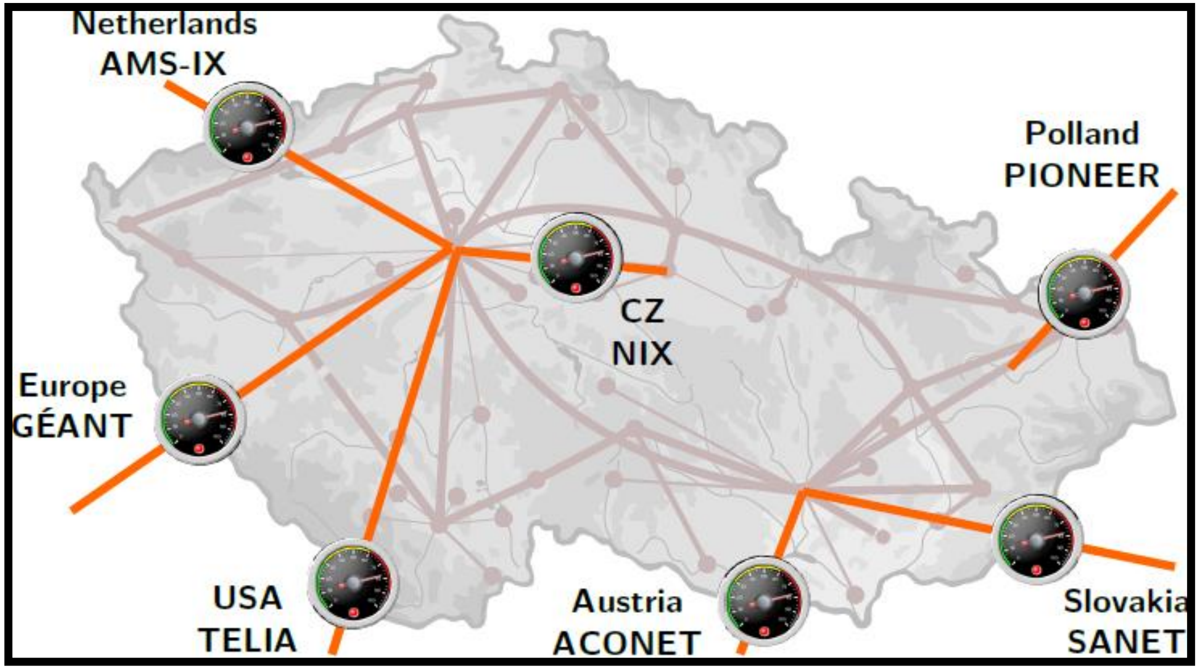
Flow data storage and retrieval utilizing big data aproach

CESNET, INVEA-TECH, MU

Motivation

- Network flow monitoring generates large amount of data – 250 GB per day
- Interactive work with data is an issue



Intro

- There are several open-source platforms enabling big data processing
  - Hadoop, (native, Hive, Pig, nfdist) – MapReduce
  - ElasticSearch
  - Vertica
  - Proprietary implementation

Queries

- Query *1*: Total number of flows, packets, bytes
- Query 2: Number of flows with port 53 and proto TCP
- Q*uery 3:* Print flows with destination port 53.
- Q*uery 4:* Print IP adresses sorted by bytes with flows, packets and bytes

Data

- One 10Gbps line
- 24 hours
- 877 million of flow records
- Records simplified to NetFlow v5 equivalent
- CSV as well as binary data representation

