Why Parallel Architecture and Programming?

Todd C. Mowry
15-418
January 11, 2011

What is Parallel Programming?

- Software with multiple threads?

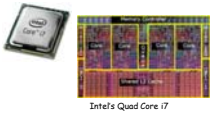
© Nintendo


© Harmonix

- Multiple threads for:
  - convenience: "concurrent programming"
  - performance: "parallel programming"

What is Parallel Architecture?

- Machines with multiple processors?


Intel's Quad Core i7

Apple MacPro


Blacklight at the PSC (4096 cores)


Hadoop cluster at Yahoo!

- "parallel" vs. "distributed" computing

One Definition of Parallel Architecture

A parallel computer is a collection of processing elements that cooperate to solve large problems fast

Some broad issues:

- Resource Allocation:
  - how large a collection?
  - how powerful are the elements?
  - how much memory?
- Data access, Communication and Synchronization
  - how do the elements cooperate and communicate?
  - how are data transmitted between processors?
  - what are the abstractions and primitives for cooperation?
- Performance and Scalability
  - how does it all translate into performance?
  - how does it scale?

Why Study Parallel Architecture and Programming?

The Answer from 10 Years Ago:

- (i.e. when the textbook was written)
- Because it allows you to achieve performance beyond what we get with CPU clock frequency scaling
  - important for applications with high performance demands

The Answer Today:

- Because it is the *only way* to achieve higher performance in the foreseeable future
- CPU clock rates are no longer increasing!
- Instruction-level-parallelism is not increasing either!
- Without parallel programming, performance becomes a zero-sum game.

Architectural Perspective

Role of a computer architect:

- To design and engineer the various levels of a computer system to maximize *performance* and *programmability* within limits of *technology* and *cost*.

Parallelism:

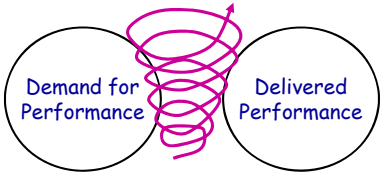
- Provides alternative to faster clock for performance
- Applies at all levels of system design

Outline

- Application Demands
- Architectural Trends
- The Impact of Technology and Power

Application Trends

There is a positive feedback cycle between delivered performance and applications' demand for performance



Example application domains:

- *Scientific computing*: CFD, Biology, Chemistry, Physics, ...
- *General-purpose computing*: Video, Graphics, CAD, Databases, ...

Speedup

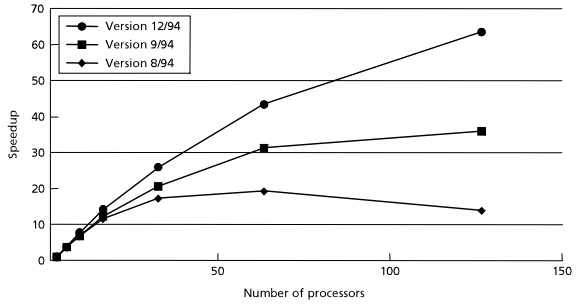
Goal of applications in using parallel machines: **Speedup**

$$\text{Speedup (p processors)} = \frac{\text{Performance (p processors)}}{\text{Performance (1 processor)}}$$

For a fixed problem size (input data set), performance = 1/time

$$\text{Speedup}_{\text{fixed problem}}\text{ (p processors)} = \frac{\text{Time (1 processor)}}{\text{Time (p processors)}}$$

Learning Curve for Parallel Programs



- AMBER molecular dynamics simulation program
- Starting point was vector code for Cray-1
- 145 MFLOP on Cray90, 406 for final version on 128-processor Paragon, 891 on 128-processor Cray T3D

Commercial Computing

Also relies on parallelism for high end

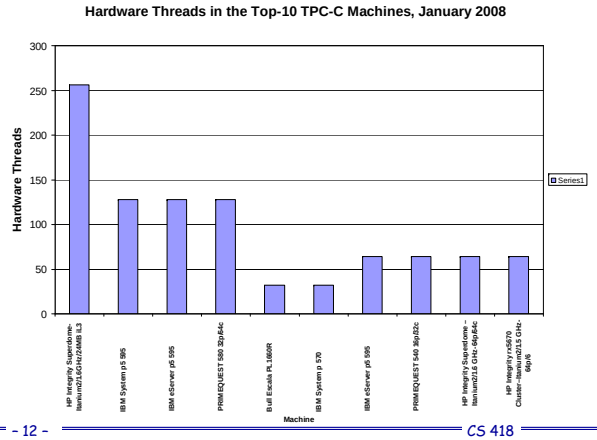
- Scale not so large, but use much more wide-spread
- Computational power determines scale of business that can be handled

Databases, online-transaction processing, decision support, data mining, data warehousing ...

