GPU-Based Network Traffic Monitoring & Analysis Tools

Wenji Wu; Phil DeMar

wenji@fnal.gov, demar@fnal.gov

CHEP 2013

October 17, 2013

Background

- Main uses for network traffic monitoring & analysis tools:
  - Operations & management
  - Capacity planning
  - Performance troubleshooting

- Levels of network traffic monitoring & analysis:
  - Device counter level (snmp data)
  - Traffic flow level (flow data)
  - At the packet inspection level $\longleftarrow$ The focus of this work
    - Security analysis
    - Application performance analysis
    - Traffic characterization studies

Coarse

$\updownarrow$

Detailed

Problem Space

- Emerging high-performance network environments:
  - 40GE/100GE link technologies now in LAN & WAN
  - Servers becoming 10GE-connected by default
  - n x 100GE / 400GE backbone links & 40GE host connections loom on the horizon.
- Current flow-based & packet-based traffic monitoring & analysis tools break down at 40GE/100GE:
  - Flow data is sampled:
    - Implemented in device hardware
    - May be too coarse for computer security forensics & detailed traffic characterization studies
  - Packet-based analysis runs into resource issues:
    - 10Gb/s = ~14M 64-bytes packets per sec

Fermilab

Packet-Based Analysis

- Our preferred choice for 40/100GE traffic analysis:
  - Flow data limitations (sampled) constrain flow-based analysis
- Characteristics of packet-based network monitoring & analysis applications
  - Time constraints on packet processing.
  - Highly compute and I/O throughput-intensive
  - High levels of data parallelism.
    - Each packet can be processed independently
  - Extremely poor temporal locality for data
    - Typically, data processed once in sequence; rarely reused

Fermilab

Platforms for Packet-Based Analysis (I)

- Computing platform requirements monitoring & analysis applications within high performance network :
  - High Compute power
  - Ample memory bandwidth
  - Capability of handing data parallelism inherent with network data
  - Easy programmability

Platforms for Packet-Based Analysis (II)

- Three types of computing platforms:
  - NPU/ASIC
  - CPU
  - GPU

Features	NPU/ASIC	CPU	GPU
High compute power	**Varies**	✖	✔
High memory bandwidth	**Varies**	✖	✔
Easy programmability	✖	✔	✔
Data-parallel execution model	✖	✖	✔

Architecture Comparison

Our Solution

- Use GPU-based Traffic Monitoring & Analysis Tools:
  - Note: This is currently a research area
- Highlights of our work:
  - Demonstrated GPUs can significantly accelerate network traffic monitoring & analysis
    - 11 million+ pkts/s without drops (single Nvidia M2070)
  - Designed/implemented a generic I/O architecture to move network traffic from wire into GPU domain
  - Implemented a GPU-accelerated library for network traffic capturing, monitoring, and analysis.
    - Dozens of CUDA kernels, which can be combined in a variety of ways to perform monitoring and analysis tasks

