Cache Coherent Synthetic Traffic Models

Mario Badr

Supervisor: Natalie Enright Jerger
Evaluating NoC Performance

Full System Simulator
- Processor
- Cache
- Disk
- Other Components

NoC Simulator
- NoC

Application

Packets Sent
Packets Arrived

Feedback!

Long Simulation Time
Very Accurate
Evaluating NoC Performance

Synthetic Traffic Driver

Traffic Pattern
- Uniform Random
- Bit Complement
- Bit Reverse
- Bit Rotation
- Shuffle
- Transpose
- Tornado
- Neighbour

NoC Simulator

Cache Coherent Synthetic Traffic Models
Spatial Behaviour of Traffic

Synthetic Traffic

Full System Simulation

Shuffle Traffic Pattern

FFT Benchmark

Cache Coherent Synthetic Traffic Models
Evaluating NoC Performance

Trace Simulator

Medium Simulation Time Not Accurate

No Feedback

NoC Simulator

Packets Sent

Application

Processor Cache Disk Other Components

Trace

NoC A

NoC B

Feedback!
NoC Performance Comparison

![Average Packet Latency Graph](image)

- Blackscholes
- Bodytrack
- Canmeal
- Facesim
- Ferret
- Fluidanimate
- Raytrace
- Streamcluster
- Swaptions
- Vips

Cache Coherent Synthetic Traffic Models
Simulation Methodology Review

• Full System Simulation
  – Model Each Component
  – Very Accurate
  – Long Simulation Time

• Synthetic Traffic
  – Traffic Patterns Based on Applications
  – Not Accurate
  – Very Short Simulation Time

• Trace Simulation
  – Most Temporal & Spatial Behaviour Captured
  – Short Simulation Time
  – Not Accurate (lack of feedback)
Overview

Full System Simulator

- Processor
- Cache
- Disk
- Other Components

NoC Simulator

- NoC

Application

Packets Sent

Packets Arrived

Feedback!
Overview

Cache Coherent Synthetic Traffic Models

Full System Simulator

NoC Simulator

Application

Packets Sent

Packets Arrived

Processor

Cache

Reactive Traffic Model

Other Components

NoC

Feedback!
1. Send Packets

Application → Reactive Traffic Model → NoC

Packets Sent → Feedback!

Packets Arrived
1. Send Packets

2. React to Packets

Traffic Model

Other Components

Packets Arrived

Feedback!

Cache Coherent Synthetic Traffic Models
1. Send Packets

2. React to Packets

3. Compare to Full System
Applications Have Time-Varying Behaviour; they go through *phases*
Modelling Time-Varying Behaviour

**Want:** Send Packets with Time-Varying Behaviour

**Need:** Methodology to create and group phases
1. Divide traffic into *Intervals*
2. Represent intervals with *Feature Vectors*
3. Group feature vectors with *Clustering*

**Need:** Methodology to transition from one phase to another
- *Markov Chains*
Dividing Into Intervals

Intervals are a fixed size.

Fluidanimate Benchmark

Packets Injected

Time Bin (500,000 cycles per bin)

Cache Coherent Synthetic Traffic Models
Dividing Into Intervals

Visually we see: High, Low+High, and Low Intervals
Creating Phases

• Characterize intervals with vectors:
  1. Total Injection (TI)
  2. Coherence Composition
  3. Node Injection (NI)
  4. Row-Column Flow (RCFlow)
  5. Per-Node Flow (Flow)

• Group vectors with clustering algorithms

\(<N_1, N_2, N_3, ..., N_9>\)
Phase Transitions

Given current state, there is some probability to transition to the next state.
Modelling Time-Varying Behaviour

**Want:** Send Packets with Time-Varying Behaviour

**Need:** Methodology to create and group phases
1. Divide traffic into *Intervals*
2. Represent intervals with *Feature Vectors*
3. Group feature vectors with *Clustering*

**Need:** Methodology to transition from one phase to another
- *Markov Chains*
• Intervals also have time-varying behaviour!