TPC benchmarks (TPC-C order entry, TPC-D decision support)

- Explicit scaling criteria provided
- Size of enterprise scales with size of system
- Problem size no longer fixed as *p* increases, so throughput is used as a performance measure (transactions per minute or *tpm*)

Recent TPC-C Results



Outline

- Application Demands
- Architectural Trends
- The Impact of Technology and Power

Architectural Trends

Architecture translates technology's gifts to performance and capability

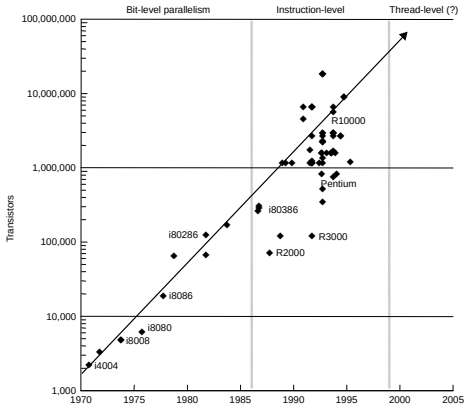
Four generations of architectural history: tube, transistor, IC, VLSI
- Here focus only on VLSI generation

Greatest delineation in VLSI has been in type of parallelism exploited

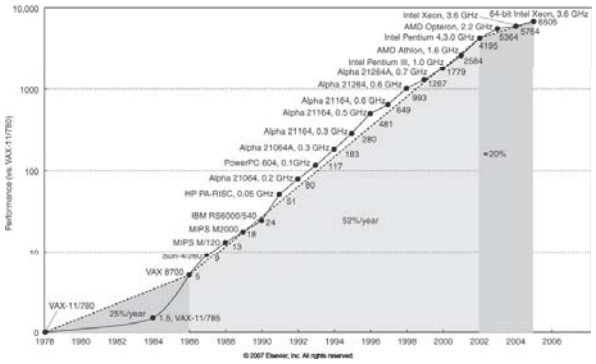
Arch. Trends: Exploiting Parallelism

Greatest trend in VLSI generation is increase in parallelism
- Up to 1985: bit level parallelism: 4-bit -> 8 bit -> 16-bit
  - slows after 32 bit
  - adoption of 64-bit now under way, 128-bit far (not performance issue)
  - great inflection point when 32-bit micro and cache fit on a chip
- Mid 80s to mid 90s: instruction level parallelism
  - pipelining and simple instruction sets, + compiler advances (RISC)
  - on-chip caches and functional units => superscalar execution
  - greater sophistication: out of order execution, speculation, prediction
    - » to deal with control transfer and latency problems
- Now: thread level parallelism

Phases in VLSI Generation



The Rate of Single-Thread Performance Improvement has Decreased

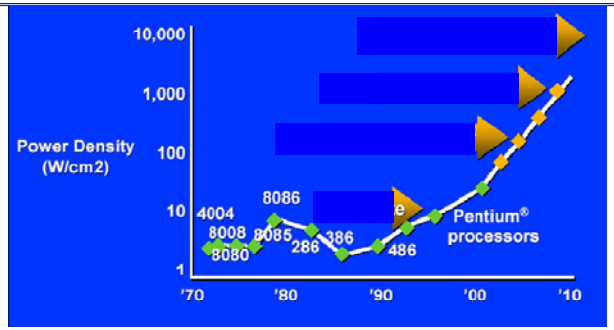


(Figure courtesy of Hennessy & Patterson, "Computer Architecture, A Quantitative Approach", V4.)

Outline

- Application Demands
- Architectural Trends
- The Impact of Technology and Power

Impact of Power Density on the Microprocessor Industry



Pat Gelsinger, ISSCC 2001

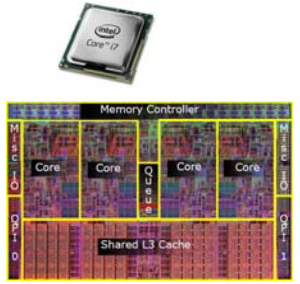
The future is not higher clock rates, but multiple cores per die.

Recent Intel Processors

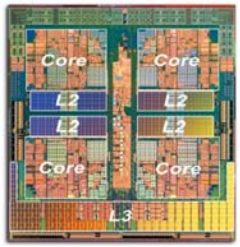
	Year	Transistors	Clock (GHz)	Power (W)
• Pentium 4	2000	42M	1.7-3.4	65-89
• Pentium M	2003	140M	1.4-2.1	21
• Core Duo	2006	151M	2.3-2.5	
• Core 2 Duo	2006	291M	2.6-2.9	
• Core 2 Quad	2006	2x291M	2.6-2.9	
• Core i7 (Quad)	2008	781M	2.93-3.6	

"We are dedicating all of our future product development to multicore designs. We believe this is a key inflection point for the industry." Intel President Paul Otellini, IDF 2005

Recent x86 Examples



Intel's Quad Core i7



AMD's Quad-Core Phenom II

- Highly integrated, commodity systems
- Current scale = 4 processors

Summary: Why Parallel Architecture?

If you can't exploit parallelism, then performance will be a zero sum game.

- If you want to add a new feature and maintain performance, you will need to remove something else.

The future is "multicore" (i.e. parallel) according to all major processor vendors.

- We expect more & more cores will be delivered with each generation.

Starting now, everyone will need to know how to write parallel programs.

- It is no longer a niche area: it is mainstream.