Improving the Efficiency of Subset Queries on Raster Images

Tanu Malik[1], Neil Best[1], Joshua Elliott[1], Ravi Madduri[1,2], Ian Foster[1,2]
Computation Institute[1] Mathematics and Computer Science[2]
University of Chicago Argonne National Lab
tanum, nbest, jelliott@ci.uchicago.edu
madduri,foster@anl.gov

ABSTRACT

We propose a parallel method to accelerate the performance of subset queries on raster images. The method, based on map-reduce paradigm, includes two principles from database management systems to improve the performance of subset queries. First, we employ column-oriented storage format for storing locationand weather variables. Second, we improve data locality by storing multidimensional attributes such as space and time in a Hilbert order instead of a serial, row-wise order. We implement the principles in a map-reduce environment, maintaining compatibility with the replication and scheduling constraints. We show through experiments that the techniques improve data locality and increase performance of subset queries, respectively, by 5x and 2x.

1. INTRODUCTION

High-resolution remote sensors and the establishment of digital photogrammetry have lead to a rapid increase in the amount of geospatial data being acquired as raster images. The data includes spatial and temporal properties of geographical regions, and at high resolution quickly leads to petabyte-sized image data repositories. Data-intensive applications often seek gigabytes and terabyte-sized subsets from the repositories for analysis. Although the geographic information science community has invested in tools that fetch subsets of data from repositories, as the size of the subsets and the number of applications seeking it grow, an important data management task is how to perform efficient data reduction, i.e., efficiently download, manage, and ingest subsets of geospatial data for data-intensive applications from petabyte-sized repositories.

The challenge is particularly demonstrated in biophysical and economic land use change and climate impact studies [1], in which complex process models or other data-intensive simulations seek daily weather, land-cover, and/or environmental data products from thousands of geospatial locations simultaneously. To perform a typical data reduction query, the scientist issues subset queries, each of the form $Q_i = \pi_{P_i}(\sigma_{R_i \wedge T}(Images)$, in which a set of user-specified weather variables P_i are sought over the ith spatial region R_i and a common time frame of T using the common database operators of selection σ and projection π. An example two-location data reduction query is "Project temperature and precipitation attributes for the selected region R_1 and project solar radiation for the selected region R_2 between a common time frame of 26th of June, 1990 and 26th of June, 2000". Typically there will be thousands of selected regions.

Raster meteorological data sets used in climate research are often stored in files in NetCDF format. Current data reduction tools, such as NetCDF operators [4] execute subset queries on files in a serial order, which makes data reduction a time-consuming task. To execute subset queries in parallel, files may be placed in a parallel file system, such as PNetCDF [9], but the communication still needs to be controlled by the programmer and high-level operators for subsetting and finding hyperslabs are not available.

Geospatial databases, such as PostGIS Raster and MySQL also provide data reduction tools through their high-level programmatic interfaces. Raster images can be ingested into the databases through the Geospatial Data Abstraction Library [2]. Performance for petabyte-sized repositories, however, suffers because the databases do not support parallel operation, and the storage model is row-oriented i.e., they store multidimensional array data as a single record. They do no leverage the performance benefit derived from a column-oriented design, which has been shown to provide orders of magnitude of performance improvement.

In this paper, we investigate the performance improvement of subset queries by executing them in parallel. We adopt Hadoop, a map-reduce platform for executing queries in parallel, especially since its framework is open-source, freely accessible, simple to program, provides built-in fault-tolerance, and is designed to analyze large data sets with good scalability.

In our parallel operation of subset queries, we noted that data locality is a determining factor for map-reduce performance. This aspect has been noted in other scenarios as well [14]. For geospatial data, this aspect is of particular concern since subset queries often seek observations nearby in space and time. In addition each subset query seeks different set of weather variables. To improve data locality, we

ACM HPDGIS'11, November 1, 2011, Chicago, IL, USA.

leverage two data organization techniques: First, we store each of the spatial, time and weather variables in a column-oriented storage format. By separating variables from each other, only the desired π_{P_i} as sought by each subset query are loaded into memory. This is contrary to row-oriented format in which all weather variables associated with a given location are loaded into memory and then processed. Second, we index the data by transforming multi-dimensional space and time attributes to a single-dimension Hilbert order. Thus data points that are closer to each other in Hilbert space are stored together.

We describe how the two techniques can be implemented in Hadoop without violating any scheduling and replication constraints. The two techniques are orthogonal to each other and if combined can lead to performance improvement by orders of magnitude. In our current experiments, we show that when applied independently, they increase performance by 5x and 2x respectively. The remainder of the paper is organized as follows. The next section describes related work. We describe our model and improvements in Section 3. In Section 4 we describe preliminary experiments showing the advantages of colocation and data organization.

