CS252
Graduate Computer Architecture

Lecture 20:
Static Pipelining #2 and
Goodbye to Computer Architecture

April 13, 2001

Prof. David A. Patterson

Computer Science 252

Spring 2001

Review #1: Hardware versus Software Speculation Mechanisms

- To speculate extensively, must be able to disambiguate memory references
  - Much easier in HW than in SW for code with pointers
- HW-based speculation works better when control flow is unpredictable, and when HW-based branch prediction is superior to SW-based branch prediction done at compile time
  - Mispredictions mean wasted speculation
- HW-based speculation maintains precise exception model even for speculated instructions
- HW-based speculation does not require compensation or bookkeeping code

Review #2: Hardware versus Software Speculation Mechanisms cont'd

- Compiler-based approaches may benefit from the ability to see further in the code sequence, resulting in better code scheduling
- HW-based speculation with dynamic scheduling does not require different code sequences to achieve good performance for different implementations of an architecture
  - may be the most important in the long run?

Review #3: Software Scheduling

- Instruction Level Parallelism (ILP) found either by compiler or hardware.
- Loop level parallelism is easiest to see
  - SW dependencies/compiler sophistication determine if compiler can unroll loops
  - Memory dependencies hardest to determine => Memory disambiguation
  - Very sophisticated transformations available
- Trace Sceduling to Parallelize If statements
- Superscalar and VLIW: CPI < 1 (IPC > 1)
  - Dynamic issue vs. Static issue
  - More instructions issue at same time => larger hazard penalty
  - Limitation is often number of instructions that you can successfully fetch and decode per cycle

VLIW in Embedded Designs

- VLIW: greater parallelism under programmer, compiler control vs. hardware in superscalar
- Used in DSPs, Multimedia processors as well as IA-64
- What about code size?
- Effectiveness, Quality of compilers for these applications?

Example VLIW for multimedia: Philips Trimedia CPU

- Every instruction contains 5 operations
- Predicated with single register value;
  if 0 => all 5 operations are canceled
- 128 64-bit registers, which contain either integer or floating point data
- Partitioned ALU (SIMD) instructions to compute on multiple instances of narrow data
- Offers both saturating arithmetic (DSPs) and 2's complement arithmetic (desktop)
- Delayed Branch with 3 branch slots

Trimedia Operations

Operation Category	Examples	No. Ops	Comment
Load/store ops	ld8, ld16, ld32, ld64,limm. st8, st16, st32, st64	39	SIMD, signed, unsigned, register indirect, indexed, scaled addressing
Byte shuffles	shift right 1-, 2- , 3-bytes, select byte, merge, pack	67	SIMD type convert
Bit shifts	asl, asr, lsl, lsr, rol,	48	round, fields, SIMD
Multiplies	mul, sum of products, sum-of-SIMD-elements	54	round, saturate, 2祖 comp, SIMD
Integer arithmetic	add, sub, min, max, abs, average, bitand, bitor, bitxor, bitinv, bitandinv, eql, neq, gtr, geq, les, leq, sign extend, zero extend, sum of absolute differences	104	saturate, 2's comp, unsigned, immediate, SIMD
Floating point	add, sub, neg, mul, div, sqrt eql, neq, gtr, geq, les, leq, IEEE flags	59	scalar and SIMD
Lookup table	SIMD gather load using registers as addresses	6	SIMD
Special ops	alloc, prefetch block, invalidate block, copy block back, read tag read, cache status, read counter	23	MMU, cache, special regs
Branch	jmpt, jmpf	10	(un)interruptible, trap
Total		410	

- large number of ops because used retargetable compilers, multiple machine descriptions, and die size estimators to explore the space to find the best cost-performance design
  - Verification time, manufacturing test, design time?