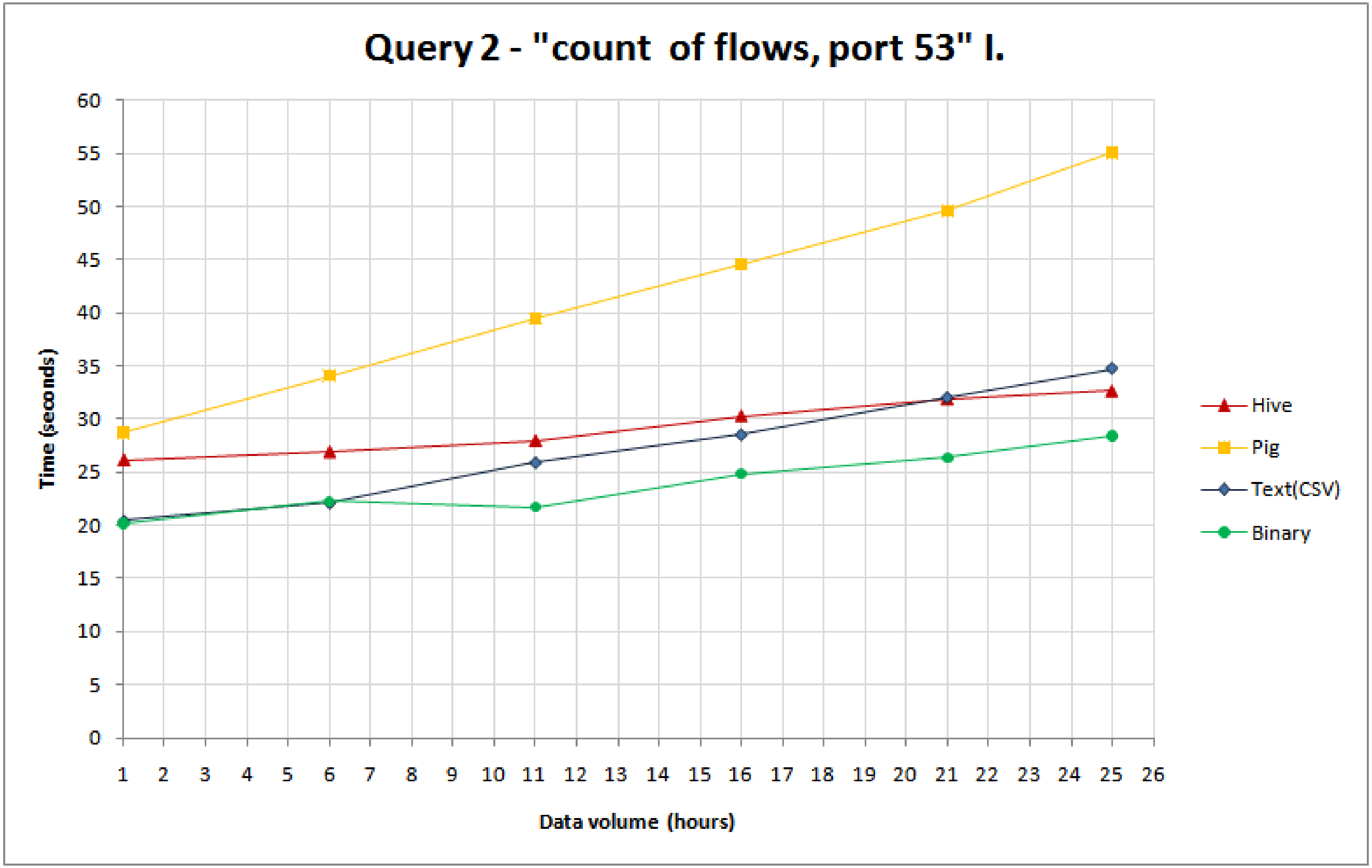
Hadoop cluster

- 24 slave + 3 master nodes
- Intel Xeon CPU E5-2630 v3 @ 2.40GHz
- 128 GB RAM each node
- Total disk capacity: 1 PB

