F

A complete system integration of stream-based IP flow-record querier

VAIBHAV BAJPAI

Masters Thesis

School of Engineering and Science
Jacobs University Bremen
Bremen, Germany

June 2012

[January 11, 2012 at 7:13]

ABSTRACT

Short summary of the contents in English...

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

IPFIX Internet Protocol Flow Information Export

HDF Hierarchical Data Format

LALR Look-Ahead LR Parser

PLY Python Lex-Yacc

Part I

INTRODUCTION

You can put some informational part preamble text here

1 TRAFFIC MEASUREMENT APPROACHES

2 FLOW EXPORT PROTOCOLS

3 LANGUAGES AND TOOLS

4

LEGAL CONSIDERATION

Part II

STATE OF THE ART

You can put some informational part preamble text here

5 FLOWY

Flowy [1][2] is the first prototype implementation of a stream-based flow record query language [3][4][5]. The query language allows to describe patterns in flow-records in a declarative and orthogonal fashion, making it easy to read and flexible enough to describe complex relationships among a given set of flows.

5.1 PROCESSING PIPELINE

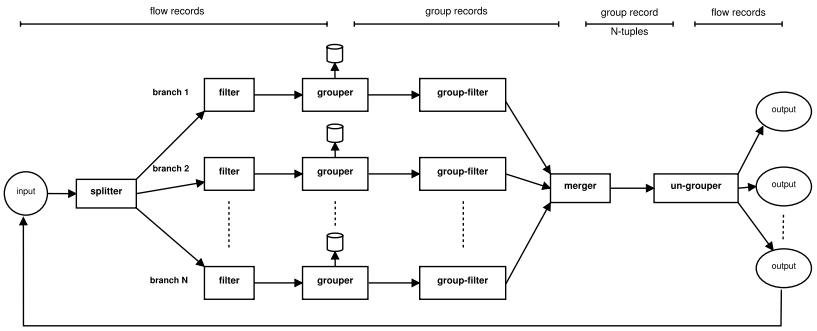

Figure 1: Flowy: Processing Pipeline [4]

The pipeline consists of a number of independent processing elements that are connected to one another using UNIX-based pipes. Each element receives the content from the previous pipe, performs an operation and pushes it to the next element in the pipeline. Figure 1 shows an overview of the processing pipeline. The flow record attributes used in this pipeline exactly correlate with the attributes defines in the Internet Protocol Flow Information Export (IPFIX) Information Model specified in RFC 5102 [6]. A complete description on the semantics of each element in the pipeline can be found in [3]

5.1.1 *Splitter*

The splitter takes the flow-records data as input in the `flow-tools` compatible format. It is responsible to duplicate the input data out to several branches without any processing whatsoever. This allows each of the branches to have an identical copy of the flow data to process it independently.

5.1.2 *Filter*

The filter performs *absolute* filtering on the input flow-records data. The flow-records that pass the filtering criterion are forwarded to the grouper, the rest of the flow-records are dropped. The filter compares separate fields of a flow-record against either a constant value or a value on a different field of the *same* flow-record. The filter cannot *relatively* compare two different incoming flow-records

5.1.3 *Grouper*

The grouper performs aggregation of the input flow-records data. It consists of a number of rule modules that correspond to a specific subgroup. A flow-record in order to be a part of the group should be a part of at-least one subgroup. A flow-record can be a part of multiple subgroups within a group. In addition a flow-record cannot be part of multiple groups. The grouping rules can be either absolute or relative. The newly formed groups which are passed on to the group filter can also contain meta-information about the flow-records contained within the group using the aggregate clause defined as part of the grouper query.

