

CS184b: Computer Architecture (Abstractions and Optimizations)

Day 15: May 4, 2005
Message Passing



Caltech CS184 Spring2005 -- DeHon

Today

- Message Passing Model
- Examples
- Performance Issues
- Design for Multiprocessing
 - Engineering “Low cost” messaging

Caltech CS184 Spring2005 -- DeHon

2

Message Passing

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

Caltech CS184 Spring2005 -- DeHon

3

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 (essentially multi-tasking model)

Caltech CS184 Spring2005 -- DeHon

4

Programming for MP

- Have a sequential language
 - C, C++, Fortran, lisp...
- Add primitives (system calls)
 - send
 - receive
 - Spawn
 - (multitasking primitives with no shared memory)

Caltech CS184 Spring2005 -- DeHon

5

Architecture for MP

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

Caltech CS184 Spring2005 -- DeHon

6

MP Architecture Virtualization

- Processes virtualize nodes [0,1,infinity]
 - size independent/scalable
- Virtual connections between processes
 - placement independent communication

MP Example and Performance Issues

N-Body Problem

- Compute pairwise gravitational forces
- Integrate positions

Coding

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

Performance

- Body Work $\sim cN$
- $\times N$ processes
- Cycle work $\sim cN^2$
- Ideal N_p processor time: cN^2/N_p

Performance Sequential

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

Performance MP

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

Caltech CS184 Spring2005 -- DeHon

13

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 CS184 Spring2005 -- DeHon

14

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:
 - $c_{\text{seq}} / c_{\text{mp}} \approx (t(\text{read}) + t(\text{write})) / (t(\text{send}) + t(\text{rcv}))$

Caltech CS184 Spring2005 -- DeHon

15

Waiting?

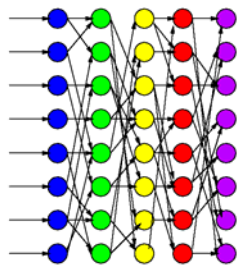
- Assume no network congestion:
- Must wait $L(\text{net})$ time after message sent to receive
- if insufficient parallelism
 - latency dominate performance

Caltech CS184 Spring2005 -- DeHon

16

Dertouzos Latency Bound

- Speedup Upper Bound
 - processes / Latency

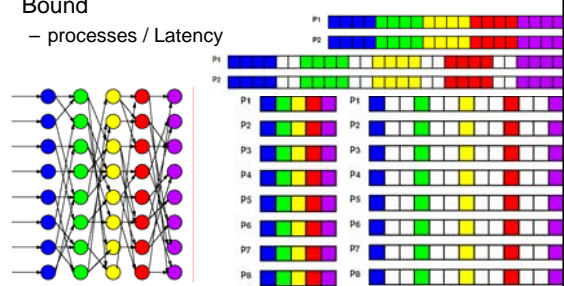


Caltech CS184 Spring2005 -- DeHon

17

Dertouzos Latency Bound

- Speedup Upper Bound
 - processes / Latency



Caltech CS184 Spring2005 -- 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?
- Is queuing optional?

More Parallelism

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

Dispatching

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

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

Seitz Coding

```
while(s.steps--) /* repeat s.steps computation cycles */
{
  body_out.buf = &host;          /* first time send out host body */
  for (i = (s.n-1)/2; i--;)      /* repeat (s.n-1)/2 times */
  {
    send_wait(&body_out);        /* send out the host/guest */
    recv_wait(&body_in);         /* receive the next guest */
    COMPUTE_FORCE(&host, &guest, FORCE); /* calculate force */
    ADD_FORCE_TO_HOST(&host, FORCE); /* may the force be with you */
    ADD_FORCE_TO_GUEST(&guest, FORCE); /* and with the guest, also */
    body_out.buf = &guest;       /* prepare to pass the guest */
  }
  body_bak.id = guest.home_id;   /* send guest back */
  send_wait(&body_bak); recv_wait(&body_bak); /* the envoy returns */
  ADD_GUEST_FORCE_TO_HOST(&host, &guest);
  UPDATE(&host);                /* integrate position */
}
```

[Seitz/CACM'85: Fig. 5]

MP Issues

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 to millions of cycles...

