

NFQL: A Tool for Querying Network Flow Records

nfql.vaibhavbajpai.com

Vaibhav Bajpai, Johannes Schauer and Jürgen Schönwälder
{v.bajpai, j.schauer, j.schoenwaelder}@jacobs-university.de

IM 2013, Ghent

Computer Networks and Distributed Systems
Jacobs University Bremen
Bremen, Germany

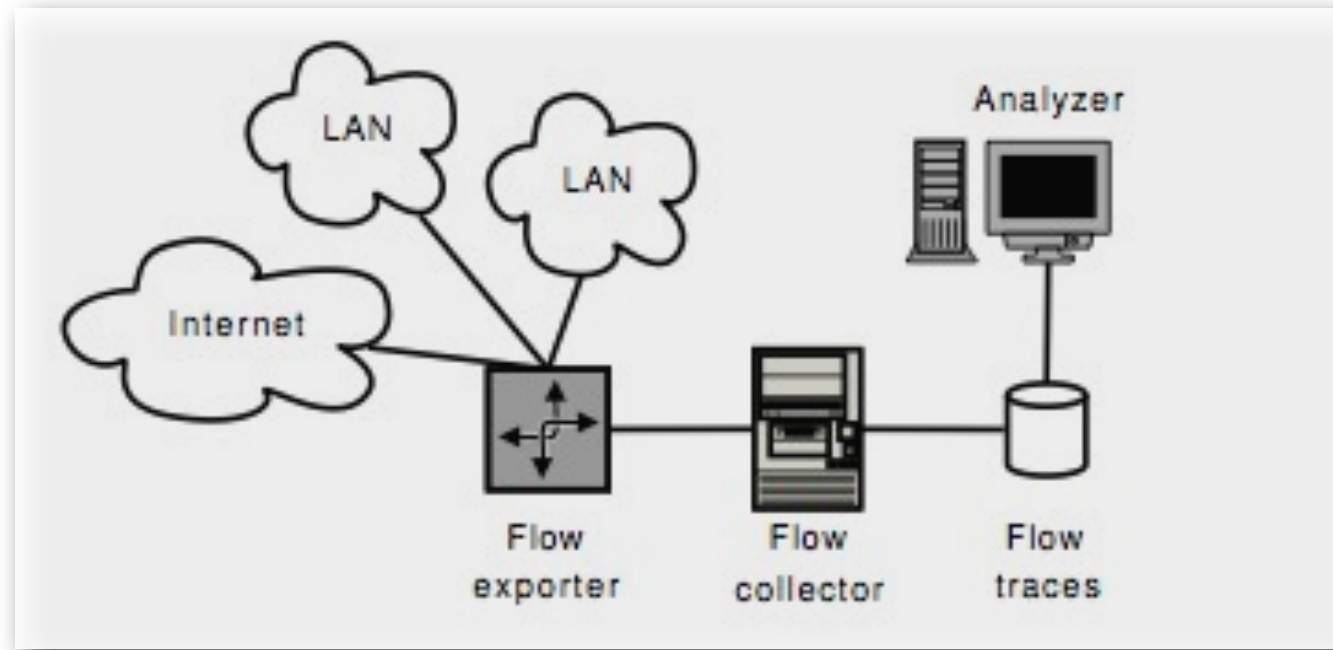
May 2013

Overview

- Motivation
- Related Work
- Flow Query Language: NFQL
- Implementation: `nfql`
- Performance Evaluations
- Conclusions

Motivation

- IP traffic flow



A set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties [\[RFC 3917\]](#).

Flow analysis use cases:

- Survey on detection of intrusion attacks [\[1\]](#).
- Survey on behavior analysis of Internet backbone traffic [\[2\]](#).

- Flow export protocols

- Cisco NetFlow [\[RFC 3954\]](#)
- IETF IPFIX [\[RFC 5101\]](#)

Version	Features
v1, {2, 3, 4}	original format with several internal releases
v5	CIDR, AS support and flow sequence numbers
v{6, 7, 8}	router-based aggregation support
v9	template-based with IPv6 and MPLS support
IPFIX	universal standard, transport-protocol agnostic

- Understanding intricate traffic patterns require sophisticated flow analysis tools.
- Current tools span a smaller use-case owing to their simplistic language designs.