2. RELATED WORK

For geoscientists, analysis is currently limited by cumbersome access and manipulation of large datasets from remote sources, and data reduction is a necessary task prior to analysis. Sometimes, data reduction is offered as part of computational systems. For instance, grid toolkits such as Globus [8] allow reduction at the granularity of files, wherein files can be to be staged to and from compute nodes. Data locality is generally ignored in favor of greater scheduling flexibility for higher throughput. Execution of data reduction queries have also gained recent attention in the form of OpenNDAP protocols [4], which provides simple, lightweight access to large, remote datasets, but their interfaces remain optimized for operations on small volumes of data.

Data reduction systems have paid less attention to organization of the underlying data to improve the performance of the reduction query, making it a rate-limiting step in analysis. Recent advancements in database management systems, have put emphasis on column-oriented organization that favors analysis on individual attributes and thus speeds up subset queries [5]. Such organizations have not been significantly analyzed in the context of spatio-temporal datasets. Space-filling curves [11] have known to provide better performance for spatial queries, and Hilbert order curves provide the best performance [6]. While parallel Hilbert order methods have been studied for load balancing [12], the performance of the methods is less known in the context of emerging distributed file systems, such as HDFS.

3. BACKGROUND

Model outputs and data products in the climate, and land-use communities can be represented as multi- dimensional arrays. NetCDF offers libraries for converting high-level data models such as arrays to low-level byte streams and vice-versa, and thus has become an important standard to store model outputs and data products. Figure 1(a) shows a typical geospatial NetCDF file in which the main data array has coordinates arrays and time (xc, xy, t). The recorded weather variables are the values of the cells, and typically there are several of them, each corresponding to the measurements taken.

Owing to the binary encoding of the scientific data files, which is not compatible with the underlying MapReduce framework, a common approach is to convert the NetCDF data into text format and store the cells of the array in row-wise order (Figure 1(b)). One or several bytes, represent the set of recorded variables in each cell.

In a typical map-reduce setup the text-formatted file is ingested into the Hadoop file system (HDFS), which divides the data into blocks and replicate those blocks across the local disks of nodes in the cluster. The distributed file system adopts a master-slave architecture in which the master maintains the file namespace (metadata, directory structure, file to block mapping, and location of blocks) and the slaves manage the actual data blocks.

Subset queries in MapReduce can be expressed by defining two functions: map and reduce. Conceptually, a set of concurrently executing map tasks scan and read all the data in parallel and filter based on the region and time clauses specified in the subset query [1]. The output of each map task is a key-value pair in which the key is the spatial location and the value is the set of chosen weather variables. The resulting output is re-partitioned, and each new partition is routed to a single reduce task for final processing.

4. DATA REORGANIZATION TO IMPROVE EFFICIENCY OF SUBSET QUERIES

Executing map-reduce, as described above, is sufficient when subset queries seek all weather variables. However, subset queries are model specific, which may seek surface variables, such as air temperature, wind speed, and pressure for certain geographical regions, and gaseous composition, such as CO_2 and methane over other regions. Since all weather variables are not used, storing the variables together and iterating over the entire record leads to significant disk I/O. To reduce the amount of disk I/O, we must introduce column-oriented storage format for the geospatial data. Thus instead of storing together all the weather variables corresponding to a geospatial location, we must store them in separate columns. In Hadoop, column-oriented data placement is not available by default and in subsection 4.1, we describe how a new format class can easily achieve this in Hadoop.

The key attributes in geospatial data are multi-dimensional, consisting of three dimensions latitude, longitude and the time dimension. The NetCDF text convertors store the 3-dimensional (xc, yc, $time$) data in a row-wise order. Importing the data file in its native format leads to poor performance because it does not correspond to the access pattern of subset queries that seek regions nearby in space and time. Traditional database management systems have avoided this problem through use of multi-dimensional indexes. To improve performance we transform multidimensional data into a single dimension by using Hilbert curves. The Hilbert or-

[1]This multi-attribute access obviates the choice of key-value stores, such as HBase, which do not natively support secondary access

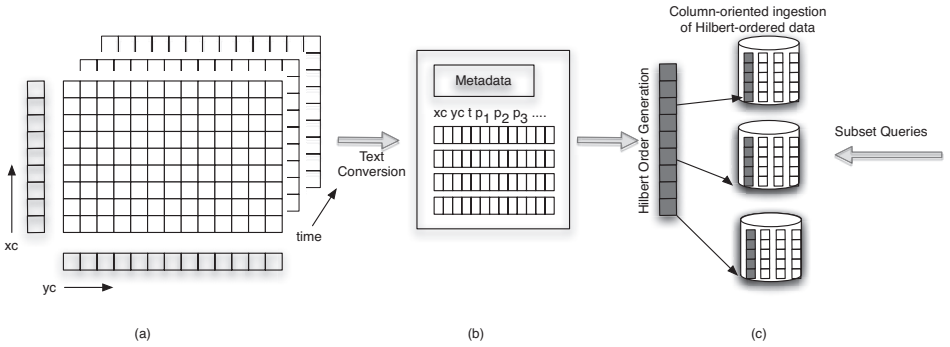

Figure 1: Architecture

dered data is then range partitioned onto parallel nodes. We describe in subsection 4.2 the map-reduce implementation of a parallel Hilbert curve . Figure 1(c) shows the application of the two methods on the text file.

