Why Parallel Architecture?
CS 418
Lecture 1

What is 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?

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

Why Study it Today?

History: diverse and innovative organizational structures, often tied to novel programming models

Rapidly maturing under strong technological constraints

- The "killer micro" is ubiquitous
- Laptops and supercomputers are fundamentally similar!
- Technological trends cause diverse approaches to converge

Technological trends make parallel computing inevitable

- In the mainstream

Need to understand fundamental principles and design tradeoffs, not just taxonomies

- Naming, Ordering, Replication, Communication performance

Inevitability of Parallel Computing

Application demands: Our insatiable need for cycles

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

Technology Trends

- Number of transistors on chip growing rapidly
- Clock rates expected to go up only slowly

Architecture Trends

- Instruction-level parallelism valuable but limited
- Coarser-level parallelism, as in MPs, the most viable approach

Economics

Current trends:

- Today's microprocessors have multiprocessor support
- Servers & even PCs becoming MP: Sun, SGI, COMPAQ, Dell,...
- Tomorrow's microprocessors are multiprocessors

Application Trends

Demand for cycles fuels advances in hardware, and vice-versa

- Cycle drives exponential increase in microprocessor performance
- Drives parallel architecture harder: most demanding applications

Range of performance demands

- Need range of system performance with progressively increasing cost
- Platform pyramid

Speedup

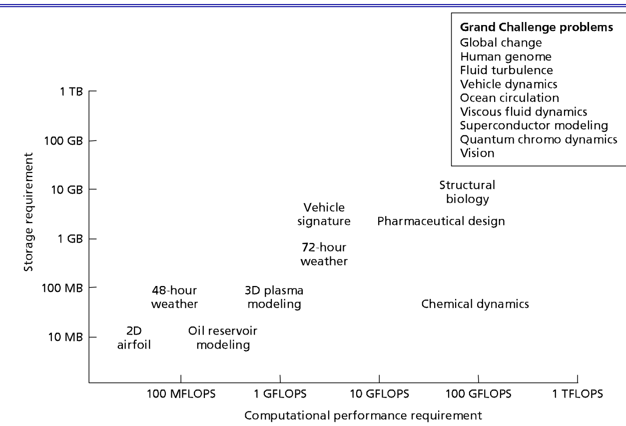
Goal of applications in using parallel machines: **Speedup**

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

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

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

Scientific Computing Demand

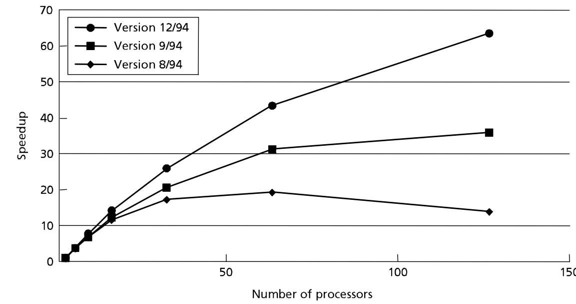


Engineering Computing Demand

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)
- **Visualization**
  - in all of the above
  - entertainment (films like Toy Story)
  - architecture (walk-throughs and rendering)
- **Financial modeling** (yield and derivative analysis)
- etc.

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

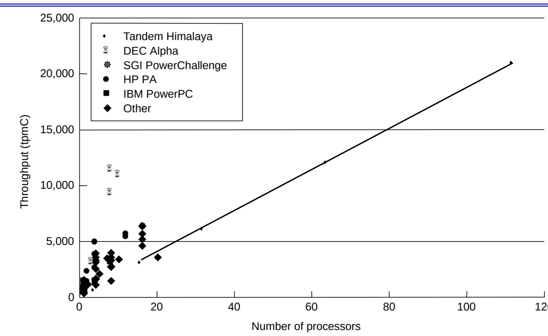
- Scale not so large, but use much more wide-spread
- Computational power determines scale of business that can be handled

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*)

TPC-C Results for March 1996



- Parallelism is pervasive
- Small to moderate scale parallelism very important
- Difficult to obtain snapshot to compare across vendor platforms

Summary of Application Trends

Transition to parallel computing has occurred for scientific and engineering computing

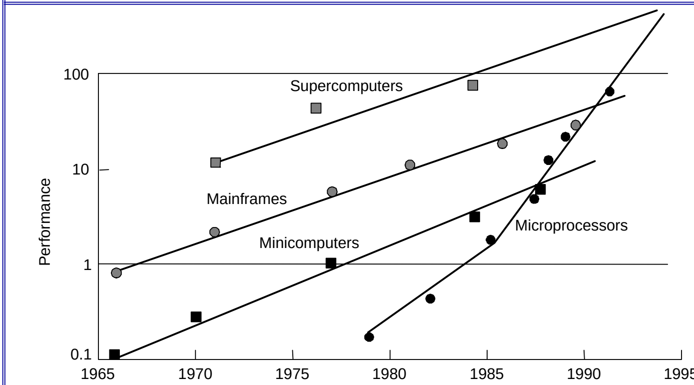
In rapid progress in commercial computing
- Database and transactions as well as financial
- Usually smaller-scale, but large-scale systems also used

Desktop also uses multithreaded programs, which are a lot like parallel programs

Demand for improving throughput on sequential workloads
- Greatest use of small-scale multiprocessors

Solid application demand exists and will increase

Technology Trends



Commodity microprocessors have caught up with supercomputers.

