Random Numbers

- The ZPL book defines the function: `llrand()`
- This is a high quality generator, yielding a pseudo-random stream of scalars, r_0, r_1, r_2 ...
- Assigning

```
A := llrand(seed); -- set whole array to
                   -- same random value
```

- How to generate an array of random numbers?

```
A := llrand(Seed); -- set elements to new
                   -- random values
```

- The question is, how to initialize Seed to produce an array of independent streams

Random Numbers, Continued

- One time initialization of an array to a random set of values works as follows ...

```
for   i := 1 to n do
  for j := 1 to m do
[i,j] A := llrand(seed);
  end;
end;
```

r_0	r_1	r_2	r_3
r_4	r_5	r	r_7

- For random arrays, pick a larger separation

```
for   i := 1 to n do
  for j := 1 to m do
[i,j] A := llrand(seed);
      for k := 1 to 9999 do   -- spin generator
        temp := llrand(seed); -- to separate
      end;                    -- samples
  end;
end;
```

r_0	r_{10000}	r_{20000}	r_{30000}
r_{40000}	r_{50000}	r_{60000}	r_{70000}

Dealing With Latency

Latency -- the time required to perform a memory operation or interprocessor communication continues to be large relative to processor speed. What can be done?

Relaxed Consistency Models

The consistency model for a shared address memory computer specifies the constraints on the order in which memory operations can appear to execute with respect to each other

- Programmers expect sequential consistency because it is "comprehensible"
- SC is rigid, resulting in poor performance ... duh
- Relaxed consistency is any alternative set of rules describing the ordering on memory operations
- Relaxed consistency models are generally hard to use and understand -- basically a bad idea

Parallel programming is already difficult enough

Basically A Good Idea

Use parallelism to cover latency

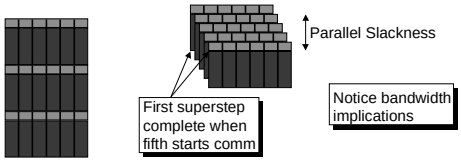
- J.T. Schwartz example:
  - Find maximum of P numbers
    - O(log P) using Ladner/Fischer algorithm
    - With nothing else to do ... wait for answer
  - Find P maxima of P sets of P numbers
    - O(log P) for each, but interleaved O(log P) for all
    - Time to perform each maximum is a constant

Another application of basic pipelining

Latency Hiding In Model Of Computation

Valiant's Bulk Synchronous Parallel (BSP) model applies latency hiding to computational model

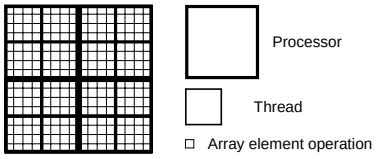
- Supersteps: [Computation; Communication]
- Parallel Slackness -- amount of parallelism needed to cover communication latency



In ZPL ...

Because ZPL's parallelism is implicit, a program can be partitioned into any number of separate parallel threads

- Example: 4P threads to run on P processors



ZPL's Efficient Code Generation

- ZPL's generated code overlaps computation with communication to the maximum extent possible
- The machine independent optimizations due to Sung-Eun Choi
- Ironman calls DR(), SR(), DN(), SV() allow ZPL to exploit whatever the latency covering features the machine may have

Specifying the computation at a high level lets the compiler deal with latency hiding

ZPL's Latency Tolerance

- There are two ways ZPL exploits blocked data transfer
  - Vectorization moves array slices as a single unit -- ZPL naturally vectorizes because it is compiling array operations
  - Combining communications to the same destination reduces the overhead, benefits from pipelining
- Communication is also pipelined, allowing communication to overlap with computation
- Goals of combining and pipelining can conflict

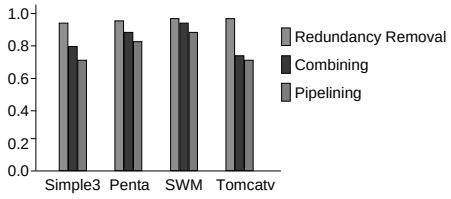
Choi's Optimizations

Schematic of Optimizations (using send/recv)

