A System Overview for netCDF-FastBit Integration

David Marks, Elias Ioup, John Sample, Kevin Shaw
Geospatial Computing
Naval Research Laboratory
Stennis Space Center, Mississippi
{dmarks,eioup,jsample,kshaw}@nrlssc.navy.mil

Mahdi Abdelguerfi
Computer Science Department
University of New Orleans
New Orleans, Louisiana
mahdi@cs.uno.edu

***Abstract*—This paper discusses the creation of a FastBit bitmap index from the contents of a netCDF file. Using a two-step transformation, netCDF is loaded into a FastBit bitmap index which can then be used to perform quick and highly efficient data queries. Metadata from the original netCDF is utilized along with the bitmap indexes and the output is extracted and visualized in several formats. The performance of the netCDF-FastBit indexes is shown to be far greater than accessing the netCDF file without.**

Keywords-netCDF; geospatial processing; bitmap indexing

I. INTRODUCTION

As highly advanced sensors began to penetrate into the awareness of the disparate scientific fields, many researchers found themselves grappling with the need to store and usably interact with vast amounts of data. Unfortunately, the real world scientific data that was being collected violated several of the classical assumptions about data in database design. The largest hurdle of all for traditional database structures was the high multidimensionality of the scientific data, lacking any field or even reasonable combination of fields that could serve as a unique identifier for a gathered data.

In contrast to traditional database design, bitmap indexes do not require a unique key identifier, and indeed work quite well with very multidimensional data. Bitmap indexes are usually overlooked in lieu of traditional databases because of their inefficiency in handling update and delete operations, but in the case of scientific data that is written once this inefficiency does not matter. The other detriment normally levied against bitmap indexes, the “Curse of Cardinality”, has largely been solved using the process of order-preserving bin-based clustering [1]. Indeed, bitmap indexing serves as an excellent form of indexing for write-once highly multidimensional data, which real world scientific data is a prime example of.

NetCDF is a popular scientific data storage format, and it provides a very compact encoding of large multidimensional data sets. It is not, however, optimized for the retrieval, analysis, and mining of the data stored within its format. Performing efficient queries over data stored within netCDF files is difficult, especially if a large number of netCDF files are involved.

FastBit is an open source bitmap indexing toolkit. FastBit creates highly efficient bitmap indexes, and provides an interactive SQL interface into its generated indexes. Its WAH based compression has been proven to provide optimal query response time with the compressed bitmaps smaller than comparable B-trees [2].

Using the system outlined within this paper, a netCDF file can be loaded into a FastBit bitmap index, providing efficient and interactive query access to the data stored within. The process requires only a one-time cost of bitmap index creation time and room to store the generated indexes. Further, a number of different ways to envision the produced query results are demonstrated.

In the next section, the method used to create bitmap indexes from netCDF data is discussed, with experimental results generated from testing provided. In System Architecture, we present the structure and functions of the proposed system. Screenshots of our proposed graphical user interface are provided, as well as a schematic for the system architecture. In Query Processing, the benefits of such a system are described, both in terms of speedup of generalized data access and by a number of advanced spatio-temporal queries. Finally, in Conclusions we reiterate the goals and benefits of this system and stress the benefits its use could provide.

II. METHODS

Because FastBit currently only accepts comma separated values as input for bitmap generation, the first step in netCDF-FastBit integration involves a transformation of the netCDF data into the comma separated value format employed by FastBit. A number of tools are already existent for the purpose of netCDF to comma separated values, but as the concept of comma separated values is only loosely defined, different tools produce outputs in different formats. One tool, ncks (short for netCDF Kitchen Sink), offers a correctly formatted comma separated value transformation however, and with the correct configuration of options our source netCDF files can be converted into a comma separated value format readable by FastBit. After FastBit ingests the transformed netCDF data, the process is complete. In testing, a 1.4 GB netCDF file required slightly over 30 minutes to transition from netCDF to FastBit bitmap index. The resulting index used up 6.8 GB of space.