Architectural Trends

Architecture translates technology's gifts to performance and capability

Resolves the tradeoff between parallelism and locality
- Current microprocessor: 1/3 compute, 1/3 cache, 1/3 off-chip connect
- Tradeoffs may change with scale and technology advances

Understanding microprocessor architectural trends
- Helps build intuition about design issues or parallel machines
- Shows fundamental role of parallelism even in "sequential" computers

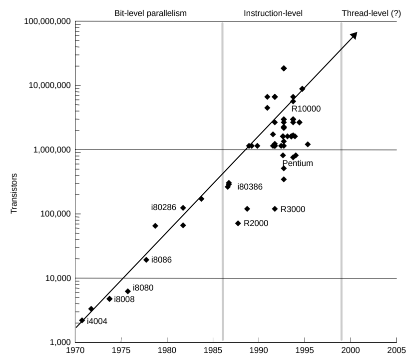
Four generations of architectural history: tube, transistor, IC, VLSI
- Here focus only on VLSI generation

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
- Next step: thread level parallelism

Phases in VLSI Generation

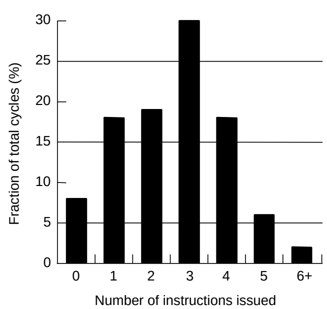


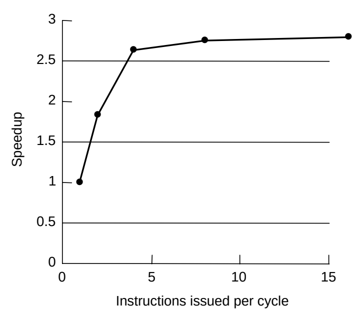
- How good is instruction-level parallelism?
- Thread-level needed in microprocessors?

Architectural Trends: ILP

- **Reported speedups for superscalar processors**
  - Horst, Harris, and Jardine [1990] 1.37
  - Wang and Wu [1988] 1.70
  - Smith, Johnson, and Horowitz [1989] 2.30
  - Murakami et al. [1989] 2.55
  - Chang et al. [1991] 2.90
  - Jouppi and Wall [1989] 3.20
  - Lee, Kwok, and Briggs [1991] 3.50
  - Wall [1991] 5
  - Melvin and Patt [1991] 8
  - Butler et al. [1991] .... 17+
- **Large variance due to difference in**
  - application domain investigated (numerical versus non-numerical)
  - capabilities of processor modeled

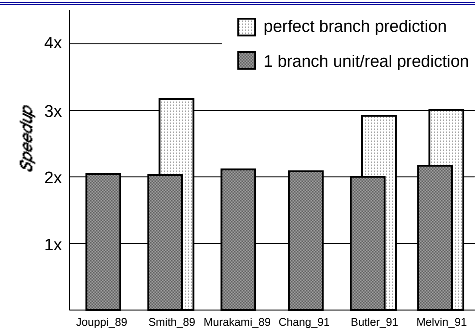
ILP Ideal Potential





- **Infinite resources and fetch bandwidth, perfect branch prediction and renaming**
  - real caches and non-zero miss latencies

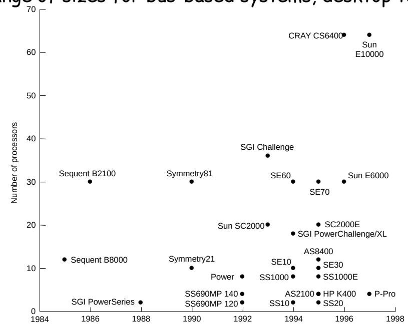
Results of ILP Studies



- **Concentrate on parallelism for 4-issue machines**
- **Realistic studies show only 2-fold speedup**
- **Recent studies show that for more parallelism, one must look across threads**

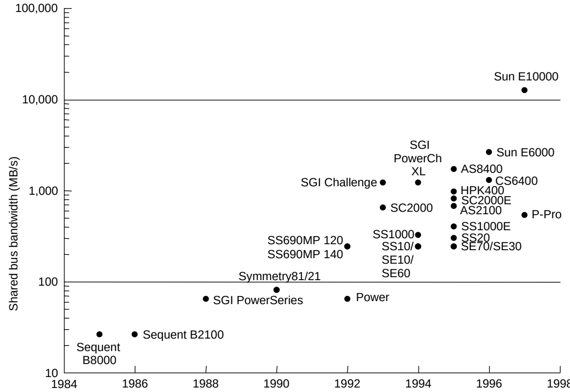
Architectural Trends: Bus-based MPs

- Micro on a chip makes it natural to connect many to shared memory
  - dominates server and enterprise market, moving down to desktop
- Faster processors began to saturate bus, then bus technology advanced
  - today, range of sizes for bus-based systems, desktop to large servers



No. of processors in fully configured commercial shared-memory systems

Bus Bandwidth



Economics

Commodity microprocessors not only fast but CHEAP

- Development cost is tens of millions of dollars (5-100 typical)
- BUT, <u>many</u> more are sold compared to supercomputers
- Crucial to take advantage of the investment, and use the commodity building block
- Exotic parallel architectures no more than special-purpose

Multiprocessors being pushed by software vendors (e.g. database) as well as hardware vendors

Standardization by Intel makes small, bus-based SMPs commodity

Desktop: few smaller processors versus one larger one?

- Multiprocessor on a chip

Summary: Why Parallel Architecture?

Increasingly attractive

- Economics, technology, architecture, application demand

Increasingly central and mainstream

Parallelism exploited at many levels

- Instruction-level parallelism
- Thread-level parallelism within a microprocessor
- Multiprocessor servers
- Large-scale multiprocessors ("MPPs")

Same story from memory system perspective

- Increase bandwidth, reduce average latency with many local memories

Wide range of parallel architectures make sense

- Different cost, performance and scalability