5.1.4 *Group-Filter*

The group-filter performs *absolute* filtering on the input group-records data. The group-records that pass the filtering criterion are forwarded to the merger, the rest of the group-records are dropped. The group-filter compares separate fields (or aggregated fields) of a flow-record against either a constant value or a value on a different field of the *same* flow-record. The group-filter cannot *relatively* compare two different incoming group-records

5.1.5 *Merger*

The merger performs relative filtering on the N-tuples of groups formed from the N stream of groups passed on from the group-filter as input. The merger rule module consists of a number of a submodules, where the output of the merger is the set difference of the output of the first submodule with the union of the output of the rest of the submodules. The relative filtering on the groups are applied to express timing and concurrency constraints using Allen interval algebra [7]

5.1.6 *Ungrouper*

The ungrouper unwraps the tuples of group-records into individual flow-records, ordered by their timestamps. The duplicate flow-records appearing from several group-records are eliminated and are sent as output only once.

5.2 PYTHON FRAMEWORK

The Python framework is subdivided into two main modules: the validator module and the execution module. The validator module is used for syntax checking and interconnecting of all the stages of the processing pipeline and the execution module is used to perform actions at each stage of the runtime operation.

5.2.1 *PyTables and PLY*

Flowy uses PyTables [8] to store the flow-records. PyTables is built on top of the Hierarchical Data Format (HDF) library and can exploit the hierarchical nature of the flow-records to efficiently handle large amounts of flow data. In addition, Flowy uses Python Lex-Yacc (PLY) for generating a Look-Ahead LR Parser (LALR) parser and providing extensive input validation, error reporting and validation on the execution module.

5.2.2 *Records*

Flow-records are the principal unit of data exchange throughout Flowy's processing pipeline. The prototype implementation allows the `Record` class to be dynamically generated allowing future implementations to easily plug in support for IPFIX or even newer versions of NetFlow [9] exports. The `Record` handles the reading of the flow-records and saving them in PyTables for future processing.

5.2.3 *Filters and Rules*

5.2.4 *Branches and Branch Masks*

6 FLOWY IMPROVEMENTS USING MAP/REDUCE

7

FLOWY 2.0

8

FLOWY: APPLICATIONS

Part III

MOTIVATION

Part IV

WORK PLAN

You can put some informational part preamble text here

9 DESIGN

10 IMPLEMENTATION

11

PERFORMANCE EVALUATION

12

CONCLUSION

Part V

IMPLEMENTATION AND EVALUATION

You can put some informational part preamble text here

13

DESIGN

14

IMPLEMENTATION

15

PERFORMANCE EVALUATION

16 FUTURE WORK

17

CONCLUSION

Part VI

APPENDIX

APPENDIX

Put your appendix here.

BIBLIOGRAPHY

[1] Vladislav Marinov and Jürgen Schönwälder. Design of a Stream-Based IP Flow Record Query Language. In *Proceedings of the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Integrated Management of Systems, Services, Processes and People in IT*, DSOM '09, pages 15–28, Berlin, Heidelberg, 2009. Springer-Verlag.

[2] Kaloyan Kanev. Flowy - Network Flow Analysis Application. Master's thesis, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany, 2009.

[3] Kaloyan Kanev, Nikolay Melnikov, and Jürgen Schönwälder. Implementation of a stream-based IP flow record query language. In *Proceedings of the Mechanisms for autonomous management of networks and services, and 4th international conference on Autonomous infrastructure, management and security*, AIMS'10, pages 147–158, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Vladislav Marinov. Design of an IP Flow Record Query Language. Master's thesis, Jacobs University Bremen, Campus Ring 1, 28759 Bremen, Germany, 2009.

[5] Vladislav Marinov and Jürgen Schönwälder. Design of an IP Flow Record Query Language. In *Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security: Resilient Networks and Services*, AIMS '08, pages 205–210, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer. Information Model for IP Flow Information Export. RFC 5102 (Proposed Standard), January 2008. Updated by RFC 6313.

[7] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, November 1983.

[8] Francesc Alted and Mercedes FernÃ¡ndez-Alonso. PyTables: Processing And Analyzing Extremely Large Amounts Of Data In Python. 2003.

[9] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational), October 2004.

DECLARATION

Put your declaration here.

Bremen, Germany, June 2012

Vaibhav Bajpai