Trimedia Functional Units, Latency, Instruction Slots

F.U.	Latency	Operation Slot					Typical operations performed by functional unit
		1	2	3	4	5	
ALU	0	X	X	X	X	X	Integer add/subtract/compare, logicals
DMem	2				X	X	Loads and stores
DMemSpec	2					X	Cache invalidate, prefetch, allocate
Shifter	0	X	X				Shifts and rotates
DSPALU	1	X		X			Simple DSP arithmetic ops
DSPMul	2		X	X			DSP ops with multiplication
Branch	3		X	X	X		Branches and jumps
FALU	2	X			X		FP add, subtract
IFMul	2		X	X			Integer and FP multiply
FComp	0			X			FP compare
FTough	16		X				FP divide, square root

- 23 functional units of 11 types,
- which of 5 slots can issue (and hence number of functional units)

Philips Trimedia CPU

- Compiler responsible for including no-ops
  - both within an instruction-- when an operation field cannot be used--and between dependent instructions
  - processor does not detect hazards, which if present will lead to incorrect execution
- Code size? compresses the code (~ Quiz #1)
  - decompresses after fetched from instruction cache

Example

- Using MIPS notation, look at code for

```
void sum (int a[], int b[], int c[],
  int n)
{ int i;
  for (i=0; i<n; i++)
      c[i] = a[i]+b[i];
```

Example

- MIPS code for loop

```
Loop: LD     R11,R0(R4)  # R11 = a[i]
      LD     R12,R0(R5)  # R12 = b[i]
      DADDU  R17,R11,R12 # R17 = a[i]+b[i]
      SD     R17,0(R6)   # c[i] = a[i]+b[i]
      DADDIU R4,R4,8     # R4 = next a[] addr
      DADDIU R5,R5,8     # R5 = next b[] addr
      DADDIU R6,R6,8     # R6 = next c[] addr
      BNE    R4,R7,Loop  # if not last go to
                                           Loop
```