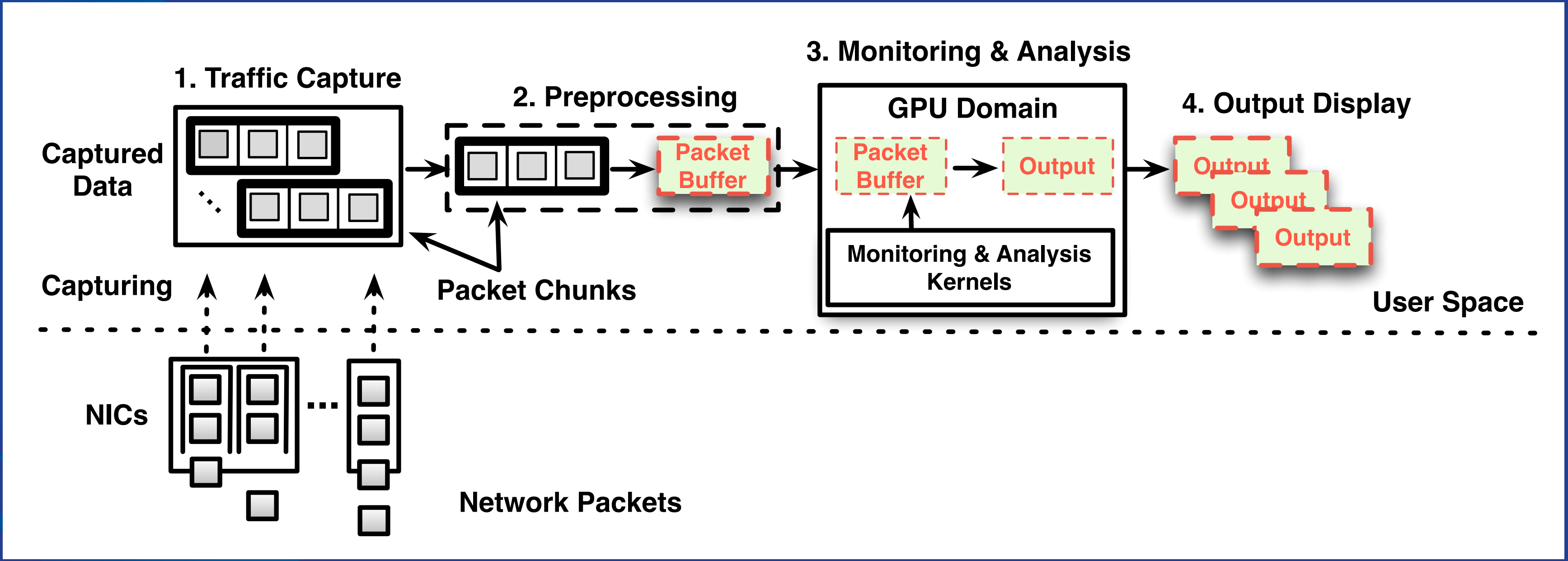
Key Technical Issues

- GPU's relatively small memory size:
  - Nvidia M2070 has 6 GB Memory
  - Workarounds:
    - Mapping host memory into GPU with zero-copy technique?
    - Partial packet capture approach ✔
- Need to capture & move packets from wire into GPU domain without packet loss
- Need to design data structures that are efficient for both CPU and GPU

System Architecture

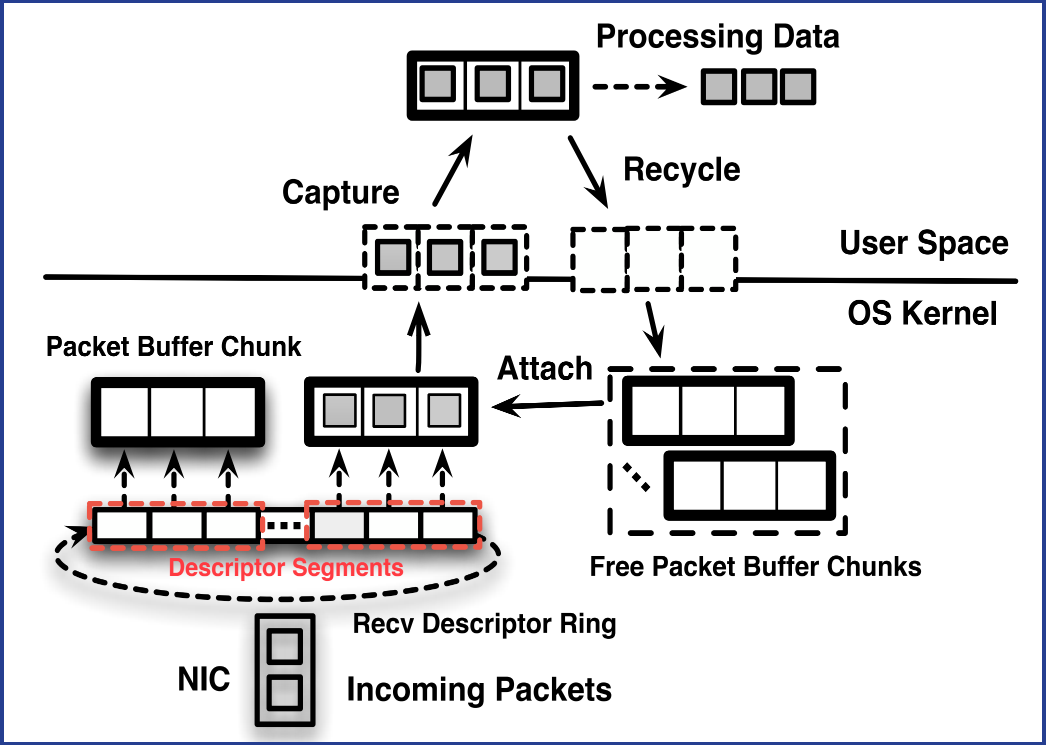
Four Types of Logical Entities:

