LBNL-57982

FastBit: An Efficient Indexing Technology For Accelerating Data-Intensive Science

Kesheng Wu

Berkeley Lab, Berkeley, CA, 94720

E-mail: `KWu@lbl.gov`

Abstract. FastBit is a software tool for searching large read-only datasets. It organizes user data in a column-oriented structure which is efficient for on-line analytical processing (OLAP), and utilizes compressed bitmap indices to further speed up query processing. Analyses have proven the compressed bitmap index used in FastBit to be theoretically optimal for one-dimensional queries. Compared with other optimal indexing methods, bitmap indices are superior because they can be efficiently combined to answer multi-dimensional queries whereas other optimal methods can not. In this paper, we first describe the searching capability of FastBit, then briefly highlight two applications that make extensive use of FastBit, namely Grid Collector and DEX.

1. Introduction

It is a significant challenge to search for key insight in the huge amount of data being produced by many data-intensive science application. For example, a high-energy physics experiment called STAR is producing nearly a petabyte of data a year and has accumulated many millions of files in last five years of operation. One of the core missions of the STAR experiment is to verify the existence of a new state of matter called the Quark Gluon Plasma (QGP) [1]. An effective strategy for this task is to find the high-energy collisions that contain signatures unique to QGP, such as a phenomenon called jet quenching. Among the hundreds of millions of collision events captured, a very small fraction of them, maybe only a few hundreds, contain clear signatures of jet quenching. Efficiently identifying these events and transferring the relevant data files to analysis programs are a great challenge. Many data-intensive science applications are facing similar challenges in searching their data.

In the past few years, we have been working on a set of strategies to address this type of searching problem. Usually, the data to be searched are read-only[1]. Our approach takes advantage of this fact. Since most database systems (DBMS) are built for frequently modified data, FastBit can perform searching operations significantly faster than those DBMS.

Conceptually, most data can be thought of as tables, where each row of the table represents an object or a record, and each column represents one attribute of the record. To accommodate frequent changes in records, a typical DBMS stores each record together on disk. This allows easy update of the records, but in many operations the DBMS effectively reads all attributes from disk in order to access a few that are relevant for a particular query. FastBit stores each attribute together on disk, which allows one to easily access the relevant columns without

[1] Another name that may characterize the data more accurately is Write-Once Read-Many (WORM).

involving any other columns. Even though an update may take longer to execute, but because the update usually come in the form of bulk append operations, the new records can usually be integrated into existing tables efficiently. In the database theory, separating out the values of a particular attribute is referred to as a projection. For this reason, using column-wise organized data to answer user queries is also known as the projection index [2].

Each column of a table is also referred to as a dimension of the data. Many scientific datasets have tens or hundreds of dimensions; they are called high-dimensional data. User queries usually involve conditions on several attributes; they are known as multi-dimensional queries. It is well-known that for multi-dimensional queries on high-dimensional data the projection index performs better than most of well-known indexing schemes including B-Tree. Since FastBit uses column-wise organization for user data, without any additional indices it is using the projection index, which is already very efficient. Our indexing technology further speeds up the searching operations. We have analyzed our bitmap index and showed it to be optimal for one-dimensional queries [7, 8]. Some of the best indexing methods including B+-tree and B*-tree have this optimality property as well. However, bitmap indices are superior because they can be efficiently combined to answer multi-dimensional queries.

2. FastBit Bitmap indices

In collaboration with Ekow Otoo and Arie Shoshani.

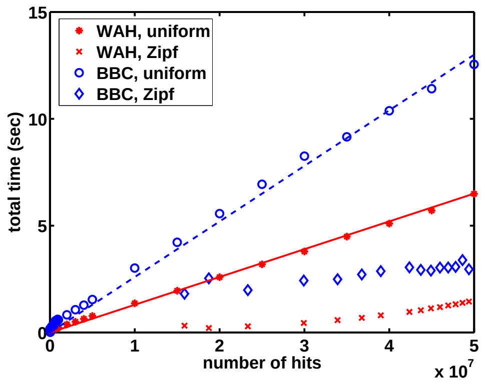
FastBit implements a number of different bitmap indices. In this section, we briefly review one of the simplest bitmap indices. Figure 1(a) illustrates a bitmap index for an integer attribute called **X**. The values of **X** can be 0, 1, 2, or 3. For each possible value, a sequence of 0s and 1s are used to represent whether the value of **X** at row i is the specified value. Each column of bits is called a bitmap. These bitmaps together with their associated information are called a bitmap index. Bitmap indices in various forms have been used before [3, 4]. One shortcoming of the basic bitmap index shown in Figure 1(a) is that the index size is proportional to the number of distinct values (also known attribute cardinality) of the attribute indexed. For low cardinality attributes such as **X**, the index size is small. However, scientific datasets frequently have attributes with cardinalities in the thousands or even millions. For these attributes, the basic bitmap index would be too large to be useful. We overcome this size problem with an efficient compression called the Word-Aligned Hybrid (WAH) code.

