Detecting distributed attacks using distributed processing frameworks

RP2 #59
Sudesh Jethoe

Advanced ICT Services

UNIVERSITEIT VAN AMSTERDAM

Overview

- Introduction
- Problem Description
- Research Questions
- Method
- Results
- Conclusion

Introduction

http://www.eweek.com/security/slideshows/verisign-sees-sharp-climb-in-ddos-attack-volume-in-q2.html/

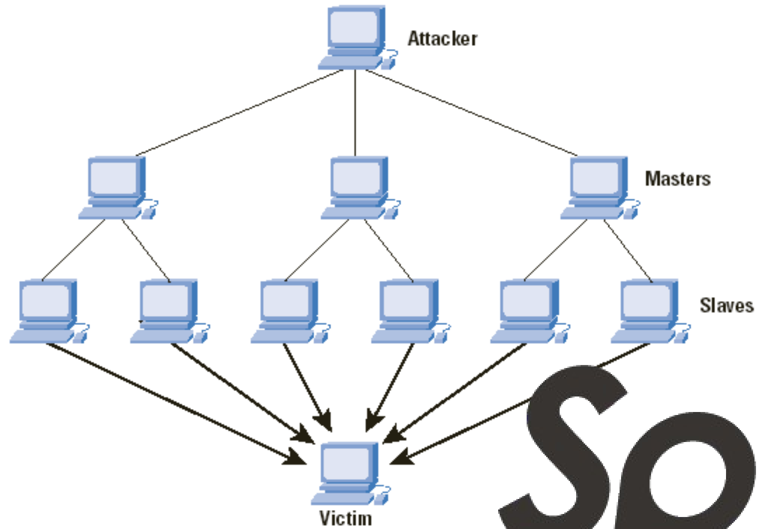
Overview

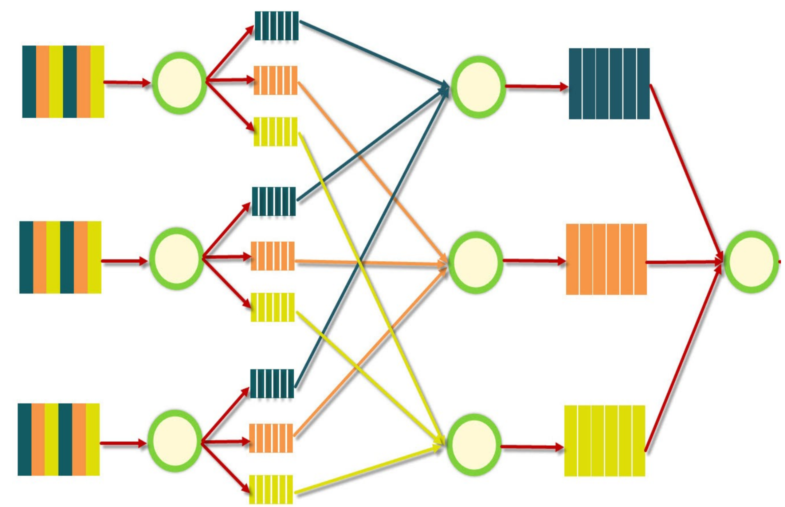
- Introduction
- Problem Description
- Research Questions
- Method
- Results
- Conclusion

Problem Description

- Analysis of large volumes of network traffic data takes time
- A lot of time
- Can we make it faster?

Solution?





Overview

- Introduction
- Problem Description
- Research Questions
- Method
- Results
- Conclusion

Research Questions

Main research question:

- How can a distributed processing framework be utilized to identify network anomalies in historical netflow data?

Sub questions:

- Which processing framework is best suited for identifying DDOS attacks?
- How can we distinguish anomalies in netflow data?
- Which algorithms for detecting network anomalies exist and how can they be applied in a distributed processing environment?