Caltech CS184 Spring2005 -- DeHon

25

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 CS184 Spring2005 -- DeHon

26

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 CS184 Spring2005 -- DeHon

27

Low-Level Model

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

Caltech CS184 Spring2005 -- DeHon

28

Low-Level Primitives

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

Caltech CS184 Spring2005 -- DeHon

29

Problem 1

- Messages take milliseconds
 - (1000 → 10⁶s of cycles)
- **Forces** use of coarse-grained parallelism
 - Speedup = $T_{seq}/T_{mp} = c_{seq} \times N_p / c_{mp}$
 - $c_{seq}/c_{mp} \approx t(comp) / (t(comm) + t(comp))$
 - driven to make $t(comp) \gg t(comm)$

Caltech CS184 Spring2005 -- DeHon

30

Problem 2

- **Potential parallelism** is costly
 - additional communication cost is born even when sequentialized (same node)
- Process to process switch expensive
- Discourages exposing maximum parallelism
 - works against simple/scalable model

Caltech CS184 Spring2005 -- DeHon

31

Bad Cost Model

- Challenge
 - give programmer a simple model of how to write good programs
- Here
 - exposing parallelism increases performance
 - but has cost
 - expose too much will decrease
 - hard for user to know which

Caltech CS184 Spring2005 -- DeHon

32

Bad Model

- **Poor User-level abstraction:** user should not be picking granularity of exploited parallelism
 - this should be done by tools

Caltech CS184 Spring2005 -- DeHon

33

Cosmic Cube

- Early 1980s
- Used commodity hardware
 - off the shelf solution
 - components not engineered for parallel scenario
- Showed
 - could get benefit out of parallelism
 - exposed issues need to address to do it right
 - ...why need to do something different

Caltech CS184 Spring2005 -- DeHon

34

Mechanisms...

Caltech CS184 Spring2005 -- DeHon

35

Design for Parallelism

- To do it right
 - need to engineer for parallelism
- Optimize key **common cases** here
- Figuring out:
 - what goes in hardware vs. software

Caltech CS184 Spring2005 -- DeHon

36

Vision: MDP/Mosaic

- Single-chip, commodity building block
 - [today, tile to step and repeat on die]
 - contains all computing components
 - compute: sequential processor
 - interconnect in space: net interface + network
 - interconnect in time: memory
- Step-and-repeat competent μ P
 - avoid diminishing returns trying to build monolithic processor

Caltech CS184 Spring2005 -- DeHon

37

Message Driven Processor

- “Mechanism” Driven Processor?
 - Study mechanisms needed for a parallel processing node
 - address problems saw in using existing
- View as low-level (hardware) model
 - underlies range of compute models
 - shared memory, dataflow, data parallel

[Dally *et. al.*/IEEE Micro, April 1992]

Caltech CS184 Spring2005 -- DeHon

38

Philosophy of MDP

- mechanisms=primitives
 - like RISC focus on primitives from which to build powerful operations
- common support not model specific
 - like RISC not language specific
- Hardware/software interface
 - what should hardware support/provide
 - vs. what should be composed in software

Caltech CS184 Spring2005 -- DeHon

39

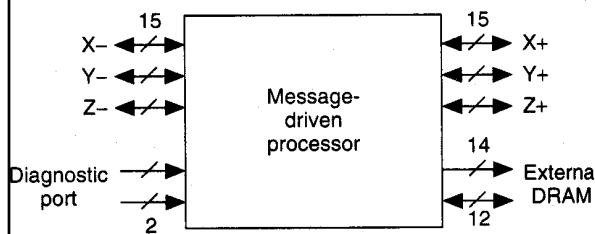
MP Primitives

- SEND message
- self [hardware] routed network
- message dispatch
- fast context switch
- naming/translation support
- synchronization

Caltech CS184 Spring2005 -- DeHon

40

MDP Components

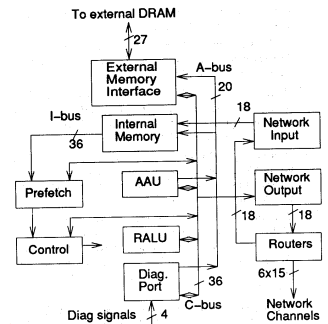


[Dally *et. al.*
IEEE Micro 4/92]

Caltech CS184 Spring2005 -- DeHon

41

MDP Organization



[Dally *et. al.*
ICCD'92]

Caltech CS184 Spring2005 -- DeHon

42

Message Send

- Ops
 - SEND, SEND2
 - SENDE, SEND2E
 - ends messages
- to make “atomic”
 - SEND{2} disable interrupts
 - SEND{2}E reenable

