Jacobs University Bremen

Implementation of a Stream-based IP Flow Record Query Language

Authors: K. Kanev, N. Melnikov & J. Schönwälder
Presentation date: 25.06.2010

Overview

- Concepts
- Implementation
- Profiling results
- Conclusion & next steps

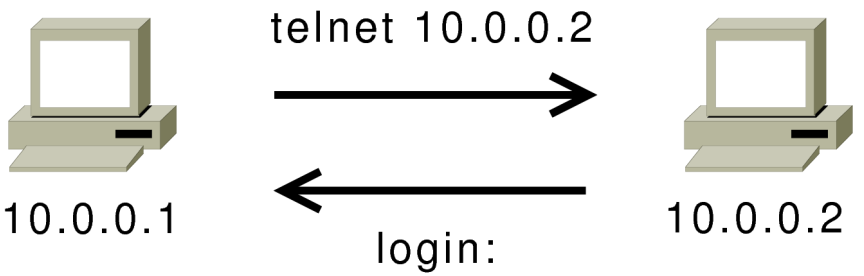
Concepts – network flows

Network flows

Packets with common attributes

Several fields

Vary based on the used tool



Formed by packets or frames that have a common attribute

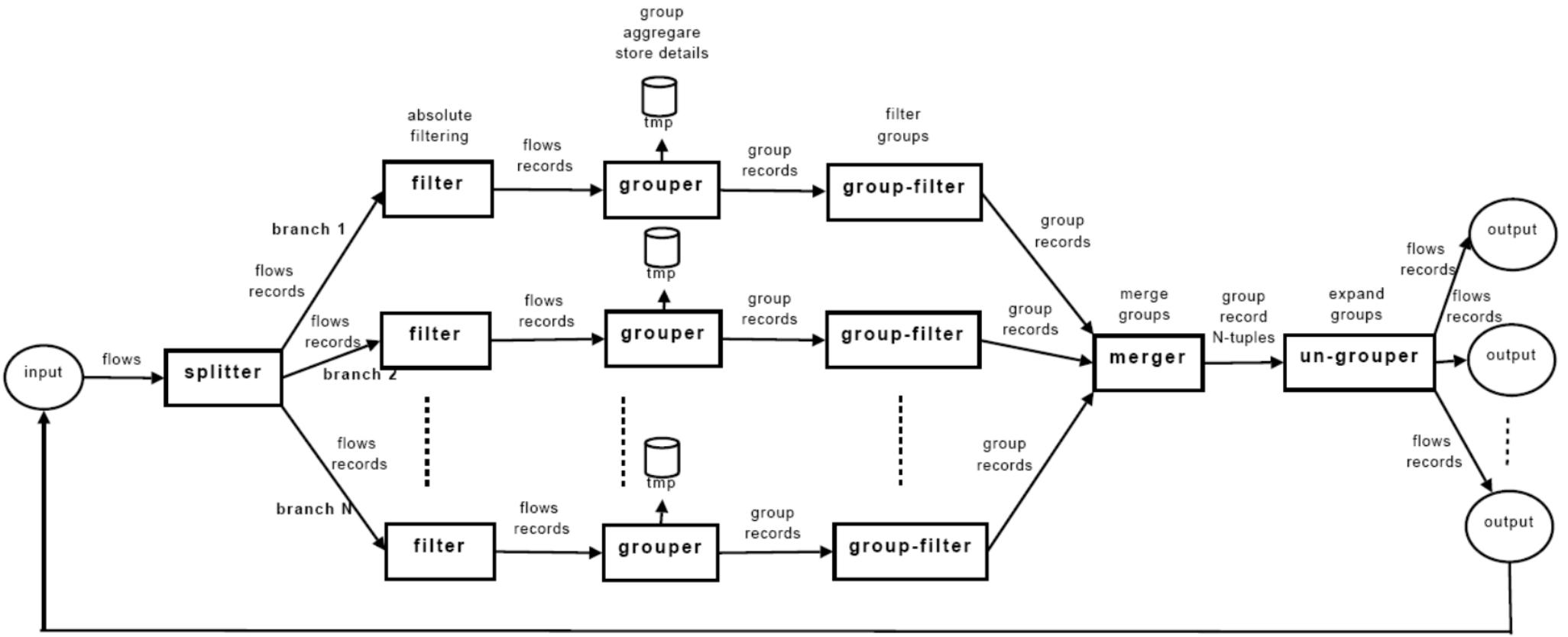
Benefits

Potential aggregation of large amounts of data

(Less) privacy violations

Concepts – language structure

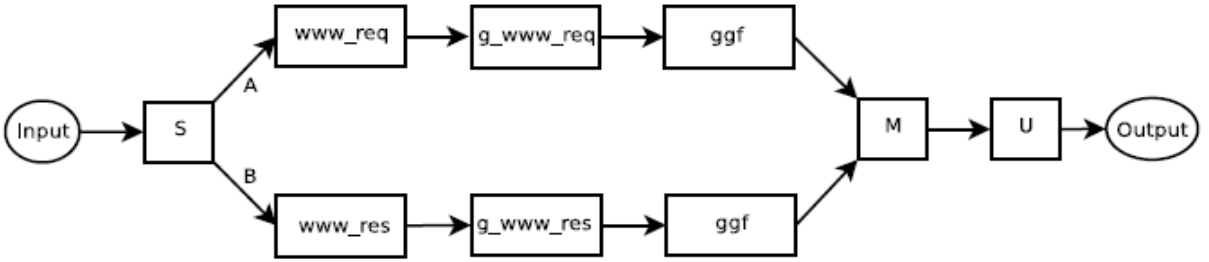
- 6 stages with parallel branches
- Absolute and relative filtering
- Allen's time-interval algebra



Concepts – query structure

- *We defined application signatures using a stream-based flow query language*

```
 1 splitter S {}
 2
 3 filter www_req {
 4    dstport = 80
 5 }
 6
 7 filter www_res {
 8    srcport = 80
 9 }
10
11 grouper g_www_req {
12    module g1 {
13       srcip = srcip
14       dstip = dstip
15       etime < stime delta 1s
16    }
17    aggregate srcip, dstip, sum(bytes) as bytes, count(rec_id) as n,
18              bitOR(tcp_flags) as flags, union(srcport) as srcports
19 }
20
21 grouper g_www_res {
22    module g1 {
23       srcip = srcip
24       dstip = dstip
25       etime < stime delta 1s
26    }
27    aggregate srcip, dstip, sum(bytes) as bytes, count(rec_id) as n,
