High-Performance Packet Classification on GPU
Shijie Zhou, Shreyas G. Singapura, and Viktor K. Prasanna

Ming Hsieh Department of Electrical Engineering
University of Southern California
Outline

• Introduction
• Background
• Contributions
• Algorithm
• Evaluation
• Conclusion and Future Work
Outline

- Introduction
- Background
- Contributions
- Algorithm
- Evaluation
- Conclusion and Future Work
Introduction (1)

• **Internet**: Global system of interconnected computer networks

• Exponentially increasing network traffic

• Future Internet
  • More network traffic
  • Large amounts of data
  • Changing more frequently
Introduction (2)

- Multi-field Packet Classification Applications
  - Routing
  - Access control in firewalls
  - Provision of differentiated qualities of service
  - OpenFlow flow table lookup
Outline

• Introduction
• Background
• Contributions
• Algorithm
• Evaluation
• Conclusion and Future Work
Multi-field Packet Classification (1)

- 5-Fields
  - Source IP address
  - Destination IP address
  - Source port number
  - Destination port number
  - Protocol

<table>
<thead>
<tr>
<th></th>
<th>Src IP</th>
<th>Des IP</th>
<th>Src Port</th>
<th>Des Port</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits</td>
<td>32</td>
<td>32</td>
<td>16</td>
<td>16</td>
<td>8</td>
</tr>
</tbody>
</table>
Multi-field Packet Classification (2)

- **Rule-set**
  - A certain number of rules
  - Matching criteria for each field
  - Wildcard bit in the rule: 1, 0 or * (do not care)
- **Priority**

Multiple matches $\rightarrow$ choose the highest priority rule
$\rightarrow$ take the action
## Multi-field Packet Classification (3)

<table>
<thead>
<tr>
<th>ID</th>
<th>Src IP</th>
<th>Des IP</th>
<th>Src Port</th>
<th>Des Port</th>
<th>Protocol</th>
<th>Priority</th>
<th>ACTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>175.77.88.155/31</td>
<td>119.106.1</td>
<td>0-65535</td>
<td>6888-6888</td>
<td>0x06</td>
<td>1</td>
<td>Act 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>58.230/32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>175.77.88.6/20</td>
<td>36.174.23</td>
<td>0-65535</td>
<td>1604-1704</td>
<td>0x06</td>
<td>2</td>
<td>Act 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.222/32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>12.2.0.0/16</td>
<td>192.1.1.0/24</td>
<td>20-30</td>
<td>1024-1024</td>
<td>0x11</td>
<td>3</td>
<td>Act 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Related Work

• Packet classification on GPU
  • Relatively less explored

• Previous GPU implementations
  • Unique Rules: small\textsuperscript{[1]}
  • Throughput or Latency not discussed\textsuperscript{[2]}
  • \textasciitilde{}11 and 5 MPPS for 500 and 2000 rules\textsuperscript{[3]}

CUDA Programming Model

- CUDA program
  - Host + Kernel
- Host function runs on CPU
- Kernel function runs on GPU
GPU Architecture
Outline

• Introduction
• Background
• Contributions
• Algorithm
• Evaluation
• Conclusion and Future Work
Contributions

• Range-tree search and bit vector (BV) based packet classifier on GPU

• Efficient range-tree search on GPU

• Optimize data layout to minimize shared memory bank conflict

• Throughput of 85 MPPS for 512-rule rule-set
Challenges

- Divergence Overhead
- Limited on-chip memory: data layout
- Classic Tree-Search: pointers to connect nodes

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>...</th>
<th>...</th>
<th>31</th>
</tr>
</thead>
</table>

```
T T F T F F F F F
T T F T F F F F F
X X X X X X X X X
X X X X X X X X X
```

True: Action_a()
False: Action_b()
Outline

• Introduction
• Background
• Contributions
• Algorithm
• Evaluation
• Conclusion and Future Work
Algorithm (1)

- Decomposition-based approach
- Range-tree Search
Algorithm (2)

- Bit Vector Representation
  - $i^{th}$ bit $\rightarrow i^{th}$ original rule
  - 1 $\rightarrow$ match
  - 0 $\rightarrow$ not match
  - Merge by Bit AND operation

```
0 1 0 1
0 1 0 1
& & & = 0 1 0 1
```

match
Algorithm (3)

• 32 threads (a warp) per packet
• Pre-processing (In CPU):
  • Partition rule-set into 32 subsets
  • Construct range-trees & BVs for each subset
• Classification (In GPU):
  • Phase 1: obtain an intermediate result (using the range-trees and BVs)
  • Phase 2: intermediate results → final result
Architecture

Note: $K = 32$
Optimizations (1)
Optimizations (2)

- Store range-trees in shared memory

![Diagram showing shared memory bank conflicts]

- Row-major

**USC Viterbi**

School of Engineering

University of Southern California
Optimizations (3)

- Minimize shared memory bank conflicts
Outline

• Introduction
• Background
• Contributions
• Algorithm
• Evaluation
• Conclusion and Future Work
Platform

- CPU (Intel E5-2665)
  - Cores: 16
  - Frequency: 2.4 GHz
- GPU (NVIDIA K20 Kepler)
  - Streaming Multi-Processor (SMX): 13
  - CUDA cores: 2496
  - Frequency: 705.5 MHz
Performance (1)

No. of rules = 512

Latency (μs)

Column-major
(With Shared-memory)

Row-major
(With Shared-memory)

Without
Shared-memory

USC Viterbi
School of Engineering

University of Southern California
Performance (2)

**Throughput**

<table>
<thead>
<tr>
<th>No. of rules</th>
<th>Throughput (MPPS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>80</td>
</tr>
<tr>
<td>1024</td>
<td>60</td>
</tr>
<tr>
<td>2048</td>
<td>40</td>
</tr>
<tr>
<td>4096</td>
<td>20</td>
</tr>
</tbody>
</table>

**Latency**

<table>
<thead>
<tr>
<th>No. of rules</th>
<th>Latency (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>5</td>
</tr>
<tr>
<td>1024</td>
<td>6</td>
</tr>
<tr>
<td>2048</td>
<td>8</td>
</tr>
<tr>
<td>4096</td>
<td>10</td>
</tr>
</tbody>
</table>
Performance (3)

- Best Case: smallest possible range-trees
- Worst Case: largest possible range-trees
Outline

• Introduction
• Background
• Summary of Contributions
• Algorithm
• Evaluation
• Conclusion and Future Work
Conclusions

- Range-trees + BV packet packet classifier on GPU: 85 MPPS for 512-rule rule-set

- Performance: throughput and latency
  - number of rules (512 ~ 4096)
  - data layout in shared memory

- Compared to state-of-the-art multi-core implementation: 2x improvement in throughput
Future Work

• Hash-based packet classification algorithms

• Other networking applications using GPUs
  • Traffic classification
  • OpenFlow flow table lookup
Thank you!

- **Group Webpage:**
  - [http://ganges.usc.edu/wiki/Parallel_Computing](http://ganges.usc.edu/wiki/Parallel_Computing)

- **Email IDs:**
  - shijiezhi@usc.edu, singapur@usc.edu,
  - prasanna@usc.edu