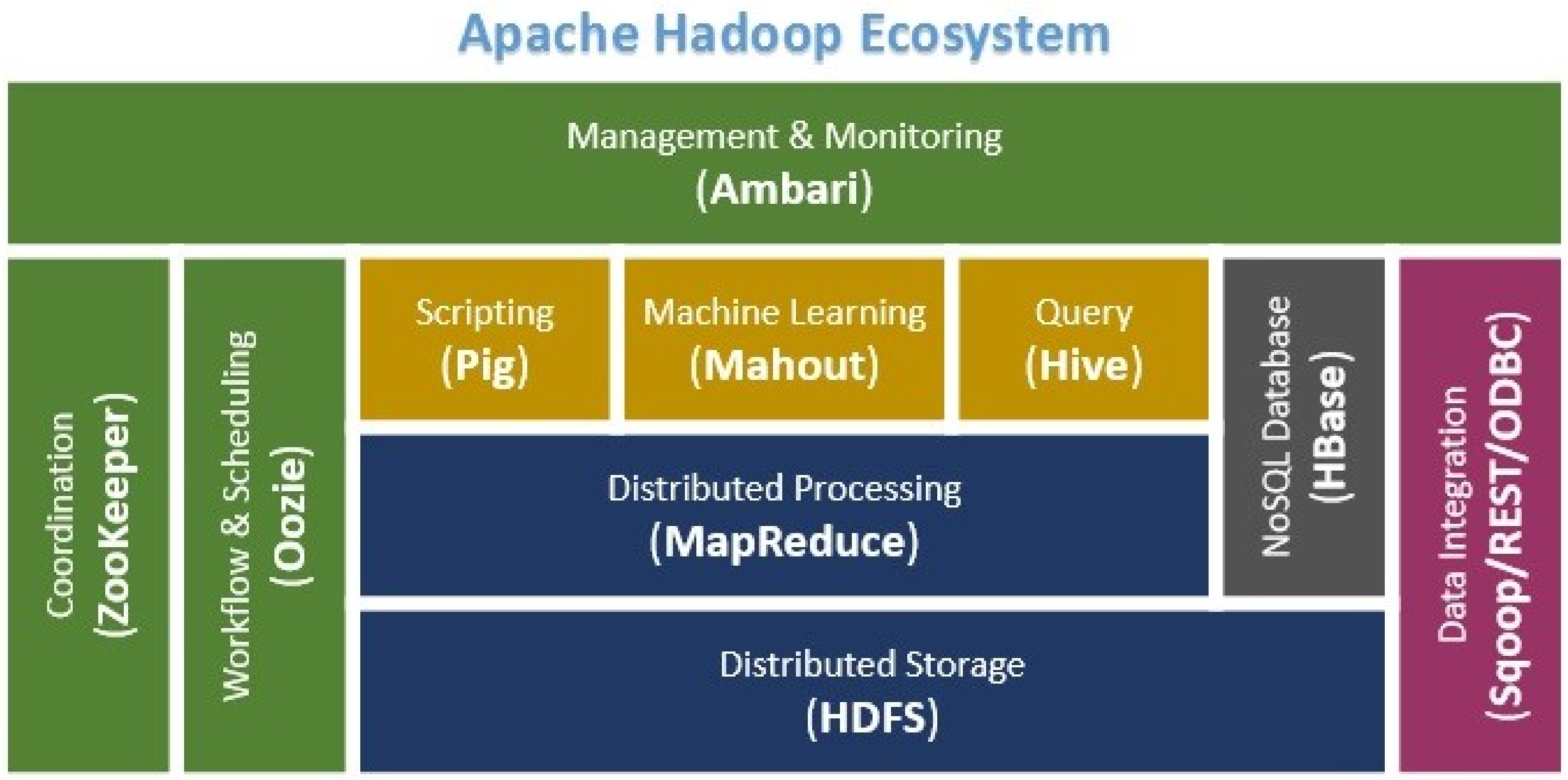
Overview

- Introduction
- Problem Description
- Research Questions
- Method
- Results
- Conclusion

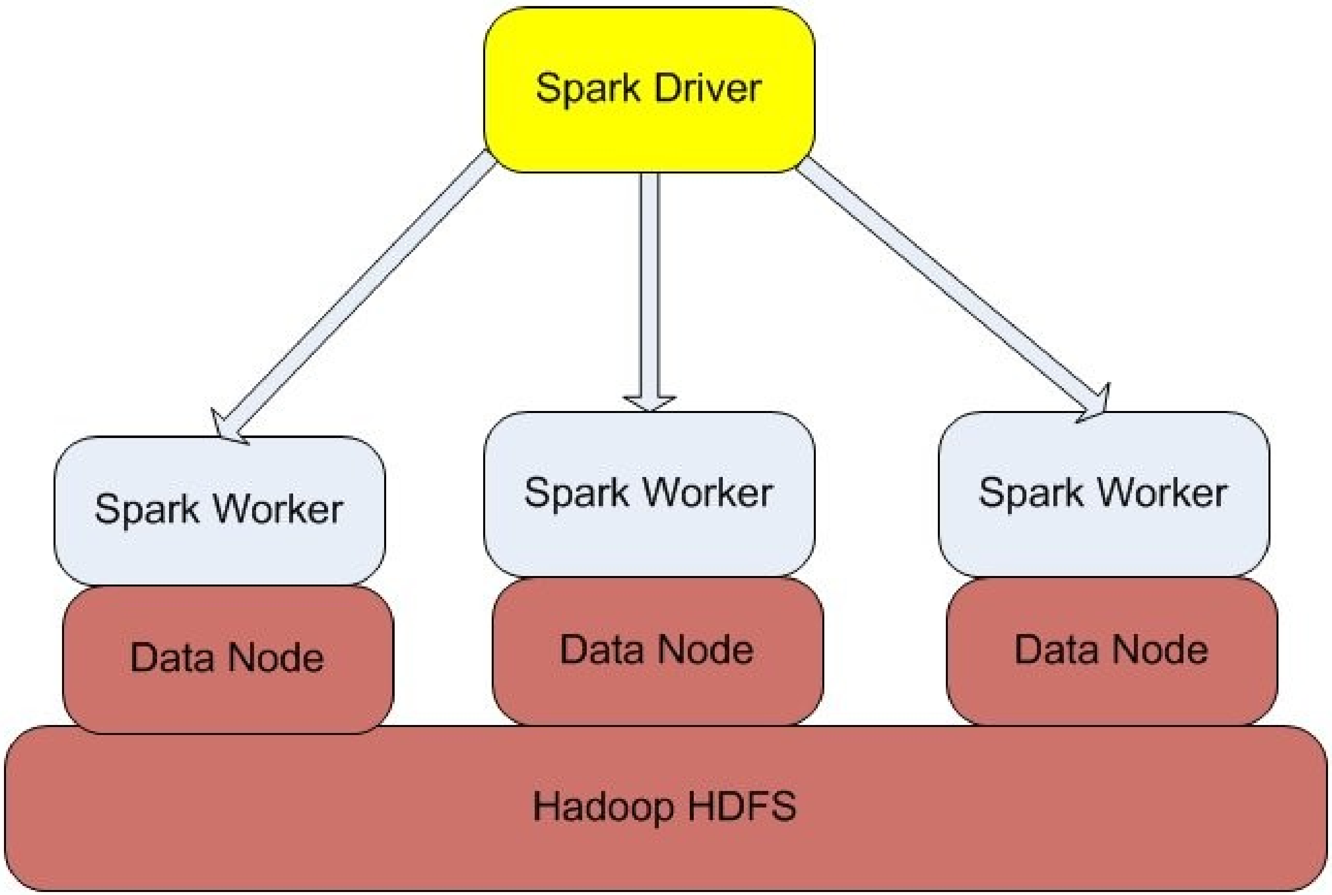
Method

1) Review distributed processing frameworks
2) Create application for distributed processing framework
3) Implement DDOS-algorithm in application

Distributed processing frameworks



Distributed processing frameworks



Distributed processing frameworks

- Hive
  - Limited to querying datasets
- Pig
  - Extend queries with scripting and ML
- Spark
  - Extract data, transform, query, extendable python

Method

1) Review distributed processing frameworks
2) Create application for distributed processing framework
3) Implement DDOS-algorithm in application

Implementing Spark

- Cluster
  - 26 nodes
  - 2x2TB disks
  - AMD Opteron 3vCPU
  - 1GB/s ethernet

- Dataset

Route r	Dataset Size
1	83,4 MiB
2	126,7 MiB
3	1,1 GiB
4	3,1 GiB
5	10 GiB
6	41,5 GiB
7	88,2 GiB
8	99,3 GiB
9	296,4 GiB
10	444,4 GiB