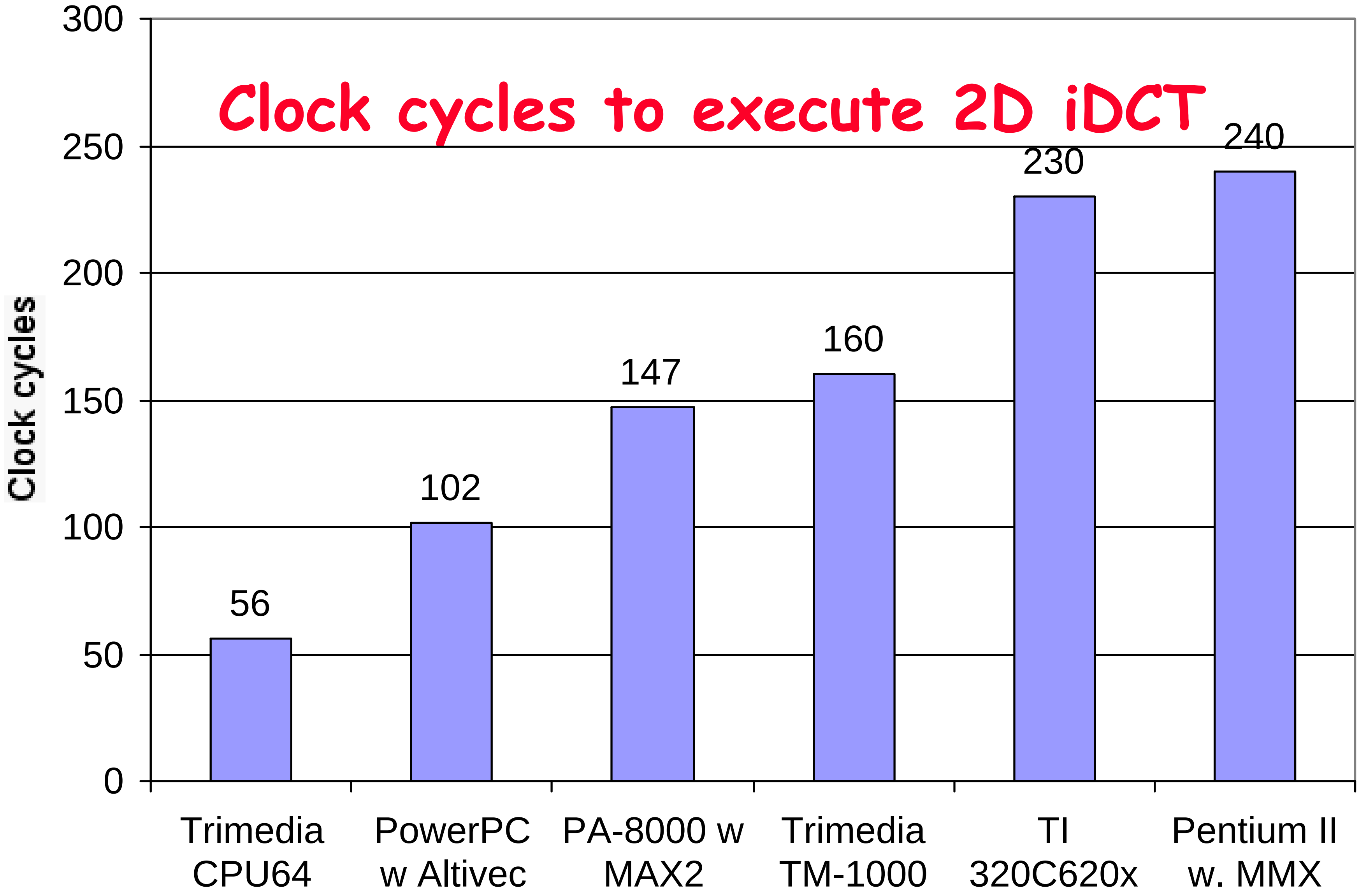
- Then unroll 4 times and schedule

Tridmedia Version

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5
			LD R11,0(R4)	LD R12,R0(R5)
DADDUI R25,R6,32			LD R14,8(R4)	LD R15,8(R5)
SETEQ R25,R25,R7			LD R19,16(R4)	LD R20,16(R5)
DADDU R17,R11,R12	DADDIU R4,R4,32		LD R22,24(R4)	LD R23,24(R5)
DADDU R18,R14,R15	JMPF R25,R30		SD R17, 0(R6)	
DADDU R21,R19,R20	DADDIU R5,R5,32		SD R18, 8(R6)	
DADDU R24,R22,R23			SD R21,16(R6)	
DADDIU R6,R6,32			SD R24, 24(R6)	

- Loop address in register 30
- Conditional jump (JMPF) so that only jump is conditional, not whole instruction predicated
- DADDUI (1st slot, 2nd instr) and SETEQ (1st slot, 3rd instr) compute loop termination test
  - Duplicate last add early enough to schedule 3 instruction branch delay
- 24/40 slots used (60%) in this example

Clock cycles to execute 2D iDCT



- Note that the Trimedia results are based on compilation, unlike many of the others. The year 2000 clock rate of the CPU64 is 300 MHz . The 1999 clock rates of the others are about 400 MHz for the PowerPC, PA-8000, and Pentium II, with the TM-1000 at 100 MHz and the TI 320620x at 200 MHz.

Administratrivia

- 3rd project meetings 4/11: good progress!
  - Meet with some on Friday
- 4/18 Wed Quiz #2 310 Soda at 5:30
- Pizza at La Val's at 8:30
- What's left:
  - 4/20 Fri, "How to Have a Bad Academic Career" (Career/Talk Advice); signup for talks
  - 4/25 Wed, Oral Presentations (8AM to 2 PM) 611 Soda (no lecture)
  - 4/27 Fri (no lecture)
  - 5/2 Wed Poster session (noon - 2); end of course

Transmeta Crusoe MPU

- 80x86 instruction set compatibility through a software system that translates from the x86 instruction set to VLIW instruction set implemented by Crusoe
- VLIW processor designed for the low-power marketplace

Crusoe processor: Basics

- VLIW with in-order execution
- 64 Integer registers
- 32 floating point registers
- Simple in-order, 6-stage integer pipeline:
  2 fetch stages, 1 decode, 1 register read,
  1 execution, and 1 register write-back
- 10-stage pipeline for floating point, which has 4 extra execute stages
- Instructions in 2 sizes: 64 bits (2 ops) and 128 bits (4 ops)