Related Work

- Simple traffic analysis tools

- ntop, FlowScan, NfSen, Stager

- Popular NetFlow analysis tools

- flow-tools: supports NetFlow v5
- nfdump: supports NetFlow v9

- Popular IPFIX analysis tools

- SiLK

- Limited to only absolute comparison of flow-keys

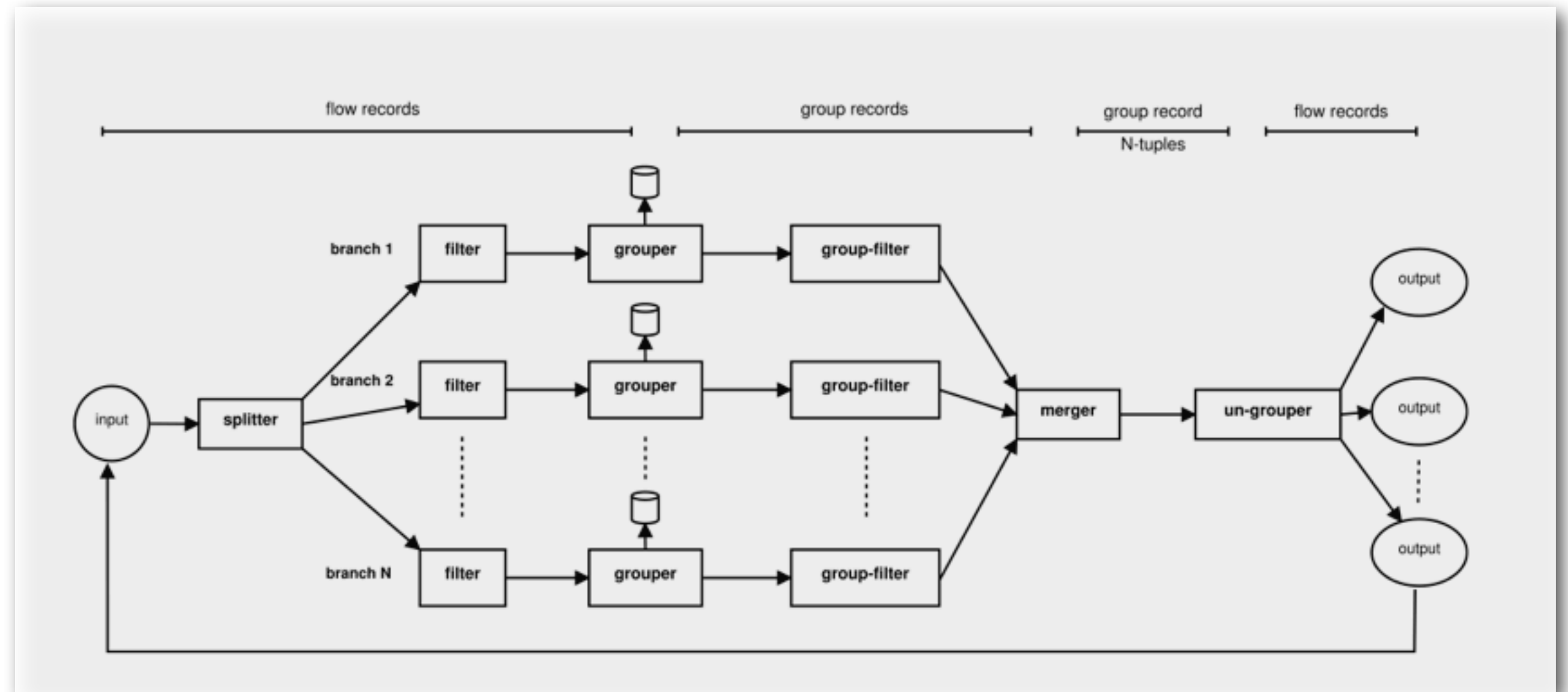
- Grouping and merging can only be performed on = operator.
- Cannot ungroup the flows once grouped.
- Stringent requirements on organization of input flows.

NFQL (Network Flow Query Language)

The expressiveness of the language can be seen from [4], where NFQL queries are used to identify application signatures.