Naive	Remove Redundant Comm	Combine	Pipeline
U.,. := ...	U.,. := ...	U.,. := ...	U.,. := ...
...	...	...	...
send(U)	send(U)		send(U,V)
recv(U)	recv(U)		
send(V)	send(V)	send(U,V)	
recv(V)	recv(V)	recv(U,V)	recv(U,V)
aux2.,. := ...	aux2.,. := ...	aux2.,. := ...	aux2.,. := ...
send(V)		send(V)	
recv(V)		recv(V)	
aux4.,. := ...	aux4.,. := ...	aux4.,. := ...	aux4.,. := ...

Choi's Numbers

Cray T3D performance scaled to naive



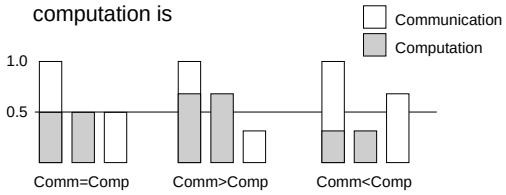
Basic LT Machine Design

Effectively tolerating latency requires some hardware assistance

- A naive hardware implementation generally doesn't have enough ability to hide latency with concurrency
  - Communication coprocessor
  - Multithreading support
  - NOWs fall short
- Where appropriate, caching is essential

Overlap Communication w/Computation

The upper bound on performance improvement by overlapping communication with computation is



Latency Tolerance In Architecture

Multithreading is an architectural approach in which multiple threads-of-execution are run "simultaneously"

- Requires no special software except more threads than processors
- Can handle both predictable and unpredictable situations
- Handles long latencies no matter what the cause
- Doesn't affect the memory consistency model

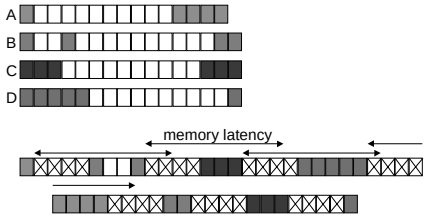
$$\text{Utilization} = \frac{\text{Busy}}{\text{Busy + Switching + Idle}}$$

Two Techniques For Multithreading

- Blocked Multithreading [Alewife], like time sharing ... continue to execute until thread is blocked, then switch
  - Has lower hardware impact
  - Good single thread performance
- Interleaved Multithreading [Tera], switch execution of threads on each cycle
  - Lower logical switching penalty
  - Greater impact on hardware design

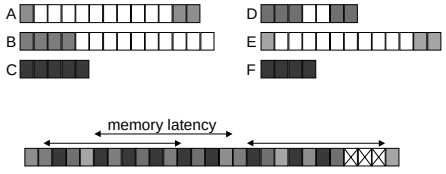
Keeping multiple contexts is essential

Four Threads, Blocked Approach



Utilization is 41%

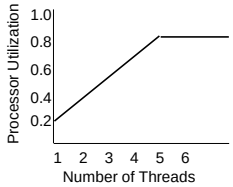
Six Threads, Interleaved Approach



Utilization is 89%

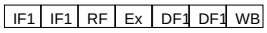
Benefits Of Available Threads

For the blocked approach the availability of ready threads improves utilization



Affects Of Pipelining

When a (memory) block comes, it is is detected in the pipeline



How to handle instructions in the pipe?