Crusoe processor: Operations

- 5 different types of operation slots:
- ALU operations: typical RISC ALU operations
- Compute: this slot may specify any integer ALU operation (2 integer ALUs), a floating point operation, or a multimedia operation
- Memory: a load or store operation
- Branch: a branch instruction
- Immediate: a 32-bit immediate used by another operation in this instruction
- For 128-bit instr: 1st 3 are Memory, Compute, ALU; last field either Branch or Immediate

80x86 Compatability

- Initially, and for lowest latency to start execution, the x86 code can be interpreted on an instruction by instruction basis
- If a code segment is executed several times, translated into an equivalent Crusoe code sequence, and the translation is cached
  - The unit of translation is at least a basic block, since we know that if any instruction is executed in the block, they will all be executed
  - Translating an entire block both improves the translated code quality and reduces the translation overhead, since the translator need only be called once per basic block
- Assumes 16MB of main memory for cache

Exception Behavior during Speculation

- Crusoe support for speculative reordering consists of 4 major parts:

1. shadowed register file
   - Shadow discarded only when x86 instruction has no exception
2. program-controlled store buffer
   - Only store when no exception; keep until OK to store
3. memory alias detection hardware with speculative loads
4. conditional move instruction (called select) that is used to do if-conversion on x86 code sequences

Crusoe Performance?

- Crusoe depends on realistic behavior to tune the code translation process, it will not perform in a predictive manner when benchmarked using simple, but unrealistic scripts
  - Needs idle time to translate
  - Profiling to find hot spots
- To remedy this factor, Transmeta has proposed a new set of benchmark scripts
  - Unfortunately, these scripts have not been released and endorsed by either a group of vendors or an independent entity

Real Time, so comparison is Energy

Workload description	Energy consumption for the workload (W/Hr.)		Relative consumption TM 3200 / Mobile Pentium III
	Mobile Pentium III @ 500 MHz	TM 3200 @400MHz 1.5V	
MP3 playback	0.672	0.214	0.32
DVD playback	1.13	0.479	0.42

Crusoe Applications?

- Notebook: Sony, others
- Compact Servers: RLX technologies

VLIW Readings

- Josh Fisher 1983 Paper + 1998 Retrospective
- What are characteristics of VLIW?
- Is ELI-512 the first VLIW?
  - How many bits in instruction of ELI-512?
- What is breakthrough?
- What expected speedup over RISC?
- What is wrong with vector?
- What benchmark results on code size, speedup?
- What limited speedups to 5X to 10X?
- What other problems faced ELI-512?
- In retrospect, what wished changed?
- In retrospect, what naïve about?

Review of Course

- Review and Goodbye to Computer Architecture, topic by topic + follow-on courses
- Future Directions for Computer Architecture?

Chapter 1: Performance and Cost

- **Amdahl's Law:**

$$\text{Speedup}_{\text{overall}} = \frac{\text{ExTime}_{\text{old}}}{\text{ExTime}_{\text{new}}} = \frac{1}{(1 - \text{Fraction}_{\text{enhanced}}) + \dfrac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}}$$

- **CPI Law:**

$$\text{CPU time} = \frac{\text{Seconds}}{\text{Program}} = \frac{\text{Instructions}}{\text{Program}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Seconds}}{\text{Cycle}}$$

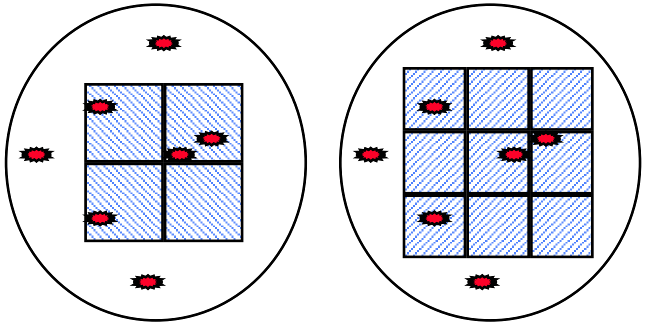
- **Designing to Last through Trends**

	Capacity	Speed
Logic	2x in 3 years	2x in 3 years
DRAM	4x in 4 years	2x in 10 years
Disk	4x in 3 years	2x in 5 years

Processor 2x every 1.5 years?

