

---

## Latency Tolerance: what to do when it just won't go away

CS 258, Spring 99  
David E. Culler  
Computer Science Division  
U.C. Berkeley

---

## Reducing Communication Cost

- Reducing effective latency
  - Avoiding Latency
  - Tolerating Latency
- 
- communication latency vs. synchronization latency vs. instruction latency
  - sender initiated vs receiver initiated communication

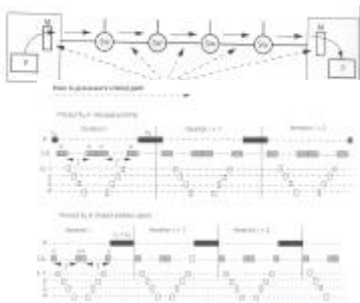
4/30/99

CS258 S99

2

---

## Communication pipeline



4/30/99

CS258 S99

3

---

## Approached to Latency Tolerance

- block data transfer
  - make individual transfers larger
- precommunication
  - generate comm before point where it is actually needed
- proceeding past an outstanding communication event
  - continue with independent work in same thread while event outstanding
- multithreading - finding independent work
  - switch processor to another thread

4/30/99

CS258 S99

4

---

## Fundamental Requirements

- extra parallelism
- bandwidth
- storage
- sophisticated protocols

4/30/99

CS258 S99

5

---

## How much can you gain?

- Overlapping computation with all communication
    - with one communication event at a time?
  - Overlapping communication with communication
    - let C be fraction of time in computation...
- 
- Let L be latency and r the run length of computation between messages, how many messages must be outstanding to hide L?
  - What limits outstanding messages
    - overhead
    - occupancy
    - bandwidth
    - network capacity?

4/30/99

CS258 S99

6

## Communication Pipeline Depth

Table 11.1 Number of One-Way Microbenchmark Messages Outstanding at a Time as Limited by I/O Rate and Rate at Which Messages Can Be Sent to the Network

Machine	One-Way Network Transmission		Remote Read		Machine	One-Way Network Transmission		Remote Read	
	L/2o	Message	L/2o	Message		L/2o	Message	L/2o	Message
DUC C64-5	7.11	250	1.75	181	NOVA-Ultra	1.76	512	1.77	127
IBM Regalis	2.42	181	5.63	133	OW-720	1.00	365	3.11	2380
Alpha C5-2	4.38	74	6.35	63.3	SG/Omega				9,000

4/30/99

CS258 S99

7

## Block Data Transfer

- **Message Passing**
  - fragmentation
- **Shared Address space**
  - local coherence problem
  - global coherence problem



4/30/99

CS258 S99

8

## Benefits Under CPS Scaling

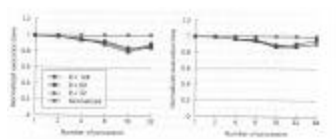


FIGURE 11.16 The benefits of block transfer in the SPECint application. The platform and data.

4/30/99

CS258 S99

9

## Proceeding Past Long Latency Events

- **MP**
  - posted sends, recvs
- **SAS**
  - write buffers and consistency models
  - non-blocking reads

4/30/99

CS258 S99

10

## Precommunication

- **Prefetching**
  - HW vs SW
- **Non-blocking loads and speculative execution**

4/30/99

CS258 S99

11

## Multithreading in SAS

- **Basic idea**
  - multiple register sets in the processor
  - fast context switch
- **Approaches**
  - switch on miss
  - switch on load
  - switch on cycle
  - hybrids
    - » switch on notice
    - » simultaneous multithreading

4/30/99

CS258 S99

12

## What is the gain?

- let  $R$  be the run length between switches,  $C$  the switch cost and  $L$  the latency?
- what's the max proc utilization?
- what the speedup as function of multithreading?
- tolerating local latency vs remote

4/30/99

CS258 S99

13

## Latency/Volume Matched Networks

- K-ary d-cubes?
- Butterflies?
- Fat-trees?
- Sparse cubes?

4/30/99

CS258 S99

14

## What about synchronization latencies?

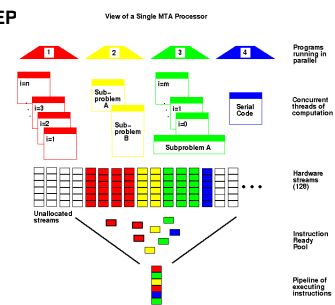
4/30/99

CS258 S99

15

## Burton Smith's MT Architectures

- Denelcor HEP
- Horizon
- Tera



4/30/99

6

## Tera System Pipeline



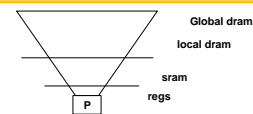
- 64-bit, 3-way LIW
- 128 threads per proc
- 32 regs per thread
- 8 branch target regs
- ~16 deep inst pipe
- 70 cycle mem
- upto 8 concurrent mem refs per thread
  - look-ahead field limit
  - speculative loads/ poison regs
- full/empty bits on every mem word

4/30/99

CS258 S99

17

## A perspective on Mem Hierarchies



- Designing for the hard case has huge value!
- Speedup due to multithreading is proportional to number of msgs outstanding
- Each takes state
  - capacity of a given level limits the number (as does occupancy and frequency)
  - achieve more tolerance by discarding lower-level
  - huge opportunity cost
- Putting it back in enhances 'local' performance and reduces frequency
- Natural inherent limits on speedup

4/30/99

CS258 S99

18