- Traffic Capture
- Preprocessing
- Monitoring & Analysis
- Output Display



Packet I/O Engine for Capture

- Key techniques
  - Pre-allocated large packet buffers
  - Packet-level batch processing
  - Memory mapping based zero-copy



Key Operations

- Open
- Capture
- Recycle
- Close

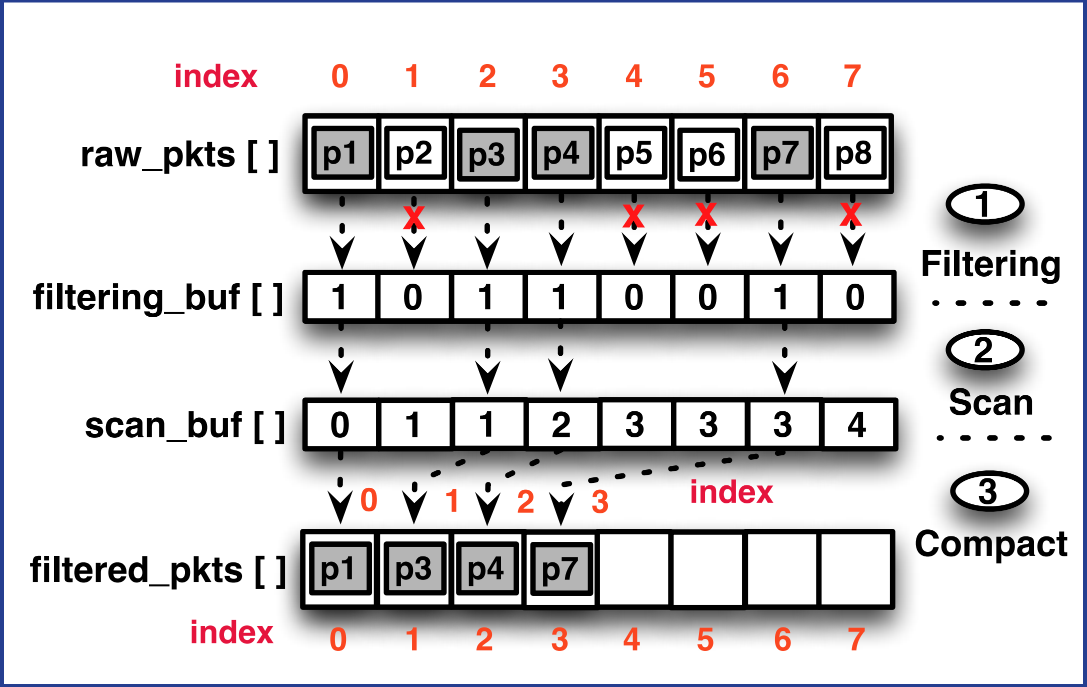
Fermilab

GPU-based Network Traffic Monitoring & Analysis Algorithms

- A GPU-accelerated library for network traffic capturing, monitoring, and analysis apps.
  - Dozens of CUDA kernels
  - Can be combined in a variety of ways to perform intended monitoring & analysis operations
  - Examples in following slides