Chapter 1: Performance and Cost

- Die Cost goes roughly with die area^4
  - Microprocessor with 1B transistors in 2005?

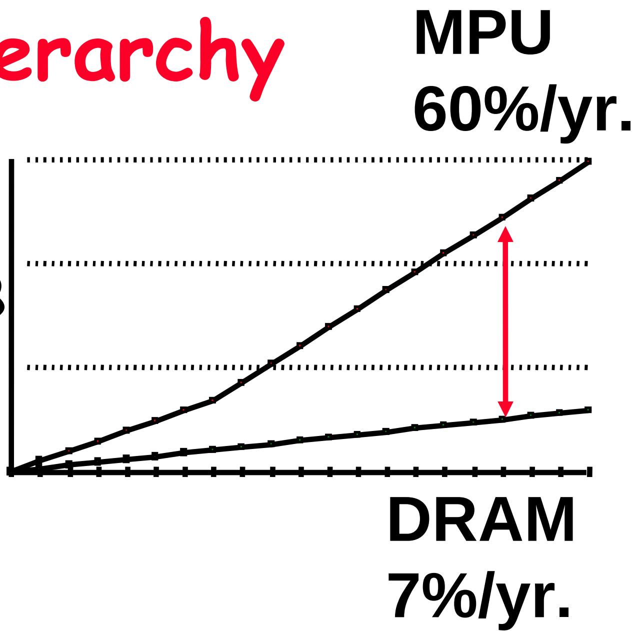


- Cost vs. Price
  - Can PC industry support engineering/research investment?
- For better or worse, benchmarks shape a field
- Interested in learning more on integrated circuits? EE 241 "Advanced Digital Integrated Circuits"
- Interested in learning more on performance? CS 266 "Introduction to Systems Performance"

Goodbye to
Performance and Cost

- Will sustain 2X every 1.5 years?
  - Can integrated circuits improve below 1.8 micron in speed as well as capacity?
- 5-6 yrs to PhD =>
  16X CPU speed, 10XDRAM Capacity, 25X Disk capacity?
  (10 GHz CPU, 1GB DRAM, 2TB disk?)

Chapter 5: Memory Hierarchy



- Processor-DRAM Performance gap
- 1/3 to 2/3 die area for caches, TLB
- Alpha 21264: 108 clock to memory
  ⇒ 648 instruction issues during miss
- 3 Cs: Compulsory, Capacity, Conflict
- 4 Questions: where, who, which, write
- Applied recursively to create multilevel caches
- Performance = *f*(hit time, miss rate, miss penalty)
  - danger of concentrating on just one when evaluating performance

Cache Optimization Summary

$$CPUtime = IC \times \left(CPI_{Execution} + \frac{Memory \ \ accesses}{Instruction} \times \textbf{Miss rate} \times Miss \ \ penalty \right) \times Clock \ \ cycle \ \ time$$

	Technique	*MR*	*MP*	*HT*	*Complexity*
miss rate	Larger Block Size	+	–		0
	Higher Associativity	+		–	1
	Victim Caches	+			2
	Pseudo-Associative Caches	+			2
	HW Prefetching of Instr/Data	+			2
	Compiler Controlled Prefetching	+			3
	Compiler Reduce Misses	+			0
miss penalty	Priority to Read Misses		+		1
	Subblock Placement		+	+	1
	Early Restart & Critical Word 1st		+		2
	Non-Blocking Caches		+		3
	Second Level Caches		+		2
hit time	Small & Simple Caches	–		+	0
	Avoiding Address Translation			+	2
	Pipelining Writes			+	1

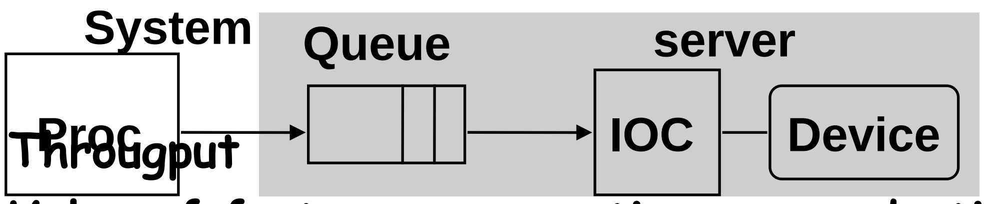
memory hierarchy art: taste in selecting between alternatives to find combination that fits well together