Caltech CS184 Spring2005 -- DeHon

43

Message Send Sequence

- Send R0,0
 - ; first word is destination node address
 - ; priority 0
- SEND2 R1,R2,0
 - ; opcode at receiver (translated to instr ptr)
 - ; data
- SEND2E R2,[3,A3],0
 - ; data and end message

Caltech CS184 Spring2005 -- DeHon

44

MDP Messages

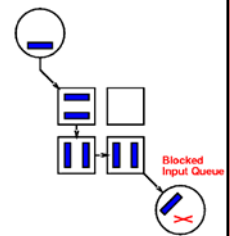
- Few cycles to inject
- Not doing translation here
 - have to map from process to processor before can send
 - done by user code?
 - Trust user code?
 - Deliver to operation (address) on other end
 - receiver translates op to address
 - no protection

Caltech CS184 Spring2005 -- DeHon

45

Network

- 3D Mesh
 - wormhole
 - minimal buffering
 - dimension order routing
- hardware routed
 - orthogonal to node except enter/exit
- messages can backup
 - ...all the way to sender



Caltech CS184 Spring2005 -- DeHon

46

Context Switch

- Why context switch expensive?
 - Exchange state (save/restore)
 - Registers
 - PC, etc.
 - TLB/cache...

Caltech CS184 Spring2005 -- DeHon

47

Fast Context Switch

- General technique:
 - internal vs. external setup
- Machine Tool analogy
- **Pattern:** Double-buffering

- Modern: SMT ... fast change among running contexts...

Caltech CS184 Spring2005 -- DeHon

48

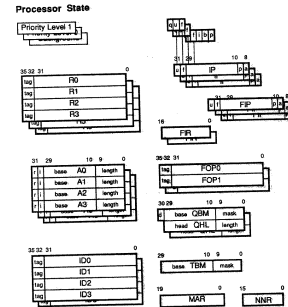
Fast Context Switch

- Provide separate sets of Registers
 - trade space (more, large registers)
 - easier for MDP with small # of regs
 - for speed
- Don't have to go through serialized load/store of state
- Probably also have to assure minimal/necessary handling code in fast memory

Caltech CS184 Spring2005 -- DeHon

49

MDP State



Caltech CS184 Spring2005 -- DeHon

50

Message Dispatch

- Incoming message queued by priority
- If higher priority than running (and interrupts enabled), will start running
 - few cycles to switch to "create" new task
- Terminated with suspend instruction
 - removes message from input queue

Caltech CS184 Spring2005 -- DeHon

51

Message Dispatch

- Idle MPD start running message after 3 cycles
 - set instruction pointer
 - create new message segment
 - A3 is message pointer

Caltech CS184 Spring2005 -- DeHon

52

Message Handler: CALL

- `MOVE [1,A3],R0 ; get method ID`
- `XLATE R0,A0 ; translate to address`
- `LDIP INITIAL_IP ; branch w/in seg`

Caltech CS184 Spring2005 -- DeHon

53

Translation

- **XLATE**
 - associative lookup
 - cache/TLB/mapping primitive
- **ENTER**
 - Used to place an entry in associative table
 - may evict entry
- **PROBE**

Caltech CS184 Spring2005 -- DeHon

54

Translation

- XLATE used to map global ids to local memory
- could be used to map processes to processors?

Synchronization

- Future tags on data
 - [we'll talk about **futures** in the future]

Example

- Combining Tree
 - Each node in tree collects up results from its children
 - Combines results (e.g. add)
 - sends combined result to parent
- Used to collect results of distributed computation

Sample code: Combining Tree

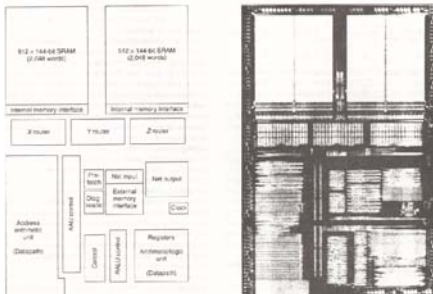
COMBINE:

- MOVE [1,A3],COMB
 - MOVE [2,A3], R1
 - ADD R1,COMB.v,R1
 - MOVE R1,COMB.v
 - MOVE COMB.cnt,R2
 - ADD R2,-1,R2
 - MOVE R2,COMB.cnt
 - BNZ R2, DONE
- MOVE HEADER,R0
 - SEND2 COMB.pnode,R0
 - SEND2E COMB.paddr,R1
- DONE:
- suspend

