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

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 \(\text{Cost}(i,j) = f(\text{dist}(i,j))\), 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 \(S = \{c_1, c_2, ..., c_n\}\) 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 \(S^* = \{c_1, c_2, ..., c_n\}\), 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.

End-to-end Elapse Time

Significant Savings under different row group size settings
- 256MB
- 512MB
- 1GB

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

Savings

End-to-end Elapse Time (milliseconds)

Query id

0 1 2 3 4 5 6 7 8 9

End-to-end Elapse Time

Savings

0 1 2 3 4 5 6 7 8 9

Query id

Savings

0 1 2 3 4 5 6 7 8 9

Query id

Significant Savings under different row group size settings

Cache Enabled
Cache Disabled

Savings

0 1 2 3 4 5 6 7 8 9

Query id