Goodbye to Memory Hierarchy

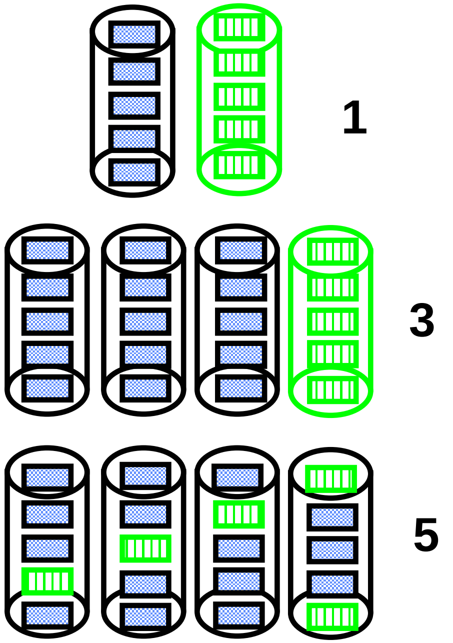
- Will L2 cache keep growing? (e.g, 64 MB L2 cache?)
- Will multilevel hierarchy get deeper? (L4 cache?)
- Will DRAM capacity/chip keep going at 4X / 4 years? (e.g., 16 Gbit chip?)
- Will processor and DRAM/Disk be unified? For which apps?
- Out-of-order CPU hides L1 data cache miss (3–5 clocks), but hide L2 miss? (>100 clocks)
- Memory hierarchy likely overriding issue in algorithm performance: do algorithms and data structures of 1960s work with machines of 2000s?

Chapter 6: Storage I/O

- Disk BW 40%/yr, areal density 60%/ yr, $/MB faster?
- Little's Law: $Length_{system} = rate \times Time_{system}$
  (Mean number customers = arrival rate x mean service time)



  - Throughput
  - Value of faster response time on productivity
- Benchmarks: scaling, cost, auditing, response time limits
- RAID: performance and reliability



- Queueing theory? IEOR 161, 267, 268
- SW storage systems? CS 286
  "Implementation of Data Base Systems"

Summary: I/O Benchmarks

- Scaling to track technological change
- TPC: price performance as nomalizing configuration feature
- Auditing to ensure no foul play
- Throughput with restricted response time is normal measure
- Benchmarks to measure Availability, Maintainability?

