goProbe: A Scalable Distributed Network Monitoring Solution

Christian Decker
Lennart Elsen
Fabian Kohn
Roger Wattenhofer
Goal
Enable quick and efficient retrieval of key pieces of information about traffic patterns in global networks
Goal
Enable quick and efficient retrieval of key pieces of information about traffic patterns in global networks

Scalability
Acquisition of Traffic Data

Packet Capture

Storage
Acquisition of Traffic Data

Packet Capture → Grouping → Information Reduction → Storage
NetFlow

Packet aggregation by set of **shared** attributes

Network packet headers & packet counters

Expiry time
Current Network Monitoring System

- **Exporter**: nProbe
- **DB**: FastBit
- **Query Tool**

**Single Host**

- **Analysts**
- **Request Traffic Metadata**
- **Formatted Results**

**Flow**

- Flow Data

**Aggregated Results**

- Queries
Challenges
Capturing Process

nProbe

FastBit

Query Tool
Challenges
Capturing Process

Immense memory footprint

nProbe → FastBit → Query Tool
Challenges
Capturing Process

One process per capture interface

nProbe nProbe nProbe

FastBit

Query Tool
Challenges
Storage Backend

- nProbe
- nProbe
- nProbe

FastBit

Query Tool
Challenges

Storage Backend

Inefficient memory management

nProbe nProbe nProbe

FastBit

Query Tool
Challenges

Storage Backend

No data compression
Challenges
Storage Backend

FastBit
Query Tool
nProbe
nProbe
nProbe

Long query execution times
Challenges

- FastBit
- nProbe

Poor Scalability
## Reduced Flow Format

<table>
<thead>
<tr>
<th>Src IP</th>
<th>Dst IP</th>
<th>IP Protocol</th>
<th>Src Port</th>
<th>Dst Port</th>
<th>Packets Rcvd</th>
<th>Packets Sent</th>
<th>Bytes Rcvd</th>
<th>Bytes Sent</th>
</tr>
</thead>
</table>

**Shared Attributes**

<table>
<thead>
<tr>
<th>Counters</th>
</tr>
</thead>
</table>

**Diagram:**

- Black boxes represent the flow format.
- Gray boxes indicate shared attributes and counters.
Reduced Flow Format

- Shared Attributes
  - Src Port
  - Dst Port
- Counters
- Deep Packet Inspection
- Appl. Layer Protocol
Reduced Flow Format

Flow in goProbe

Source Port Aggregation

Stored Flow

Deep Packet Inspection

Appl. Layer Protocol

Src Port Dst Port

Appl. Layer Protocol Dst Port
Collection of Flow Information — goProbe

Written in Google Go

One capture routine per interface

Packet capture using modified libpcap

Database flush in regular intervals
goProbe – Concept (Multiple Interfaces)
How does it Compare?

![Graph showing memory consumption over time for nProbe and goProbe.]
Database Performance Evaluation

Reference DB

7.8 GB

Runtime
CPU utilization
Disk I/O
Memory

120 Million Entries

Aggregation Queries

Conditional Queries
<table>
<thead>
<tr>
<th></th>
<th>Runtime [s]</th>
<th>Reserved Memory [MB]</th>
<th>CPU Utilization [%]</th>
<th>Data Read From Disk [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FastBit</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>3300</td>
<td>23</td>
<td>2200</td>
</tr>
<tr>
<td><strong>InfoBright EE</strong></td>
<td>9</td>
<td>1399</td>
<td>213</td>
<td>352</td>
</tr>
<tr>
<td><strong>InfiniDB</strong></td>
<td>17</td>
<td>83</td>
<td>83</td>
<td>302</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>74</td>
<td>350</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>63</td>
<td>105</td>
<td>1405</td>
</tr>
</tbody>
</table>
Tailored Column Store

- File Based
- Compression
- Concurrency
- Independent Processing
Tailored Column Store — goDB