Packet-Filtering Kernel

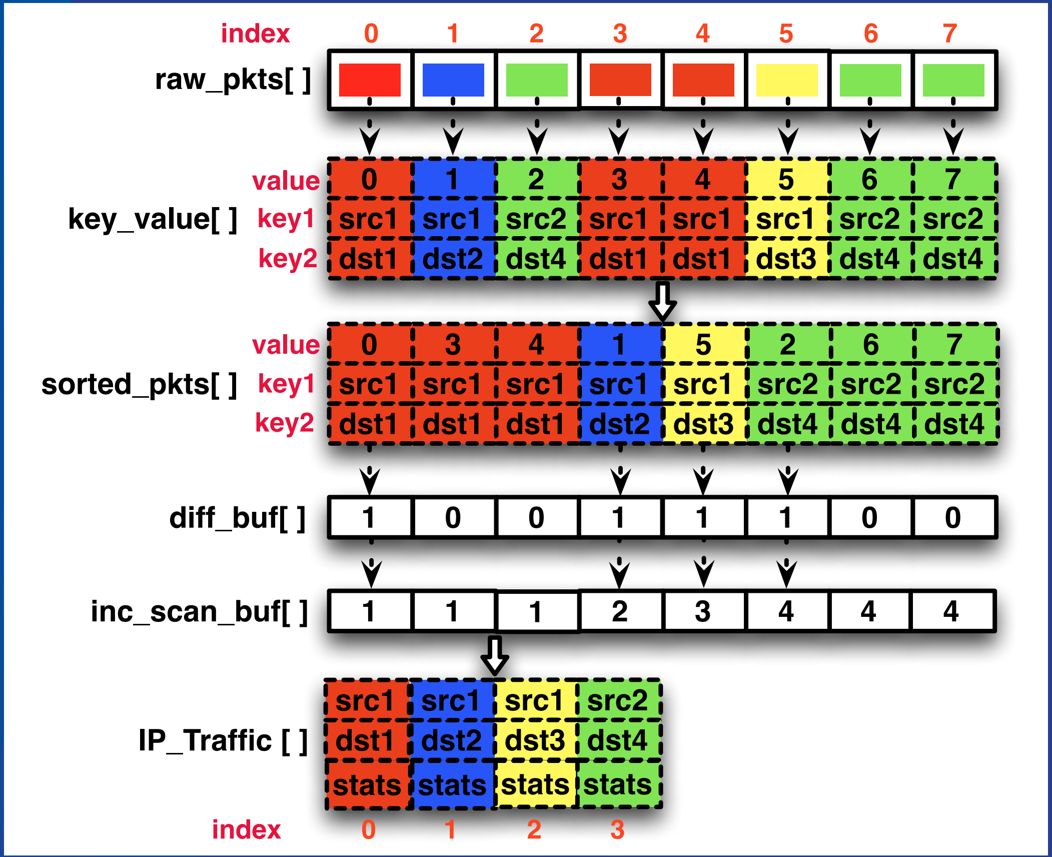
- Advanced packet filtering capabilities necessary so that we only analyze packets of interest
  - We use Berkeley Packet Filter (BPF) as the packet filter
  - A few basic GPU operations, such as sort, prefix-sum, and compact.