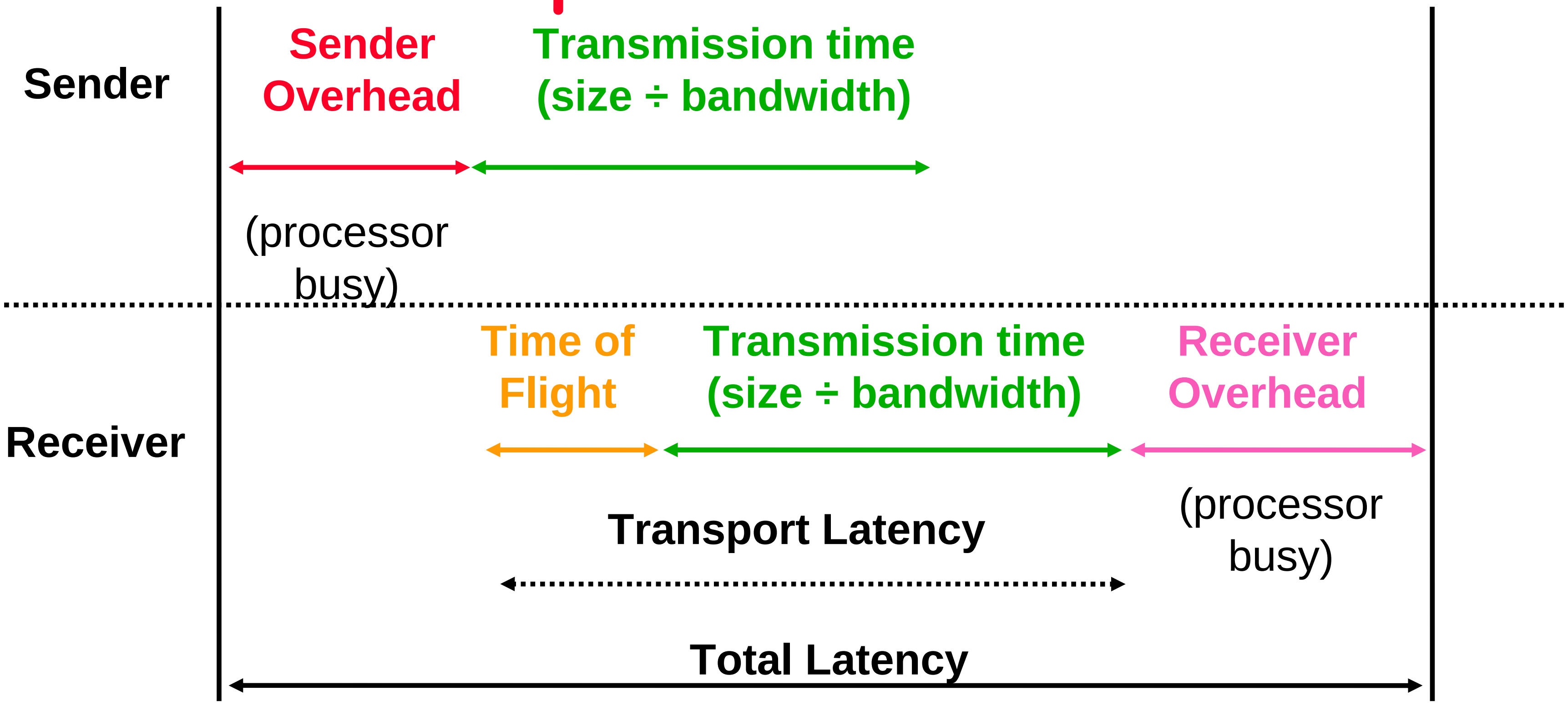
Goodbye to Storage I/O

- Disks attached directly to networks, avoiding the file server? ("Network Attached Storage Devices")
- Disks:
  - Extraodinary advance in capacity/drive, $/GB
  - Currently 17 Gbit/sq. in. ; can continue past 100 Gbit/sq. in.?
  - Bandwidth, seek time not keeping up: 3.5 inch form factor makes sense? 2.5 inch form factor in near future? 1.0 inch form factor in long term?
- Tapes
  - No investment, must be backwards compatible
  - Are they already dead?
  - What is a tapeless backup system?

Goodbye to Storage I/O

- Terminology of Fault/Error/Failure
- Is Availability the killer metric for Service oriented world?
- Can we construct systems that will actually achieve 99.999% availability, including software and people?
- Disks growing at 2X/ 1 years recently: Will Patterson continue get email messages to reduce file storage for the rest of my career?
- Heading towards a personal terabyte: hierarchical file systems vs. database to organize personal storage?
- What going to do when can have video record of entire life on line?

Chapter 7: Networks



Total Latency = Sender Overhead + Time of Flight + Message Size ÷ BW + Receiver Overhead

High BW networks + high overheads violate of Amdahl's Law

Chapter 7: Networks

- Similarities of SANs, LANs, WANs
- Integrated circuit revolutionizing networks as well as processors
- Switch is a specialized computer
- Protocols allow hetereogeneous networking , handle normal and abnormal events
- Interested in learning more on networks?
  EE 122 “Introduction to Computer Networks” (Stoika)
  CS 268 “Computer Networks” (Stoika)

Review: Networking

- Clusters +: fault isolation and repair, scaling, cost
- Clusters -: maintenance, network interface performance, memory efficiency
- Google as cluster example:
  - scaling (6000 PCs, 1 petabyte storage)
  - fault isolation (2 failures per day yet available)
  - repair (replace failures weekly/repair offline)
  - Maintenance: 8 people for 6000 PCs
- Cell phone as portable network device
  - # Handsets >> # PCs
  - Univerisal mobile interface?
- Is future services built on Google-like clusters delivered to gadgets like cell phone handset?