Implementing Spark

- 3 methods
  - Traditional
  - Parallelised
  - Single MapReduce

Implementing Spark

- Traditional

1) retrieve unique intervals
2) partition the data by interval
3) for each interval create counts of packets for each found socket

- Result

  > 1,5 hour / 84,4 MiB

Implementing Spark

- Parallelised

1) retrieve unique intervals
2) partition the data by interval
3) Parallel: for each interval create counts of packets for each found socket

- Result

~ 10 mins / 126,7 MiB

Implementing Spark

- Single MapReduce

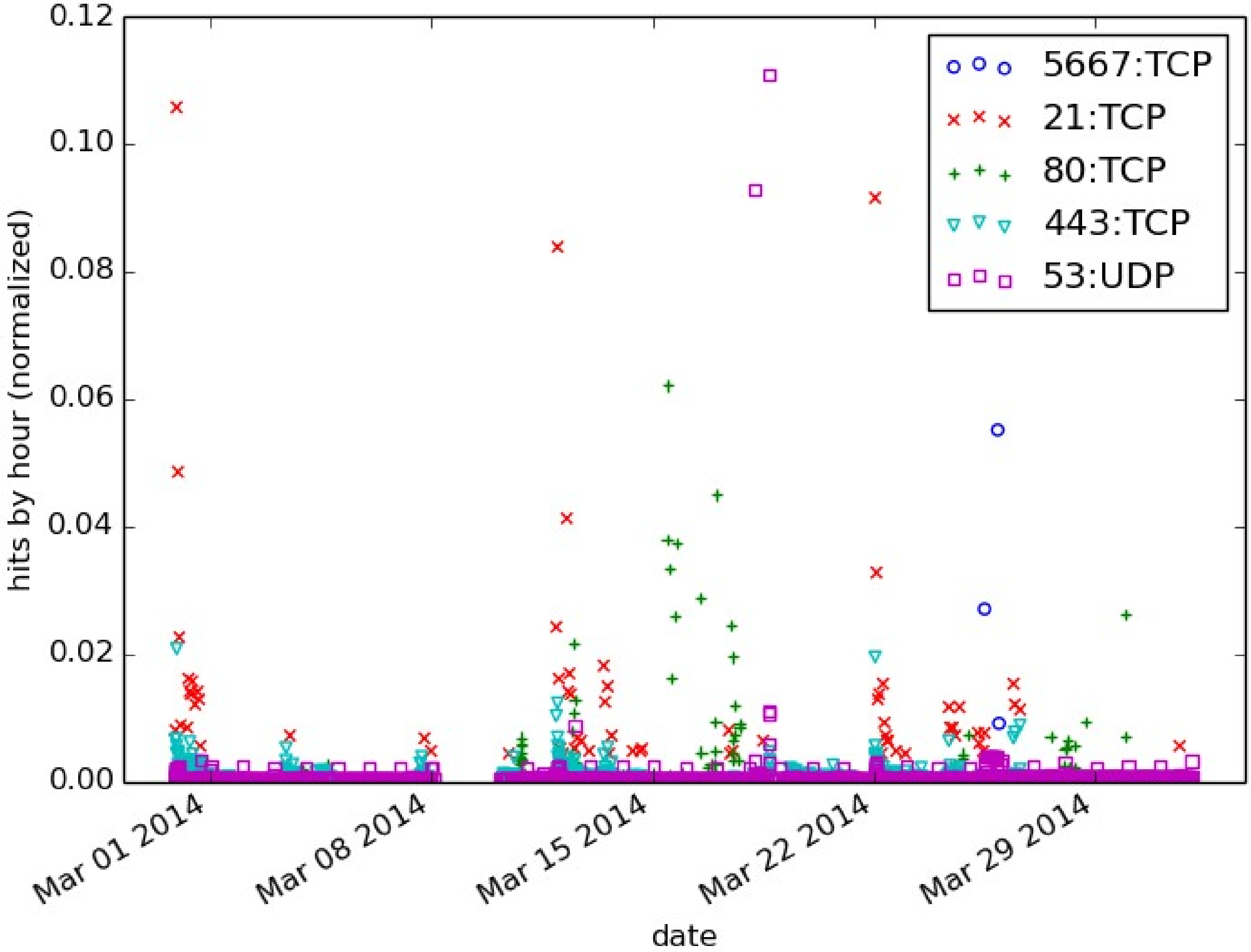
1) Initialize cluster
2) Read network traffic data from HDFS
3) Apply map/reduce to get flow counts for “dest IP:port:protocol:hour”
4) Filter out all counts < #threshold
5) Group results by “port:protocol”
6) Filter out all combinations < #min results
7) Normalize results by “port:protocol
8) Plot all hits for remaining “port:protocol” combinations