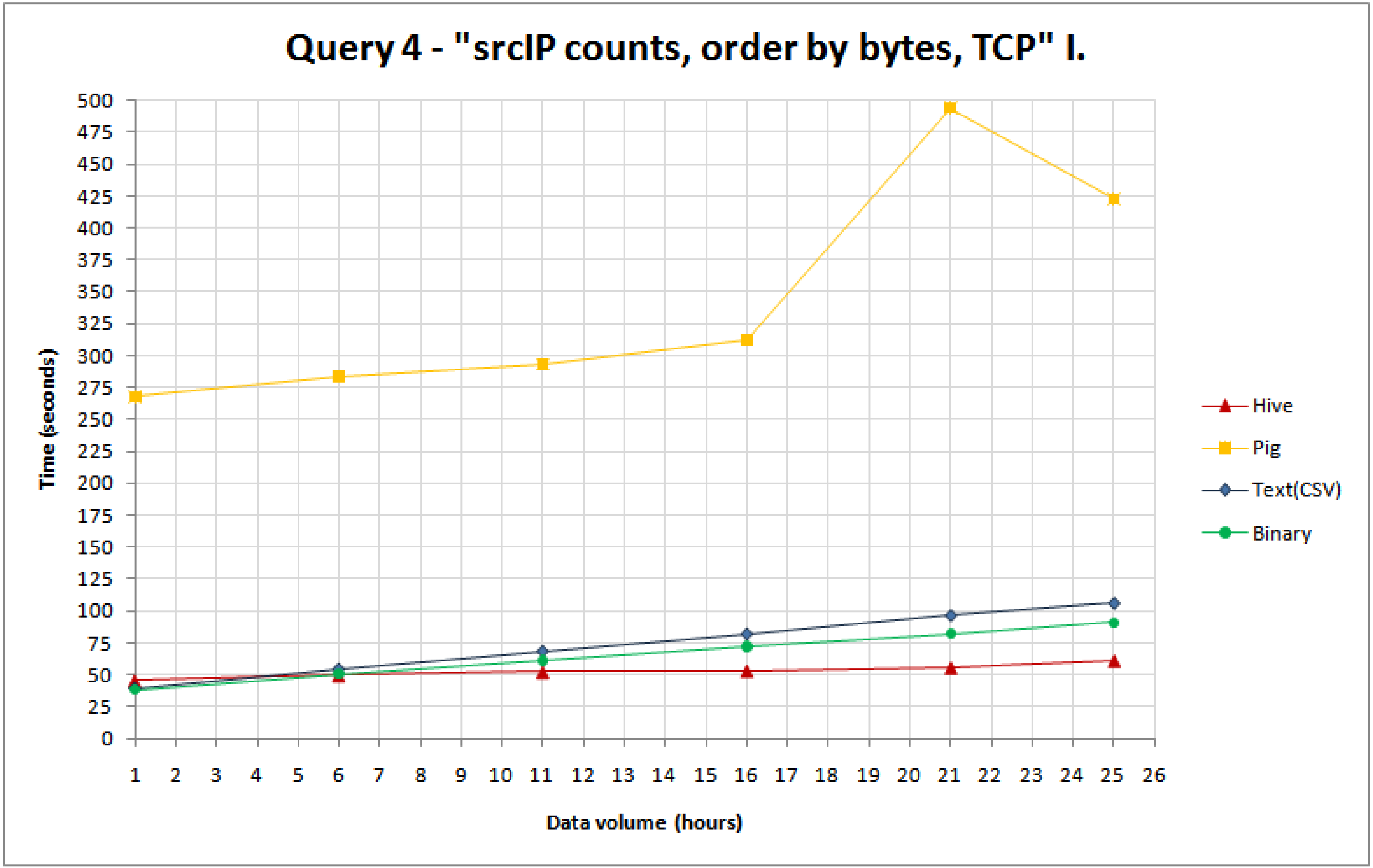
Hadoop, Hive, Pig

- Hadoop configuration can be customized, e.g. replication factor, heartbeat
- Queries in Hadoop are written in Java as MapReduce operations, text and binary format
- Hive is an SQL interface into Hadoop, data are uploaded into Hive representation
- Pig is a functional interface into Hadoop, data are stored in CSV format

