Cloud Computing for Satellite Data Processing on High End Compute Clusters

N. Golpayegani
University of Maryland, Baltimore County
golpa1@umbc.edu

Prof. M. Halem
University of Maryland, Baltimore County
halem@umbc.edu

Abstract

Hadoop is a Distributed Filesystem and MapReduce framework originally developed for search applications by Google and subsequently adopted by the Apache foundation as an open source system. We propose that this parallel computing framework is well suited for a variety of service oriented science applications and, in particular, for satellite data processing of remote sensing systems. We show that, by installing Hadoop on a cluster of IBM PowerPC blade clusters, we can efficiently process multi-year remote sensing data, expect to see speed performance improvements over conventional multi-processor methodologies, and have more memory efficient implementation allowing for finer grid resolutions. Moreover, these improvements can be met without significant changes in coding structure.

1. Introduction

Cloud computing as originally proposed by Google [1], [2] and subsequently offered as open source software by the Apache foundation as Hadoop [3] is based on a parallel programming paradigm MapReduce employing a distributed file system for implementation on very large clusters of low performance processors (i.e. perhaps as many as a thousand or more processors) aimed at text based searching. One of the substantial advantages to programmers utilizing this software system is that it enables a simple parallel programming structure while automating all the data management and movement operations. The purpose of this paper is to explore the feasibility of Hadoop MapReduce parallel programming paradigm to a wider domain of scientific data processing problems when installed in a high-end compute cluster. We describe the Multicore Computational Center compute blade cluster[4] and the installation of Hadoop on its IBM Power PC cluster based on the JS20 blades in Section 2.0. We choose in this paper a scientific application from the field of remote sensing requiring data intensive processing of satellite observations. This application has properties similar to document searching in that the observations, taken each day or each hour from an instrument on board a satellite, are, as in the case of web documents, independent of each other. The instrument gathers new data as the satellite orbits the Earth but contains measurements of electromagnetic spectral energies (i.e. radiance observations) in the same format which are similar in content to words occuring on a page in a text document.

In this paper, we specifically focus on data produced from an instrument on the Aqua satellite, namely, the Visible/Near Infrared (VIS/NIR) data collected by the AIRS instrument [5, 6]. We describe the problem and relevance of gridding satellite data [7] and present the current non-Hadoop method for gridding these data. We also describe, for comparison, a proposed Hadoop MapReduce approach [8]. The remainder of the paper is organized as follows; First, we briefly describe the non-Hadoop algorithm which we will call the standard algorithm and then Hadoop algorithm. We then compare the runtime results of the two algorithms.

2. Hadoop Compute Cluster

The Multicore Computational Center at UMBC manages a computer research facility that maintains two very high end multi-core clusters that are dedicated to researchers who need the computational power available on these advanced hardware systems. Both clusters are accessed through a common management node and connected to each other and externally through a 10 Gb fiberoptic router. The system name for both clusters is BlueGrit. One system consists of 33 IBM dual-core 2.2 GHz PowerPC JS20 compute blades with 2 GB of ram, 14 dual-core processor 2.5GHz PowerPC JS21 compute blades with 4GB of ram and 4 JS22 quad-core 4.0 GHz Power PC blades with 32GB of Ram providing UMBC with a 138 core IBM PowerPC cluster. Each blade contains two such processor chips. The other configuration consists of 14 IBM QS20 Cell blades and 10 QS 22 blades. Each QS20/22 blade consists of 2 Cell processors and each Cell processor has a 3.0 GHz Power PC processor and 8 SPU processors. The QS 20 has 1 GB of local RAM with access speeds of 25 GB/s with SPU

978-0-7695-3840-2/09 $26.00 © 2009 IEEE
DOI 10.1109/CLOUD.2009.71

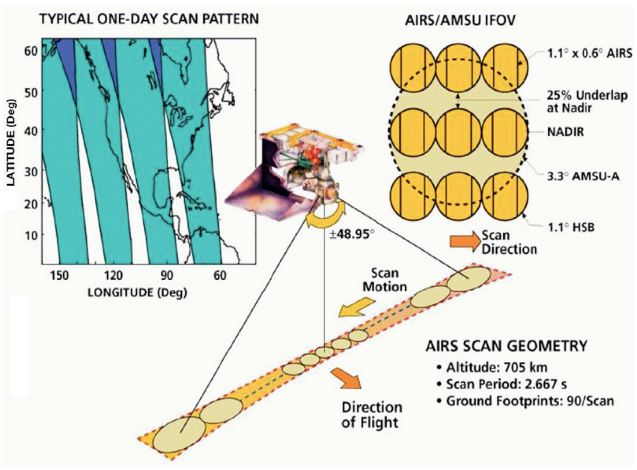

Figure 1: Cross track scans, FOV patterns, and overlap for two atmospheric radiance instruments, AIRS and AMSU, currently flying on the NASA research satellite, Aquay. Courtesy of http://airs.jpl.nasa.gov/technology/how_AIRS_works/how_AIRS_works_detail/