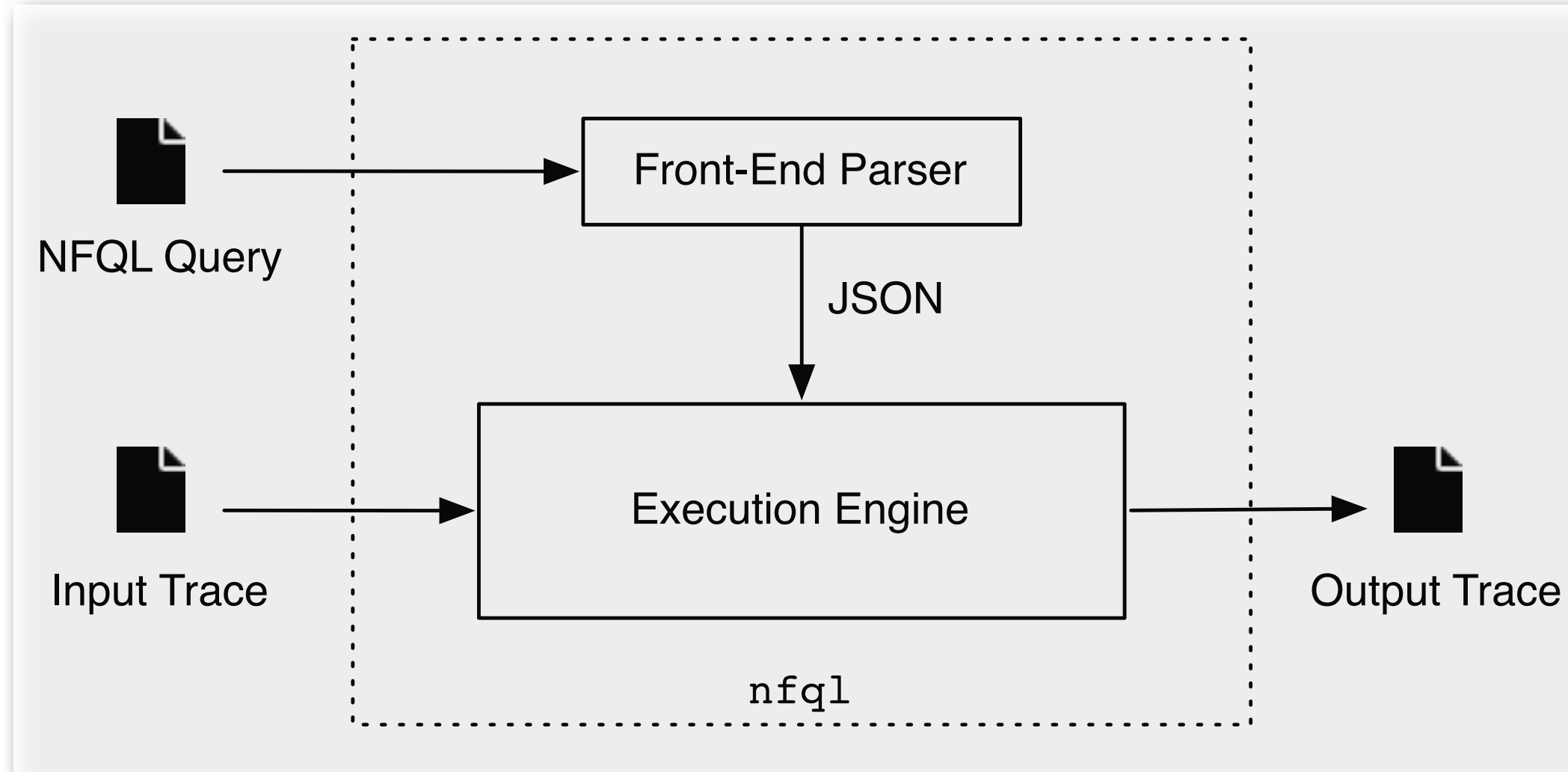
- Features

- Filter flows.
- Combine flows into groups.
- Aggregate flows on flow-keys as one grouped flow aggregate.
- Invoke Allen interval algebra on flows.
- Merge grouped flows.
- Apply absolute or relative filters when grouping or merging.
- Unfold grouped flows back into individual flows.



