Scalable, Hardware-Accelerated Network Analytics

Oliver Michel
University of Colorado Boulder

John Sonchack
University of Pennsylvania

Adam J. Aviv
U.S. Naval Academy

Eric Keller
University of Colorado Boulder

University of Colorado Boulder

Network Monitoring & Analytics

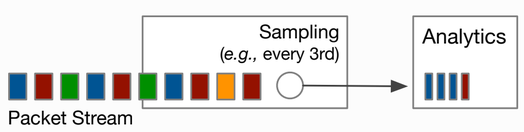
Applications
- Network debugging
- Performance analysis
- Security

Data Collection
- Netflow/IPFIX
- Packet traces
- P4 telemetry systems

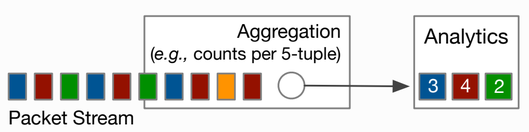
Data Analysis
- Generally post-hoc
- Batch processing, e.g., MapReduce
- Heavyweight deployments

Compromises in Today's Systems

Traffic Filtering



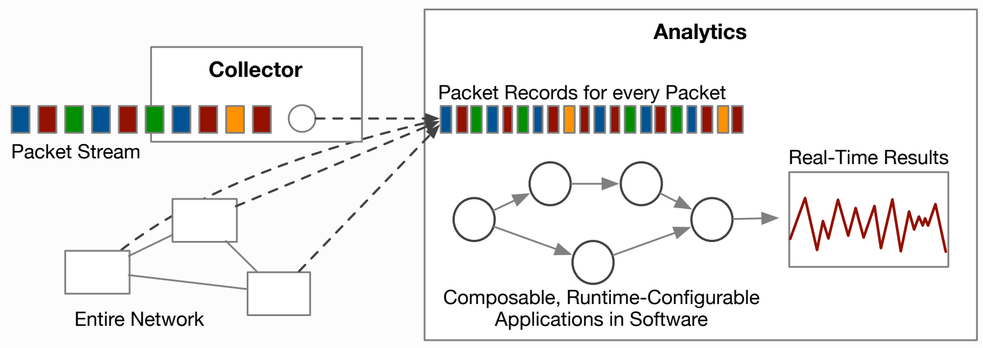
Sampling



Aggregation

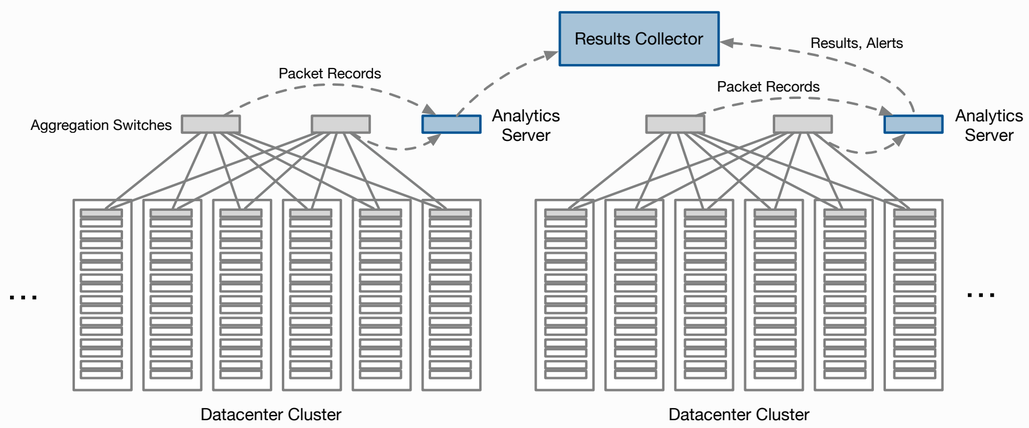
Limited Expressiveness

An Ideal Network Analytics System

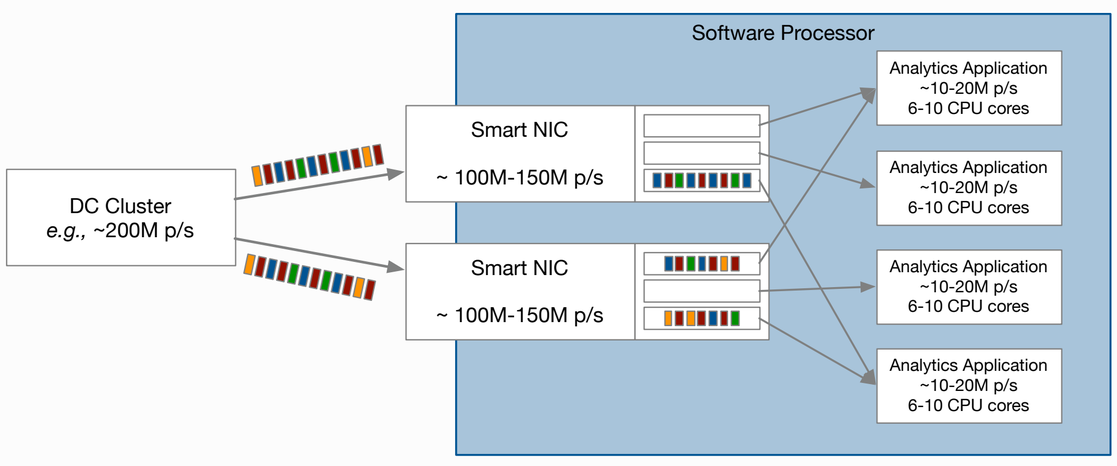


- Every single packet in software (no filtering, no sampling)
- Per packet data (no aggregation)
- In real time (processing as packets traverse the network)
- Composable, flexible analytics applications in software
- Analytics configurable at runtime without downtime