Fermilab

Traffic-Aggregation Kernel

- Reads an array of n packets at pkts[] and aggregates traffic for same src & dst IP addresses. Exports list of entries; each entry records a src/dst address pair, with associated traffic statistics



index 0 1 2 3 4 5 6 7
raw_pkts[ ]
key_value[ ]
value 0 1 2 3 4 5 6 7
key1 src1 src1 src2 src1 src1 src1 src2 src2
key2 dst1 dst2 dst4 dst1 dst1 dst3 dst4 dst4
sorted_pkts[ ]
value 0 3 4 1 5 2 6 7
key1 src1 src1 src1 src1 src1 src2 src2 src2
key2 dst1 dst1 dst1 dst2 dst3 dst4 dst4 dst4
diff_buf[ ] 1 0 0 1 1 1 0 0
inc_scan_buf[ ] 1 1 1 2 3 4 4 4
IP_Traffic [ ]
src1 src1 src1 src2
dst1 dst2 dst3 dst4
stats stats stats stats
index 0 1 2 3

Multikey-Value Sort

Inclusive Scan

Use to build IP conversations

Unique-IP-Addresses Kernel

- Reads an array of n packets at pkts[] and outputs a list of unique src or dst IP addresses seen on the packets

CUDA

```
1.  for each $i \in [0,n-1]$ in parallel do
        IPs[i] ≔ src or dst addr of pkts[i];
    end for
2.  perform sort on IPs[] to determine sorted_IPs[];
3.  diff_results[0] =1;
    for each $i \in [1,n-1]$ in parallel do
        if(sorted_IPs[i] ≠sorted_IPs[i-1]) diff_buf[i]=1;
        else diff_buf[i]=0;
    end for
4.  perform exclusive prefix sum on diff_buf[];
5.  for each $i \in [0,n-1]$ in parallel do
        if(diff_buf[i] ==1) Output[scan_buf[i]]=sorted_IPs[i];
    end for
```

A Sample Use Case

- Using our GPU-accelerated library, we developed a sample use case to monitor network status:
  - Monitor networks at different levels:
    - from aggregate of entire network down to one node
  - Monitor network traffic by protocol
  - Monitor network traffic information per node:
    - Determine who is sending/receiving the most traffic
    - For both local and remote addresses
  - Monitor IP conversations:
    - Characterizing by volume, or other traits.

A Sample Use Case – Data Structures

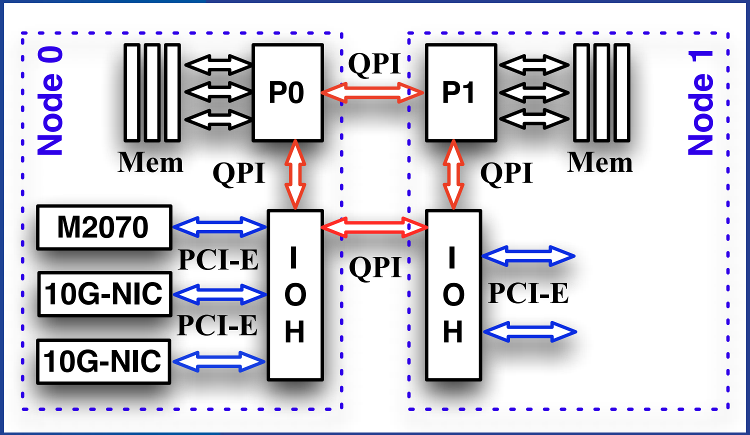
- Three key data structures were created at GPU:
  - *protocol_stat[]*
    - an array used to store protocol statistics for network traffic, with each entry associated with a specific protocol.
  - *ip_snd[] and ip_rcv[]*
    - arrays that are used to store traffic statistics for IP conversations in send & receive directions respectively
  - *ip_table*
    - a hash table that is used to keep track of network traffic information of each IP address node

These data structures are designed to reference themselves and each other with relative offsets such as array indexes

A Sample Use Case – Algorithm

1. Call Packet-filtering kernel to filter packets of interest
2. Call Unique-IP-addresses kernel to obtain IP addresses
3. Build the ip_table with a parallel hashing algorithm
4. Collect traffic statistics for each protocol and each IP node
5. Call Traffic-Aggregation kernel to build IP conversations

