

CS184c: Computer Architecture [Parallel and Multithreaded]

Day 1: April 3, 2001
Overview and Message Passing



CALTECH cs184c Spring2001 -- DeHon

Today

- This Class
- Why/Overview
- Message Passing

CALTECH cs184c Spring2001 -- DeHon

CS184 Sequence

- A - structure and organization
 - raw components, building blocks
 - design space
- B - single threaded architecture
 - emphasis on abstractions and optimizations including quantification
- C - multithreaded architecture

CALTECH cs184c Spring2001 -- DeHon

CS184b

“Architecture”

- “attributes of a system as seen by the programmer”
- “conceptual structure and functional behavior”
- Defines the visible interface between the hardware and software
- Defines the semantics of the program (machine code)

CALTECH cs184c Spring2001 -- DeHon

CS184b

Conventional, Single-Threaded Abstraction

- Single, large, flat memory
- sequential, control-flow execution
- instruction-by-instruction sequential execution
- atomic instructions
- single-thread “owns” entire machine
- byte addressability
- unbounded memory, call depth

CALTECH cs184c Spring2001 -- DeHon

This Term

- Different models of computation
 - different microarchitectures
- Big Difference: Parallelism
 - previously **model** was sequential
- Mostly:
 - Multiple Program Counters
 - threads of control

CALTECH cs184c Spring2001 -- DeHon

Architecture Instruction Taxonomy

Control Threads (PCs)				
<i>pins</i> per Control Thread				
Instruction Depth				
Granularity				
Architecture/Examples				
0	0	n/a	Hardwired Functional Unit (e.g. ECC/EDC Unit, FP MPY)	
0	n	1	FPGA	
		w	Reconfigurable ALUs	
1	c	$n_v \cdot 1$	Bitwise SIMD	
		w	Traditional Processors	
	n	$n_v \cdot w$	Vector Processors	
		c	1	DPGA
m	n	8	PADDI	
		16	PADDI	
	c	w	VIW	
		n	1	HSRA/SCORE
m	1	c	$n_v \cdot w$	MSIMD
		c	1	VEGA
	1	8	16	PADDI-2
c		w	MIMD (traditional)	

CALTECH cs184c Spring2001 -- DeHon

Why?

- Why do we need a different model?
 - Different architecture?

CALTECH cs184c Spring2001 -- DeHon

Why?

- Density:
 - Superscalars scaling super-linear with increasing instructions/cycle
 - cost from maintaining sequential model
 - dependence analysis
 - renaming/reordering
 - single memory/RF access
 - VLIW lack of model/scalability problem
- Maybe there's a better way?

CALTECH cs184c Spring2001 -- DeHon

CS184a

Consider

- Two network data ports
 - states: idle, first-datum, receiving, closing
 - data arrival uncorrelated between ports

CALTECH cs184c Spring2001 -- DeHon

Instruction Control

- If FSMs advance orthogonally
 - (really independent control)
 - context depth => product of states
 - for full partition
 - *i.e.* w/ single controller (PC)
 - must create product FSM
 - which may lead to state explosion
 - N FSMs, with S states => S^N product states
 - This example:
 - 4 states, 2 FSMs => 16 state composite FSM

Why?

- Scalability
 - compose more capable machine from building blocks
 - compose from modular building blocks
 - multiple chips

Why?

- Expose/exploit parallelism better
 - saw non-local parallelism when looking at IPC
 - saw need for large memory to exploit

CALTECH cs184c Spring2001 -- DeHon

Models?

- Message Passing (week 1)
- Dataflow (week 2)
- Shared Memory (week 3)
- Data Parallel (week 4)
- Multithreaded (week 5)
- Interface Special and Heterogeneous functional units (week 6)

CALTECH cs184c Spring2001 -- DeHon

Additional Key Issues

- How Interconnect? (week 7-8)
- Cope with defects and Faults? (week 9)

CALTECH cs184c Spring2001 -- DeHon

Message Passing

CALTECH cs184c Spring2001 -- DeHon

Message Passing

- Simple extension to Models
 - Compute Model
 - Programming Model
 - Architecture
- Low-level

CALTECH cs184c Spring2001 -- DeHon

Message Passing Model

- Collection of sequential processes
- Processes may communicate with each other (messages)
 - send
 - receive
- Each process runs sequentially
 - has own address space
- Abstraction is each process gets own processor

CALTECH cs184c Spring2001 -- DeHon

Programming for MP

- Have a sequential language
 - C, C++, Fortran, lisp...
- Add primitives (system calls)
 - send
 - receive
 - spawn

CALTECH cs184c Spring2001 -- DeHon

Architecture for MP

- Sequential Architecture for processing node
 - add network interfaces
 - process have own address space
- Add network connecting
- ...minimally sufficient...

CALTECH cs184c Spring2001 -- DeHon

MP Architecture Virtualization

- Processes virtualize nodes
 - size independent/scalable
- Virtual connections between processes
 - placement independent communication

CALTECH cs184c Spring2001 -- DeHon

MP Example and Performance Issues

CALTECH cs184c Spring2001 -- DeHon

N-Body Problem

- Compute pairwise gravitational forces
- Integrate positions

CALTECH cs184c Spring2001 -- DeHon

Coding

- // params position, mass....
- $F=0$
- For $I = 1$ to N
 - send my params to $p[\text{body}[I]]$
 - get params from $p[\text{body}[I]]$
 - $F += \text{force}(\text{my params}, \text{params})$
- Update pos, velocity
- Repeat

CALTECH cs184c Spring2001 -- DeHon

Performance

- Body Work $\approx cN$
- Cycle work $\approx cN^2$
- Ideal N_p processors: cN^2/N_p