File Based
Compression
Concurrency
Independent Processing
One File per Attribute

- Destination IP
- Source IP
- Destination Port
- IP Protocol
- Appl. Layer Protocol
- Bytes Received
- Bytes Sent
- Packets Received
- Packets Sent
<table>
<thead>
<tr>
<th>Day 1</th>
<th>172.0.50.4</th>
<th>10.30.0.3</th>
<th>8145</th>
<th>6</th>
<th>128</th>
<th>1024</th>
<th>1</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Destination IP</td>
<td>Source IP</td>
<td>Destination Port</td>
<td>IP Protocol</td>
<td>Appl. Layer Protocol</td>
<td>Bytes Received</td>
<td>Bytes Sent</td>
<td>Packets Received</td>
</tr>
</tbody>
</table>
Block-wise Writing and Reading

- Header
- 5 min
- 5 min
- 5 min
- Compressed Block
- Attribute File
- Position
- Length of Uncompressed Block
- Block Timestamps
Concurrent Processing

Full Database

Day 1

Day d
Concurrent Processing
Day 1

Worker 1

Decompress
Aggregate

Day d

Worker d

Partial Result
Block i, Day 1

<table>
<thead>
<tr>
<th>sip</th>
<th>dip</th>
<th>counters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Partial Result
Block j, Day d

<table>
<thead>
<tr>
<th>sip</th>
<th>dip</th>
<th>counters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concurrent Processing

Day 1

Worker 1

Decompress Aggregate

Worker d

Day d

Partial Result Block i, Day 1

<table>
<thead>
<tr>
<th>sip</th>
<th>dip</th>
<th>counters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Partial Result Block j, Day d

<table>
<thead>
<tr>
<th>sip</th>
<th>dip</th>
<th>counters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Merge Routine

Combined Result

<table>
<thead>
<tr>
<th>sip</th>
<th>dip</th>
<th>counters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concurrent Processing

Day 1
Worker 1
Decompress
Aggregate

Worker d

Day d

Partial Result
Block i, Day 1

Partial Result
Block j, Day d

Merge Routine

Combined Result

Format
Sort
Limit

Concurrent Processing

Decompress
Aggregate

Partial Result
Block i, Day 1

Partial Result
Block j, Day d

Merge Routine

Combined Result

Format
Sort
Limit

Concurrent Processing

Decompress
Aggregate

Partial Result
Block i, Day 1

Partial Result
Block j, Day d

Merge Routine

Combined Result

Format
Sort
Limit
### How does it Compare?

<table>
<thead>
<tr>
<th></th>
<th>Runtime [s]</th>
<th>Reserved Memory [MB]</th>
<th>CPU Utilization [%]</th>
<th>Data Read From Disk [MB]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FastBit</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runtime</td>
<td>60</td>
<td>3300</td>
<td>123</td>
<td>2200</td>
</tr>
<tr>
<td>Reserved Memory</td>
<td>63</td>
<td>1399</td>
<td>17</td>
<td>2000</td>
</tr>
<tr>
<td>CPU Utilization</td>
<td>23</td>
<td>47</td>
<td>237</td>
<td>494</td>
</tr>
<tr>
<td>Data Read From Disk</td>
<td>494</td>
<td>760</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>goDB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Runtime</td>
<td>13</td>
<td>47</td>
<td>20</td>
<td>13</td>
</tr>
<tr>
<td>Reserved Memory</td>
<td>20</td>
<td>50</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>CPU Utilization</td>
<td>237</td>
<td>50</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>Data Read From Disk</td>
<td>123</td>
<td></td>
<td></td>
<td>760</td>
</tr>
</tbody>
</table>
Traffic Portfolio of an NGO Customer
Global Breakdown of Ports

External Traffic

(a) UDP: 32.3 GB

(b) TCP: 4.7 TB

Internal Traffic

(c) UDP: 674.1 GB

(d) TCP: 20.7 TB
Global Breakdown of Ports

External Traffic

- HTTPS
  - 5407: 34.4%
  - 514: 6.9%
  - 49156: 5.6%
  - Other: 45.9%

- HTTP
  - 58.7%
  - Other: 2.5%

Internal Traffic

- DNS
  - 5405: 32.7%
  - 2000: 1.5%
  - 5247: 28.1%
  - Other: 31.4%

- SMB
  - 443: 47.7%
  - 80: 21.5%
  - 8530: 10.5%
  - Other: 16.1%
Conclusion

Improved capturing and flow logic

High performance DB written from scratch

Global deployment

Open source:

https://github.com/open-ch/