System Overview



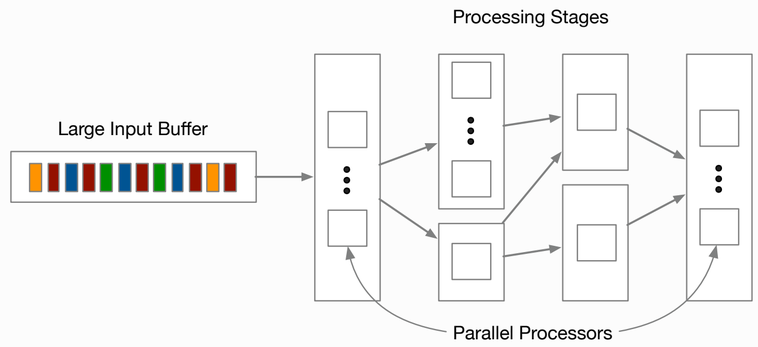
Hardware Support



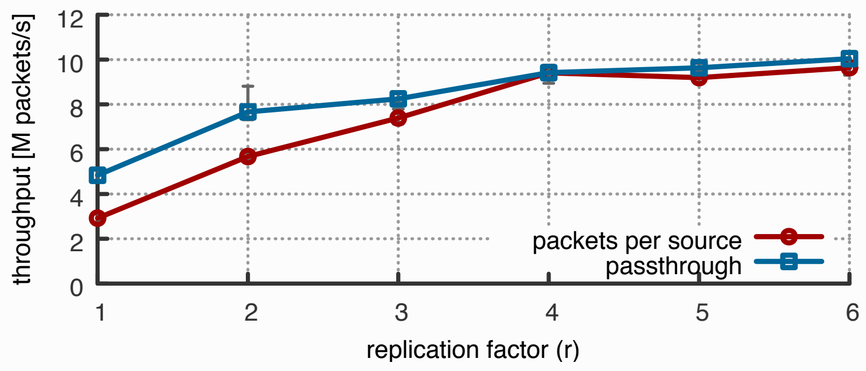
- Initial pre-processing in hardware (e.g., Smart NIC, PFE)
  - Different queues per target application
  - Batching of low-priority traffic
- Zero-copy read into user-space (e.g., DPDK)
- Netronome NFP-4000: 148M packet/s throughput

This work was supported in part by the NSF grants 1652698 (CAREER) and 1406192 (SaTC), and by the NSF and VMWare grant 1700527 (SDI-CSCS).

High-Performance Software Analytics



- Processing system based on streaming analytics paradigm
- Processing elements organized in scalable stages
- Run in parallel
- Data is passed between processors through queues
- C++ API with pre-defined elements
- Custom elements easy to implement



API Example:

```
jetstream::app app;
auto source = app.add_stage<source>(1, "eth0");
auto per_src_counter = app.add_stage<pkts_per_src>(3);
auto threshold = app.add_stage<threshold>(1);
app.connect<pkt>(source, per_src_counter);
app.connect<src_count>(per_src_counter, threshold);
app();
```