Advanced Science and Technology Letters
Vol.120 (CCSC 2015), pp.735-741
http://dx.doi.org/10.14257/astl.2015.120.146

Dynamic Page Layout Management for Efficient Big Data Analytics

Kyounghyun Park, Choon Seo Park, Hee Sun Won

Big Data Software Research Department,
Electronics and Telecommunications Research Institute, Daejeon, Korea
{hareton, parkcs, hswon}@etri.re.kr

Abstract. Traditional page storage models are suitable for a specific workload such as OLAP or OLTP workloads. However workload for big data analytics usually contains OLAP and OLTP queries. Therefore, these page storage models do not show good performance in case of big data analytics. In this paper, we propose dynamic page layout for big data analytics. The proposed page layout analyzes workload and periodically reorganizes data pages. As a result, the proposed page layout management shows better performance, compared to the previous page storage models in the mixed workload environment.

Keywords: OLAP, OLTP, mixed workload, page storage model, page layout

1 Introduction

Relational database systems have been developed in two fields - Online Analytics Processing (OLAP) and Online Transaction Processing(OLTP)[1]. The purpose of OLAP is mainly to help with planning, problem solving, and decision support by analyzing high dimensional data online, while the main emphasis for OLTP is fast query processing, data integrity in multi-access environments, and an effectiveness measured by number of transactions per second. Therefore, OLAP is characterized by relatively low volume of transactions and OLTP is characterized by a large number of short on-line transactions (insert, update, delete). Although difference between OLAP and OLTP is clear, enterprise database markets are getting to require real-time analytics of big data that needs to perform mixed OLAP and OLTP operations. However, the current database systems are not enough to support mixed workload. Therefore they are not suited for real-time big data analytics. In this paper, we introduce a flexible page storage model and dynamic page layout mangement. Dynamic page layout analyzes a mixed workload in realtime and extracts attribute group.

This paper is organized as follows. In the next section, we introduce the overview of hybrid database systems and main page storage models. In section 3, we explain a column group-based page storage model(CGM) organized by the layout manager. In section 4, we show the performance on CGM, compared with the current page models. Finally, we conclude in Section 5.

ISSN: 2287-1233 ASTL

2 Related Works

In this section, we briefly describe the overview of hybrid database systems. We also address main page storage models in database systems.

3.1 Hybrid Database Systems

Hybrid database systems are emerging OLAP and OLTP database systems that allow to process analytical queries on the transactional data. Many researches on hybrid database systems have been performed since 2009 in which [2] suggested that in-memory column stores are suited for OLAP and OLTP. [2] explains that in-memory column stores enable to process OLAP and OLTP due to enterprise software and system hardware environments. Octopus[3] is an one-size-fits-all database system that automatically adapts to various workloads with one database engine. Octopus creates various materialized views to mimic OLTP, OLAP, streaming systems as well as other types of databases. Octopus improves the overall performance by morphing into any hybrid combination of these systems.

Hyrise[4] is an in-memory hybrid storage engine that paritions tables into vertical partitions of varying widths depending on how columns accessed. The key idea of hyrise is to provide dynamic vertical partitioning of the tables. Hyrise improves the system performance by minimizing the number of cache misses for the given workload while it does not support SQL. ES^2(Elastic data Storage System of epiC)[5] is a data storage system of epiC(Elastic Poser-aware data-Intensive Cloud platform). epiC is an data-intensive cloud platform for supporting OLAP and OLTP and ES^2 supports column group-based vertical partitioning and horizontal partitioning into vertical partition.

Hyper[6] is an in-memory database system that uses the MMU/OS support to separate OLTP and OLAP. Hyper performs fork operation when it needs OLAP processing. Hyper avoids any interaction between OLTP and OLAP using lazy copy on update technique. Scyper[7] is a distributed hybrid in-memory database system and is derived from Hyper. Scyper uses OLTP process for OLTP and coordinator process for OLAP. Scyper consists of 1 primary node and several secondary nodes. The primary node uses hearbeat to check status of secondary nodes.

SAP HANA[8] is an in-memory database system for OLAP and OLTP and combines a row store engine and a column store engine. In-memory colum store is based on SAP TREX text engine and SAP BIA and in-memory row store is derived from P*Time. Relation data resides in tables in column or row layout and can be converted from one layout to the other.

3.2 Page Storage Model

NSM(N-ary Storage Model)[9] is a relation database page model in which records are stored sequentially. NSM stores all attributes of a record together so does not show good performant in case of OLAP that accesses a part of attributes in tables.

DSM(Decomposition Storage Model)[10] is a page storage model that stores sub-relation by partitioning vertically data record. Sub-relation contains id and one attribute value so it offer a high spatial locality when sequentially accessing one attribute. However, DSM deteriorate significantly for queries that involve multiple attributes.

PAX(Partition Attribute Access)[11] is a page storage model that partitions each record within the page for spatial locality. Each attribute is grouped together on minipages respectively. Compared to DSM, PAX has fewer cache misses and lower reconstruction cost.