Using general purpose compression schemes such as LZ77 can effectively compress the bitmaps. However, the operations on bitmaps compressed this way are much slower than the same operations on the uncompressed ones. This makes the query response time longer with compression than without compression. To reduce the query response time, specialized compression schemes have been developed, the most well-known of which is the Byte-aligned Bitmap Code (BBC) [5, 6]. BBC compresses very well; BBC compressed indices are not much larger than those compressed with LZ77. However, query processing time using BBC compressed indices can be much less than that using LZ77 compressed indices. In the same spirit of trading some space for faster operations, the Word-Aligned Hybrid (WAH) compression uses moderately more space than BBC but is much faster in answering queries [7, 8]. Both analysis and timing measurements confirm that the query response time using WAH compressed indices grows linearly as the number of query results (i.e., hits) increases as shown in Figure 1b). This is theoretically optimal. In addition, in all tests, WAH compressed indices are faster in answering the same query as BBC compressed indices.

Figure 2 shows the query processing time of multi-dimensional queries. It also includes two different implementations of BBC, one is our own and the other is from a popular commercial DBMS, labeled "BBC" and "DBMS" respectively. The horizontal axis is the query box sizes which are the fraction of attributes' domain selected by the query conditions. For each query box size, 1000 queries were used to compute the average time. On average, the WAH compressed

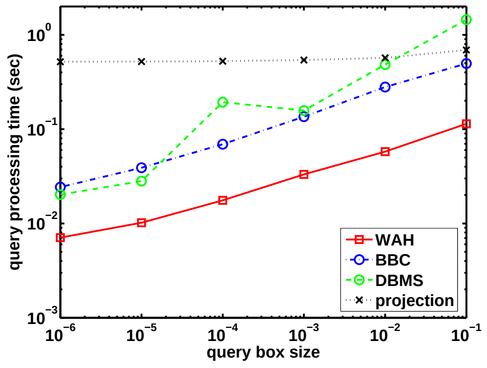
row ID	X	bitmap index b_0 =0	b_1 =1	b_2 =2	b_3 =3
1	0	1	0	0	0
2	1	0	1	0	0
3	3	0	0	0	1
4	2	0	0	1	0
5	3	0	0	0	1
6	3	0	0	0	1
7	1	0	1	0	0
8	3	0	0	0	1

(a) A sample bitmap index

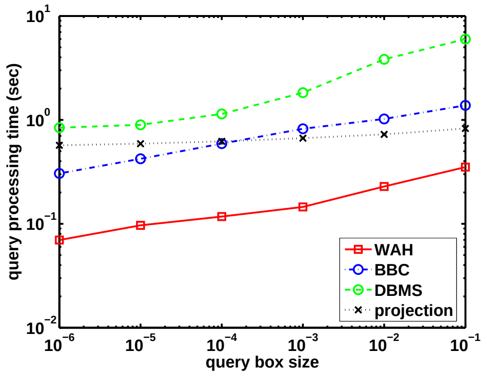


(b) Time to process one-dimensional query

Figure 1. The WAH compressed bitmap index is optimal. The timing plot on the right shows that in the worst case (indices for uniform random attributes) the query processing time used by the WAH compressed indices is a linear function of the number of hits. For more realistic data distributions, such as the Zipf distribution, the query processing time is significantly less than in the worst case.



(a) 2-dimensional queries



(b) 5-dimensional queries

Figure 2. The average query processing time of random range queries on the 12 most queried attributes of a subset of STAR data.

index is about 5 times as fast as our own implementation of the BBC compressed index and about 12 times as fast as the commercial implementation.