- Complete while fetching new thread -- complex
- Complete before fetching new thread
- Squash the instructions

Basics of Denelcor HEP

- First interleaved multithreaded machine (78-85)
- Each processor had 64 user contexts and 64 privileged contexts, 128-way replicated register file and state
- Contention-free memory (20-40 cycles) in a dancehall design
- Processor had 8 deep pipeline, but only one memory, branch or divide could be in pipe at a time

Basics Of Tera Design

Instructions are [arithmetic, control, memory] or [arithmetic, arithmetic, memory]

- Ready instructions issue on each tick, but there is a 16 tick minimum issue delay for consecutive instructions from a thread
- Each (memory) instruction has a 3 bit tag telling how many instructions forward are independent of this memory reference
- Average memory latency w/o contention 70 cycles

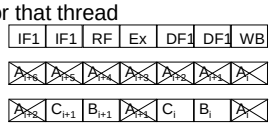
More On Tera

- Since there is a 16 instruction minimum issue it takes 16 threads to keep utilize the processor without hiding latency
- Each processor has 128 fully replicated contexts
- Synchronization latency can even be covered
- When everything works, the Tera should approximate a PRAM

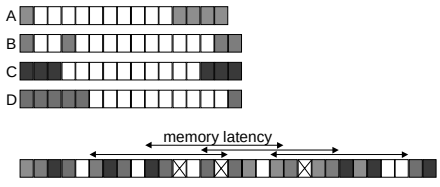
An Alternative Design

Combine the best of the blocked and interleaved approaches

- Use a standard processor
- Issue instructions from each ready thread, fairly
- When a memory operation makes tread unready, squash any later issued instructions for that thread



Four Threads For Interleaved Scheme



Utilization is 70%

Latency Tolerance Summary

- Two main approaches: blocked & interleaved
- Approaches differ in their single thread performance
- It may be tough to find all those threads w/o language or programmer assistance
- Programming on the assumption of aggressive latency tolerance may yield a very unportable program
- Some further discussion in Section 11.7

Reading

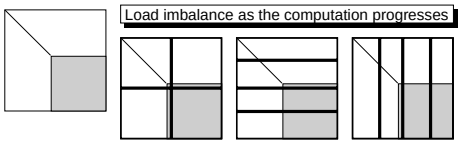
- J. T. Schwartz, Ultracomputers, ACM ToPLAS
- Valiant BSP
- Sung-Eun Choi, "Machine Independent Communication Optimization", PhD Dissertation, University of Washington, 1999
- B. J. Smith, Architecture and Applications of the HEP Multiprocessor, Proc. SPIE: Real Time Signal Processing IV 298, pp 241-248

Parallel Algorithmic Techniques

The goal in (practical) parallel algorithm design is to express parameterized parallelism (so it can be scaled to the actual number of processors available) that minimizes communication and synchronization, and has good load balance

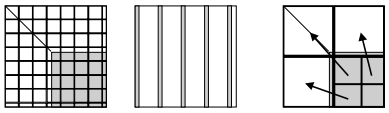
Parallel Algorithms: LU Decomposition

- Solving systems of linear equations is a critical part of many scientific computations
- Recall that the standard solution "marches" to the lower right corner of the matrix, leading to poor load balance



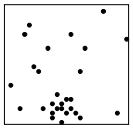
Solutions To Load Balance

- The most common balancing scheme is to allocate the array block cyclically
- Lennart Johnsson has observed that marching to the corner is not necessary, that the eliminations can be strided
- And it's always possible to reallocate



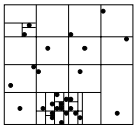
Algorithms: N-body Computations

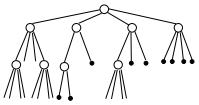
- Some N-body computations require all n^2 pairwise interactions to be calculated
- For others interactions involving distant bodies can be ignored or approximated by a point mass, leading to more efficient execution
- Allocating bodies spatially eases communication load



N-body Representation

To exploit the fact that only nearby attractions need to be explicitly calculated, partition space, inducing an oct-tree, traverse the oct tree computing the attractions, update positions





...

The 2D version would uses a quad tree

N-body (Barnes Hut) Algorithm

- Construct the tree
- Compute the attractions of the other points by traversing the tree; at a node, if the bodies are close, computer pairwise attractions; if they are distant, compute approximation and do not traverse any lower
- Totality of attractions induces a new position
- Variations --
  - Alternative tree structures
  - Salmon uses an out of core algorithm using a space filling curve to promote locality