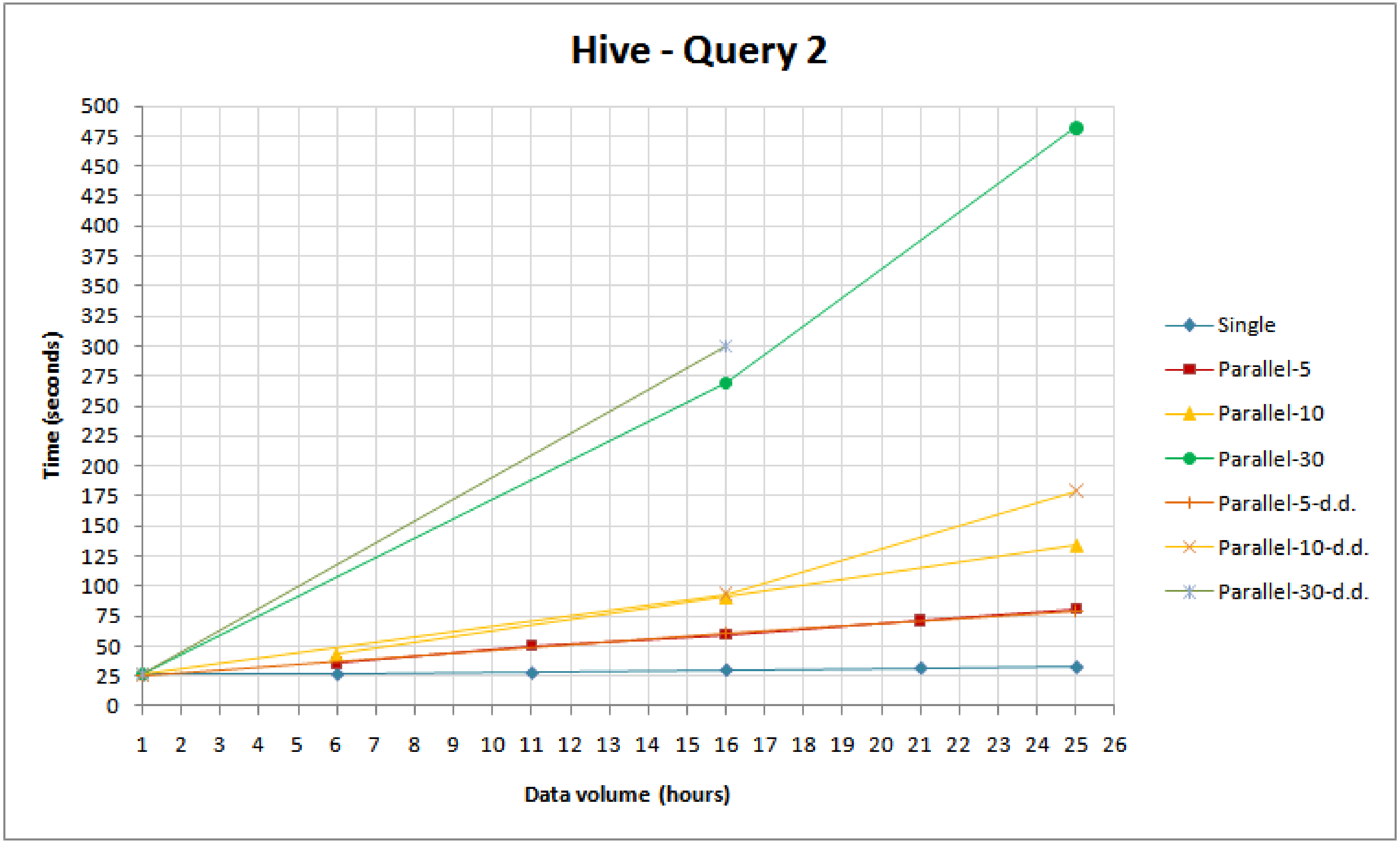
Results


Query 2 - "count of flows, port 53" I.
Time (seconds)
0
5
10
15
20
25
30
35
40
45
50
55
60
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
Data volume (hours)
Hive
Pig
Text(CSV)
Binary

Results


Query 4 - "srcIP counts, order by bytes, TCP" I.
Time (seconds)
500
475
450
425
400
375
350
325
300
275
250
225
200
175
150
125
100
75
50
25
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
Data volume (hours)
Hive
Pig
Text(CSV)
Binary

Results: Hive parallel



Hadoop summary

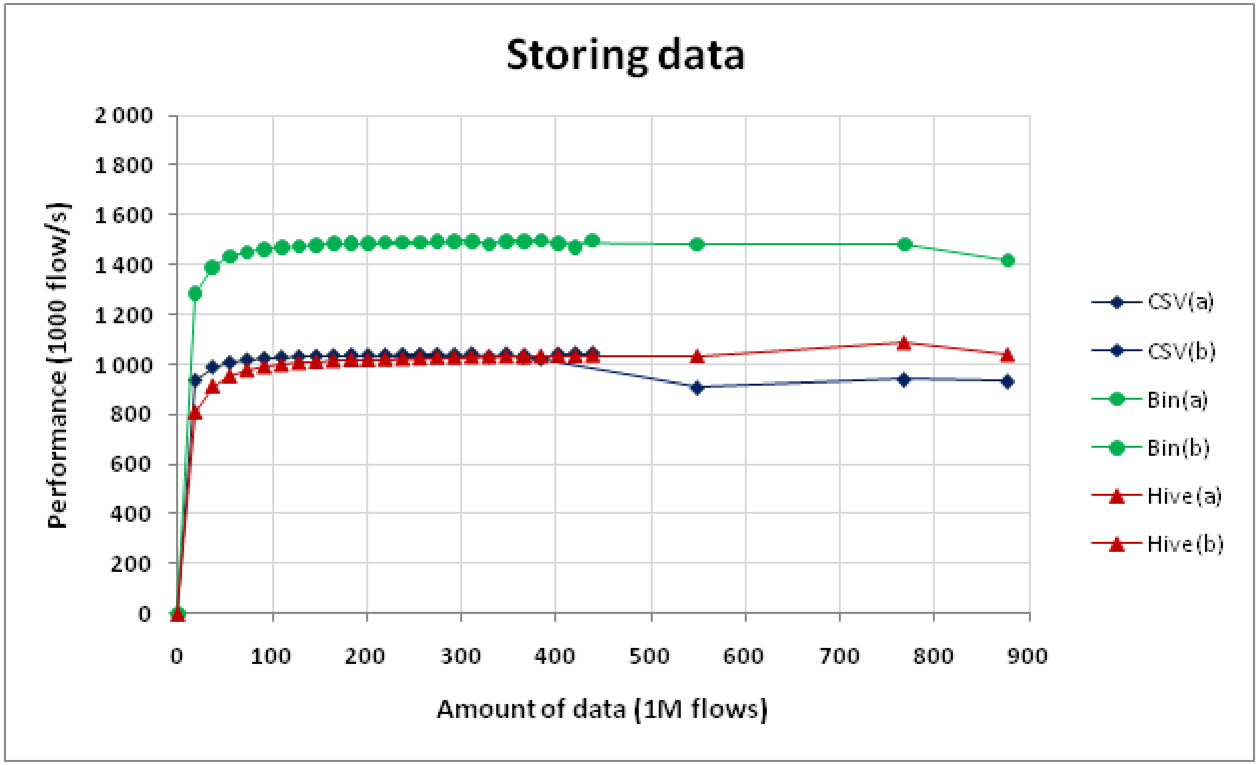
- No significant differences between native Hadoop query implementation and Hive
- Pig is worse and fails arbitrarily
- Hadoop utilize heartbeat messages not only to liveness detection but also to distribute jobs and collect results – this cause long latencies before retrieving first data (around 20s)