4.1 Co-location

The default HDFS block placement policy uses 3-way block-level replication to provide fault tolerance on commodity servers, but the default block placement policy does not provide any co-location guarantees. To change the default data placement policy, Hadoop provides configuration settings, which when initialized with the appropriate class can allow Hadoop to place the incoming data on nodes as defined in the class. To implement a column-oriented storage we initialize the Hadoop configuration with a `ColumnFormat` class.

The `ColumnFormat` class reads in metadata from the NetCDF common format file and initializes as many files as the number of variables in the CDF file. Thus given a NetCDF text file with 50 variables, the `ColumnFormat` class reads in data in a row order but outputs data in separate files under HDFS. The column files in a given directory are scanned sequentially and the records are reassembled using values from corresponding positions in the files. The class implements methods for selecting data from a specified list of columns that are mentioned in the subset query. Column-oriented storage in Hadoop has also been described in [7]. However, their method does not learn the schema automatically and the user must define it a priori.

Such a storage format assumes that the application is willing to pay a one-time loading cost to organize the input data in the appropriate column-oriented fashion. As argued in earlier papers [10], this is a reasonable assumption to make for datasets that are expected to be analyzed multiple times. In addition, our experiments show that the system incurs minimal penalty for reconstructing records from the constituent columns.

4.2 Hilbert order-based key organization

To linearize multidimensional geospatial data into a single dimension, we use Hilbert curves. A Hilbert curve is a continuous curve that passes through each point in space exactly once. It enables one to continuously map multidimensional data onto a line and is an excellent technique for image-to-line mapping. Th Hilbert curve is a self-similar curve that is created recursively. The basic curve H_1 has 2×2 vertices. The Hilbert curve H_i can be derived directly from H_{i-1} by replacing each vertex of H_{i-1} with a basic curve, which may be rotated or rejected. We direct the reader to the Wikipedia entry on Hilbert curves that shows interactively the shape of H_1, H_2 and H_3 [3].

The Hilbert curve must be big enough to cover all the pixels of the image by its vertices. By definition, it is easy to see that a Hilbert curve with order i has $2^i \times 2^i$ vertices. Therefore, for an image with $m \times n$ cells, we must build a Hilbert curve with order k in order to cover all pixels where $k = log^2(max(m, n))$ After the Hilbert curve is generated, all the cells are on the vertices of the curve. The position of each cell on the mapped curve is called the Hilbert order of that pixel. More details are described in [6, 11, 12].

We perform the Hilbert order mapping as a map reduce operation, in which the map phase outputs a Hilbert order key for each record. In the reduce phase we group Hilbert order keys based on the following procedure: Given that we have generated a Hilbert order of k and initialized reducers in multiples of four, $4r$, keys are distributed such that those who have Hilbert order of $4rk$ go to reducer 1, $4rk+1$ goes to reducer 2, and so on. This way keys of a square region, viewed as composed of 4 parts, arranged 2 by 2 are distributed to each reducer. Keys on each reducer form a part of Hilbert curve too, but one with a lower order. For the subset queries, we only have to check the start positions and the length of the string pieces on one reducer. For the remaining reducers, the result is the same, such that the subset query workload is balanced on all nodes.

5. EXPERIMENTS

We show through experiments the performance of subset queries in a a parallel environment. Our current experimental framework consists of a small cluster of one master node, which is the name node and the job tracker and six slave data nodes. All nodes are homogeneous with 2.0GHz/CPU, 4GB RAM and 120GB storage capacity. The OS stack is Red Hat Linux 5.0 accompanied with Java 1.6.0.6 and Hadoop 0.21.2 as running environment.

The total amount of data is 13GB in NetCDF format expanded to 200GB in text format. Our datasets consist of 24 weather variables that are of high importance to crop models and coordinate and time attributes. The files stores daily values of each variable. We use `netdump` to convert

NetCDF files to CDL files. The files are further formatted to translate array points into a database like record. The metadata is maintained in a separate file to be read in by the ColumnFormat class. We used the more efficient non-recursive procedures to generate Hilbert curves of order 32. The query workload that specifies the weather variables and the set of ranges is generated synthetically, but closely corresponds to standard crop model simulations.

Currently our experimental setup is modest and is primarily done to test the performance of the techniques. We are currently in the process of scaling up our experimental testbed to Amazon EC2 and perform experiments with much larger dataset sizes.