Implementing Spark

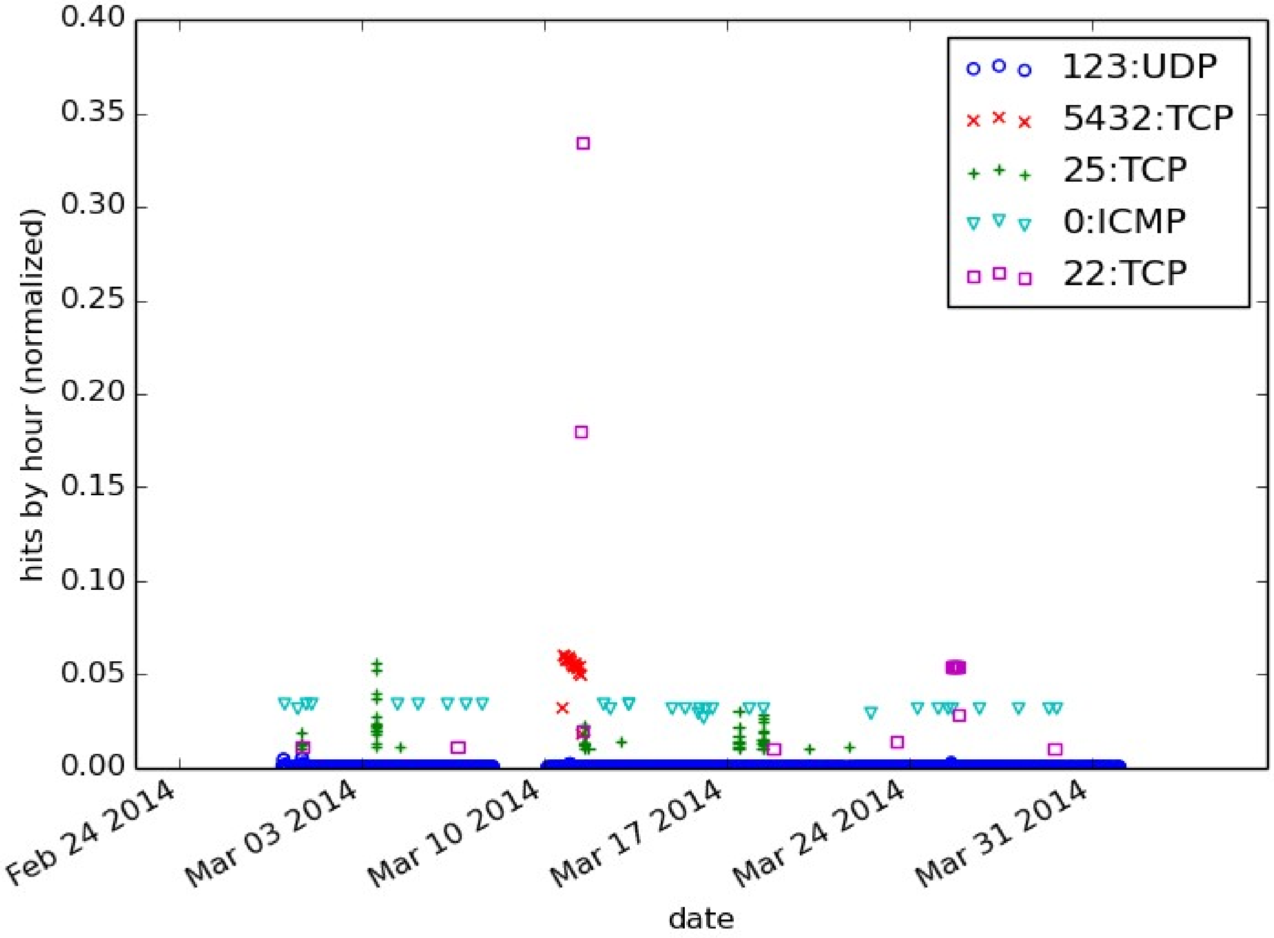
- Results

Dataset Size (GiB)	Execution Time (seconds)	Rate (MiB/seconds)
0,128	28	4,57
1,1	45,6	4,07
99,3	430,4	231
444,4	/	/

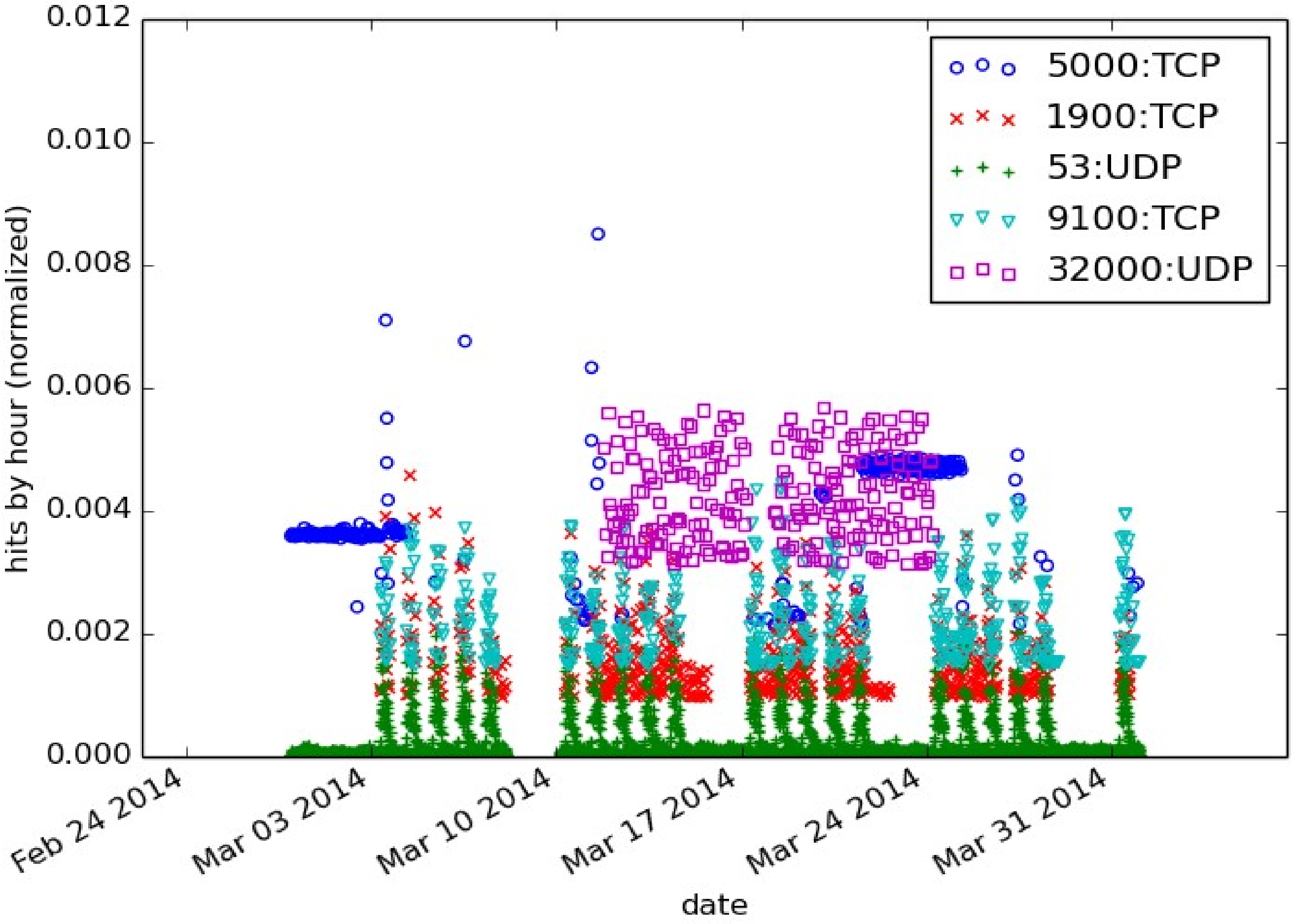
Results (126,7 MiB)