Hadoop summary

- Java and long latency means low performance/effectivity per single node
  - Query over 877. mil records – performance lower than 1 mil. records/s per node
  - Query over 8 billion records – performance lower than 2 mil. records/s per node
  - nfdump on single node reaches 4+ mil. records/s per node
- Parallel queries improve single node perf.

Hadoop results

- Data upload (877 mil. toků)



nfdist

- Tool utilizing hdfs as a storage
- NfDump files are upload to hdfs
- Distributed nfdumps retrives data from hdfs and results are merged by nfcat tool

NfDist summary

- Outdated with limitations
- Old nfdump format
- Limited by HDFS block size
- Performance per single node simillar to Hadoop

ElasticSearch

- 9+1 nodes
- Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz
- Each node 8GB RAM

Results ElasticSearch

- Queries over whole data set
- Query 2 – 1x 30s, 3x10s
- Query 3 – 1-2s
- Query 4 – 5200s

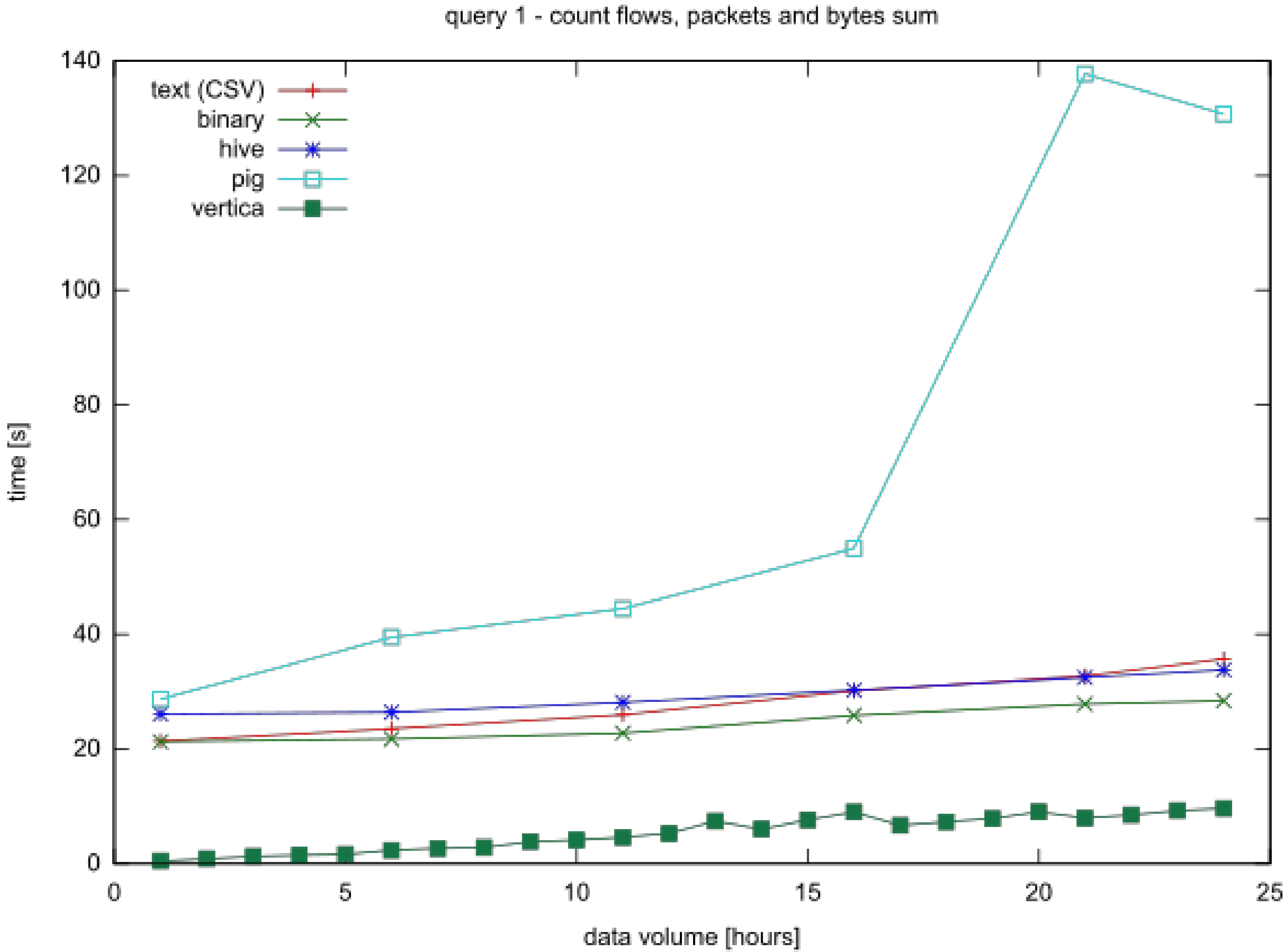
ElasticSearch summary

- Extremely slow upload due to indexing
  - 877 mil. toků in 9hours without replication
  - 46 hours with replication
- Large index
  - Index is 4 times larger than data
- Fast response to filtration queries
  - Around 1s
- Limited by RAM

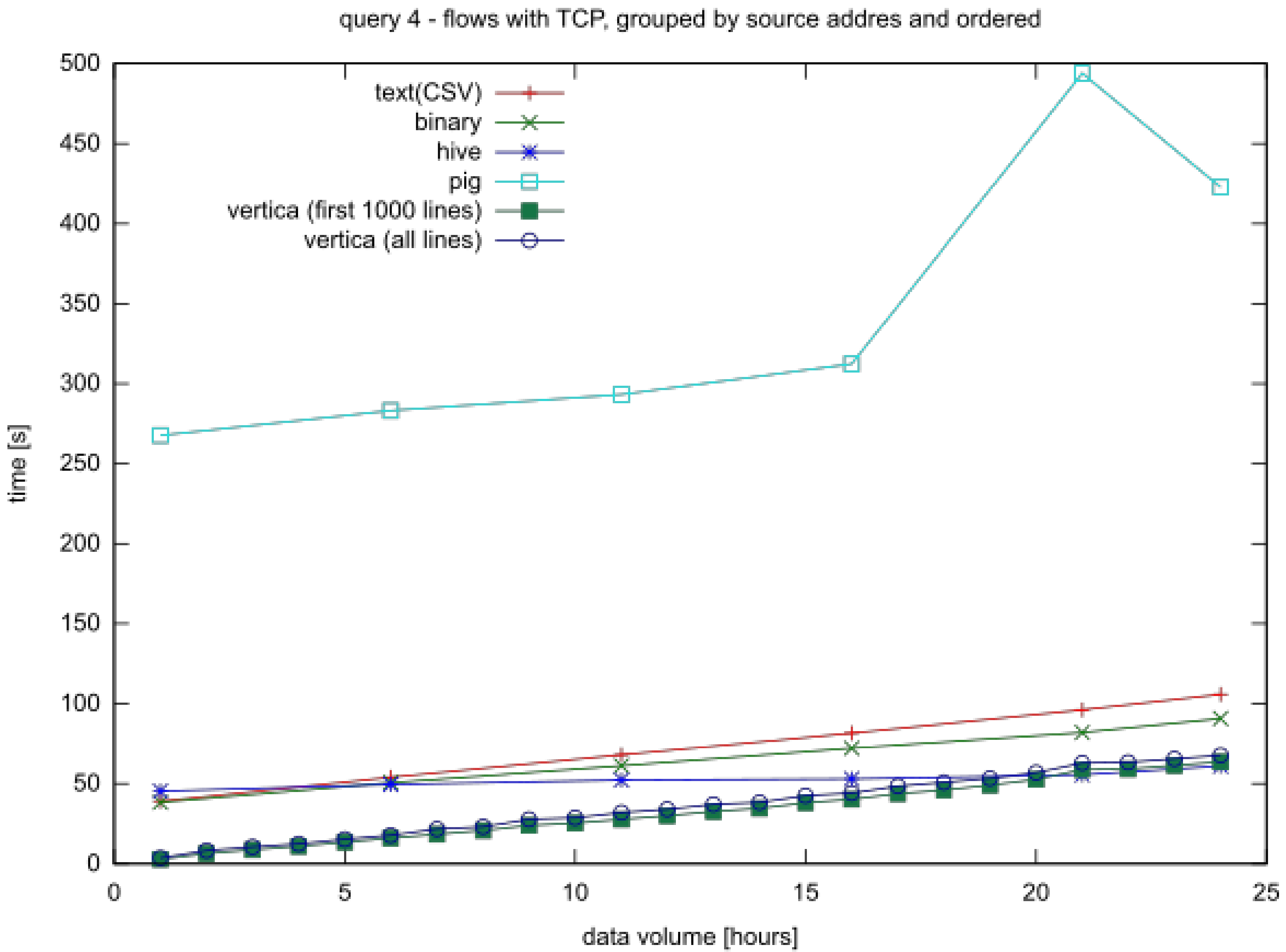
Vertica

- 3 nodes
- CPU: 2 cores z Intel E5-2670 @ 2600 MHz
- Each node 4 GB RAM

Vertica


query 1 - count flows, packets and bytes sum
text (CSV)
binary
hive
pig
vertica
time [s]
0
20
40
60
80
100
120
140
0
5
10
15
20
25
data volume [hours]