NFQL processing pipeline [3]

nfql Tool



nfql architecture

- Execution engine is written in C.
- Parser is written in Python.
- The input and output traces are in NetFlow v5 format.

- **JSON intermediate format**

- Each pipeline stage of the JSON query is a DNF expression.
- JSON query can disable the pipeline stages at RUNTIME.
- Execution engine uses `json-c` to parse the JSON query.

nfql Tool

- Execution workflow

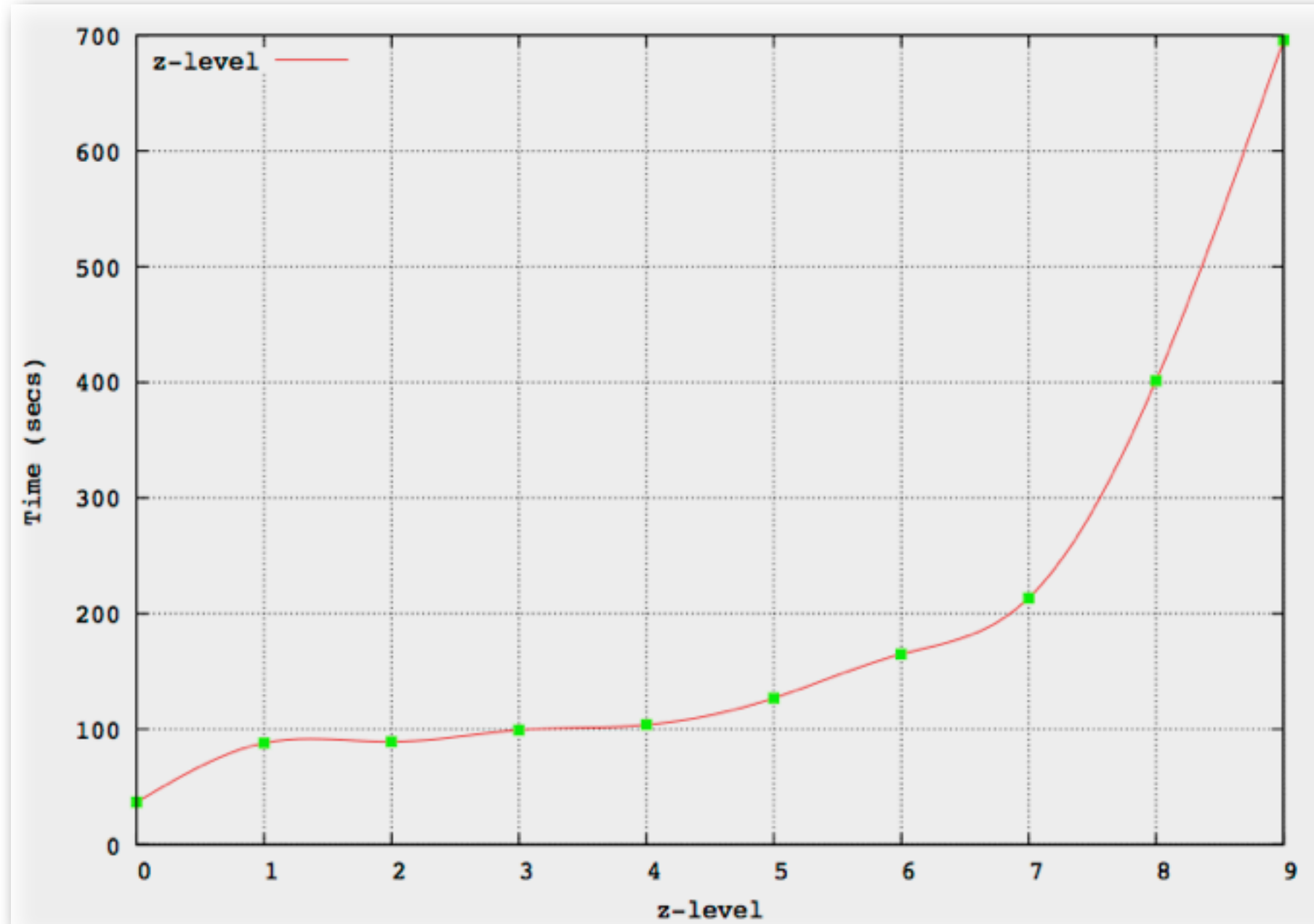
- A custom C library has been written to read/write traces in flow-tools format.
- Flows are read in memory and indexed to allow retrieval in $O(1)$ time.
- Each branch is run in separate thread.

