Performance Evaluation

- **Why?**
  - Evaluate tradeoffs
  - Estimate machine performance
  - Measure application behavior

- **How?**
  - Ask an expert --- hire a consultant?
  - Measure existing machines --- (What?!)
  - Build simulators
  - Analytical models
  - Hybrids --- combination of 2-3-4
In the lab

Parallel Traces

Network Model

Parallel Address Trace

Cache coherence simulator

Event frequencies

Compute

Msg probability, size

Network Model

Processor Utilization
Example

1. Trace simulation, awk filter

<table>
<thead>
<tr>
<th>Event</th>
<th>%Prob</th>
<th>out-msg</th>
<th>out-siz</th>
<th>in-msg</th>
<th>in-siz</th>
</tr>
</thead>
<tbody>
<tr>
<td>read miss in write mode</td>
<td>1.2507</td>
<td>yes</td>
<td>4</td>
<td>yes</td>
<td>flits</td>
</tr>
<tr>
<td>other msgs</td>
<td>2.8483</td>
<td>yes</td>
<td>4</td>
<td>yes</td>
<td>4</td>
</tr>
</tbody>
</table>

* : in-siz: 4+ (block size/flit size)

2. Compute network parameters \( m, B \)

\[
m = \sum_{events} \left( \text{msgs. per event} \right) \times \left( \frac{\% \text{ prob}}{100} \right)
\]

\[
B = \frac{\sum_{events} \left[ \sum_{msgs} \text{msgsize} \right]}{m} \times \left( \frac{\% \text{ prob}}{100} \right)
\]

3. Compute processor utilization

Given \( m, B \) and \( k_d, n, N \)

Derive \( U \).
**Evaluation**

- Barriers implemented using distributed trees
- Read-only sharing marked

**Weather**

Processor utilization

<table>
<thead>
<tr>
<th>Pointer</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td>64</td>
<td>1.0</td>
</tr>
</tbody>
</table>

- LimitLESS  
- No coherence  
- 1 chain  
- 2 chain  
- Write through = limited dir

**Speech**

Processor utilization

<table>
<thead>
<tr>
<th>Pointer</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
</tr>
<tr>
<td>4</td>
<td>0.8</td>
</tr>
<tr>
<td>64</td>
<td>1.0</td>
</tr>
</tbody>
</table>

- LimitLESS  
- No coherence  
- 1 chain  
- 2 chain  
- Write through = limited dir
Evaluation
Full system simulation (coupled)

- Processor simulator
- Cache and memory systems simulator
- Interconnection network simulator

Wait, traps

Memory requests

Synchronization requests

Compiled application

Wait, traps

Synchronization requests

Acknowledgements, responses

Network requests

Measures:
- Speedup, runtime
- proc. util.
- net latency
- cache miss rate

4000x slowdown per node

Very accurate

ECE669 L20: Evaluation and Message Passing
Trace-driven simulation (coupled)

- Sequential address trace
- Trace scheduler
- Port ready
- Address trace
- Proc0 Addr
- Proc1 Addr
- Proc2 Addr
- Address
- Network requests
- Network simulator
- Acknowledgements, responses
- Cache and memory systems simulator
- Processor
- Memory
- Inter-connection network
- 2000x slowdown per node
- Very accurate

E.g. From M.I.T. many parallel traces exist

ECE669 L20: Evaluation and Message Passing
Hybrid Network Model

Processor simulator

Cache and memory system simulator

Network model

Processor

Memory

Interconnection network

Compiled application

Wait

Response, latency

Requests

Time window average

800x slowdown per node
Fair accuracy

Compiled application

Parallel address trace

Requests
Many hybrids possible

ECE669  L20: Evaluation and Message Passing

200x slowdown per node
Fair accuracy
Hybrid - Network model

Compiled application

Direct execution
- Round robin/process
- Switch on mem. req.

Switch

Memory requests

Cache and memory system simulator

Responses, latency

Network requests, time window ave

Network model

Processor

Memory

Inter-connection network

100x slowdown (30x with threads)
Fair accuracy
Trace-driven (decoupled)

Processor

Memory

Interconnection network

Cache and memory system simulator

Network simulator

No trace scheduler!

No feedback

Responses

Network requests

Address trace

Synchronization constraints may be violated

Garbage!
Message passing

- **Bulk transfers**

- **Complex synchronization semantics**
  - more complex protocols
  - More complex action

- **Synchronous**
  - Send completes after matching recv and source data sent
  - Receive completes after data transfer complete from matching send

- **Asynchronous**
  - Send completes after send buffer may be reused
Synchronous Message Passing

- Constrained programming model.
- Deterministic! What happens when threads added?
- Destination contention very limited.
- User/System boundary?

1. Initiate send
2. Address translation on $P_{src}$
3. Local/remote check
4. Send-ready request
5. Remote check for posted receive (assume success)
6. Reply transaction
7. Bulk data transfer
   Source VA $\rightarrow$ Dest VA or ID

Source

Destination

Processor Action?

Send $P_{dest}$ local VA, len

Send-rcy req

Wait

Tag check

Send-rcy reply

Data-xfer req

Bulk data transfer

Time

ECE669  L20: Evaluation and Message Passing

April 13, 2004
Asynchronous Message Passing: Optimistic

- More powerful programming model
- Wildcard receive => non-deterministic
- Storage required within msg layer?
Asynchronous Message Passing: Conservative

- Where is the buffering?
- Contention control? Receiver initiated protocol?
- Short message optimizations
Key Features of Message Passing Abstraction

- **Source knows send data address, dest. knows receive data address**
  - after handshake they both know both

- **Arbitrary storage “outside the local address spaces”**
  - may post many sends before any receives
  - non-blocking asynchronous sends reduces the requirement to an arbitrary number of descriptors
    - fine print says these are limited too

- **Fundamentally a 3-phase transaction**
  - includes a request / response
  - can use optimisitic 1-phase in limited “Safe” cases
    - credit scheme
Active Messages

° User-level analog of network transaction
  • transfer data packet and invoke handler to extract it from the network and integrate with on-going computation

° Request/Reply

° Event notification: interrupts, polling, events?

° May also perform memory-to-memory transfer
Common Challenges

- **Input buffer overflow**
  - N-1 queue over-commitment => must slow sources
  - reserve space per source (credit)
    - when available for reuse?
      - Ack or Higher level
  - Refuse input when full
    - backpressure in reliable network
    - tree saturation
    - deadlock free
    - what happens to traffic not bound for congested dest?
  - Reserve ack back channel
  - drop packets
  - Utilize higher-level semantics of programming model
Summary

- Evaluation – important to understand intermediate messages in cache protocol
- Message sizes may vary based on function
- Two main types of message passing protocols
  - Synchronous and asynchronous
- Active messages involve remote operations
- Message techniques depend on network reliability