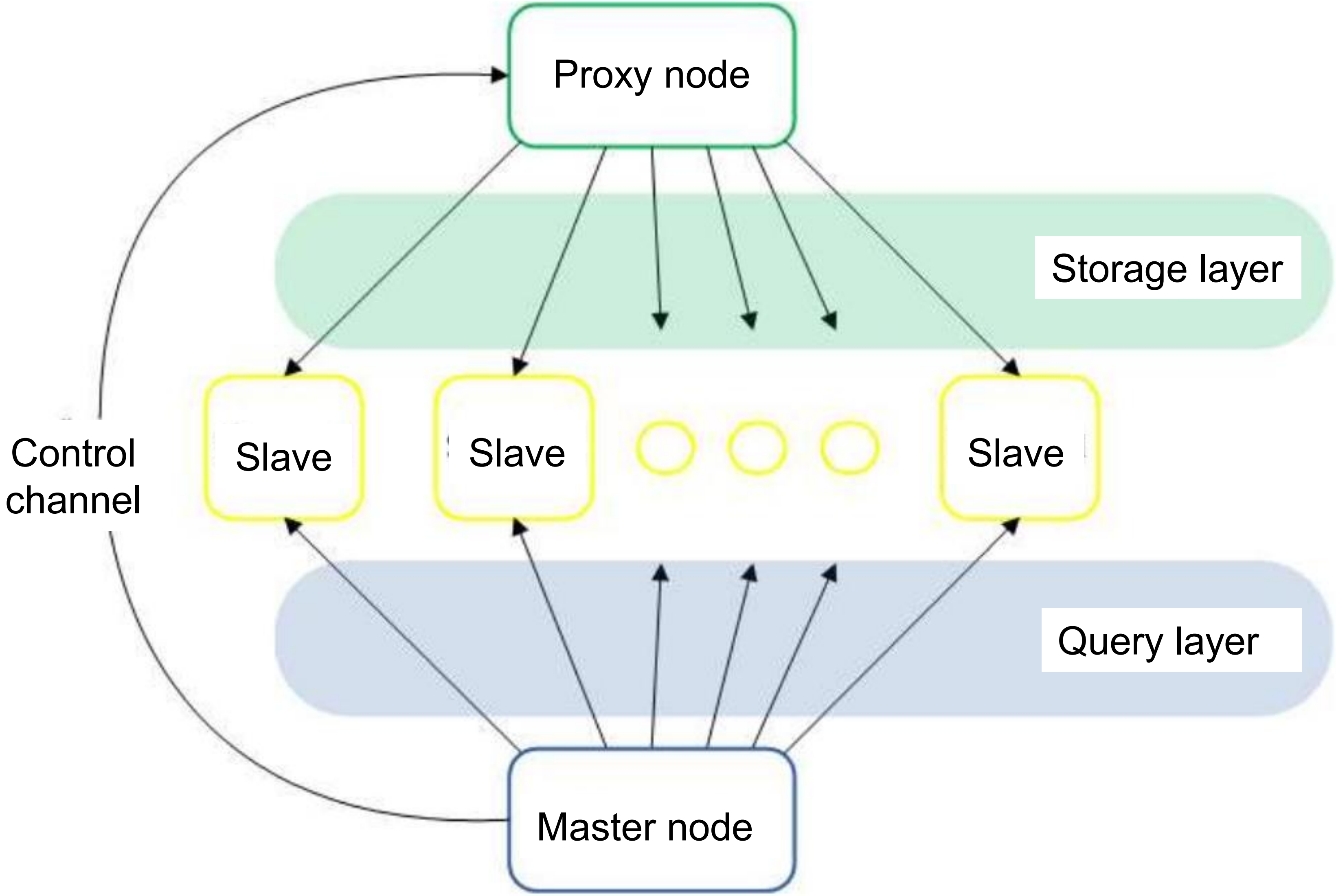
Vertica


query 4 - flows with TCP, grouped by source addres and ordered
text(CSV)
binary
hive
pig
vertica (first 1000 lines)
vertica (all lines)
time [s]
0
50
100
150
200
250
300
350
400
450
500
data volume [hours]
0
5
10
15
20
25