- Performance optimizations

- No splitter: Using indexes to reference flows in each branch.
- Inline filter: Flows are filtered as soon as they are read in memory.
- Faster grouper lookups: Sort on group keys and perform a nested binary search.
- Faster merger matches: Sort on merger keys to skip iterator permutations.

Filter (worst case)	$O(n)$ where $n = \text{num}(\text{flows})$
Grouper (average case)	$O(n \times \lg(k)) + O(p \times n \times \lg(n))$ where $k = \text{num}(\text{unique}(\text{flows}))$, $p = \text{num}(\text{terms})$
Grouper aggregations (worst case)	$O(n)$
Group Filter (worst case)	$O(g)$ where $g = \text{num}(\text{groups})$
Merger (worst case)	$O(g^m)$ where $m = \text{num}(\text{branches})$
Ungrouper (worst case)	$O(g)$

nfql Tool



- Output traces are compressed using zlib library. nfdump uses lzo compression.
- Compression level is configurable at RUNTIME. nfql uses ZLIB_LEVEL 5 by default.
- Each compression level adds its own performance overhead when writing output traces to files.

- **Additional Features**

- Each pipeline stage results can be written out as flow-tools files.
- Capability to read multiple input traces from stdin: `$ flow-cat $TRACES | nfql $QUERY -`

Demo

Thread A

```
branch A {  
  
  filter f1 {  
    dstport=80  
    protocol=TCP  
  }  
  
  grouper g1 {  
    srcaddr = srcaddr  
    dstaddr = dstaddr  
    aggregation {  
      sum(dPkts)  
      sum(dOctets)  
    }  
  }  
  
  groupfilter gf1 {  
    dPkts > 200  
  }  
}
```