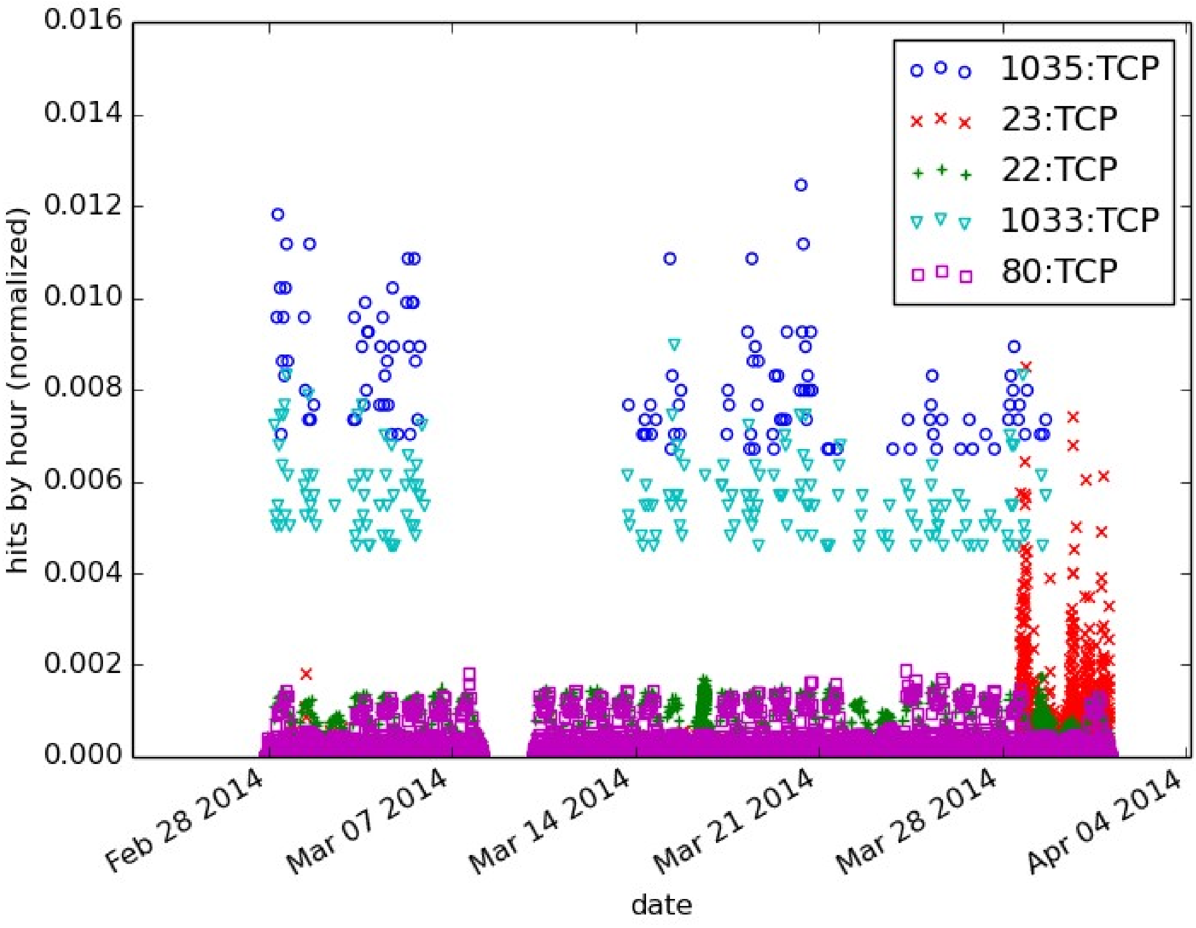
Results (126,7 MiB)



Results (88,2 GiB)



Results (10,0 GiB)



Method

1) Review distributed processing frameworks
2) Create application for distributed processing framework
3) Implement DDOS-algorithm in application

Implement DDOS-algorithm in application

- Weighted Moving Average

$$\hat{x}_{(i+1)} = y x_i + (1 - y)\hat{x}_i$$

$x_i : current\ value\ of\ x$

$\hat{x} : estimation\ x$

$y : smoothing\ factor$

Implement DDOS-algorithm in application

- Adaptive threshold
  - Uses weighted average
  - Threshold: Multiple of expected value of the average

