String Figure: A Scalable and Elastic Memory Network Architecture

Matheus Ogleari
Ye Yu, Chen Qian, Ethan Miller, Jishen Zhao
HPCA 2019
Issues with Memory Capacity Bottleneck

Training neural networks is data intensive, and increasingly so.
Why Memory Network

• Memory Networks
Why Memory Network

- Memory Networks
- More sockets, more problems

[Zhao, TACO 2015]
Design Goals

Our Goals

- Scalability
- Arbitrary Network Scale
- Elastic Network Scale
Design Goals

Scalability

Our Goals

Arbitrary Network Scale

Elastic Network Scale
Design Goals

Scalability

- Arbitrary Network Scale
- Elastic Network Scale

A Quad-socket Server System

Memory Node

Processor
Design Goals

Our Goals

- Scalability
- Arbitrary Network Scale
- Elastic Network Scale

A Quad-socket Server System
Design Goals

Scalability
Arbitrary Network Scale
Elastic Network Scale

A Quad-socket Server System
Memory Node
Processor
Design Goals

Our Goals

- Scalability
- Arbitrary Network Scale
- Elastic Network Scale

A Quad-socket Server System
Design Goals

Scalability

Arbitrary Network Scale

Elastic Network Scale

A Quad-socket Server System

Memory Node

Processor
Design Goals

Our Goals

Scalability

Arbitrary Network Scale

Elastic Network Scale

A Quad-socket Server System
String Figure:
A Scalable and Elastic Memory Network Design

Our Goals
- Scalability
- Arbitrary Network Scale
- Elastic Network Scale
Topology design

Our Goals

- Scalability
- Arbitrary Network Scale
- Elastic Network Scale

Topology is a function of number of router ports

Random Topology
Random topology generation

Number of spaces = Number of ports \div 2

Random number
Short cuts

Our Goals

- Scalability
- Arbitrary Network Scale
- Elastic Network Scale

Random Topology
Reconfigurable router

Output

Random Connections

Short Cuts

Input
# Greedy routing protocol

<table>
<thead>
<tr>
<th>Node #</th>
<th>Node # Block. Valid</th>
<th>Hop</th>
<th>Space#</th>
<th>Coordi.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Minimum distance (MD) to Node-2 from Node-7 and Node-7’s neighbors

Routing table entries
Implementation

Design Parameters:
• Payload Size (bits)
• # of net ports
• # of routers
• # of router ports
Simulation

Parameters

- S2 Topology gen script (*.py)
- Topology File (*.Topo)
- Network RTL (*.v)
- PyMTL Wrapper (*.py)

workloads → McSimA+ simulator → net sim (*.py) → Unit Tests (*.py)

memory trace → results

- waveform (*.vcd)
- waveform viewer

- Future Work
  - SystemVerilog to PyMTL (Python)
  - Open source
### Evaluation

<table>
<thead>
<tr>
<th>Topology</th>
<th>Number of Nodes (N), Number of Ports per Router (p)</th>
<th>Routing Scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed-Mesh (DM) / Optimized DM (ODM)</td>
<td>N 16 17 32 61 64 113 128 256 512 1024 1296</td>
<td>Greedy + adaptive</td>
</tr>
<tr>
<td></td>
<td>p 4 n/a 4 n/a 4 n/a 4 4 4 4 4</td>
<td></td>
</tr>
<tr>
<td>Flattened Butterfly (FB)</td>
<td>p 20 24 31 33</td>
<td>minimal + adaptive</td>
</tr>
<tr>
<td>Adaptive FB (AFB)</td>
<td>p 13 17 23 25</td>
<td>minimal + adaptive</td>
</tr>
<tr>
<td>Space Shuffle Ideal (S2-ideal)</td>
<td>p 4 4 4 4 4 4 8 8 8</td>
<td>look-up table (LUT)</td>
</tr>
<tr>
<td>String Figure (SF)</td>
<td>p 4 4 4 4 4 4 8 8 8</td>
<td>LUT + greedy + adaptive</td>
</tr>
</tbody>
</table>
Experiment Setup

- **Traffic Patterns** – specific routing behaviors in networks
  - Uniform Random
  - Tornado
  - Hotspot
  - Opposite

- **Workloads** – real-world applications for memory networks
  - Spark-wordcount
  - Spark-grep
  - Spark-sort
  - Pagerank
  - Nearest Neighbor
  - Complement
  - Partition
  - Redis
  - Memcached
  - Matrix Multiply
  - Kmeans
Results
Results

Latency (cycles) vs. Injection Rate (%) for different configurations:
- DM-32
- ODM-32
- S2-32
- SF-32
- DM-1024
- ODM-1024
- S2-1024
- SF-1024
- AFB-1024
- FB-1024

Uniform Random pattern
Results

![Graph showing latency vs. injection rate for different cases.]

- **Ideal case**
- **Uniform Random**
Results

![Diagram showing hop count vs. number of memory nodes for different designs: DM, ODM, FB, AFB, S2-ideal, SF.](image)

- The hop count increases as the number of memory nodes increases.
- The designs show varying performance with DM, ODM, FB, AFB, S2-ideal, and SF demonstrating different patterns.

**Note:** Real-memory-based designs such as DM, ODM, FB scales better than the other approaches that rely on a network of short-cut routers as the number of nodes increases.
Results

- More in the paper!
  - Traffic pattern latencies
  - Real workload performance
  - Network saturation
  - Energy-Delay Product (EDP)
  - Network scaling
  - Deadlock avoidance
Summary of String Figure

Benefits

- Scalability
- Arbitrary Network Scale
- Elastic Network Scale
String Figure: A Scalable and Elastic Memory Network Architecture

Matheus Ogleari
Ye Yu, Chen Qian, Ethan Miller, Jishen Zhao
HPCA 2019
A Quad-socket Server System

Memory Node

Processor
A Quad-socket Server System

Memory Node

Processor