Prototyped System

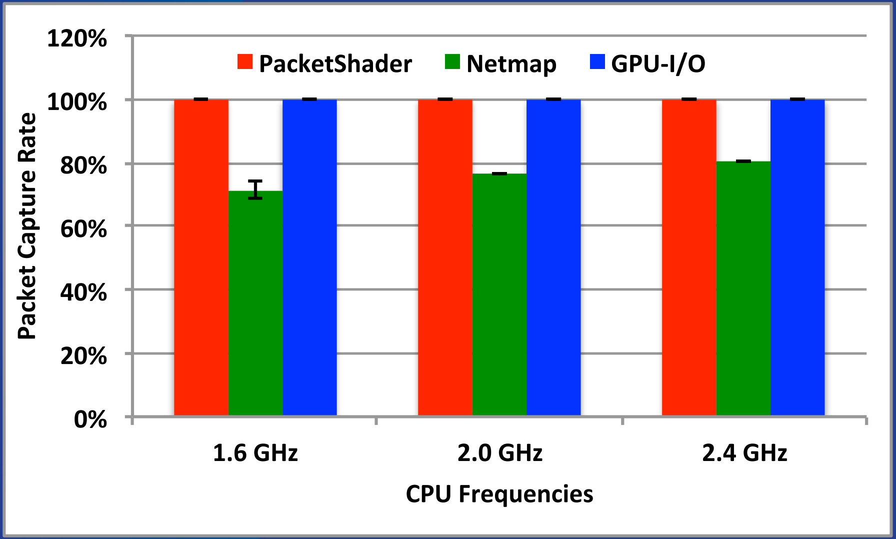


Prototyped System

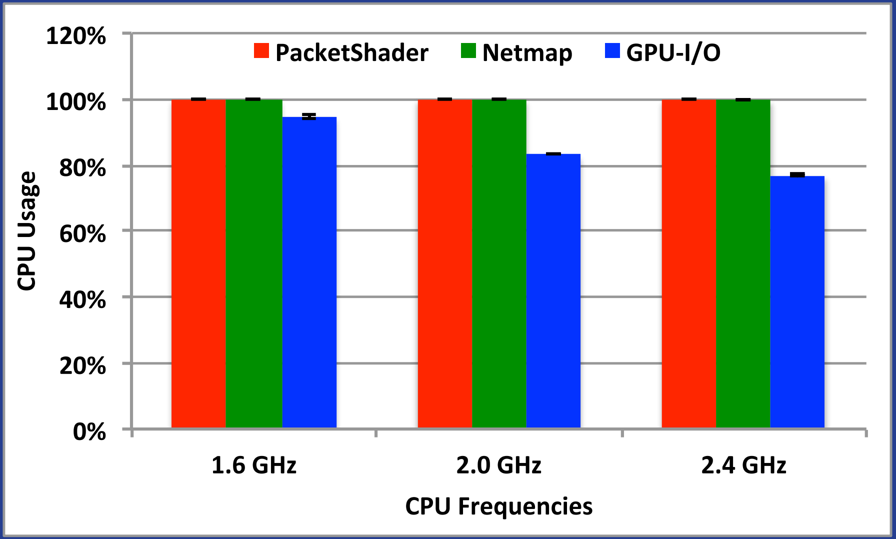
- Our application is developed on Linux.
- CUDA 4.2 programming environment.
- The packet I/O engine is implemented on Intel 82599 10GigE NIC

- A two-node NUMA system
  - Two 8-core 2.67GHz Intel X5650 processors.
- Two Intel 82599-based 10GigE NICs
- One Nvidia M2070 GPU.

