Taming Big Wide Tables: Layout Optimization based on Column Ordering*

1. Motivation

Modern data analytical tasks often witness very wide tables, from a few hundred columns to a few thousands. Wide table is very efficient for big data analytics because it avoid joining multiple tables which is much more costly. In SQL-on-Hadoop systems, wide tables are stored in columnar formats such as Parquet or ORC-File on HDFS. In such formats, tables are horizontally partitioned into a set of row groups [1]. Each HDFS block may contain one or more row group. Within each row group, rows are partitioned by column. Data in one column is stored continuously.

Such columnar data layout manner has been proved to be efficient for analytical workloads. Because data in the same column can be compressed better and queries can avoid reading unnecessary columns [2]. In production wide table workload at Microsoft Bing, compared to row data layout, columnar data layout saves 40% storage space and improves 50%-93% query efficiency. However, columnar data layout can be further improved. We find that the order of columns also matters the I/O efficiency of wide table analytics. Because in wide tables, there are more disk seeks, and by reordering the columns, the frequently accessed column can be put nearby so that the seek cost is reduced.

2. Column Ordering Problem

Seek cost in hard disk drives (HDD) is highly related to the seek distance, and longer seek distance generally means longer seek time [3]. The pattern of the seek cost function is general for all disks. In this work, we studies the general patterns of seek cost function on disk-based HDFS, and defined the seek cost model for HDFS. Given the seek cost model, the workload, and the data schema and statistics, we can define the column order problem as finding the optimal column order which makes the seek cost of all the queries in the workload minimal. Formal definition of the problem is shown in the poster.

3. Solution

We devise a probabilistic algorithm, called Simulated Annealing Based Column Order Algorithm (*SCOA*). Simulated Annealing (SA) is a well-known meta-heuristic to solve hard problems [4]. It is an iterative procedure that continuously updates the current state until the system reaches a state that is good enough, or until a given computation budget has been exhausted.

In each iteration of SCOA, a neighboring column order strategy S_0 of the current *column order strategy* S is generated by perturbing the current column order strategy. The algorithm probabilistically decides between moving to S_0 or staying in S. The *AcceptColumnOrder* function utilizes a Temperature function and an Energy function to determine whether the neighbor column order strategy should be accepted or not. The temperature returned by the Temperature function decreases (cools down) during the iteration process following a specific distribution, which is called annealing schedule. The Energy function is used to calculate the seek cost of a column order strategy for the given workload. The objective of SCOA is to achieve a state (column order strategy) with lowest energy. So the neighbour column order strategy with lower energy is generally more likely to be accepted.

Our column ordering algorithm shows up to 50% efficiency gain in our evaluation and is currently implemented into Microsoft Bing log analysis pipeline.

References

[1]. Y. Huai, S. Ma, R. Lee, O. O'Malley, and X. Zhang. Understanding insights into the basic structure and essential issues of table placement methods in clusters. PVLDB, 6(14), 2013.

[2]. D. J. Abadi, S. Madden, and N. Hachem. Column-stores vs. row-stores: how different are they really? In SIGMOD, 2008.

[3]. R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau. Hard disk drives. In Operating Systems: Three Easy Pieces. Arpaci-Dusseau Books, 0.80 edition, May 2014.

[4]. S. Kirkpatrick et al. Optimization by simulated annealing. Science, 220(4598), 1983.

---

* This work is supported by the Outstanding Innovative Talents Cultivation Funded Programs 2014 of Renmin University of China.

Taming Big Wide Tables: Layout Optimization based on Column Ordering

Haoqiong Bian, **Ying Yan**, **Liang Jeff Chen**, *Yueguo Chen*, **Thomas Moscibroda**
Microsoft Research, *Renmin University of China*
{ying.yan, jeche, moscitho}@microsoft.com

Summary

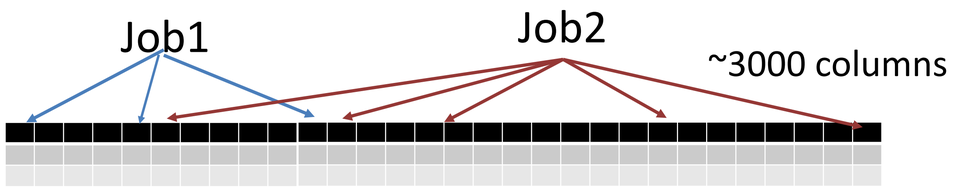
- Column store is widely used for efficient data analytics. However, the order of columns has not received much attention because it was believed that the number of columns in a big table is small, usually less than one hundred.
- Based on our investigation, the order of columns can affect much of the I/O performance especially when the table is big and wide.
- Our proposed column ordering algorithm - SCOA, shows up to 50% efficiency gain under real production data and workload.
- Our SCOA has been implemented into Microsoft Bing log analysis pipeline.

