Evaluating GPUs for Network Packet Signature Matching

Randy Smith, Neelam Goyal, Justin Ormont
Karu Sankaralingam, Cristian Estan

Department of Computer Sciences
University of Wisconsin—Madison
Network Communication
Network Communication
Network Communication
Network Communication
Network Communication
Network Communication

Thousands of packets, match header bytes at line rates

Thousands of packets, match *full payloads* at line rates
Signature Matching Mechanics

- An intrusion prevention system is a filter
  - Functionality defined by user-supplied signatures

- Problem: find all matching signature occurrences up to the currently scanned byte

Signatures = {
  \( s_1 \): /(.*)shadow/ ,
  \( s_2 \): /(.*)user(.*)root/ ,
  \( s_3 \): /(.*)[Pp][Aa][Ss][Ss][Ww][Dd]/  }

<table>
<thead>
<tr>
<th>← Header</th>
<th>Payload →</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0 A8 64 01</td>
<td>60 19 00 15</td>
</tr>
<tr>
<td>user</td>
<td>passwd</td>
</tr>
</tbody>
</table>

- \( s_3 \) match!
- \( s_2 \) match!
- \( s_3 \) match!
Processor Architectures

![Graph comparing Processor Architectures]

- **ASIC**
  - Cost: Low
  - Area Eff.: High
  - Flexibility: Low
  - Performance: High

- **FPGA**
  - Cost: Medium
  - Area Eff.: Medium
  - Flexibility: High
  - Performance: High

- **Network Processor**
  - Cost: Low
  - Area Eff.: Low
  - Flexibility: Medium
  - Performance: High

- **General Purpose CPU**
  - Cost: High
  - Area Eff.: Low
  - Flexibility: High
  - Performance: Medium

- **SIMD/GPU**
  - Cost: High
  - Area Eff.: Low
  - Flexibility: High
  - Performance: High
Why Consider GPUs?

- “The future of GPUs is programmable processing”
- GPUs specialized for compute-intensive highly parallel computation (not just for graphics)
GPUs for Signature Matching: Challenges

- Recursive data access patterns
- Not purely data parallel (divergence occurs)

Key Questions:

- *Do GPU/SIMD architectures have necessary flexibility?*
- *Does mapping to such architectures negate performance gains?*
This Work

- Overall Goal: Assess suitability of GPUs for signature matching

- Conclusion: GPUs/SIMD provide higher performance than CPUs at similar costs

- In support of this:
  - Characterize signature matching in terms of control flow, memory access patterns, and concurrency;
  - Build and evaluate fully-functional prototype signature matcher on an NVIDIA G80 GPU;
Outline

- Signature Matching Characteristics
- NVIDIA G80 Prototype
- Experimental Results
- Discussion and Conclusions
Signature Matching Components

- Several MB in size
- DMA, etc. to copy from NIC to memory
- Regular Memory Access
Signature Matching Components

- 1 KB per state, thousands to millions of states
- Recursive data structures, irregular accesses
- Accesses serially dependent, driven by input
Signature Matching Components

- Packet buffer

- Holds temporary data, acceptance indicators
- Typically less than 10K in size

- State Machine Memory

- Auxiliary Memory
Signature Matching Components

- Reads packet input
- Selects and traverses state machine
- Updates auxiliary data
Matching with DFAs

- Regular Expression: `/^retr[^\n]{100}/`

- Simple computation model, easy to combine
- One table look-up per input byte
- Subject to state-space explosion when combined
Matching with XFAs

- Regular Expression: `/^retr[^\n]{100}/`

- Use variables to reduce DFA state explosion
- Manipulate variables with instructions attached to states
- Same semantics, complicated execution model
Matching with XFAs

- Regular Expression: \(/.*ab.*cd/\)

**DFA**

- States: 0, 1, 2, 3, 4
- Transitions:
  - 0 \(\rightarrow\) 1: \[^a\]
  - 1 \(\rightarrow\) 2: a
  - 1 \(\rightarrow\) 2: b
  - 2 \(\rightarrow\) 3: \[^c\]
  - 2 \(\rightarrow\) 3: c
  - 3 \(\rightarrow\) 4: c
  - 3 \(\rightarrow\) 4: d

**XFA**

- States: 1, 2, 3, 4
- Transitions:
  - 1 \(\rightarrow\) 2: a
  - 2 \(\rightarrow\) 3: b
  - 3 \(\rightarrow\) 4: c
  - 3 \(\rightarrow\) 4: d