Overall, all page models are static page storage models with fixed page layout. Therefore, they are not efficient when we should manage and analyze big data characterized by mixed OLAP and OLTP.

3 CGM: Column Group-based Page Storage Model

We intoduce our column group-based page model (CGM). CGM is a page storage model to efficiently process mixed OLAP and OLTP queries using dynamic page layout management. In this section, we explain CGM page structure and basic page operations.

3.1 CGM Page Structure

In CGM page model, each page consists of a page header and a set of subpages. The page header includes Page ID, the number of subpages, offset array of subpages, the number of attributes and records, and attribute-subpage mapping information.

Each subpage has a subpage header, an array that stores Item objects, and tuple data. A subpage header has four pointers - start, end, lower, and upper. The start and end fields point to the start and the end of the page. The lower field points to the end of the Item array and the upper field points to the start position of the last tuple. Each Item object consists of an offset value and a length value of the corresponding tuple.

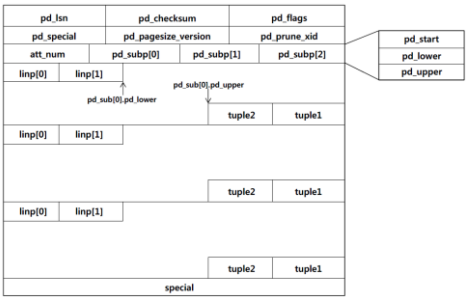

Fig. 1. A CGM Page Model Structure

Fig. 2 shows the tuple structure in CGM. Each tuple consists of a tuple header and a set of attribute values. The tuple header has the number of attributes and tuple

length. In addition, the tuple header has an object array that consists of value length and value offset of each attribute. Each object in the array points to each attribute value.

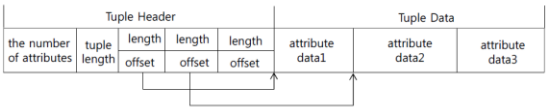

Fig. 2. A CGM Tuple Structure

3.2 CGM Page Operations

An insert operation checks if there is space available in the page. If there is no more space to insert data, a new page is created and the record is inserted into the new page. To search a record or a part of attributes in a record, the system finds the subpage to access by interpreting the page header. Then, it reads items and get data using offset and length values. For an update operation, if the new data's size is equal to or smaller than the existing data's size in the tuple, the operation is done immediately. Otherwise, the system stores new data in available space in the page or reorganizes the page first and then updates the data. To delete a record, the system removes the item entry in the item array. This may cause fragmentation in pages. Therefore, the system periodically reorganizes pages by running page compaction.

4 Dynamic Page Layout Management

In this section, we describe page layout management for efficient big data analytics. Dynamic page layout consists of page monitoring, page layout management, page storage management.

3.1 Page Monitoring

Page monitoring collects the names and selectivity of the attributes accessed by the query processor. In addition, it collects IDs of the pages accessed by the system when evaluating the queries. In short, the page monitor collects and manages information that is necessary to create column groups of each page.

3.2 Page Layout Management

The page layout manager periodically extracts the efficient column groups from the current workload and provides the storage manager with column group information. To select the best column groups, the page layout manager requires a cost model and a column group selection algorithm. The cost model is based on cache miss so the page layout manager computes the number of cache misses when processing the

workload. The process for extracting column groups is as follows. First, the page layout manager receives a page list accessed by query evaluation from the page monitor. Second, the page layout computes all possible costs about all possible attribute combinations and extracts the best column groups.

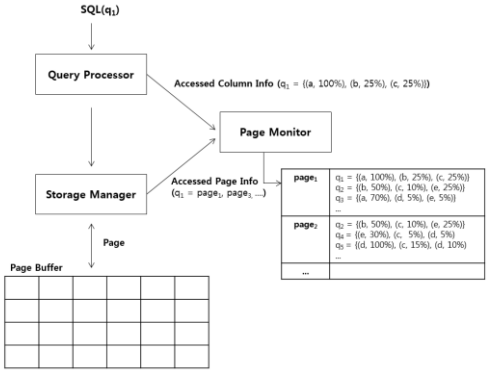

Fig. 3. Page Monitoring

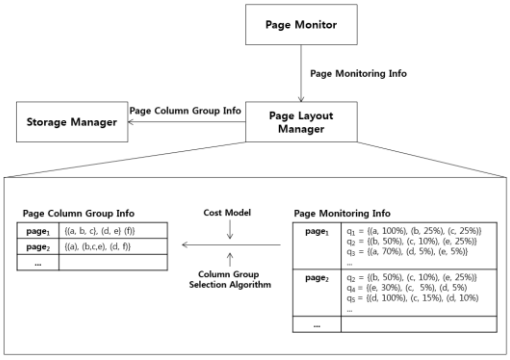

Fig. 4. Page Layout Management

3.3 Page Storage Management

The purpose of the dynamic page layout reorganizer periodically is to get new column groups of the page from the layout manager and reorganizes the page. The page layout reorganizer does not reorganize all pages. Instead, it reorganizes only hot pages. A page is considered to be “hot” if the number of accesses is greater than a threshold. The dynamic page layout reorganizer consists of a candidate page manager and a candidate page reorganizer. The candidate page manager decides if a page is a hot or not. The selected hot pages are sent to the candidate page queue. The role of the candidate page reorganizer is to reorganize old pages. For the page reorganization, first, it reads a candidate page from the candidate page queue. Second, it reads a page header and checks if the current column group information is equal to the new column

group information. If the new column group information is equal to the current, page reorganization is ignored and it reads the next page from the candidate page queue. Otherwise, the candidate page reorganizer starts reorganizing the page according to the new column group information.