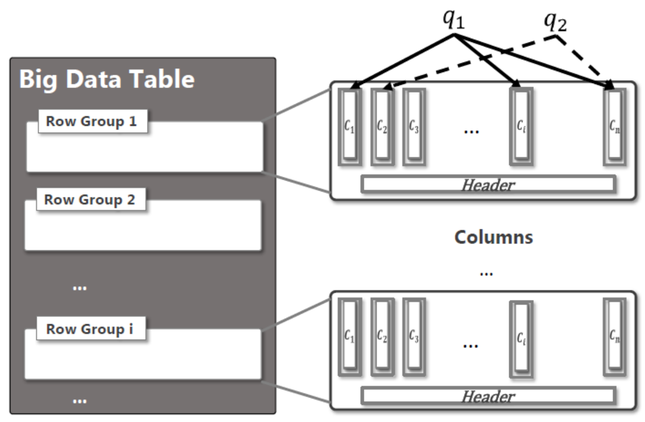
Big Wide Table and Column Ordering

The Importance of Column Ordering

Wide tables are stored as a set of columnar format files. (E.g. thousands of columns in Microsoft)

Thousands of daily queries running





Disk seeks become the main part:
up to 70% of I/O cost
(≈ 100 M$/day)

Problem Definition

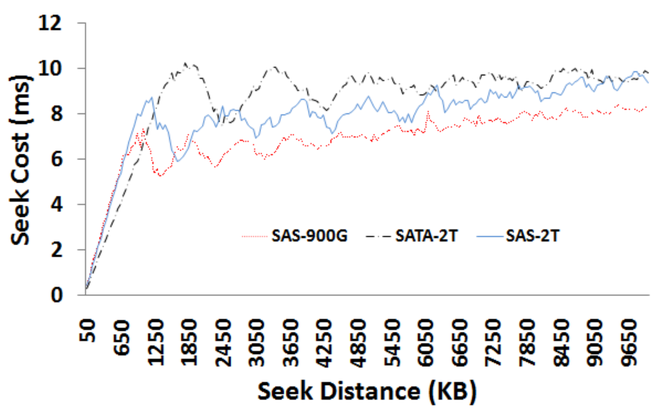
Seek Cost: Given two data objects *i* and *j*, the seek cost from *i* to *j* is denoted as $Cost(i, j) = f(dist(i, j))$, where *f* is the seek cost function which depends on the hardware.

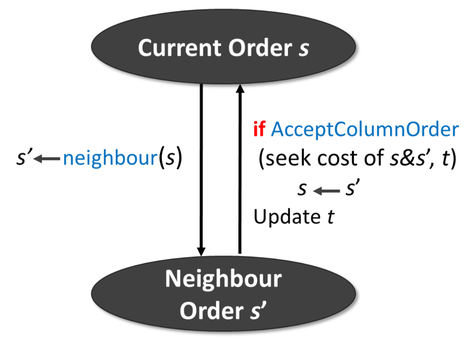
Column Order Strategy: Given a table with n columns, a column order strategy $S = \langle c_1, c_2, ..., c_n \rangle$ is an ordered sequence of those columns.

Column Ordering Problem: Given a workload *Q* containing a set of queries, finding an optimal column order strategy $S^* = \langle c_1, c_2, ..., c_n \rangle$, such that the overall seek cost of Q is minimized.