Goodbye to Networks

- Will network interfaces follow example of graphics interfaces and become first class citizens in microprocessors, thereby avoiding the I/O bus?
- Will Ethernet standard keep winning the LAN wars? e.g., 1 Gbit/sec, 10 Gbit/sec, wireless (802.11B)...

Chapter 8: Multiprocessors

Programming Model
Communication Abstraction
Interconnection SW/OS
Interconnection HW

- Layers:
  - Programming Model:
    - Multiprogramming : lots of jobs, no communication
    - Shared address space: communicate via memory
    - Message passing: send and recieve messages
    - Data Parallel: several agents operate on several data sets simultaneously and then exchange information globally and simultaneously (shared or message passing)
  - Communication Abstraction:
    - Shared address space: e.g., load, store, atomic swap
    - Message passing: e.g., send, recieve library calls
    - Debate over this topic (ease of programming, scaling) => many hardware designs 1:1 programming model
- Interested in learning more on multiprocessors: CS 258 "Parallel Computer Architecture"
- E 267 "Programming Parallel Computers"

Goodbye to Multiprocessors

- Successful today for file servers, time sharing, databases, graphics; will parallel programming become standard for production programs?
  If so, what enabled it: new programming languauges, new data structures, new hardware, new coures, ...?
- Which won large scale number crunching, databases: Clusters of independent computers connected via switched LAN vs. large shared NUMA machines? Why?

Chapter 2: Instruction Set Architecture

- What ISA looks like to pipeline?
  - Cray: load/store machine; registers; simple instr. format
- RISC: Making an ISA that supports pipelined execution
- 80x86: importance of being their first
- VLIW/EPIC: compiler controls Instruction Level Parallelism (ILP)
- Interested in learning more on compilers and ISA?
  CS 264/5 "Advanced Programming Language Design and Optimization"

Goodbye to Instruction Set Architecture

- What did IA-64/EPIC do well besides floating point programs?
  - Was the only difference the 64-bit address v. 32-bit address?
  - What happened to the AMD 64-bit address 80x86 proposal?
- What happened on EPIC code size vs. x86?
- Did Intel Oregon increase x86 performance so as to make Intel Santa Clara EPIC performance similar?

Goodbye to Dynamic Execution

- Did Transmeta-like compiler-oriented translation survive vs. hardware translation into more efficient internal instruction set?
- Did ILP limits really restrict practical machines to 4-issue, 4-commit?
- Did we ever really get CPI below 1.0?
- Did value prediction become practical?
- Branch prediction: How accurate did it become?
  - For real programs, how much better than 2 bit table?
- Did Simultaneous Multithreading (SMT) exploit underutilized Dynamic Execution HW to get higher throughput at low extra cost?
  - For multiprogrammed workload (servers) or for parallelized single program?

Goodbye to Static, Embedded

- Did VLIW become popular in embedded? What happened on code size?
- Did vector become popular for media applications, or simply evolve SIMD?
- Did DSP and general purpose microprocessors remain separate cultures, or did ISAs and cultures merge?
  - Compiler oriented?
  - Benchmark oriented?
  - Library oriented?
  - Saturation + 2's complement

Goodbye to Computer Architecture

- Did emphasis switch from cost-performance to cost-performance-availability?
- What support for improving software reliability? Security?

Goodbye to Computer Architecture

- 1985-2000: 1000X performance
  - Moore's Law transistors/chip => Moore's Law for Performance/MPU
- Hennessy: industry been following a roadmap of ideas known in 1985 to exploit Instruction Level Parallelism to get 1.55X/year
  - Caches, Pipelining, Superscalar, Branch Prediction, Out-of-order execution, …
- ILP limits: To make performance progress in future need to have explicit parallelism from programmer vs. implicit parallelism of ILP exploited by compiler, HW?
- Did Moore's Law in transistors stop predicting microprocessor performance? Did it drop to old rate of 1.3X per year?
  - Less because of processor-memory performance gap?