$$alert\ if\ x_i > threshold * \hat{x}_i$$

Implement DDOS-algorithm in application

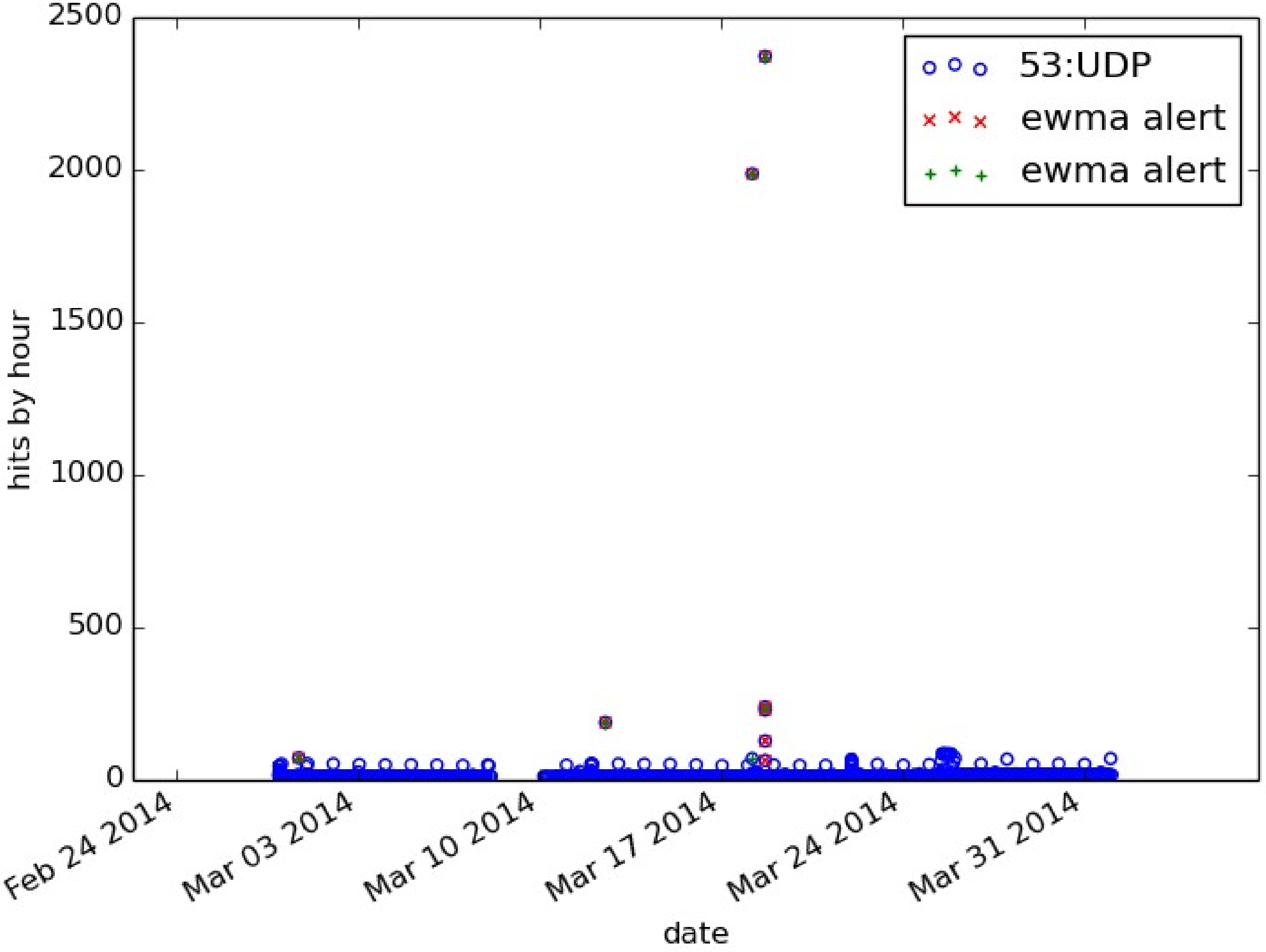
- Exponential Weighted Moving Average (EWMA)
- Threshold

```
Gap = 0, avg = X0, Max_Gap = #
If Xi < AVG:
    update(AVG, Xi)
If Xi > AVG:
    Alert()
    If Gap >= Max_Gap:
        Gap = 0
        update(AVG, Xi)
    Gap +=1
```

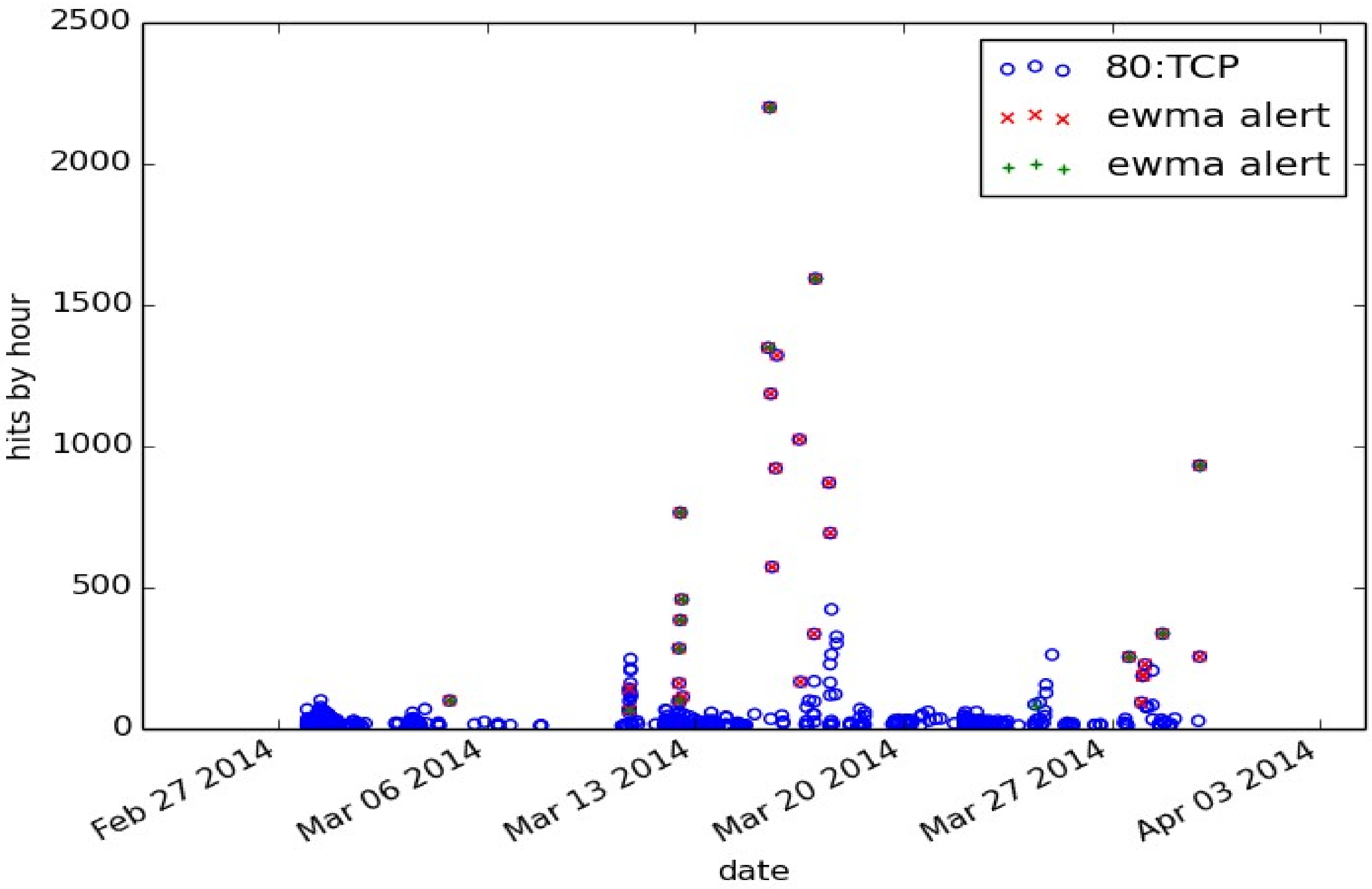
Overview

- Introduction
- Problem Description
- Research Questions
- Method
- Results
- Conclusion

