  - Increase the number of cores in a single chip (Chip level Multiprocessing)

Parallel computing is omnipresent (ubiquitous)

- Over the next few years, all computers will be *somehow parallel* computers.
  - Servers
  - Laptops
  - Cell phones
- What about software?
  - Herb Sutter of Microsoft said in Dr. Dobbs' Journal:
    - The free lunch is over: Fundamental Turn towards Concurrency in software
  - Performance will no longer rapidly increase from one generation to the next as hardware improves … unless the software is parallelized

Application performance will become a competitive feature for Independent Software Vendors.