operations able to sustain 24Gflops/proc. The QS22 has 32GB of Ram. Adittionally, the 8 SPU processors on the chip are connected with a Broadband network at 200 GB/s offering unprecedented access for moving data between processors and memory. In addition, BlueGrit contains 2 Google rack servers with dual Intel Pentium 4 processors and 2.25 TBs each with Hadoop software installed. An additional 16 TB of disc is attached with PVFS.

The BlueGrit cluster has one Intel based head node, and a large Intel based NFS server. Each JS20 blade has two 2.2GHz PowerPC 970 processors, and 512MB of RAM. The dual core nodes have 2.3GHz processors and 2GB of RAM. The NFS server (storage001) currently has 2.2TB of storage total, 2.5GB of memory, and two 3GHz Intel Xeon processors. The management node is a two processor 2.8GHz Xeon system with 256MB of RAM and has 5.4 TB of shared storage. The operating system uses Red Hat Enterprise 4.0 Linux.

We installed Hadoop on the IBM JS20 blades. The version of Hadoop used, as of this writing, is 0.18.1. One of the JS20 blades is setup as the master Hadoop node. The remainder of the JS20 blades are setup as file and compute nodes. Each blade is installed with Fedora release 8. The Java Runtime Environment, version 1.5.0, was obtained as a binary release from IBM. The configuration files of Hadoop were mostly left at their default values. The only modifications made in the hadoop-site.conf file were to the following variables:

- hadoop.tmp.dir: Set to a local disk to store data.
- fs.default.name: Set to IP of master Hadoop node.
- mapred.job.tracker: Set to IP of master Hadoop node.

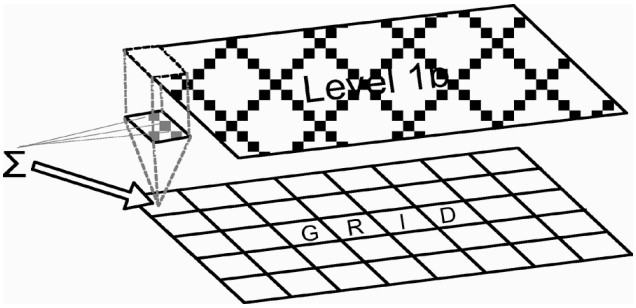

Figure 2: Reduction of satellite imagery in kilometer resolution to a gridded dataset of courser resolution.

- dfs.replication: Set to 1 to disable replication.
- mapred.reduce.tasks: Set to 30 for 30 node cluster
- mapred.task.tracker.http.address: Set to 31337

Unfortunately, Hadoop currently has a limitation in that it can only process text files. Future versions of Hadoop will be able to process binary files but, as of this writing, Hadoop is only capable of reading text files. As a result, our experiments were setup to read radiances from text files. This causes a significant slowdown in processing since files have to be read in as text, converted to binary, and then processed. Additionally, with text files the size of the input is dependent on the values stored. We therefore cannot read the entire file with one operation and had to implement multiple read statements. This also causes a significant slowdown in processing. To maintain a good comparison, both algorithms were tested against the same input files. In the future, when Hadoop supports binary files, we intend to measure the performance differences with binary data as input.