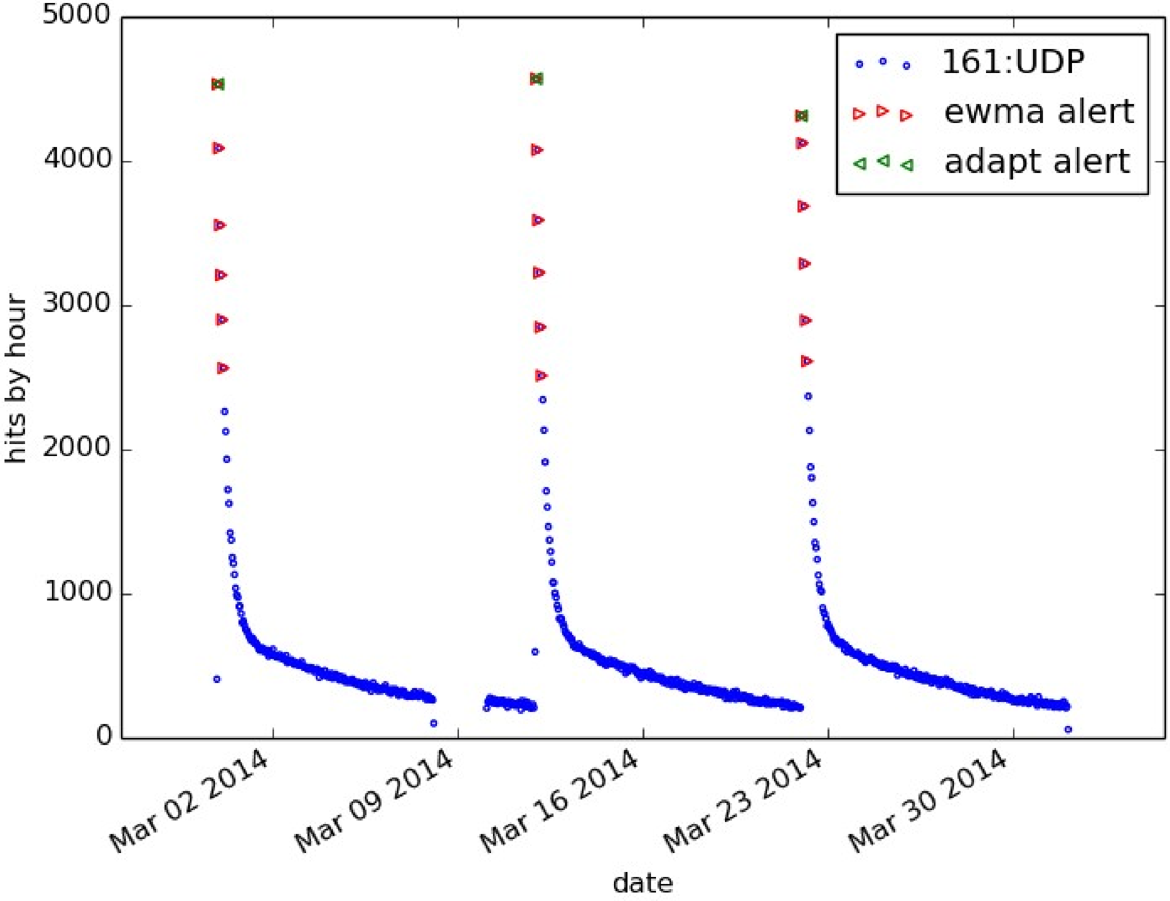
Results (training 126,7MiB)



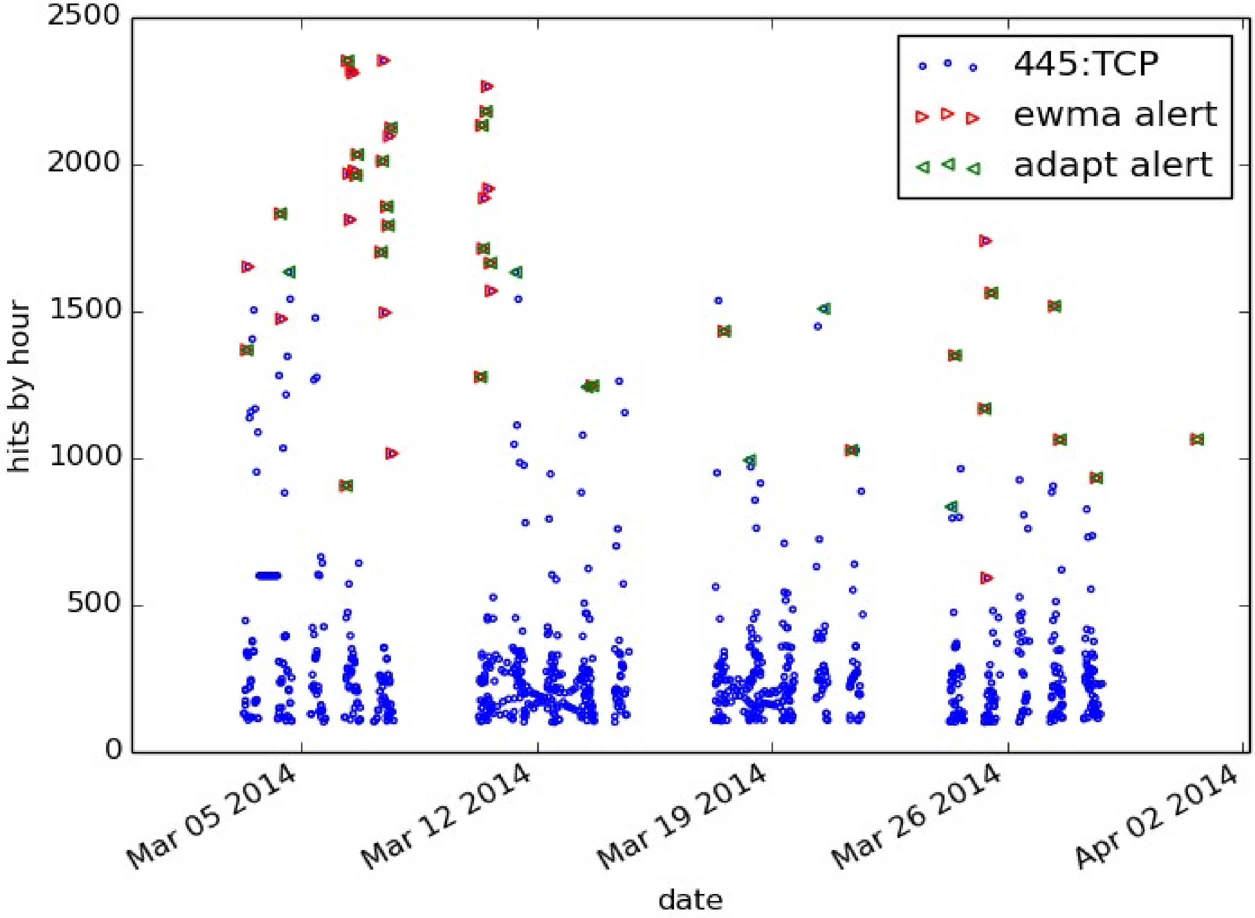
Results (training 126,7MiB)