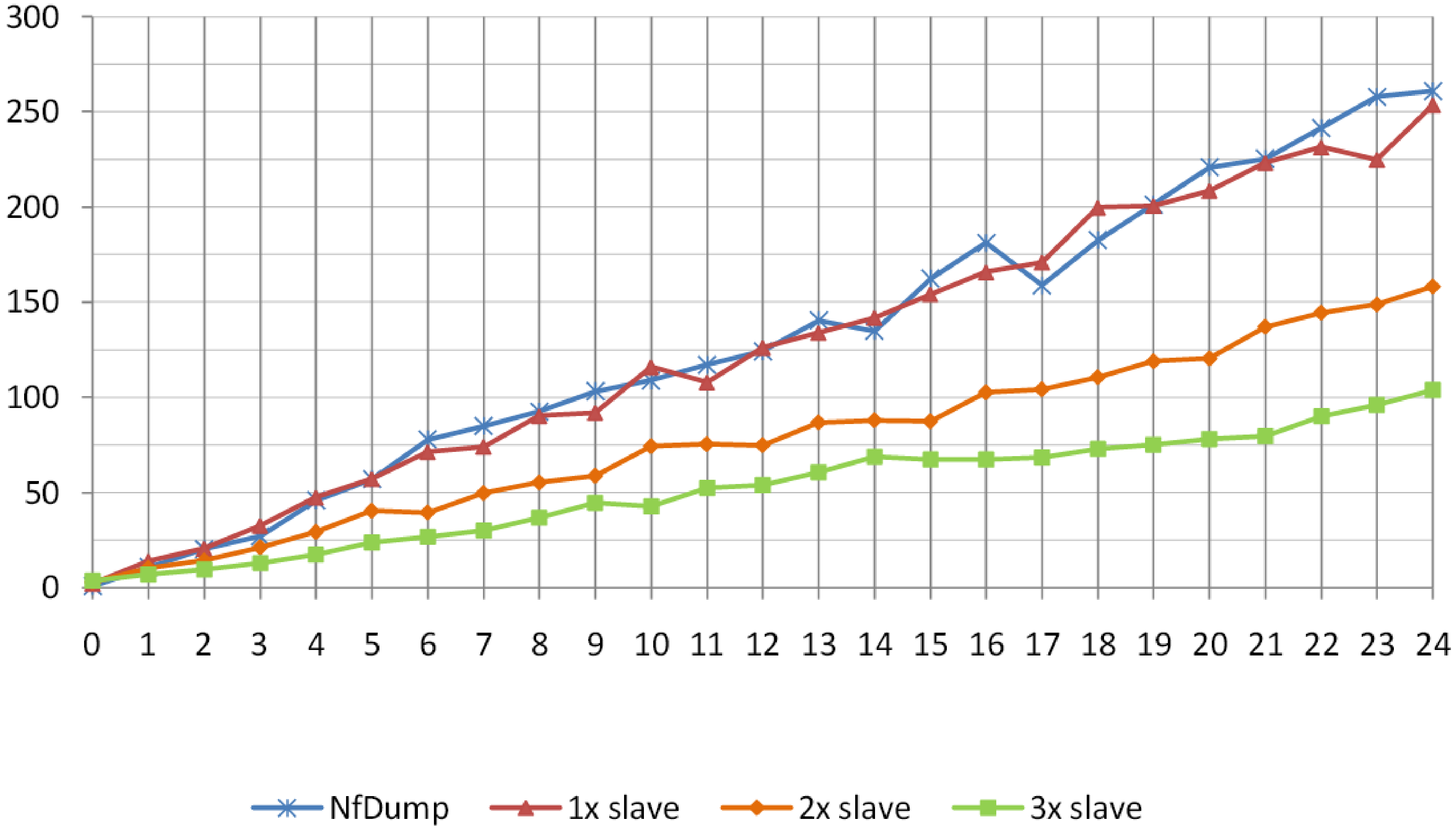
Vertica summary

- Vertica is a column based DB
  - Allows to read only necessary fields from the record
  - Exploit thread paralellism
  - Deals with realibility
  - Publicly available up to 3 nodes

Proprietary implementation



Results


300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
NfDump
1x slave
2x slave
3x slave

Summary

- Proprietary implementation achieves high performance per single node in both tasks storage and queries.
- Does not support high-availability features and multi-thread support so far

Conclusion

- Publicly available platforms exhibit certain limitations
- Flow collector deals with specific data and queries as such proprietary solution will always offer better parameters
- SecurityCloud project implements open source „big data“ flow collector which will be available 2015

Acknowledgement

- This work is partially supported by Technological Agency of Czech Republic

T A

Č R