Multidimensional index structures for NetFlow records processing

Alexander Sherikov, Dr. Yuri A. Bogoiavlensky

Department of Computer Science,
University of Petrozavodsk

Our work is devoted to application of multidimensional index structures for NetFlow records processing.

1 NetFlow

NetFlow[1] is a technology, developed by Cisco Systems Inc., for monitoring network traffic. It presents network traffic as flows.

A flow is identified as unidirectional stream of packets between given source and destination - both defined by network-layer IP address and transport-layer source and destination port number.

A flow record provides information (a set of attributes) about an observed IP flow (see fig. 1 for example).

Number of attributes depends on NetFlow version (average is 15 - 20). NetFlow version 9 introduces flexibility in grouping flow attributes and currently predefines 89[2] types of attributes.

NetFlow records analysis includes gathtering statistics and finding interesting network activity by performing queries to set of flows.

Usually only small subset of attributes is affected by the queries. Often these queries contains interval searching conditions.

It is also important to note nonuniform distribution of records by value of the attributes.

2 Motivation

Huge disk space is crucial for processing flows, because typically ones need to process several millions of flows at a time.

	Description	Length (bytes)
1	Source IP address	**4**
2	Destination IP address	**4**
3	Next hop router‘s IP address	**4**
4	Ingres interface SNMP ifindex	**2**
5	Egress interface SNMP ifindex	**2**
6	Packets in the flow	**4**
7	Octets (bytes) in the flow	**4**
8	SysUptime at start of the flow	**4**
9	SysUptime at end of the flow	**4**
10	Layer 4 source port number or equivalent	**2**
11	Layer 4 destination port number or equivalent	**2**
12	Cumulative OR of TCP flags	**1**
13	Layer 4 protocol	**1**
14	IP type-of-service byte	**1**
15	Autonomous system number of the source	**2**
16	Autonomous system number of the destination	**2**
17	Source address prefix mask bits	**1**
18	Destination address prefix mask bits	**1**

Figure 1: NetFlow version 5 record attributes

Typically captured flows are stored on disk sequentially, in order of their observation.

It is quite obvious that queries can be speeded up by indexing flow records in some way.

3 Data structures overview

We are interested in data structures that use external (slow) memory. In this case query processing time significantly depends on accessing rate to external memory.

Most widespread data structures are one-dimensional. These structures use only one key attribute.

One-dimensional data structures still can be used for multi-dimensional queries, but they do not fit this task as well as multi-dimensional data structures.

Mulit-dimensional data structures can be divided into two groups:

- First group of structures use mapping of N-dimensional space to one dimensional space;
- Second group of structures use distribution of points (or their approximations) between multidimensional domains.

Most popular multidimensional data structures suffer from dimensionality growth, which leads to poor performance. This phenomenon is called "curse of dimensionality".

For the majority of multidimensional data structures processing data with 15 - 20 number of dimensions, leads to performance worse than using sequential record processing.

Following methods[3] can help to cope with "curse of dimensionality":

- data approximation, which helps to increase density of data;
- query approximation allows to fetch data with some sort of error in result;
- perform queries with group of low-dimensional data structures, which show good results;
- hybrid data structures.

4 Requirements for data structures for indexing flow records

- processing huge data sets;
- indexing data without performing any kind of prior analysis;
- effective processing of range queries;
- acquire exact result without an error;
- effective processing queries with conditions that cover subset of attributes;

- high performance search in the event of nonuniform distribution of points;
- sufficient indexing speed for 15 - 25 dimensionality;

5 Examined data structures

- One-dimensional data structures:
  - inverted B-tree [4]
  - hashing [5]
- Multi-dimensional data structures:
  - Hashing:
    - grid-file [5]
    - MOLPHE [5] (multidimensional linear hashing)
  - Mapping:
    - Pyramid technique[6]
    - UB-tree [7]
  - Sequential processing with approximation:
    - VA-file[8]
    - VA+file[9]
  - Multi-dimensional bounding boxes:
    - R-tree[10]
    - R*-tree[11]
    - PR-tree[12]
    - SR-tree[13]
  - Hybrid structures:
    - X-tree[14]
  - Multi-dimensional bounding boxes with approximation:
    - IQ-tree[15]
    - A-tree.[16]

  - Partition data space by hyperplanes on each tree level:
    * KDB-tree [3, 5]
  - Group of low-dimensional structures:
    * high-dimensional KDB-tree[17] (KDB_{HD}-tree)
  - OLAP-specific structures:
    * DWARF [18]
    * QC-tree [19]

6 Examined data indexing software

- BerkeleyDB (B-tree, hash)[20]
- libgist (GiST)[21]
- GiMP (Generalized multidimensional data mapping and query processing)[22]
- HnSRTree (the SR-tree library)[23]
- eXtremeDB (B-tree, R-tree)[24]
- PostgreSQL (GiST, GiN)[25]
- Spatial Index Library (R*-tree, MVR-tree (PPR-tree), TPR-tree)[26]

7 Conclusion

None of examined data indexing software tools completely suites our needs.

Among all examined data structures A-tree seems to be the most suitable data structure for indexing NetFlow records, because it provide high performance in case of high-dimesional data.