3. Grid Collector

In collaboration with Jerome Lauret, Wei-Ming Zhang, Alexander Sim, Junmin Gu, Arie Shoshani, Arthur Poskanzer, and Victor Perevoztchikov.

Grid Collector is a software system that extends the STAR analysis framework to provide event-based and location-transparent data access. A high-level sketch is shown in Figure 3. Without Grid Collector, all data files needed by an analysis job have to be on disk before it can start. In addition, all collision events stored in a file has to be read into memory so that the user code can decide which ones to further analyze. In most cases, the total execution time is dominated by the time required to read the events. Grid Collector provides the end user with a convenient way to specify events to be analyzed and at the same time avoids reading the

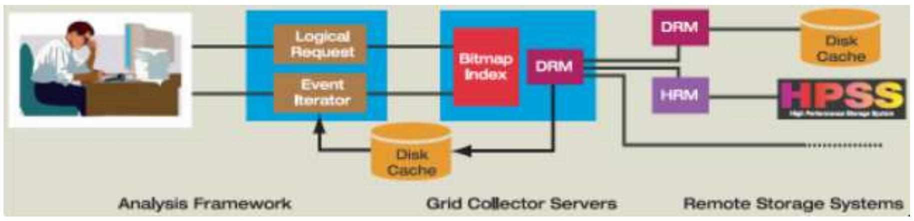


Figure 3. A high-level illustration of the Grid Collector.

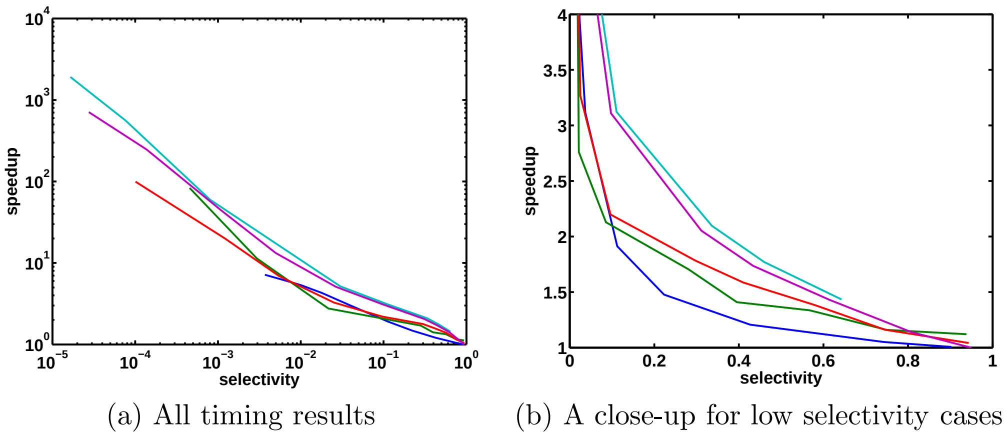


Figure 4. The speedup of using Grid Collector measured on five different subsets of STAR data.

unwanted events. Figure 4 shows the speedup values of using Grid Collector on five different subsets of the STAR data. When an analysis is highly selective, say selecting 1 out of 1000 events, i.e., selectivity of 0.001, using Grid Collector can speed up analysis jobs by a factor of 10 to 100. It is common for many analysis jobs to select about 10% of the events from a subset of data. In this case, the speedup is between 2 and 3 [9, 10].

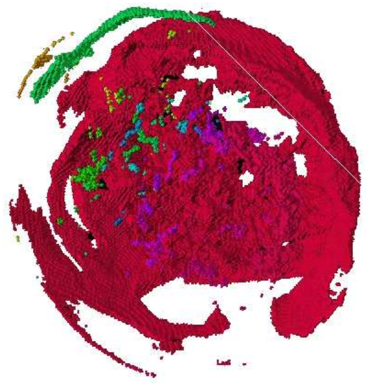
4. DEX

In collaboration with Kurt Stockinger, John Shalf, Wes Bethel. Based on an earlier project with Wendy Koegler, Jacqueline Chen and Arie Shoshani.