Performance Evaluation
Packet I/O Engine



Packet Capture Rate

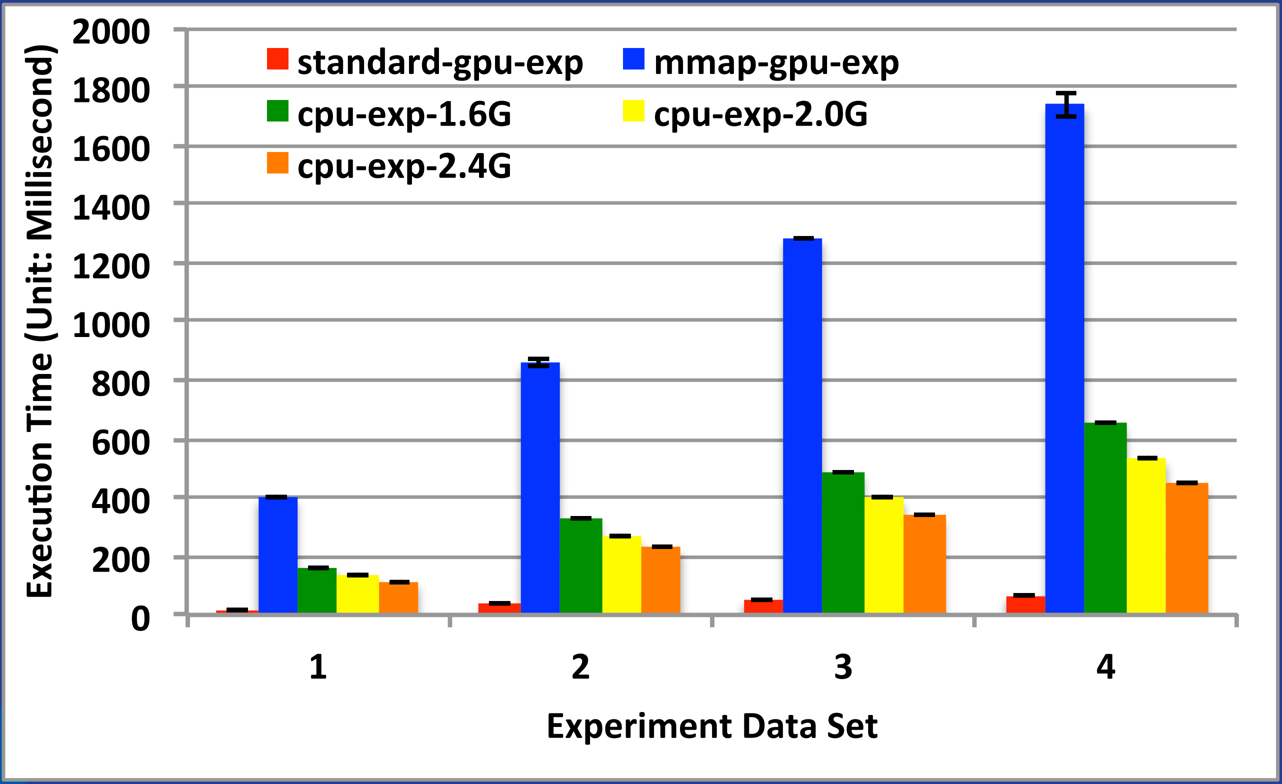


CPU Usage

- Our Packet I/O engine (GPU-I/O) vs CPU-based packet analysis tools
  - No packet drops
  - Least CPU usage

