Improving the Performance of Parallelized Bitmap Index Compression Through Data Striping

Alexia Ingerson and David Chiu (Advisor)
Department of Mathematics and Computer Science, University of Puget Sound

Introduction

As technology continues to be fundamental to our society, the abundance of data has increased exponentially. Because computers' hard drives are slow, the more data is stored, the longer it takes to access useful information. For this reason, it is imperative to use efficient data structures to provide fast data access, such as bitmap indices.

What is a bitmap index?

A bitmap index is a way of representing large data sets by expanding its columns, or attributes, into "bins" and representing the values of each row, or tuple, by a 0 or 1 depending on whether it falls into each bin.

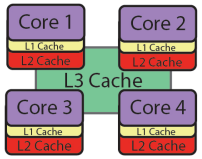
Sample Database Table:

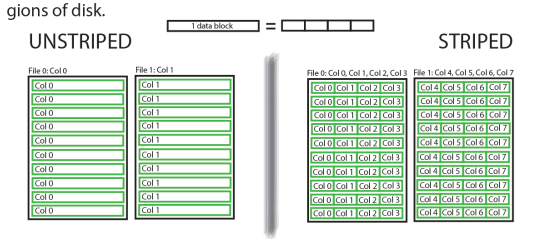
Tuple	Name	Sex	Age
t1	Brian	M	28
t2	Alex	F	35
t3	Kris	F	12
…	…	…	…

Corresponding Bitmap Index:

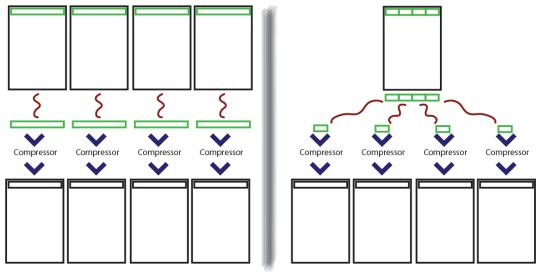
Tuple	Name			Sex		Age		
	Brian	Alex	Kris	M	F	0–18	19–50	51–99
t1	1	0	0	1	0	0	1	0
t2	0	1	0	0	1	0	1	0
t3	0	0	1	0	1	1	0	0
…	…	…	…	…	…	…	…	…

To combat this problem, we split the index into N columns, and dispatch a thread to compress each column separately. Our research investigates the data-access bottlenecks and explores ways to reorganize our data set to increase performance. We hypothesize that the contention for shared last-level cache and disk I/O among the threads led to slowdowns in execution. To this end, we reformatted the bitmap files into striped files, that is, instead of storing one column per file, each striped file contains multiple columns interleaved, thereby decreasing the latency of accessing files from different regions of disk.



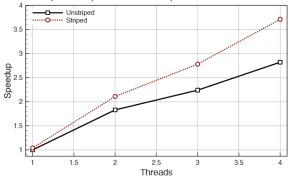


Threaded compression:

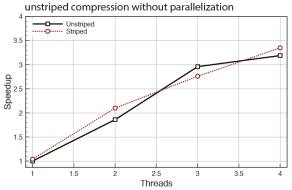


Speedup

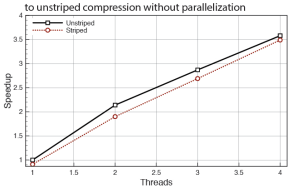
Speedup achieved on bitmap_stock.txt compared to unstriped compression without parallelization



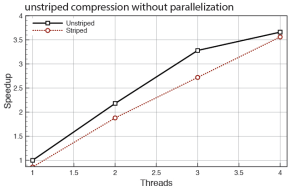
Speedup achieved on bitmap_landsat.txt compared to unstriped compression without parallelization



Speedup achieved on kddcup_sorted.bitmap compared to unstriped compression without parallelization

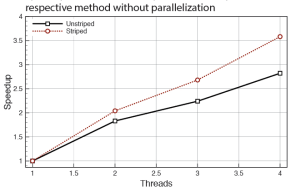


Speedup achieved on bitmap_energy.txt compared to unstriped compression without parallelization

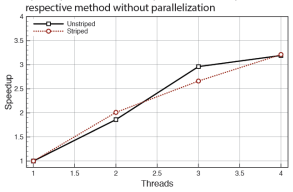


Scalability

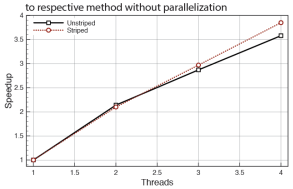
Speedup achieved on bitmap_stock.txt compared to respective method without parallelization



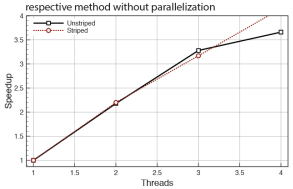
Speedup achieved on bitmap_landsat.txt compared to respective method without parallelization



Speedup achieved on kddcup_sorted.bitmap compared to respective method without parallelization



Speedup achieved on bitmap_energy.txt compared to respective method without parallelization



Operating System	OS X
Processor	2.2 GHz
Number of Cores	4
Memory	16 GB
L2 Cache (per core)	256 kB
L3 Cache	6 MB

Data Set Name	Size	Number of Columns
bitmap_stock.txt	7.1 MB	1081
bitmap_landsat.txt	250.3 MB	901
kddcup_sorted.bitmap	2.37 GB	475
bitmap_energy.txt	10.01 GB	1367

Results

While the striped method did result in a larger speedup some of the time, changing the format of the files for compression did not achieve a consistent or significant speedup up when compared to the unstriped, unparallelized results.

However, striping the files consistently resulted in a larger speedup relative to its own unparallelized version. This result shows promise for the method if optimized.

Future Work

Though the overall speedup gained from striping the files was not what we hypothesized, the algorithm shows a greater ability for scalability (relative speedup to its own unparallelized version). Because of this, we believe it is possible to optimize this method of compression to demonstrate a consistent benefit to striping bitmap indices before compression.

Acknowledgements

Many thanks to Google Summer of Code for supporting the open-source development of this project over the summer of 2015 and to the University of Puget Sound for providing resources to print and present my research. A huge thanks to my advisor, David Chiu, for his support and guidance throughout the course of this project.