III. SYSTEM ARCHITECTURE

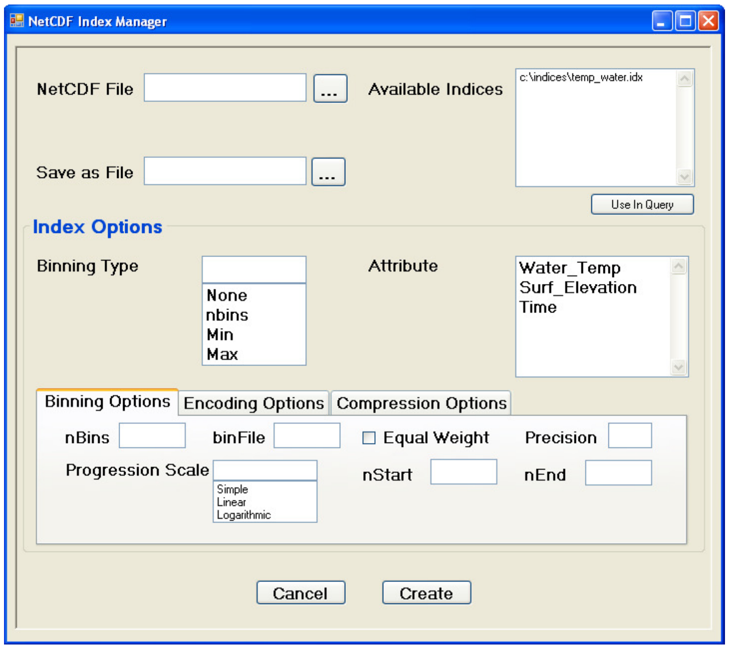
The main purpose of the netCDF-FastBit integration is to provide an efficient and interactive SQL interface into the data stored within. To maximize the search capability of a user's queries, however, requires more than just the netCDF extracted indexes. Further, retrieved data may require different formats for different questions. Thus, the below system in Fig. 3 is proposed.

A. *Internal Components*

The *Bitmap Index Manager* is the component charged with the building and maintenance of the bitmap indexes as described in Methods above. The *Manager* allows for both single and multiple file indexing for netCDF scientific data. Further, the netCDF files may be accessed either locally or across a network.

The *Metadata Repository* holds any relevant metadata found within the netCDF file's header. Often these headers contain comments and notes about the data that would not be immediately obvious in an examination of the data itself, such as the data's provenance. This metadata will aid in maintaining organization between the bitmap indexes of several netCDF files, as well as providing a boost to query speed by screening out queries from which no point (or perhaps all points) meets the desired criteria.

The *Query Processor*'s role is to transform the user submitted high level queries into sequences of bitwise logical operations executed in the reduced search space generated using the *Metadata Repository*. Further, the *Query Processor* will optimize the processing of a number



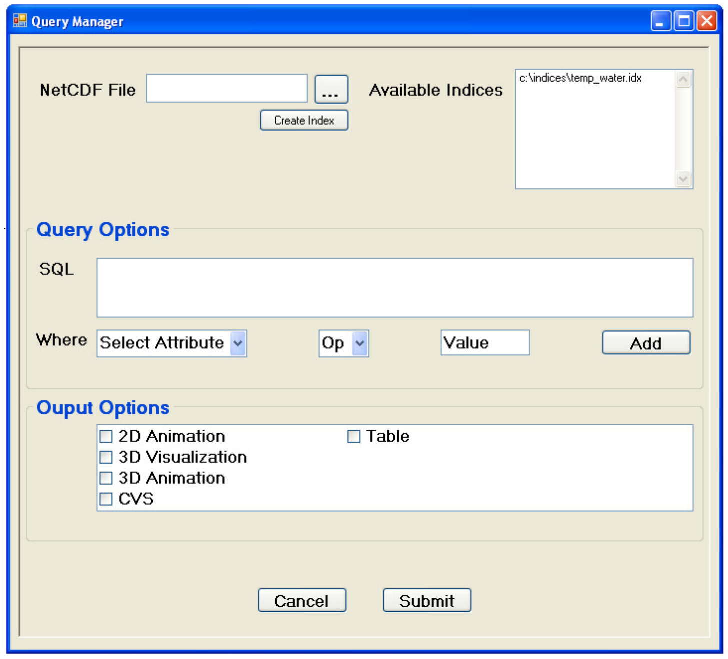
c:\indices\temp_water.idx

Binning Type: None, nbins, Min, Max

Attribute: Water_Temp, Surf_Elevation, Time

Progression Scale: Simple, Linear, Logarithmic

Figure 1. Index Manager Screen



c:\indices\temp_water.idx

Figure 2. Query Manager Screen

of primitive operations found in a recent study on query processing of mesh data in forming the basis of higher level queries [3].

B. *Graphical User Interface*

To utilize the outlined system, a graphical user interface is provided, as seen in Fig. 1 and Fig. 2. A user either selects a locally available netCDF file(s) or points to one available over the network. A list of already processed indexes saved with the *Metadata Repository* are then displayed for the selected netCDF file, and can be instantly used via the "Use In Query" button.

Alternatively, a new index can be created from a selected netCDF file, with a number of different bitmap indexing options made available via the index options tabs. These options will alter the generated bitmap index's underlying structure, and can be used to fine tune the created index for specific tasks and queries. The default values will serve for most general purposes, however, and the most common usage will simply consist of selecting which attributes to include in the generated index.

Once an index to query have been selected, the user can submit a query either freehand with the SQL textbox or by using the dropdown Where boxes to programmatically generate one. When submitting their query, a user will also select which kind of output is desired, either in the form of text or as a visualization of the results. Note that more than one form can, of course, be chosen, although some forms are not appropriate for all results. Animations, for example, are useless unless the results include some concept of passing time.