28              bitOR(tcp_flags) as flags, union(dstport) as dstports
```



Concepts – query structure

- *We defined application signatures using a stream-based flow query language*

```
31 groupfilter ggf {
32    bitAND(flags, 0x13) = 0x13
33 }
34
35 merger M {
36    module m1 {
37       branches B, A
38       A.srcip = B.dstip
39       A.srcports = B.dstports
40       A.bytes < B.bytes
41       B oi A OR B d A
42    }
43    export m1
44 }
45
46 ungrouper U {}
47
48 "./netflow-trace.h5" -> S
49 S branch A -> www_req -> g_www_req -> ggf -> M
50 S branch B -> www_res -> g_www_res -> ggf -> M
51 M->U->"./ungrouped.h5"
```

Implementation

- Python
  - Rapid prototype development
- PyTables
  - Based on Hierarchical Data Format (HDF v.5)
  - Memory-efficient performance at initial testings
- PLY
  - Enough, given the non-complicated query structure
- Flow records
  - Main unit of data exchange through processing pipeline
  - Record class (reading, storing of records) is created dynamically
  - Tool flexibility to process different versions of NetFlow

Implementation – Splitter & Filter

Branch A

Branch B

```
1 splitter S {}
2
3 filter www_req {
4    dstport = 80
5 }
6
7 filter www_res {
8    srcport = 80
9 }
10
```

- Splitter/Filter interchange
- Filter matches every record against all filter rules in the branches

  Adding record mask to a record (record, [True, False]) pairs
- Each stage consists of two modules