• Create two levels:
  1. Macro Interval (big)
  2. Micro Interval (small)

• Repeat phase grouping for second level

• Hierarchical Model
Macro-Scale Model

Blackscholes Benchmark

Apply Micro-Scale Model

Total Number of Packets

Time
Micro-Scale Model

Swaptions Benchmark

- Reads
- Invalidates
- Writes

Number of Packets

Micro-Phase (Based on Clustering)

Time
Modelling Cache Coherent Behaviour

**Want:** Send Packets in a Cache Coherent manner

**Need:** Model for *initiating Cache Coherent transactions*
1. Write Requests
2. Read Requests

**Need:** Model for Cache Coherence *reactions*
1. Accessing Memory
2. Invalidate Behaviour
3. Other Behaviour
Cache Coherence Example

Two Sharers. Write Request initiates a transaction.
Write Request arrives at $D_A$
Invalidate arrives at $C_B$. Forward Request arrives at $C_C$. 

3) Data Response
3) Acknowledgement
Cache Coherence Example

Unblock signals end of the transaction.
Modelling Cache Coherent Behaviour

Active Model
• Write/Read Requests
  – How many?
  – Destination?
• Injected uniformly over a micro interval

Reactive Model
• Forward Requests
  – Probability of forwarding a request
  – Destination?
• Invalidate(s)
  – How many? (Could be zero)
  – Destination(s)?
• Other
  – Simplified Coherence Protocol
Modelling Cache Coherent Behaviour

**Want:** Send Packets in a Cache Coherent manner

**Need:** Model for *initiating Cache Coherent transactions*
1. Write Requests
2. Read Requests

**Need:** Model for Cache Coherence *reactions*
1. Accessing Memory
2. Invalidate Behaviour
3. Other Behaviour
Model Parameters

• Clustering Quality
  – Which feature vector?
  – Number of traffic phases
  – Which interval size?

<table>
<thead>
<tr>
<th></th>
<th>Macro-Level</th>
<th>Micro-Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Vector</td>
<td>Node Injection</td>
<td>RCFlow</td>
</tr>
<tr>
<td>Formal Method</td>
<td>Calinski-Harabasz</td>
<td>L-Method</td>
</tr>
<tr>
<td>Interval Size</td>
<td>500,000</td>
<td>200</td>
</tr>
</tbody>
</table>
## Network Configurations

<table>
<thead>
<tr>
<th>Network</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topology</td>
<td>Mesh</td>
<td>Mesh</td>
<td>Flattened Butterfly</td>
</tr>
<tr>
<td>Channel Width</td>
<td>8 bytes</td>
<td>4 bytes</td>
<td>4 bytes</td>
</tr>
<tr>
<td>Virtual Channels</td>
<td>2 per port</td>
<td>2 per port</td>
<td>4 per port</td>
</tr>
<tr>
<td>Routing Algorithm</td>
<td>XY</td>
<td>Adaptive YX-XY</td>
<td>UGAL</td>
</tr>
<tr>
<td>Buffer Depth</td>
<td></td>
<td>8 flits</td>
<td></td>
</tr>
<tr>
<td>Router Pipeline</td>
<td></td>
<td>4 stages</td>
<td></td>
</tr>
</tbody>
</table>

Cache Coherent Synthetic Traffic Models
NoC Performance – Latency

Network A

Average Packet Latency

[Bar chart showing latency comparison for different workloads and traffic models.]
NoC Performance – Latency
NoC Performance – Latency

![Chart showing latency comparison for different benchmarks and models.]
NoC Performance – Latency Distribution

Network A

Hellinger Distance

barnes  blackscholes  bodytrack  cholesky  facesim  fft  fluidanimate  lu_cb  lu_ncb  radiosity  radix  raytrace  swaptions  volrend  water_nsquared  water_spatial

Synthetic  Trace.Dependency
NoC Performance – Latency Distribution

- Synthetic
- Trace.Dependency

Network B

Hellinger Distance

barnes blackscholes bodytrack cholesky facesim fft fluidanimate lu_cb lu_ncb radiosity radix raytrace swaptions volrend water nsquared water spatial
NoC Performance – Latency Distribution

- Synthetic
- Trace.Dependency

Network C

Hellinger Distance

- barnes
- blackscholes
- bodytrack
- cholesky
- facesim
- fft
- fluidanimate
- lu_cb
- lu_ncb
- radiosity
- radix
- raytrace
- swaptions
- vold
- water_nsqared
- water_spatial
Speed Up

![Graph showing Speed Up for different benchmarks with categories: Synthetic, Synthetic_SS, and Trace.Dependency.]
Conclusion

• Implemented Synthetic Traffic Models that are
  – Accurate: 10.5% average erro
  – Fast: Over 50x average speed up
  – Coherent: Packets resemble cache coherent traffic

• Future Work
  – Sweeping micro-architectural configurations