Seek Pattern Learning + Ordering Algorithm

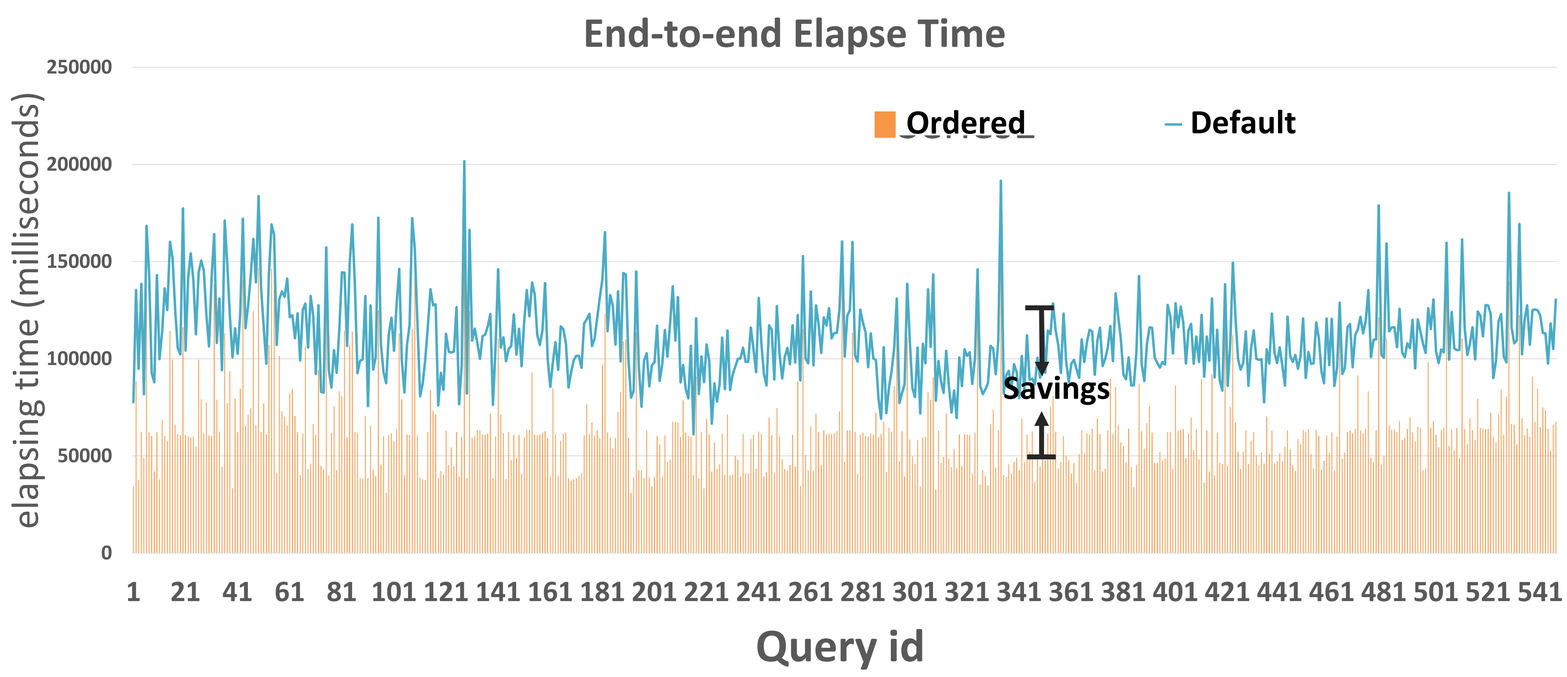
Study the cost model of column access + **Propose a Simulated Annealing Based Ordering Algorithm**



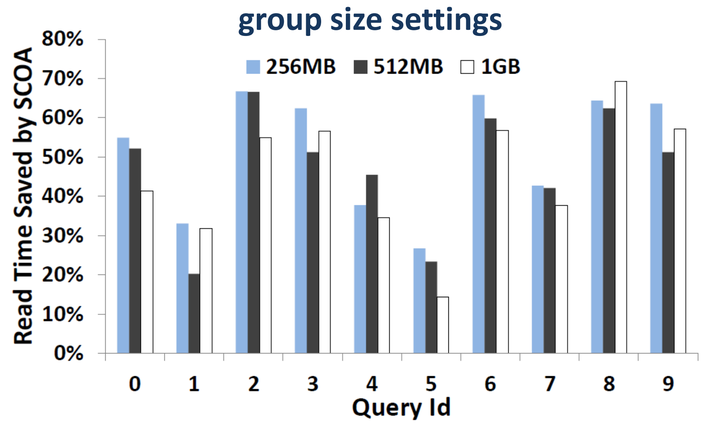


Experimental Results

End-to-end performance
(5-Node Cluster: HDFS, Spark, Disk SAS-2TB, 6T data)
Achieve 43.2% gain on average.



Significant Savings under different row group size settings



Different OS cache policies make no significant effects on the saving of column ordering