Implementation - Rules

Rules are created from conditions

```
filter a {
   prot = protocol("TCP")
   dstport = 80
}
```

```
filter b {
   prot = protocol("TCP")
   bytes > 1024
}
```

```
# protocol("TCP") is evaluated during parsing to its numeric value 6
Rule(((True), (True)), EQ, [Field("prot"), 6])
Rule(((True), (False)), EQ, [Field("srcport"), 80])
Rule(((False), (True)), GT, [Field("bytes"), 1024])
```

Rule(<branch mask>, <operation>, <argument>)

Splitter copies records to branches with True branch mask

Each branch uses a separate thread (avoid record starvation)

Implementation - Grouper

- Forms groups of flow records: using rules and aggregations:

  Aggregated meta-data of a group

```
11 grouper g_www_req {
12    module g1 {
13       srcip = srcip
14       dstip = dstip
15       etime < stime delta 1s
16    }
17    aggregate srcip, dstip, sum(bytes) as bytes, count(rec_id) as n,
18              bitOR(tcp_flags) as flags, union(srcport) as srcports
19 }
```

- Iteration over groups and records is reversed from the concept idea

  Keeps groups list in memory, and checks against them first

  Removes need for random reads from slow permanent storage

- Users can import aggregation operations using `--aggr-import`

Implementation - Group-filter

```
31 groupfilter ggf {
32    bitAND(flags, 0x13) = 0x13
33 }
```

- Group-filter performs absolute filtering
- Not using branch-mask mechanism (i.e., always in the same branch)
- Adds records to the group record time index of the branch
  - Used for the time interval algebra in the merger
  - Returns only groups that satisfy Allen relations

Implementation - Merger

```
35 merger M {
36    module m1 {
37       branches B, A
38       A.srcip = B.dstip
39       A.srcports = B.dstports
40       A.bytes < B.bytes
41       B oi A OR B d A
42    }
43    export m1
44 }
```

- Organized as nested branch loops

  Match a tuple with a rule, pass to the lower level + new group

- At least one Allen operator is required

  Group time index of a branch is scanned

Implementation – Ungrouper & Connection

```
46 ungrouper U {}
47
48 "./netflow-trace.h5" -> S
49 S branch A -> www_req -> g_www_req -> ggf -> M
50 S branch B -> www_res -> g_www_res -> ggf -> M
51 M->U->"./ungrouped.h5"
```

- Ungrouper outputs records that satisfied merger loop
- Input and the output files are fed and obtained from the stage interconnection

Profiling Results

- Multi-threaded profiling of the program execution (number of calls, wall clock, CPU time)

  (('flush','/usr/local/[...]/tables/table.py',2408), (12, 0.02, 0.01))

