A Performance Benchmark for NetFlow Data Analysis on Distributed Stream Processing Systems

NOMS 2016

Wednesday 27th April, 2016

Milan Čermák

Daniel Tovarňák, Martin Laštovička, Pavel Čeleda

T A
Č R

This research was supported by the Technology Agency of the Czech Republic under No. TA04010062 Technology for processing and analysis of network data in big data concept.

NetFlow/IPFIX Monitoring and Analysis

Network Monitoring using NetFlow/IPFIX

Flow Monitoring

- Groups packets into n-tuples that have common properties.
- From the IP point of view we know who communicates with whom, when, and for how long.
- Used for network traffic measurement in high-speed and large-scale networks.

RFC 7011

A flow is defined as “a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties”

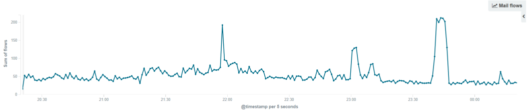
Disadvantages of Flow Data Analysis

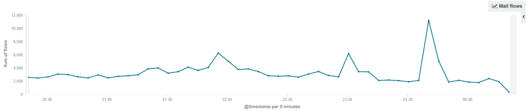
Not real-time

- Flow data typically analysed in 5 minute intervals
- Delayed detection of serious network attacks

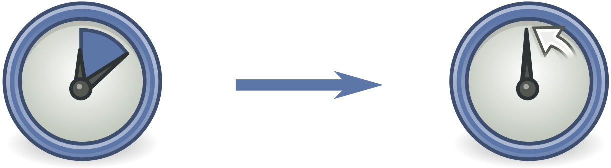
Hidden network traffic characteristics

- Invisible peaks
- Distorted traffic statistics





Solution?



Solution?

samza
Spark
STORM
EsperTech
streambase
Flink

Distributed Stream Processing Systems

	Samza	Storm	Spark
Data source	Consumer	Spout	Receiver
Cluster manager	YARN, Mesos	YARN, Mesos	Standalone, YARN, Mesos
Parallelism	Stream partitions based	Configured in Topology	Configured in SparkContext
Message processing	Sequential	Sequential	Small batches
Data sharing between nodes	Database, User implemented communication	Database, User implemented communication	Proprietary – SparkContext, Tachyon
Programming language	Java, Scala	Java, Clojure, Scala, any other using JSON API	Java, Scala, Python
Time window	Proprietary	User definition of Spout	Proprietary
Count window	Separate Job	User definition of Bolt	Accumulator

Table: Characteristics of Distributed Stream Processing Systems

Benchmark of Distributed Stream Processing Systems

Benchmark of Stream Processing Systems

Benchmark characteristics

- Follows the universal Stream Bench benchmark by Lu et al.
- Focus only on the flow throughput, not on fault tolerance or durability.
- Using real network data and common operations.
- Benchmark of standard systems without specific optimizations.
- Throughput measured using dataset size, time between computation start and arrival of predetermined computation result.

Benchmark of Stream Processing Systems

Dataset

- Based on the CAIDA network traffic public dataset.
- PCAP transformed into flows represented in the JSON format (∼270 bytes).
- Basis formed from one million flows of the one IP address.
- Final dataset consist repetitive insertions of the basis corresponding to the number of available processor cores.

```
{"date_first_seen":"2015-07-18T18:07:33.475+01:00",
 "date_last_seen":"2015-07-18T18:07:33.475+01:00",
 "duration":0.000,"src_ip_addr":"86.135.210.175",
 "dst_ip_addr":"31.157.1.1","src_port":54700,
 "dst_port":80,"protocol":6,"flags":".A....",
 "tos":0,"packets":1,"bytes":56}
```

Benchmark of Stream Processing Systems

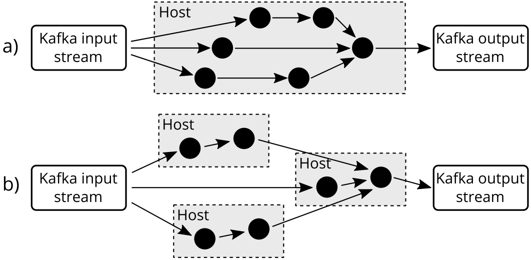
Selected operations

1. **Identity:** Input data processing without executing any operation on them.
2. **Filter:** Only flows fitting a filtering rule are selected from the input dataset and sent to the output.
3. **Count:** Flows containing a given value are filtered and their count is returned as a result.
4. **Aggregation:** Contrary to the count operation, the aggregation sums specific values over all flows.
5. **TOP N:** An extension of the aggregation returning only a given number of flows with the highest sums of values.
6. **SYN DoS:** The detection of an attack represented by a high number of flows from one source IP address with TCP SYN packets only.

Benchmark of Stream Processing Systems

Benchmark architecture

- Corresponds to a typical deployment architecture of the distributed stream processing systems.
- Utilization of the Kafka as the messaging system.
- Two environments: a) *single host* and b) *multiple hosts*.