**Accept Signature**

- DFA: bit = 0
- XFA: bit = 1

- if (bit)
  - Accept Signature;
- <10% states contribute to >90% accesses
- Conclusion: Irregular access but caching, multithreading can hide access latencies
Control Flow

```
StateMachine *SM = read_signatures();

for each packet in trace:
    apply(SM, packet.bytes, packet.len);

apply(StateMachine *SM, char *buf, int len):
    state *CS = SM.start_state;
    execute_instrs(CS->instrs);
    for i=0 to len
        CS = CS->nextState(buf[i]);
        execute_instrs(CS->instrs);
```
Divergence

- Occurs when distinct conditional branches are taken in processing elements
  - Measured as the number of distinct conditional branches in SIMD instruction

- Sources:
  - Variations in packet size
  - Acceptance of *some* regular expressions
  - Execution of distinct XFA instructions

- What are common divergence levels?
## Control Flow and Divergence

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Prediction Accuracy</th>
<th>Divergence</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DFA</td>
<td>XFA</td>
</tr>
<tr>
<td>FTP</td>
<td>97.5</td>
<td>97.6</td>
</tr>
<tr>
<td>HTTP</td>
<td>99.2</td>
<td>99.3</td>
</tr>
<tr>
<td>SMTP</td>
<td>98.3</td>
<td>98.2</td>
</tr>
</tbody>
</table>
Signature Matching Requirements

- **Memory Requirements**
  - Efficient regular access for moderate sized-memory
  - Fast access for small, local memory
  - Hide latency of irregular accesses

- **Control Flow**
  - Divergence occurs, but not prevalent
  - Support for data-dependent branching
G80 Architecture

- 16 core, 32 way SIMD
- Fast on-chip shared memory, texture cache
- Off-chip memory large but slow (400-600 cycle access times)
- Texture Cache read-only by GPU
G80 Architecture

- **CUDA** provides API, drivers to interface

- **Kernels** execute same instruction on all PEs in core

- Time-sliced multi-threading per core

- 31-cycle penalty for data-dependent branching
Two kernels:
- `fa_build` – build state machine on GPU
- `trace_apply` – perform DFA and XFA matching

Sort packets into “groups of 32”

Batch execution model:
- Copy packets to GPU
- Perform matching
- Retrieve result vector
Outline

- Signature Matching Characteristics
- NVIDIA G80 Prototype
- Experimental Results
- Discussion and Limitations
Environment

- **Test sets and data**
  - Extracted FTP, HTTP, and SMTP regular expressions from Cisco IPS signatures
  - Collected 10 GB trace from edge of academic network

- **Platforms**
  - Prototype: NVIDIA G880 GTX, Pentium 4 host
  - Baseline: Software implementation on P4 at 3.0GHz
  - Also: Intel Core2, Niagara

- All code implemented in “flat” C++
Observed Speedup

![Graph showing observed speedup for different protocols and platforms.](image-url)
Ideal Speedup

The graph shows the ideal speedup for different systems as the number of threads increases. The systems compared include:

- G80
- Pentium4
- Core2
- Niagara (32 threads)

The y-axis represents the number of nanoseconds per byte (ns/byte), and the x-axis represents the number of threads (1000s).

Key observations:

- The 1-state DFA has a significant speedup across all systems.
- At 2048 threads, the 1-state DFA shows a noticeable drop in ns/byte, indicating improved efficiency.
- Idealized packet reads are also shown, highlighting the performance gains.

The graph provides a visual representation of how these systems perform under varying thread loads, emphasizing the efficiency of the 1-state DFA in particular.
Results Summary

- G80 achieves 8.6x (DFAs) and 6.7x (XFAs) speedup over Pentium 4 baseline.
  - Key: many threads hide memory access latency

- Peak performance estimate is 36x speedup

- Better memory utilization (texture memory) may move closer to ideal
Discussion and Limitations

- Batch processing and (no) texture cache usage
  - Artifact of prototype environment
  - Can resolve with a shared address space

- Sorted packets
  - Solution 1: consider approximate sorting or binning
  - Solution 2: break serial dependency of state traversal for DFAs

- Other functionality
  - Must maintain per-flow state, perform normalization
Evaluating GPUs for Network Packet Signature Matching

Thank you