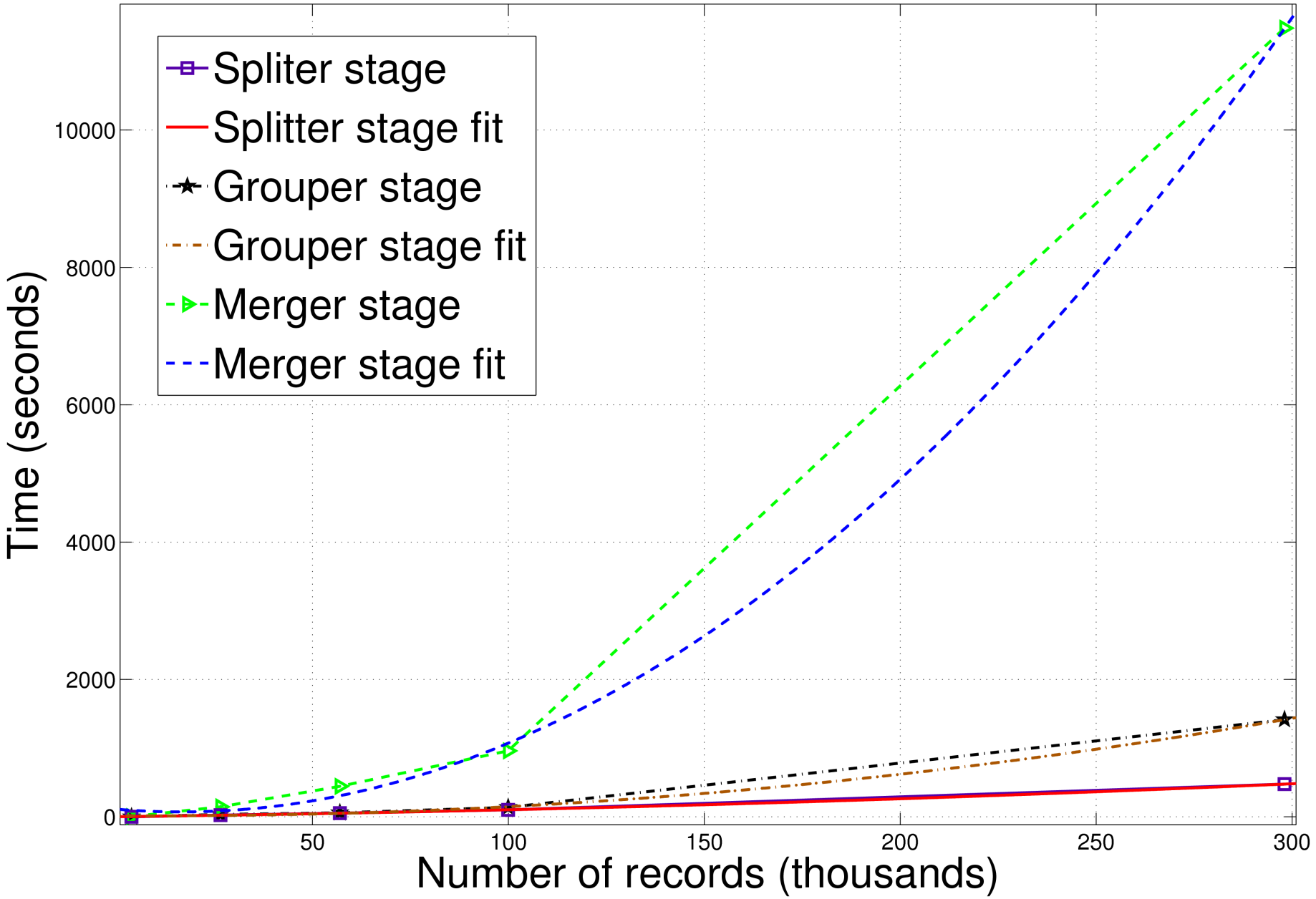
- Queries with variable load on different stages evaluated on traces with:

