Efficient Query Execution over Large Databases through Semantic Caching of Bitmap Indices

Sarah McClain

Follow this and additional works at: https://soundideas.pugetsound.edu/summer_research

Part of the Databases and Information Systems Commons

Recommended Citation
McClain, Sarah, "Efficient Query Execution over Large Databases through Semantic Caching of Bitmap Indices" (2019). Summer Research. 361.
https://soundideas.pugetsound.edu/summer_research/361

This Article is brought to you for free and open access by Sound Ideas. It has been accepted for inclusion in Summer Research by an authorized administrator of Sound Ideas. For more information, please contact soundideas@pugetsound.edu.
Efficient Query Execution over Large Databases through Semantic Caching of Bitmap Indices
Sarah McClain and David Chiu (advisor)
Department of Mathematics and Computer Science, University of Puget Sound

Background
As popular applications become increasingly data-intensive the need for novel techniques to query large-scale data stores becomes more prevalent. 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 such as bitmap indices and caches to provide fast data access.

What is a bitmap index?
A bitmap index is a way of consolidating a large data set by representing certain columns using a single bit (0=false, or 1=true) for each row. The data is then represented by expanding the columns, or attributes, into what are called bins and represent the value of the row, or tuple, by a 0 or 1 depending on whether it falls in a bin.

Table Name | Sex | Age
---|---|---
Kelsie | F | 9
Megan | F | 26
Nalin | M | 23
Zac | M | 80

Sample Database

What is a semantic cache?
Due to limited space on the (CPU), most data is stored in secondary storage, such as the disk, which is very slow. A semantic cache is a form of architecture that caches the result of selection queries after being fetched from the disk. Semantic caching involves splitting a query into two pieces, a probe query and a remainder query. The probe query addresses cached regions that contribute to the new query. The remainder query refers to data not contained in the cache (but relevant to the query) that much be fetched from secondary memory. Semantic caching allows the computer to reduce access to the disk by reusing partial matches in the cache.

Algorithm

Algorithm 1: cacheLookup(R, startQ, endQ)
1: \( R \) = Set of result vectors
2: \( startQ \) = the starting column of full query
3: \( endQ \) = the ending column of full query
4: Output
5: \( R \) = Result vectors for probe query \([startQ, endQ]\)
6: \( best \) = list
7: for all \([startQ, endQ]\) in list do
8: \( R' \) = getdata([startQ, endQ])
9: \( best \) = \([startQ\text{,} endQ]\)
10: if \( endQ \geq bestQ \) then
11: \( R' \) = getdata([startQ\text{,} endQ])
12: \( R' \) = \([startQ\text{,} endQ]\)
13: return \([startQ\text{,} endQ]\)

Plots

Total Time for Executing Query File With Cache and Without Cache

Results

The semantic cache tremendously decreased the amount of time needed to query the database. As shown in the figures, as the number of queries grows, it becomes more efficient to leverage this caching architecture. The semantic cache also has a greater speedup as the number of range queries increases. There is still more to be studied about semantic caching. More work on the effect of different lookup algorithms and caching compound ranges could increase efficiency even more.

Acknowledgements

I would like to thank the University of Puget Sound for providing me the opportunity to complete this project as well as the resources to print and present my research. I would also like to give a special thanks to my advisor, David Chiu, for his time and support as he guided me throughout the course of this project.