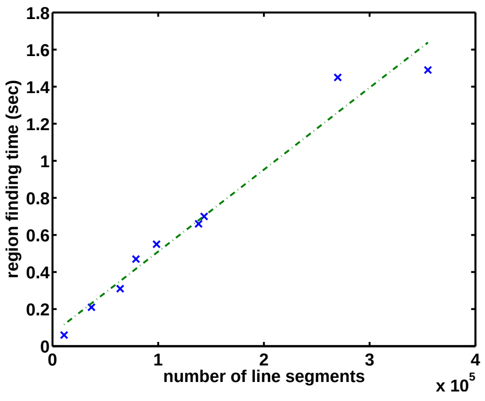
The search results produced by FastBit are internally represented as compressed bitmaps. They can be easily converted into consecutive mesh points (called line segments) for data produced from regular meshes [11, 12]. Because a line segment is a compact representation of the regions of interest, we have devised algorithms that can find regions of interest in theoretically optimal time. More specifically, the time required to identify a region of interest on a regular mesh is a linear function of the number of line segments as shown in Figure 5(b). This makes it possible to perform on-line ad hoc exploration on very large simulation data sets.

5. Summary

By taking advantage of the read-only nature of scientific data, our FastBit searching software provides significantly faster searching operations than commonly used DBMS. One of the key technology is the Word-Aligned Hybrid (WAH) code for compressing bitmap indices. WAH compressed indices have not only been proven to be theoretically optimal for one-dimensional



(a) A layer of an exploding Supernova



(b) The time to find connected regions of various sizes

Figure 5. DEX can efficiently find connected regions in 3D mesh data. Finding regions of interest in a 480 x 480 x 480 supernova simulation never took more than 2 seconds.

queries, but also been shown to be more efficient than any known methods for multi-dimensional queries as well. Applying this searching tool in a number of data-intensive applications has demonstrated it be extremely efficient.

Acknowledgments

This work was supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

References

[1] S. A. Bass, M. Gyulassy, H. Stöcker, and W. Greinerk. Signatures of quark-gluon plasma formation in high energy heavy-ion collisions: a critical review. *J. Phys. G: Nucl. Part. Phys.*, 25:R1–R57, 1999.

[2] P. O'Neil and D. Quass. Improved query performance with variant indices. In *Proceedings of SIGMOD'97*, pages 38–49. ACM Press, 1997.

[3] P. O'Neil. Model 204 architecture and performance. In *2nd International Workshop in High Performance Transaction Systems, Asilomar, CA*, pages 40–59. Springer-Verlag, September 1987.

[4] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. In *Proceedings of SIGMOD'98*, pages 355–366. ACM press, 1998.

[5] G. Antoshenkov. Byte-aligned bitmap compression. Technical report, Oracle Corp., 1994. U.S. Patent number 5,363,098.

[6] T. Johnson. Performance measurements of compressed bitmap indices. In *VLDB'99*, pages 278–289, San Francisco, 1999. Morgan Kaufmann.

[7] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. On the performance of bitmap indices for high cardinality attributes. In *Proceedings of VLDB 2004*, pages 24–35. Morgan Kaufmann, 2004.

[8] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. An efficient compression scheme for bitmap indices. Technical Report LBNL-49626, Lawrence Berkeley National Laboratory, Berkeley, CA, 2002.

[9] Kesheng Wu, Junmin Gu, Jerome Lauret, Arthur M. Poskanzer, Arie Shoshani, Alexander Sim, and Wei-Ming Zhang. Grid collector: Facilitating efficient selective access from data grids. In *Proceedings of International Supercomputer Conference In Heidelberg, 2005*, 2005. A draft appeared as LBNL report LBNL-57677.

[10] Kesheng Wu, Wei-Ming Zhang, Victor Perevoztchikov, Jerome Lauret, and Arie Shoshani. The grid collector: Using an event catalog to speed up user analysis in distributed environment. In *Proceedings of CHEP*, 2004.

[11] Kurt Stockinger, John Shalf, Wes Bethel, and Kesheng Wu. Dex: Increasing the capability of scientific data analysis pipelines by using efficient bitmap indices to accelerate scientific visualization. In *Proceedings of SSDBM 2005*, 2005. A draft appeared as LBNL report LBNL-57023.

[12] Kesheng Wu, Wendy Koegler, Jacqueline Chen, and Arie Shoshani. Using bitmap index for interactive exploration of large datasets. In *Proceedings of SSDBM 2003*, pages 65–74, Cambridge, MA, USA, 2003.