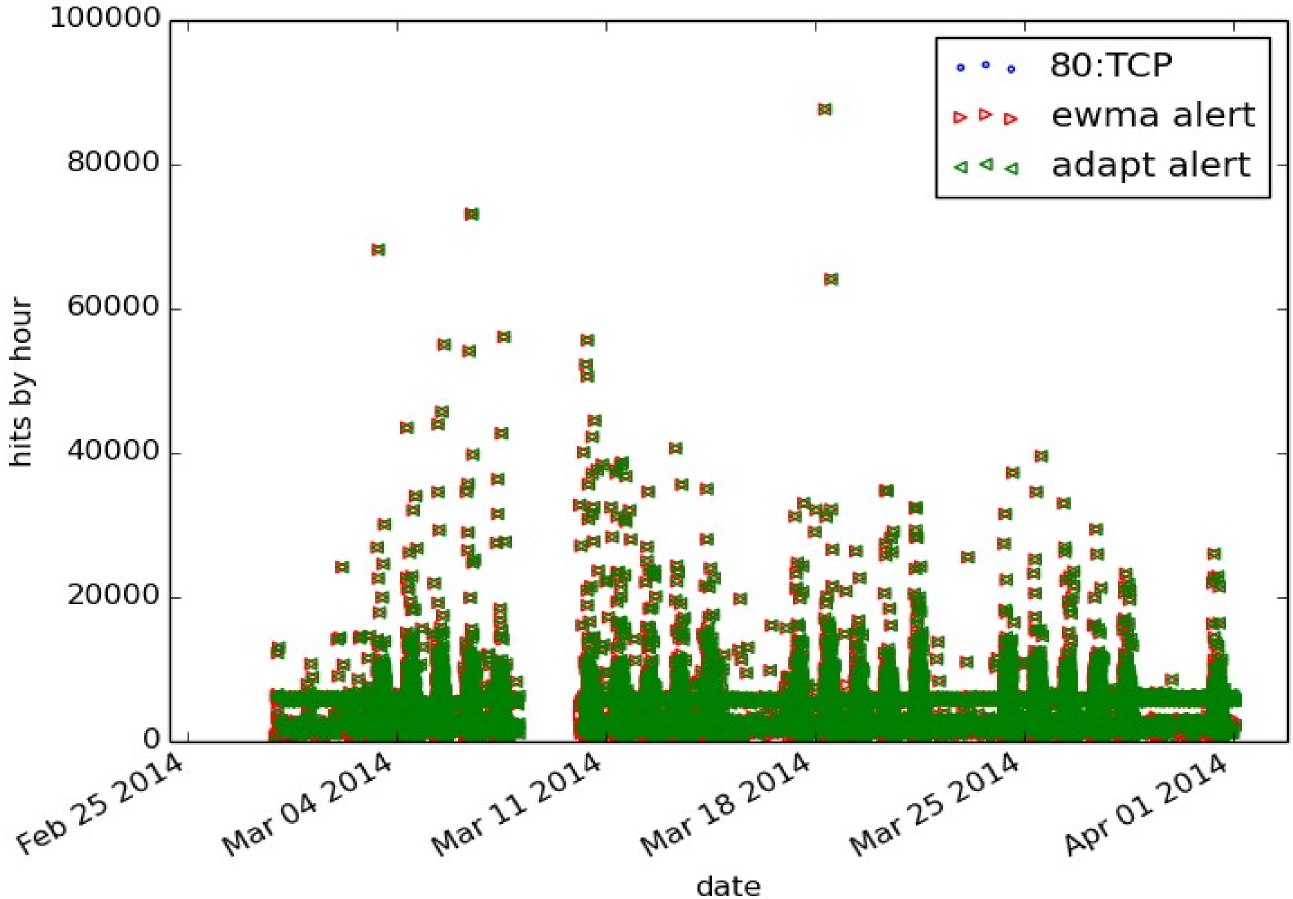
Results (84,3MiB)



Results (88,2 GiB)



Results (88,2 GiB)



Overview

- Introduction
- Problem Description
- Research Questions
- Method
- Results
- Conclusion

Conclusion

- ~ 100 GiB < 10 minutes
- Traffic from different routers require different parameters
- Traffic patterns differ per router and service

Future work

- Optimize framework to handle datasets > 100 GiB
- Test other algorithms on framework
- Apply tuned algorithms to live data
- Identify usage of irregular ports

Questions

- ?