  ≈ 26K, ≈ 57K, ≈ 100K and ≈ 300K records

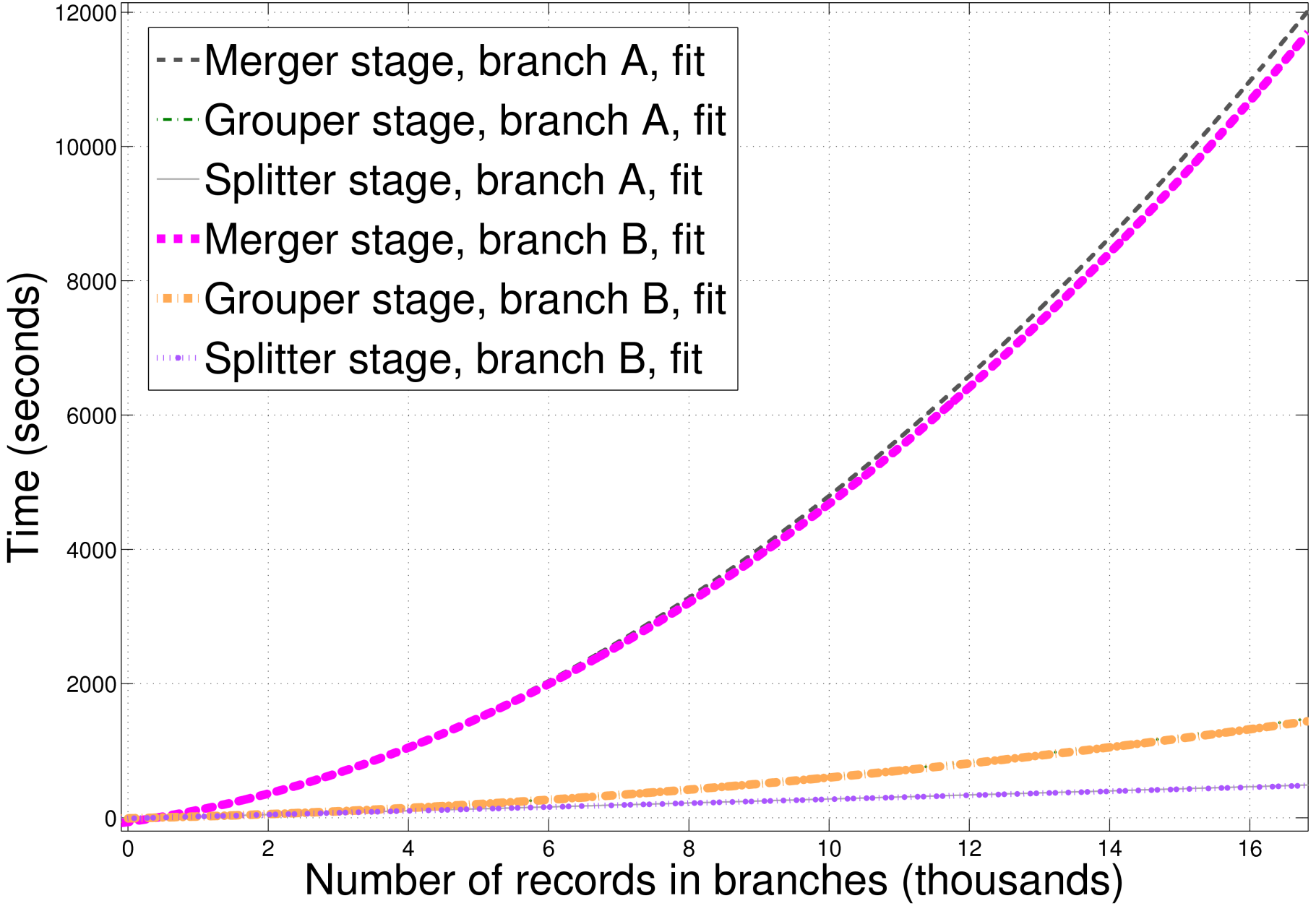
- Heaviest processing load on: filter, grouper and merger stages
- Filter - ***(O(n))*** – n is # of records
- Grouper - ***(O(n^2))*** – n is # of records, worst-case is self-contained record groups
- Merger - ***(O(m^n))*** – n is # of branches, m is # of groups
- Internal functions like: deepcopy(), number of rules at each of the stages

original - (('reset', '/[...]/filter.py', 45), (56992, 255.76, 262.48))
modified - (('reset', '/[...]/filter.py', 64), (56992, 31.72, 31.21))

Profiling Results



Profiling Results



Conclusion and Next Steps

- Exploit concurrency by processing branches in different threads
- Performance

  High-level language

  Number of branches and groups
- Improve performance:

  Subdivision of input flow records

  Map/Reduce framework
- Creation and evaluation of certain queries (next page)

Conclusion and Next Steps

User	Skype	Opera	Amarok	Chrome	Live
U0	⊙	○	⊡	○	○
U1	⊙	○	○	○	○
U2	○	○	○	○	○
U3	⊙	○	⊡	○	○
U4	○	○	○	○	○
U5	⊙	○	⊙	⊙	○
U6	○	○	○	○	○
U7	○	⊙	⊙	○	○
U8	○	○	○	○	○
U9	⊙	⊙	⊙	⊙	○

TABLE I

RESULTS OF APPLICATION SIGNATURE IDENTIFICATIONS

¡Thank you for attention!