IV. QUERY PROCESSING

The system outlined above allows for an unprecedented level of access into the contents of a netCDF file. Absent the created bitmap indexes, any kind of search on the netCDF could only be accomplished via a linear search of the file itself. Using our test data, this linear search took approximately 20 minutes to complete, and all values of speedup given below are based on this figure.

The simplest types of queries examined were simple equality queries, such as "return all points where the temperature was X". These queries returned in only 1 second, a 1200x speedup. Retrieving all temperatures for a single point in time returned in 4 seconds, a 300x speedup. The slowest execution time was found in inequality queries, such as "return all points where the temperature was above Y", which took 4 minutes to compute. That still represented a 5x speedup, however.

But this system is not limited to only simple queries. More complex queries are possible given the powerful SQL interface afforded by FastBit. A range query was created and implemented to return all points within a specified distance of a chosen point. Using a two-step bounding box filter and the haversine function, this range query often returned results within a second or less, resulting in a 1200+x speedup. Further, by ordering the returned results based on distance to the chosen point and returning only a specified k number of results, a pseudo-KNN search was built, with equivalent performance results [4].

V. Conclusion

This system provides a way to efficiently process data stored within netCDF files and to produce data output in a number of formats. The graphical user interface allows for an easy to use interface into the system and allows a user to leverage a number of different bitmap index customization options in the creation of their index. Beyond providing an otherwise unavailable amount of access into the contents of a netCDF file, our query results demonstrate the massive gains in performance inherent in this approach. Using this system, querying the contents of a netCDF file not only becomes possible, but easy and efficient.

References

[1] Kesheng Wu, Kurt Stockinger, and Arie Shoshani. "Breaking the curse of cardinality on bitmap indexes." Scientific and Statistical Database Management. Ed. Bertram Ludäscher & Nikos Mamoulis. Springer, 2008, pp. 348–365.

[2] Kesheng Wu, Ekow J. Otoo, and Arie Shoshani. *An efficient compression scheme for bitmap indices.* ACM Transactions on Database Systems 31:1 (2006), pp. 1-38

[3] B.S. Lee and R. Musick. *MeshSQL: the query language for simulation mesh data*. Information Sciences, volume 159, issue 3-4, 2004, pp. 177-202.

[4] David Marks, "An Analysis of netCDF-FastBit Integration and Primitive Spatial-Temporal Operations" (2009). *University of New Orleans Theses and Dissertations.* Paper 984. http://scholarworks.uno.edu/td/984 Retrieved 12/2011

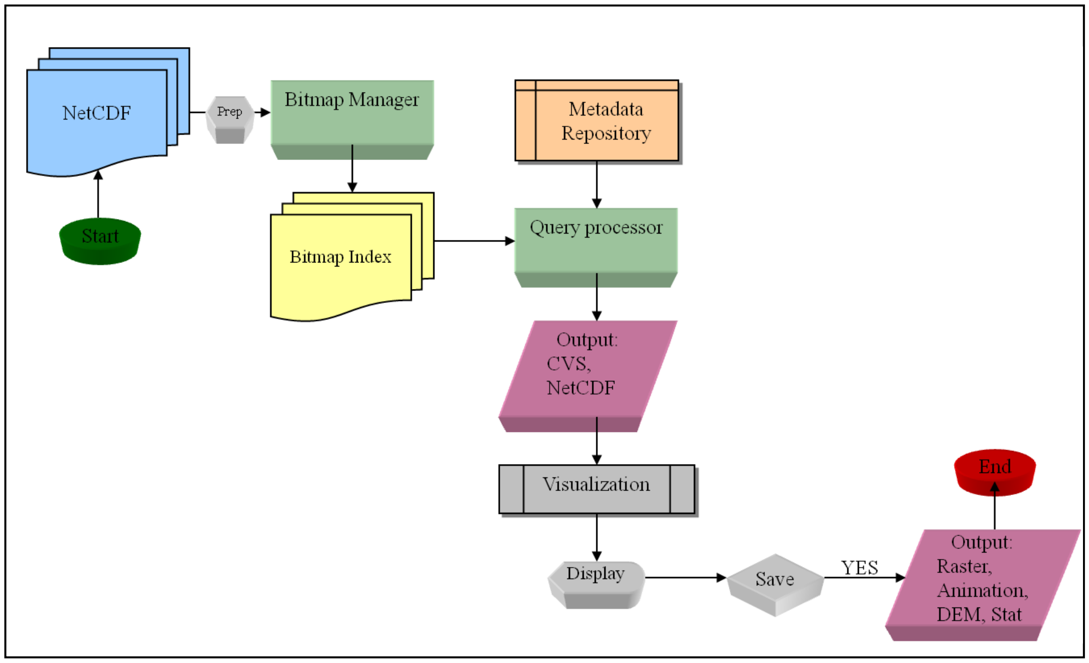


Figure 3. System Architecture