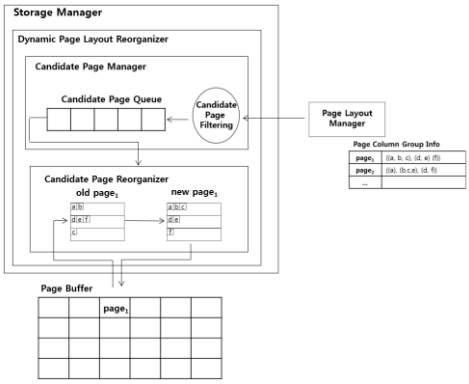

Fig. 5. Page Storage Management

5 Experimental Result

We compared the performance of the proposed CGM and the existing NSM and PAX models. All experiments were performed on a PC with 2.83 GHz Intel(R) Core(TM) 2 Quad Processor and 4 GB of main memory. The operating system was Ubuntu 12.04 64-bit with kernel version 3.5.0. The page size of all pages is 8 KB.

Firstly, we compared the performance of all storage models with OLAP workload. PAX and CGM are superior to NSM. It is because NSM reads all attributes to search for only one attribute, thus causes more cache misses than PAX and CGM. When the number of attributes increases, the improvement ratios of PAX and CGM over NSM also rise. CGM resulted in 36% faster evaluation time as compared to NSM. When the size of the dataset becomes bigger, the performance of PAX and CGM is also 30% better than NSM.

Secondly, we compared the OLTP performance. Because PAX has to reconstruct the record with significantly high costs to search for data, its performance is worst among all models. With regard to the number of attributes (tested with 200,000 records), CGM is 16% faster than PAX. As for the number of records, CGM performs 6% better than PAX.

Finally, we compared with the mixed OLAP and OLTP workloads. CGM is superior to the other models because it consists of subpages that are optimized for the mixed workload and causes less cache misses. In summary, for the mixed workload, the column group-based page model shows the best performance while, for pure OLAP or OLTP workload, the performance of our model is acceptable.

6 Conclusions

In this paper, we have proposed an efficient page storage model and dynamic page layout for processing mixed OLAP and OLTP workloads. To overcome the drawbacks of previous page models, which are good for only OLAP or OLTP, we organize tuple data into subpages dynamically. Different subpages contain data of different column groups. We also monitor page access, re-compute column groups, and reorganize the layout of hot pages periodically to improve the system's efficiency.

Acknowledgement. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0101-15-233, Smart Networking Core Technology Development).

References

1. Michael Stonebraker, Chuck Bear, Ugur Cetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, Stanley B. Zdonik, One Size Fits All? Part 2: Benchmarking Studies, CIDR 2007, 173-184
2. Hasso Plattner, A common database approach for OLTP and OLAP using an in-memory column database, SIGMOD Conference, 2009.
3. Alekh Jindal, "The Mimicking Octopus: Towards a one-size-fits-all Database Architecture," VLDB 2010 PhD Workshop, Sep. 13, 2010
4. Martin Grund, Jens Kruger, Hasso Plattner, Alexander Zeier, Philippe Cudre-Mauroux, Samuel Madden, "HYRISE - A Main Memory Hybrid Storage Engine," PVLDB, 4(2): 105-116, 2010
5. Yu Cao, Chun Chen, Fei Guo, Dawei Jiang, Yuting Lin, Beng Chin Ooi, Hoang Tam Vo, Sai Wu, Quanqing Xu, "ES2: A cloud data storage system for supporting both OLTP and OLAP," ICDE 2011, 291-302, 2011
6. Alfons Kemper, Thomas Neumann, "HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots," ICDE 2011, 195-206, 2011
7. Tobias Mühlbauer, Wolf Rödiger, Angelika Reiser, Alfons Kemper, Thomas Neumann, "ScyPer: A Hybrid OLTP&OLAP Distributed Main Memory Database System for Scalable Real-Time Analytics," BTW 2013, 499-502, 2013
8. Franz Färber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, Jonathan Dees, "The SAP HANA Database - An Architecture Overview," IEEE Data Eng. Bull., 35(1): 28-33 , 2012
9. R. Ramakrishnan and J. Gehrke, "Database Management Systems," WCB/McGraw-Hill, 2nd edition, 2000
10. George P. Copeland, Setrag Khoshafian, A Decomposition Storage Model. SIGMOD Conference 1985, 268-279
11. Anastassia Ailamaki, David J. DeWitt, Mark D. Hill, Marios Skounakis, Weaving Relations for Cache Performance. VLDB 2001, 169-180