MDP Area

Size/tech
Of 80386

1.2µm
CMOS



MDP Area

- Memory ~50%
- Processor ~33%
- Net ~10%

Module	Dimensions (mm)	Area (mm ²)	Transistors (x10 ³)
AAU	3.7 × 7.0	25.9	75.0
RALU	3.7 × 2.9	10.7	39.0
Diagnostic	0.9 × 1.1	1.0	3.7
Prefetch	0.9 × 1.1	1.0	3.2
Control	1.1 × 2.6	2.9	8.7
Internal memory interface	7.8 × 0.5	3.9	13.0
External memory interface	1.6 × 1.8	2.9	9.0
Net input	1.8 × 0.7	1.3	4.4
Net output	2.1 × 1.8	3.8	18.0
Routers	8.4 × 1.3	10.9	29.0
RAM	8.8 × 4.9	43.1	880.0
Clock	0.7 × 0.8	0.6	0.1
Pads	50.5 × 0.2	8.4	2.6
Full chip	10.2 × 15.0	153.0*	1,087.0

* Includes wiring between modules.

J-Machine



Caltech CS184 Spring2005 -- DeHon

Performance

- Base communication: $1\mu\text{s}$ node to node
- Empty ping: $3\text{--}7\mu\text{s}$ round trip
 - depends on distance
 - 43 cycles round trip for node pinging self
- MDP 12.5 MIPs
 - 2 MIPs when fetching instructions from external memory

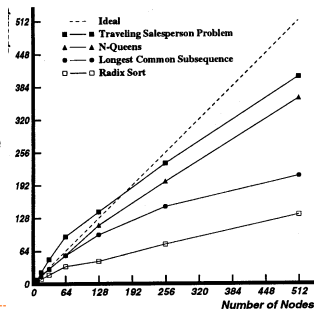
Caltech CS184 Spring2005 -- DeHon

62

Performance Results

Note:

all relative to MDP;
not show slowdown to parallel code and MDP.

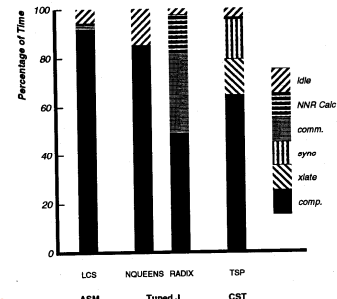


[Noakes, Wallach Dally ISCA'93]

63

Caltech CS184 Spring2005 --

Time Decomposition



[Noakes, Wallach Dally ISCA'93]

64

Caltech CS184 Spring

Other Lessons

- “Mechanisms” important for uniprocessor performance important here as well
 - hardware memory hierarchy management
 - caching, TLB
 - floating point hardware
 - large register set

Caltech CS184 Spring2005 -- DeHon

65

Observation

- Anything with a different programming model is hard to sell
- ...especially if some component of your machine is **worse** than conventional alternatives
 - communication in Cosmic Cube
 - scalar (esp. FP) performance in J-Machine

Caltech CS184 Spring2005 -- DeHon

66

Modern Design

- Doesn't need completely custom ISA
 - (at least, MDP wasn't benefiting from)
 - needed: send, suspend
- Hardware managed hierarchy
 - cache, TLB
- Similar hardware for process/processor mapping

Caltech CS184 Spring2005 -- DeHon

67

Non-Lessons

- Balance
 - network overpowered for node
 - 3× speed of external memory
- Network
 - dimension order routing
 - “efficiency” of wire utilization
 - [will return to ...]

Caltech CS184 Spring2005 -- DeHon

68

Follow ons...

- M-Machine (research)
 - Clustered (VLIW-like) node
- Cray T3D/T3E
 - Alpha's for nodes, 3-cube packet net
- ASCII Red
- MPI
- Myrinet

Caltech CS184 Spring2005 -- DeHon

69

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...

Caltech CS184 Spring2005 -- DeHon

70

Big Ideas

- Common Case
- Primitives
- Highly specialized instructions [hardware mechanisms?] brittle
- Design pulls
 - simplify processor implementation
 - simplify coding

Caltech CS184 Spring2005 -- DeHon

71

Big Ideas

- **Compiler: fill in gap between user and hardware architecture**
 - good idea, not being exploited here
- Need different/additional primitives for handling parallel cooperation efficiently
 - communication
 - cheap process virtualization

Caltech CS184 Spring2005 -- DeHon

72