In this paper, we include two performance results. The first experiment shows the performance improvement only due to column organization. The second, shows the performance improvement only due to Hilbert order organization. In both we compare it with direct execution of subset queries on data with no organization. Experiments show that column organization gives on average a factor of $5x$ improvement and linear ordering improves performance by factor of about $2x$.

In the first experiment (Figure 2) for column-oriented storage, we varied the number of weather variables in the query, *i.e.,* the set π_{P_i} for a given region R_i from 3 to 12. Real queries normally ask between 4-5 query variables and so the range provides sufficient characterization. The size of the region is kept such that the subset query seeks 10GB of data. The result shows that as the number of variables are increased the execution time of queries increases but is significantly less (about 5x) in comparison to row-oriented storage. The result is not surprising given the performance improvements due to column storage but a validation of previously reported results.

In the second experiment for linear ordering (Figure 3), we vary R_i, the size of the region sought. The number of variables are kept constant at 4. Performance results show linear ordering to perform better than row-ordering of multi-dimensional arrays. However, improvement is not drastic. In typical database management systems indexing often gains higher performance improvements. We continue to investigate the result further in terms of load balancing and the impact on the selectivity of the query.

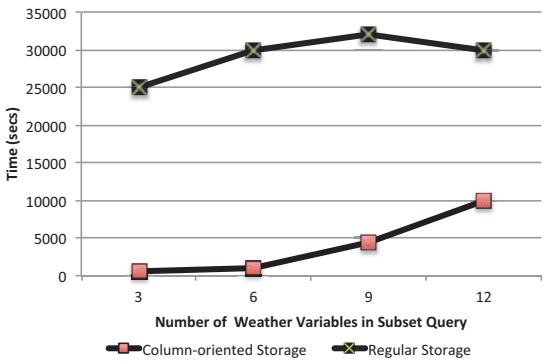

Figure 2: Impact of column organization

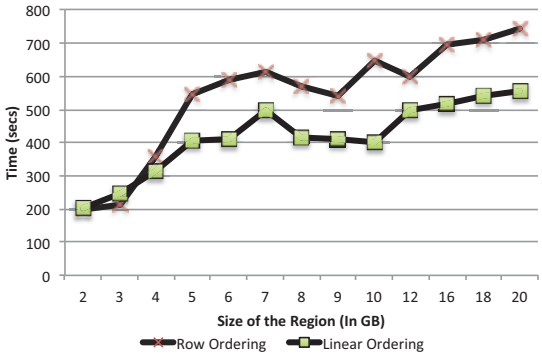

Figure 3: Impact of linear ordering, i.e., mapping multi-dimensional data to a single dimension

6. CONCLUSIONS

Data reduction continues to remain a time consuming data management task. In this work, we have emphasized on parallel operation for subset queries and used two data organization techniques to improve their performance. Our preliminary results show performance improvement. We continue to develop the data reduction system that can be used by the wider community.

7. REFERENCES

[1] `http://www.cimearth.org/`

[2] `http://www.gdal.org/`

[3] `http://en.wikipedia.org/wiki/Hilbert_curve`

[4] `http://opendap.org/`

[5] D. Abadi, S. R. Madden, and N. Hachem. Column-Stores vs. Row-Stores: How Different Are They Really? SIGMOD, 2008.

[6] C. Faloutos and S. Roseman. Fractals for secondary key retrieval. SIGMOD, 1989.

[7] A. Floratou, J. Patel, E. J. Shekita and S. Tata, Column-Oriented Storage Techniques for MapReduce, VLDB Endowment, 2011.

[8] I. Foster and C. Kesselman, Globus: A Metacomputing Infrastructure Toolkit. Journal of Supercomputer Applications. Vol: 11(2), 1997.

[9] J. Li, W. Liao, A. Choudhary, et. al., Parallel NetCDF: A High Performance Scientific I/O Interface, Supercomputing, 2003.

[10] A. Pavlo, E. Paulson, A. Rasin, D. J. Abadi, D. J. DeWitt, S. Madden, and M. Stonebraker. A Comparison of Approaches to Large-Scale Data Analysis. SIGMOD, 2009.

[11] H. Samet. The Design and Analysis of Spatial Data Structures. Addison-Wesley, 1989.

[12] Z. Song and N. Roussopoulos, Using Hilbert curve in image storing and retrieving, Information Systems, Vol: 27, 2002.

[13] A Woolf, K. Haines and C Liu, A Web-service Model for Climate Data Access on the Grid, Journal of High Performance Computing Applications, Vol 17(3), 2003.

[14] H.Yang, A. Dasdan, R. Hsaio and D.S. Parker, Map-reduce-merge. Simplified relational processing on large clusters, SIGMOD, 2007.