Multi-Node Programming – Longest IP Prefix Matching: A Stream Application using Multiple Imagines in Different Configurations

Henry Fu, Harn Hua Ng, Yeow Cheng Ong
Stanford University
EE482C Project Presentation
Thursday, May 30, 2002
Outline

• Motivation
• Goals
• Application: IP routing
• Setup
• Test methods, data, metric
• Results
• Challenges
• Conclusions
Motivation

• Develop and evaluate methods to efficiently map stream programs over multiple stream processing nodes
• Examine ways to partition data and/or instructions across the nodes
• Develop methods to coordinate multiple nodes and to communicate data
• Evaluate methods for load balancing
Goals

• Multi-node programming using multiple Imagines
  – Provide more computing power and higher performance
    • Requires more memory bandwidth and higher communication overhead

➢ Investigate different configurations that give best performance with least overhead
Introduction

• IP packet routing commonly used and can be mapped as a stream application
  – Each packet is independent
    • Data Level Parallelism (DLP)
  – Multiple flows of packets in router can be mapped as different streams of data
    • Thread Level Parallelism (TLP)
  – Same instruction can be distributed to multiple ALUs to perform multiple operations in parallel
    • Instruction Level parallelism (ILP)
Overview

• IP Routing
  – Extract IP address information from each packet, compared against a routing table, and re-routed to appropriate nexthop address
  – IP Packet traffic modeled as data stream
  – After each lookup, each processor passes longest match result, along with current packet to a neighboring processor of another node to continue longest prefix matching
Algorithm used for IP address matching

– Within a Kernel:
  • Distribute routing table entries to all clusters
    – i.e. mask, destination address, nexthop
  • Find mask length for each routing table entries
  • Find match
    – (Packet address) AND (mask) XOR (destination address)
  • Keep track of length of longest prefix match, and corresponding next hop
Setup

• Baseline case
  – Use 1 host processor and 1 Imagine
  – 1 parallel data lane, 1 pipeline stage
  – All results normalized according to baseline case results
Setup (More)

• 2 Imagines
  – Use 1 host processor and 2 Imagines
  – 1 parallel data lane, 2 pipeline stages
  – 2 parallel data lanes, 1 pipeline stage
Setup (More)

• 4 Imagines
  – Use 1 host processor and 4 Imagines
  – 1 parallel data lane, 4 pipeline stages
Setup (More)

• 4 Imagines
  – 2 parallel data lanes, 2 pipeline stages
  – 4 Parallel data lanes, 1 pipeline stage
Configurations

- Pipelined configuration: total # of routing table entries distributed evenly to all Imagine processors in each pipeline stage
  - Static load balancing

- Parallel configuration: total # of destination addresses distributed evenly to all data streams
  - Static load balancing
Test Methods

- Program written in StreamC and KernelC
- Profiling used to estimate cycle count in each kernel and total execution time

- Number of Imagines used: 1, 2, and 4
- Number of test packets used: 8, 32, 1024
- Number of routing entries used: 8, 32, 1024
Test Data

• Randomly-generated destination addresses
• Routing table entries captured from major router in ISP
  – ner-routes.bbnplanet.net
  – 119, 967 entries captured
  – Subset of total entries randomly picked for experiment
  – C program to generate correct results and to verify output of stream program
Test Metric

• Execution time of single Stream Processor configuration vs. that of multi-node configuration
  – 1, 2, 4 Imagines arranged in pipelined configuration vs. 1 Imagine configuration
  – 1, 2, 4 Imagines arranged in parallel configuration vs. 1 Imagine configuration

• Communication overhead examined in > 1 Imagine configuration
Test Results

- Pipelined Configuration
  - Almost ideal speed up for large data set
  - Significant overhead for small data set

<table>
<thead>
<tr>
<th>Pipelined</th>
<th>Execution Time</th>
<th># Packets</th>
<th># Entries</th>
<th># Imagines</th>
<th>Imagine 0</th>
<th>Imagine 1</th>
<th>Imagine 2</th>
<th>Imagine 3</th>
<th>Avg/Img</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Imagine 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1024</td>
<td>1</td>
<td>52325</td>
<td>52325</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25636</td>
<td>2</td>
<td>26721</td>
<td>26178.5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.99877762</td>
</tr>
<tr>
<td>4</td>
<td>12260</td>
<td>4</td>
<td>13408</td>
<td>13408</td>
<td>13345</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>3.99267469</td>
</tr>
<tr>
<td>32</td>
<td>1024</td>
<td>1</td>
<td>209300</td>
<td>209300</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>102544</td>
<td>2</td>
<td>106884</td>
<td>104714</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.99877762</td>
</tr>
<tr>
<td>4</td>
<td>49040</td>
<td>4</td>
<td>53632</td>
<td>53632</td>
<td>53380</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>3.99267469</td>
</tr>
<tr>
<td>1024</td>
<td>1024</td>
<td>1</td>
<td>6697600</td>
<td>6697600</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3281408</td>
<td>2</td>
<td>3420288</td>
<td>3350848</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.99877762</td>
</tr>
<tr>
<td>4</td>
<td>1569280</td>
<td>4</td>
<td>1716224</td>
<td>1716224</td>
<td>1708160</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>3.99267469</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>1</td>
<td>1669</td>
<td>1669</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>833</td>
<td>2</td>
<td>868</td>
<td>850.5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.96237507</td>
</tr>
<tr>
<td>4</td>
<td>415</td>
<td>4</td>
<td>450</td>
<td>441.25</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.78243626</td>
</tr>
</tbody>
</table>
Test Results (More)

Pipelined - Speed Up Vs. # Imagines

Speed Up

# Imagines

8 Packets, 1024 Entries
32 Packets, 1024 Entries
1024 Packets, 1024 Entries
8 Packets, 32 Entries
Test Results (More)

- Parallel Configuration
  - Almost ideal speed up for large data set
  - Slight overhead for large data set

<table>
<thead>
<tr>
<th># Packets</th>
<th># Entries</th>
<th># Imagines</th>
<th>Execution Time</th>
<th>Imagine 0</th>
<th>Imagine 1</th>
<th>Imagine 2</th>
<th>Imagine 3</th>
<th>Avg/Img</th>
<th>Speed Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1024</td>
<td>1</td>
<td></td>
<td>209311</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>104650</td>
<td>104661</td>
<td></td>
<td>104655.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>52325</td>
<td>52325</td>
<td>52325</td>
<td>52336</td>
<td>52327.75</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>1024</td>
<td>1024</td>
<td>1</td>
<td>6697701</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6697701</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3348800</td>
<td>3348901</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3348850.5</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1674400</td>
<td>1674400</td>
<td>1674400</td>
<td>1674501</td>
<td>1674425.25</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>
Test Results (More)

Parallel - Speed Up Vs. # Imagines

Speed Up

# Imagines

0 0.5 1 1.5 2 2.5 3 3.5 4 4.5

1 2 4

32 Packets, 1024 Entries

1024 Packets, 1024 Entries
Challenges

• Limitation on # of imagines (max. 4) when 1 host used
• Complexity in multiple hosts simulation
  – Out of order execution
• Profiling has restrictions
• Problems with communication and synchronization among multiple imagines
Conclusions

• Speedup increases with number of processing nodes
  – Communication and synchronization overheads
• Better to distribute data and instructions across multiple nodes to increase parallelism
• Additional configurations to be tested