Thread B

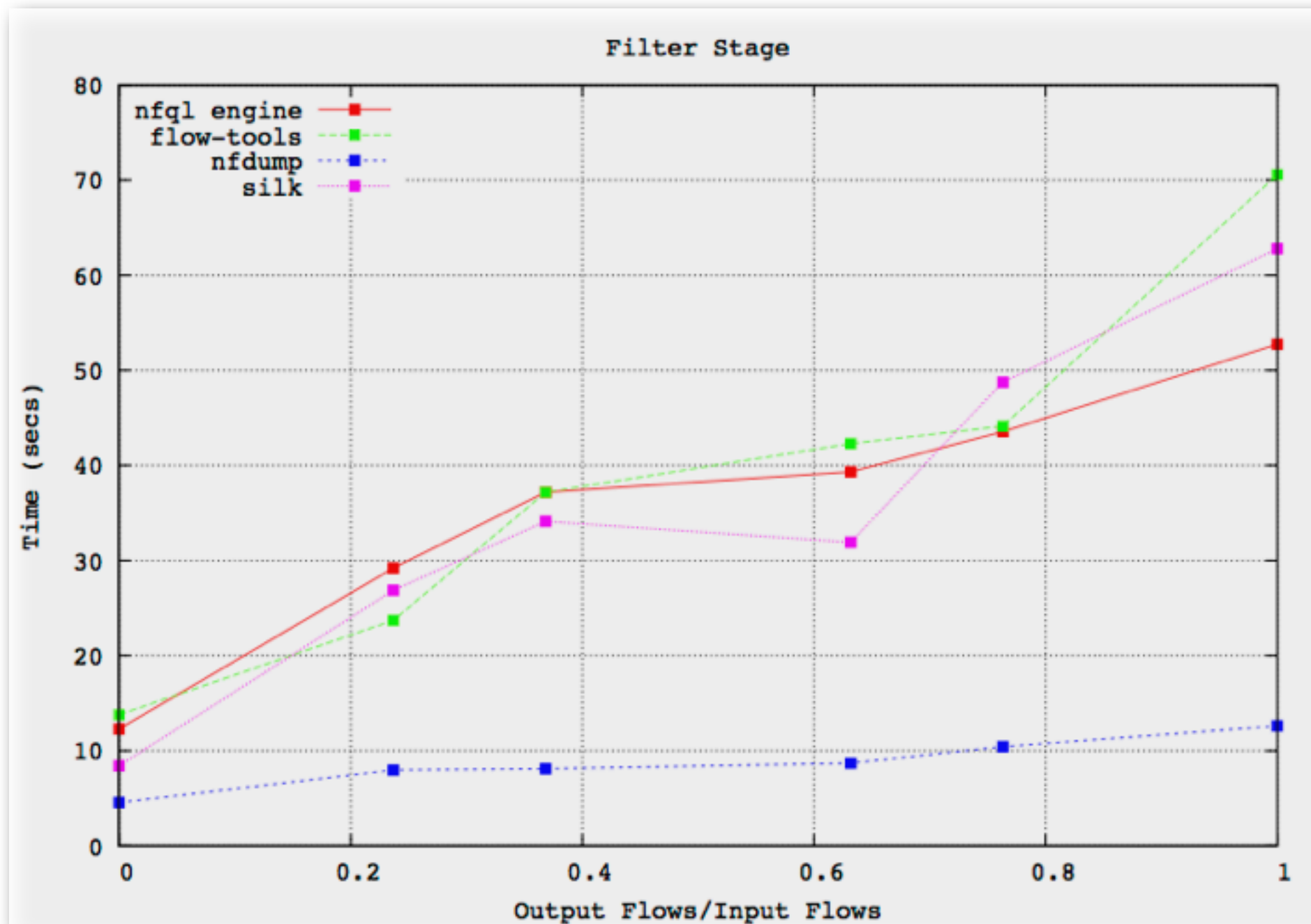
```
branch B {  
  
  filter f1 {  
    srcport=80  
    protocol=TCP  
  }  
  
  grouper g1 {  
    srcaddr = srcaddr  
    dstaddr = dstaddr  
    aggregation {  
      sum(dPkts)  
      sum(dOctets)  
    }  
  }  
  
  groupfilter gf1 {  
    dPkts > 200  
  }  
}
```

```
graph TD; A[Thread A] --> M[merger M]; B[Thread B] --> M; M --> U[ungrouper U]
```

```
merger M {  
  A.srcaddr = B.dstaddr  
  A.dstaddr = B.srcaddr  
}
```

```
ungrouper U {  
}
```

Performance Evaluations



- Used first 20M flows from Trace 7 in the SimpleWeb repository [5].
- Input trace was compressed at ZLIB_LEVEL 5.
- Ran on a machine with 24 cores, 2.5 GHz clock speed and 18 MiB of physical memory.
- nfdump uses lzo compression to trade output trace size with RUNTIME speed.

- Stressing the rest of the pipeline stages (please refer to the paper)
 - flow-tools and nfdump do not have the equivalent functionality to participate.
 - SiLK does not have equivalent Ungrouper functionality.

Conclusion

- NFQL' richer language capabilities allow sophisticated flow queries.
- nfql can process such complex queries in minutes.
- nfql has comparable execution times when processing real-world traces.
- nfql has expanded the scope of current flow-processing tools.
- Evaluation queries developed as part of this research can become input towards a generic benchmarking suite for flow-processing tools.

References

- [1] A. Sperotto, et al., An overview of IP flow-based intrusion detection, IEEE Communication Surveys and Tutorials, 2010.
- [2] A. Callado, et al., A survey on Internet traffic identification, IEEE Communication Surveys and Tutorials, 2009.
- [3] V. Marinov, et al., Design of a stream-based IP Flow Record Query Language, Distributed Systems: Operations & Management, 2009
- [4] V. Perelman, et al., Flow Signatures of Popular Applications, Symposium on Integrated Network Management, 2011
- [5] R. Barbosa, et al., Simpleweb/University of Twente Traffic Traces Data Repository, <http://www.simpleweb.org/wiki/Traces> [Last Accessed: May 25, 2013]