Parallelization and Characterization of Pattern Matching using GPUs

Giorgos Vasiliadis, FORTH-ICS, Greece
Michalis Polychronakis, Columbia University, USA
Sotiris Ioannidis, FORTH-ICS, Greece

IIISWC’2011, 8 November 2011
Problem Statement

- *Pattern matching* is a core operation in *deep packet inspection* applications
  - Network intrusion detection/prevention systems
  - Traffic classification
  - Spam filtering
  - Content routing

*Given a set of patterns, how to quickly scan network packets to determine which are matched?*
Challenges

• **Traffic rates** are increasing
  – 10 Gbit/s Ethernet speeds are common in metro networks
  – Up to 40 Gbit/s at the core

• Increasing **number of patterns**
  – L7-filter: ~1K rules
  – Snort IDS: ~10K rules
Hardware or Software?

- Prior regular expression matching algorithms are either hardware-based or software-based.

- Hardware-based algorithms:
  - FPGA/TCAM/ASIC based
  - Usually tied to a specific implementation
  - Throughput: High

- Software-based algorithms
  - Processing by general-purpose processors
  - Throughput: Low
Our Approach

• We propose an implementation of *string searching* and *regular expression matching* on the GPU
  – Flexible and programmable
  – Powerful and ubiquitous
  – Constant innovation
    • Thanks to video-game industry 😊
  – Data-parallel model
Outline

• Background
• Implementation
• Performance
• Conclusions
Pattern Matching

**Exact-match string**
- Fixed size patterns
  - “GET / HTTP/1.1”
  - “GNUTELLA”
  - “BitTorrent”
  - etc.

**Regular expressions**
- Character sets
  - [ci-cjck]
- Repetitions
  - , c+, c*
- Wildcards
  - .*, [^ci-ci]*
- Counters
  - c{m, n}, [^ci-ci]{m, n},

*Not expressive enough*  
*Provide flexibility and expressiveness*
Pattern Matching

Both string searching and regular expression matching can be matched efficiently by combining the patterns into **Deterministic Finite Automata (DFA)**

Example: \( P=\{\text{he, she, his, hers}\} \)
DFA matching

- Move over the input data stream one byte at a time
- Switch the *current state* according to the state table
- When a *final-state* is reached, a match has been found
Outline

• Background
• Implementation
• Performance
• Conclusions
Data-Parallelism in Packet Processing

• The key insight
  – Data level parallelism = packet level parallelism

Network Interface

1. Batching

2. Parallel Processing in GPU
Pattern matching on the GPU

- Uniformly one thread for each network packet
Optimizing Packet Processing for GPU

1) Memory access latency

2) Memory bandwidth

3) Memory hierarchies
1) Memory access latency

- Improve memory utilization by running many threads
2) Memory bandwidth

- Only 1/32th of the total bandwidth is utilized
  - Device memory transaction is 32 bytes (minimum)
2) Memory bandwidth

- Packet reading is boosted 4x with 4-byte fetches
2) Memory bandwidth

- Packet reading is boosted 4x with 4-byte fetches

16-byte accesses
3) Exploring memory hierarchies

<table>
<thead>
<tr>
<th>What?</th>
<th>Where?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network packets</td>
<td>Global Memory</td>
<td>1-Dimension (linear)</td>
</tr>
<tr>
<td>State tables</td>
<td>Texture Memory</td>
<td>2-Dimensions</td>
</tr>
<tr>
<td></td>
<td>Constant Memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shared Memory</td>
<td></td>
</tr>
</tbody>
</table>
3) Exploring memory hierarchies

<table>
<thead>
<tr>
<th>What?</th>
<th>Where?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network packets</td>
<td>Global Memory</td>
<td>1-Dimension (linear)</td>
</tr>
<tr>
<td>State tables</td>
<td>Texture Memory</td>
<td>2-Dimensions</td>
</tr>
<tr>
<td></td>
<td>Constant Memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shared Memory</td>
<td></td>
</tr>
</tbody>
</table>
3) Exploring memory hierarchies

<table>
<thead>
<tr>
<th>What?</th>
<th>Where?</th>
<th>How?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network packets</td>
<td>Global Memory</td>
<td>1-Dimension (linear)</td>
</tr>
<tr>
<td>State tables</td>
<td>Texture Memory</td>
<td>2-Dimensions</td>
</tr>
<tr>
<td></td>
<td>Constant Memory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shared Memory</td>
<td></td>
</tr>
</tbody>
</table>

- **Rule of thumb**
  - *Texture memory* for packets
  - *Global memory* for state table

Both L1- and texture caches are utilized
Putting it *all* together

- Pattern matching on GPU is really fast

<table>
<thead>
<tr>
<th>Packet size</th>
<th>Gbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>200b</td>
<td>1240.1</td>
</tr>
<tr>
<td>400b</td>
<td>687.6</td>
</tr>
<tr>
<td>800b</td>
<td>330.4</td>
</tr>
<tr>
<td>1500b</td>
<td>186.3</td>
</tr>
</tbody>
</table>

- *Unfortunately*, packets have to be transferred to the GPU, over the PCIe bus
Transferring overheads

- PCIe has evolved over the last versions
  - 64 Gbit/s for a PCIe x16 graphics card
- Unfortunately, PCIe suffers from small data transfers

<table>
<thead>
<tr>
<th>Buffer Size</th>
<th>1KB</th>
<th>4KB</th>
<th>64KB</th>
<th>256KB</th>
<th>1MB</th>
<th>16MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host to Device</td>
<td>2.04</td>
<td>7.1</td>
<td>34.4</td>
<td>42.1</td>
<td>44.6</td>
<td>45.7</td>
</tr>
</tbody>
</table>

⇒ Store many packets to a single buffer (CPU-side), and transfer it to the GPU at once
How to store network packets?

• **Fixed-buckets buffer**

```
| Bucket 0:   | Pkt 0 |
| Bucket 1:   | Pkt 1 |
| Bucket 2:   | Pkt 2 |
| Bucket N:   | Pkt N |
```

• **Performance**

![Throughput graph](image)

PCIe x16

Contact: gvasil@ics.forth.gr
How to store network packets?

• Indexed buffer

• Performance

PCIe x16

5x better
Outline

• Background
• Implementation
• Performance
• Conclusions
Overall Performance

GPU Speedup: 8.75x 21.2x 30.2x 35.0x 41.2x
Scalability to number of patterns

- Constant throughput
  - Independently of the number of patterns
What to expect?

- GPU throughput increased 6 times, in less than two years
  - From **28.1 Gbit/s** to over **180 Gbit/s**
Conclusions

• An *efficient* pattern matching implementation on the GPU

• Several *device-level* optimizations
  – Explore different memory hierarchies
  – Alleviate memory congestions

• Improve *transferring* of small packets
Thank you!

gvasil@ics.forth.gr