Benchmark Results

Testbed Configuration

Common configuration of nodes

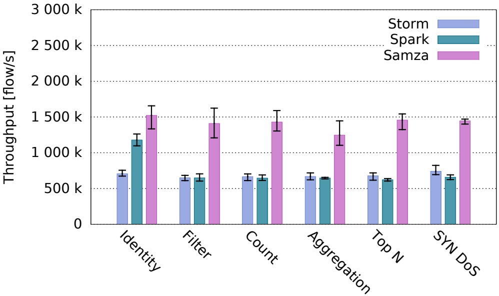
- 2 x Intel® Xeon® E5-2670 (16/32 HT cores in total),
- 192 GB 1600M MHz RDIMM ECC RAM,
- 2 x HDD 600 GB SAS 10k RPM, 2,5" (RAID1),
- 10 Gbit/s network connection, 1 Gbit/s virtual NICs.

Virtual machines configuration

Type	vCPUs	Memory	Hard Drive
vm_large	32	128 GB	300 GB
vm_normal	16	64 GB	300 GB
vm_medium	8	32 GB	300 GB
vm_small	4	16 GB	300 GB

Benchmark Results

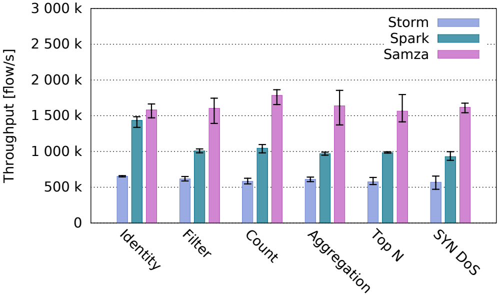
One vm_large node (32 vCPUs in total)



- Samza provides almost constant throughput for all operations.
- Strom and Spark decreases to 700 k flows/s.
- Throughput slowdown probably caused by shuffling of incoming messages, which led to input socket overloading.

Benchmark Results

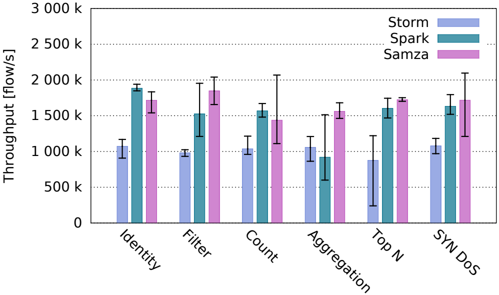
One vm_normal node (16 vCPUs in total)



- Lower computational resources reduce the internal data processing speed and shuffling of messages.
- Input socket not overloaded.
- Significant increase in Spark throughput.

Benchmark Results

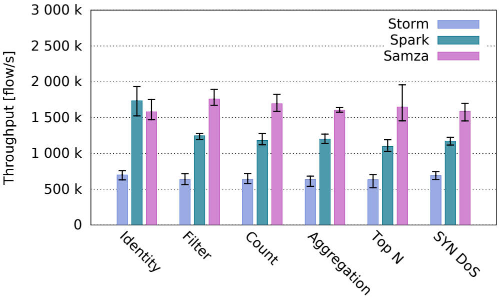
Four vm_medium nodes (32 vCPUs in total)



- Systems are better adapted to deployment in a cluster mode.
- Spark provides similar throughput as Samza.
- Large throughput variance probably caused by the network load or systems errors.

Benchmark Results

Four vm_small nodes (16 vCPUs in total)



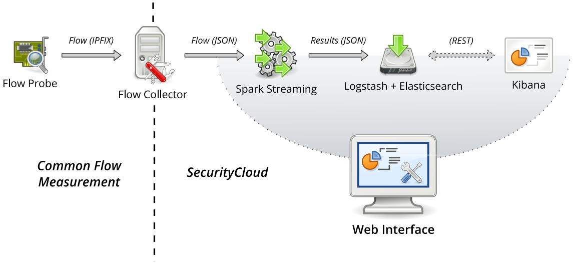
- No increase in data processing speed.
- Throughput of Storm reduced by half.
- Samza, deployed on 32 vCPUs was probably limited by a network bandwidth saturation.

Benchmark Summary

- Benchmarked systems are able to process at least 500 k flows/s.
- Spark and Samza offer much higher throughput than Storm.
- Possibility of a higher throughput using more efficient data format than JSON (MessagePack).
- Hight throughput on single node offers to combine stream processing with standard flow processing tools like NFDUMP.
- Each of tested systems have specific behaviour depending on the cluster setup.
- Samza has the best throughput but restricts number of partitions to number of available cores.

Framework for Real-time Analysis of NetFlow Data

Real-time Analysis of NetFlow Data



- Framework for the real-time generation of network traffic statistics using Apache Spark Streaming.
- Possibility to implement the same basic methods for flow data analysis.
- Will be presented on the Demo Session on Thursday.

Conclusion

- Proposed the novel performance benchmark of a flow data analysis on distributed stream processing systems.
- Testing using real network traffic dataset and common data analysis operations.
- Only Samza and Spark provides a high-enough flow throughput.
- The benchmark source code and dataset preparations scripts are available on: `https://is.muni.cz/repo/1323006`

A PERFORMANCE BENCHMARK FOR NETFLOW DATA ANALYSIS ON DISTRIBUTED STREAM PROCESSING SYSTEMS

Milan Čermák

cermak@ics.muni.cz