Fermilab

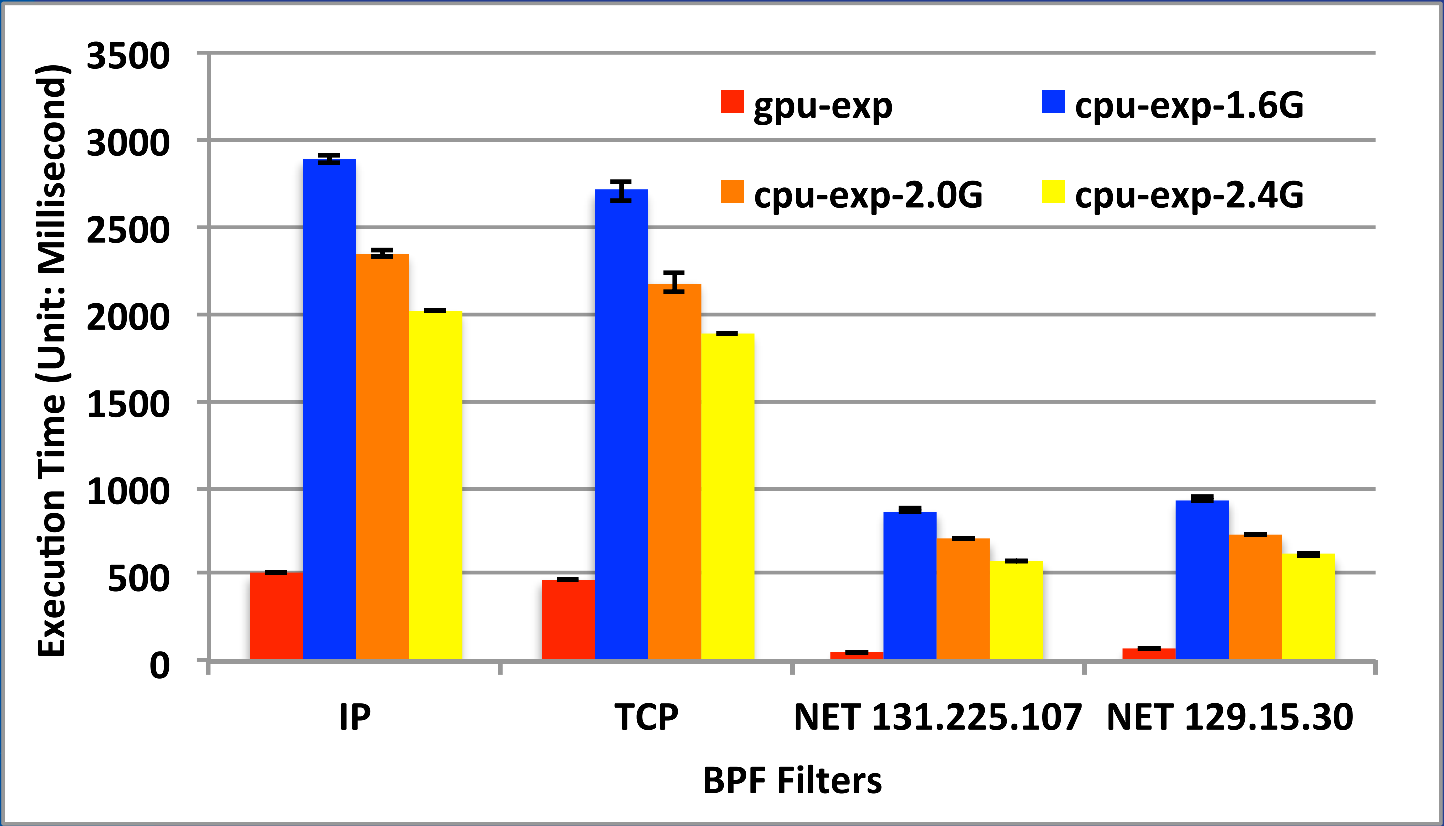
Performance Evaluation
GPU-based Packet Filtering Algorithm



- Our packet filtering algorithm (red) vs CPU-based & memory-mapped GPU-based tools

Fermilab

Performance Evaluation GPU-based Sample Use Case



- Our GPU-analysis (red) vs CPU-based analysis

Fermilab

Conclusion

- Our GPU-based network traffic monitoring & analysis tools seem effective in high-performance network technology environments
- Next steps:
  - Continue to develop these tools toward production-quality
  - Investigate ways to work around limitations (ie., IDS) of partial packet capture
- Always looking for potential collaborators in this technology area

Questions

?

Thank You!

Email: wenji@fnal.gov and demar@fnal.gov