CALTECH cs184c Spring2001 -- DeHon

Performance Sequential

- Body work:
 - read N values
 - compute N force updates
 - compute pos/vel from F and params
- $c = t(\text{read value}) + t(\text{compute force})$

CALTECH cs184c Spring2001 -- DeHon

Performance MP

- Body work:
 - send N messages
 - receive N messages
 - compute N force updates
 - compute pos/vel from F and params
- $c = t(\text{send message}) + t(\text{receive message}) + t(\text{compute force})$

CALTECH cs184c Spring2001 -- DeHon

Send/receive

- $t(\text{receive})$
 - wait on message delivery
 - swap to kernel
 - copy data
 - return to process
- $t(\text{send})$
 - similar
- $t(\text{send}), t(\text{receive}) \gg t(\text{read value})$

CALTECH cs184c Spring2001 -- DeHon

Sequential vs. MP

- $T_{\text{seq}} = c_{\text{seq}} N^2$
- $T_{\text{mp}} = c_{\text{mp}} N^2 / N_p$
- $\text{Speedup} = T_{\text{seq}} / T_{\text{mp}} = c_{\text{seq}} \times N_p / c_{\text{mp}}$
- Assuming no waiting:
 - $c_{\text{seq}} / c_{\text{mp}} \approx t(\text{read value}) / (t(\text{send}) + t(\text{rcv}))$

CALTECH cs184c Spring2001 -- DeHon

Waiting?

- Shared bus interconnect:
 - wait $O(N)$ time for N sends (receives) across the machine
- Non-blocking interconnect:
 - wait $L(\text{net})$ time after message send to receive
 - if insufficient parallelism
 - latency dominate performance

CALTECH cs184c Spring2001 -- DeHon

Dertouzous Latency Bound

- Speedup Upper Bound
 - processes / Latency

CALTECH cs184c Spring2001 -- DeHon

Waiting: data availability

- Also wait for data to be sent

CALTECH cs184c Spring2001 -- DeHon

Coding/Waiting

- For $I = 1$ to N
 - send my params to $p[\text{body}[I]]$
 - get params from $p[\text{body}[I]]$
 - $F += \text{force}(\text{my params}, \text{params})$
- How long processor I wait for first datum?
 - Parallelism profile?

CALTECH cs184c Spring2001 -- DeHon

More Parallelism

- For $I = 1$ to N
 - send my params to $p[\text{body}[I]]$
- For $I = 1$ to N
 - get params from $p[\text{body}[I]]$
 - $F += \text{force}(\text{my params}, \text{params})$

CALTECH cs184c Spring2001 -- DeHon

Queuing?

- For $I = 1$ to N
 - send my params to $p[\text{body}[I]]$
 - get params from $p[\text{body}[I]]$
 - $F += \text{force}(\text{my params}, \text{params})$
- No queuing?
- Queuing?

CALTECH cs184c Spring2001 -- DeHon

Dispatching

- Multiple processes on node
- Who to run?
 - Can a receive block waiting?

CALTECH cs184c Spring2001 -- DeHon

Dispatching

- Abstraction is each process gets own processor
- If receive blocks (holds processor)
 - may prevent another process from running upon which it depends
- Consider 2-body problem on 1 node

CALTECH cs184c Spring2001 -- DeHon

Seitz Coding

- [see reading]

CALTECH cs184c Spring2001 -- DeHon

MP Issues

CALTECH cs184c Spring2001 -- DeHon

Expensive Communication

- Process to process communication goes through operating system
 - system call, process switch
 - exit processor, network, enter processor
 - system call, processes switch
- Milliseconds?
 - Thousands of cycles...

CALTECH cs184c Spring2001 -- DeHon

Why OS involved?

- Protection/Isolation
 - can this process send/receive with this other process?
- Translation
 - where does this message need to go?
- Scheduling
 - who can/should run now?

CALTECH cs184c Spring2001 -- DeHon

Issues

- Process Placement
 - locality
 - load balancing
- Cost for excessive parallelism
 - *E.g.* N-body on $N_p < N$ processor ?
- Message hygiene
 - ordering, single delivery, buffering
- Deadlock
 - user introduce, system introduce

CALTECH cs184c Spring 2001 -- DeHon

Low-Level Model

- Places burden on user [too much]
 - decompose problem explicitly
 - sequential chunk size not abstract
 - scale weakness in architecture
 - guarantee correctness in face of non-determinism
 - placement/load-balancing
 - in some systems
- Gives considerable explicit control

CALTECH cs184c Spring2001 -- DeHon

Low-Level Primitives

- Has the necessary primitives for multiprocessor cooperation
- Maybe an appropriate compiler target?
 - Architecture model, but not programming/compute model?

CALTECH cs184c Spring2001 -- DeHon

Announcements

- Note: CS25 next Monday/Tuesday
 - Seitz speaking on Tuesday
 - Dally speaking on Monday
 - (also Mead)
 - [...even DeHon... :-)]
- Changing schedule (...already...)
 - Network Interface bumped up to next Mon.
 - von Eicken et. Al., Active Messages
 - Henry and Joerg, Tightly couple P-NI

CALTECH cs184c Spring2001 -- DeHon

Big Ideas

- Value of Architectural Abstraction
- Sequential abstraction
 - limits implementation freedom
 - requires large cost to support
 - semantic mismatch between model and execution
- Parallel models expose more opportunities

CALTECH cs184c Spring2001 -- DeHon

Big Ideas

- MP has minimal primitives
 - appropriate low-level model
 - too raw/primitive for user model
- Communication essential component
 - can be expensive
 - doing well is necessary to get good performance (come out ahead)
 - watch OS cost...