8 Current work

Currently we are working on A-tree implementation which is adapted to process NetFlow data.

We have planned number of considerable modifications that can improve A-tree performance. Further tests will show their influence on performance.

References

[1] *Claise, B.* Cisco systems netflow services export version 9. "— RFC 3954 (Informational). "— 2004. "— oct.

[2] Url: http://www.cisco.com/en/us/tech/tk648/tk362/technologies_white_paper09186a00800a3db9.shtml.

[3] *Moenne-Loccoz, N.* High-dimensional access methods for efficient similarity queries: Tech. rep. / N. Moenne-Loccoz: Computer Vision Group Computing Science Center, University of Geneva 24 rue du General Dufour, CH - 1211 Geneva 4, Switzerland, 2005.

[4] *Knuth, D. E.* Art of Computer Programming, Volume 3: Sorting and Searching / D. E. Knuth. "— 2 edition. "— Williams, 2001. "— Vol. 3. "— Pp. 597–618.

[5] *Gaede, V.* Multidimensional access methods / V. Gaede, O. Günther // *ACM Computing Surveys.* "— 1998. "— Vol. 30, no. 2. "— Pp. 170–231.

[6] *Berchtold, S.* The pyramid-technique: Towards breaking the curse of dimensionality / S. Berchtold, C. Böhm, H.-P. Kriegel // SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA / Ed. by L. M. Haas, A. Tiwary. "— ACM Press, 1998. "— Pp. 142–153.

[7] *Bayer, R.* The universal b-tree for multidimensional indexing: general concepts / R. Bayer // WWCA '97: Proceedings of the

International Conference on Worldwide Computing and Its Applications. "— London, UK: Springer-Verlag, 1997. "— Pp. 198–209.

[8] *Weber, R.* A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces / R. Weber, H.-J. Schek, S. Blott // VLDB'98, Proceedings of 24rd International Conference on Very Large Data Bases, August 24-27, 1998, New York City, New York, USA / Ed. by A. Gupta, O. Shmueli, J. Widom. "— Morgan Kaufmann, 1998. "— Pp. 194–205.

[9] High dimensional nearest neighbor searching / H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, A. E. Abbadi // *Inf. Syst.* "— 2006. "— Vol. 31, no. 6. "— Pp. 512–540.

[10] *Guttman, A.* R-trees: A dynamic index structure for spatial searching. / A. Guttman // SIGMOD Conference. "— 1984. "— Pp. 47–57.

[11] The r*-tree: An efficient and robust access method for points and rectangles. / N. Beckmann, H.-P. Kriegel, R. Schneider, B. Seeger // SIGMOD Conference. "— 1990. "— Pp. 322–331.

[12] The priority r-tree: A practically efficient and worst-case optimal r-tree. / L. Arge, M. de Berg, H. J. Haverkort, K. Yi // SIGMOD Conference. "— 2004. "— Pp. 347–358.

[13] *Katayama, N.* The sr-tree: An index structure for high-dimensional nearest neighbor queries / N. Katayama, S. Satoh // SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA / Ed. by J. Peckham. "— ACM Press, 1997. "— Pp. 369–380.

[14] *Berchtold, S.* The X-tree: An index structure for high-dimensional data / S. Berchtold, D. A. Keim, H.-P. Kriegel // Proceedings of the 22nd International Conference on Very Large Databases / Ed. by T. M. Vijayaraman, A. P. Buchmann, C. Mohan, N. L. Sarda. "— San Francisco, U.S.A.: Morgan Kaufmann Publishers, 1996. "— Pp. 28–39.

[15] Independent quantization: An index compression technique for high-dimensional data spaces. / S. Berchtold, C. Böhm, H. V. Jagadish et al. // ICDE. "— 2000. "— Pp. 577–588.

[16] The a-tree: An index structure for high-dimensional spaces using relative approximation / Y. Sakurai, M. Yoshikawa, S. Uemura, H. Kojima // VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt / Ed. by A. E. Abbadi, M. L. Brodie, S. Chakravarthy et al. "— Morgan Kaufmann, 2000. "— Pp. 516–526.

[17] *Orlandic, R.* A retrieval technique for high-dimensional data and partially specified queries / R. Orlandic, B. Yu // *Data Knowl. Eng.* "— 2002. "— Vol. 42, no. 1. "— Pp. 1–21.

[18] Dwarf: Shrinking the petacube / Y. Sismanis, N. Roussopoulos, A. Deligianannakis, Y. Kotidis // SIGMOD Conference. "— 2002.

[19] *Zhao, Y.* "— Qoutient Cube and QC-Tree: Efficient Summarizations for Semantic OLAP. "— Master's thesis, The University of British Columbia, 2003.

[20] Url: http://www.oracle.com/database/berkeley-db/index.html.

[21] Url: http://gist.cs.berkeley.edu/.

[22] Url: http://www.cs.mu.oz.au/~rui/.

[23] Url: http://research.nii.ac.jp/~katayama/homepage/research/srtree/.

[24] Url: http://www.mcobject.com/.

[25] Url: http://www.sai.msu.su/~megera/postgres/gist/.

[26] Url: http://research.att.com/~marioh/spatialindex/index.html.