3. Algorithmic Implementation

The Atmospheric Infrared Sounder (AIRS) instrument on the Aqua satellite is in a sun-synchronous, near polar circular, orbit traveling at ~700 km above the Earth passing over 78ºN and 78ºS. The Earth makes one full rotation underneath the satellite every day, while the satellite makes ~12 orbits per day, crossing the Equator at ~30 degrees separation each ascending part of the orbit. The instrument scans ~1400 km across the sub-satellite track making observations with a spatial resolution of ~14 km at nadir and ~40km at the far end of the scan. The instrument takes observations at each footprint for 2374 channels. In addition, for four additional channels in the Visible and Near Infrared (Vis/Nir), the instrument takes an array of 8x9 observations at 1km resolution. The data is collected and stored in 6 minute intervals called granules, and 240 granules are collected in a day. Figure 1 shows the number of observations per scan line and the geometry

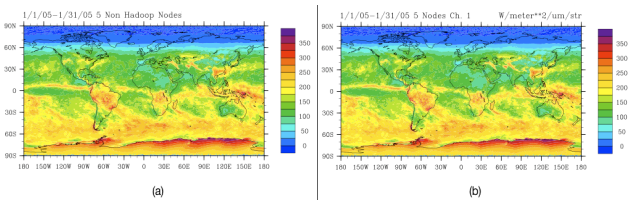

Figure 3: One Month of gridding done without Hadoop (a) and with Hadoop (b). The images look very similar suggesting that the two algorithms have the same results. The Hadoop algorithm is slightly more accurate because it does not compute an aveage of averages.

relative to the Earth. The gridding problem consists of defining a regular lat/lon grid on the surface of the Earth, say with a 1 degree of longitude interval and a 0.5 degree latitude interval, and determining which observations lie in each grid cell. The observations falling within each cell shown in Figure 2 are then averaged for certain time periods, say a day, month, year or a decade.

3.1 Non-Hadoop Approach

The standard gridding approach currently used is divided into two parts. The first part is to perform gridding for a single day. This part can then be parallelized to allow for multi-day gridding. In this section we describe the algorithm. We defer actual run time informations until the next section where we compare the runtime of this algorithm to the Hadoop algorithm.

The standard parallel processing gridding algorithm consists of assigning a given processor the responsibility to grid a complete days worth of observations. Since each day of satellite observations are independent of another day, multiple processors can process multiple days simultaneously. The unit of a days worth of data was chosen for several reasons; (i) scientists often want to analyze daily variations related to weather, (ii) gridded

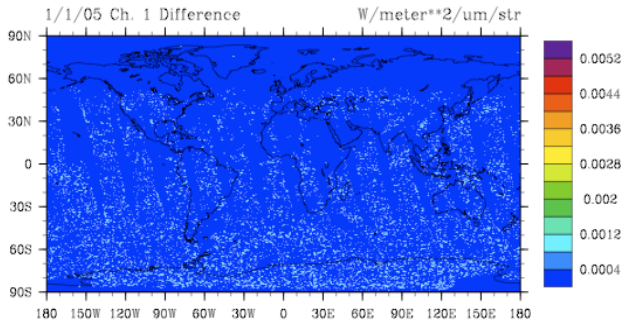

Figure 4: Subtracting the Hadoop computed grid from the standard method yields differences that can be explained by rounding errors. Therefore, the two grids are identical.

data can be used to validate weather forecasts on a daily basis and (iii) because remote network transfer of larger blocks of data from NASA archives can create computer error checking bottlenecks and/or dropped lines. Each processor's work consists of reading each granule for a particular day. Each granule contains the latitude, longitude, and the radiances for each footprint as well as each spectral channel. The latitude/longitude determines to which grid cell the radiance values belong. The radiances belonging to that cell are summed and a count is incremented on the number of radiances added in that particular grid cell. Once an entire day is read and gridded, the total radiance value for each grid cell is divided by the number of radiances added to that grid cell to obtain the average. The final value is written out to disk and can be used for computations and visualization.

When multiple days are required for the gridding, the standard algorithm is repeated for each day. The resulting grids for each day are then averaged for each cell in the grid which produces the final result. This method can be easily parallelized by performing the gridding for each day on a different computer and computing the final average over all days on a single computer. The results of this computation are not entirely accurate since they rely on averaging daily averages. To be more accurate, additional computation is required. When single day grids are produced, the algorithm needs to additionally keep track of the number of radiances falling into a particular grid cell. The additional information allows the final gridding to be accurate by multiplying the single day averages by their respective grid cell radiance counts and then averaging all days.

3.2 Hadoop Approach

With MapReduce, computations are divided into two functions the Map function and the Reduce function. In our implementation, the Map function performs the initial processing. It is provided with the input data and is responsible for processing it into key/value output pairs. These key/value pairs are sorted and grouped by key. All values for a particular key are provided as input to the Reduce function. The reduce function is responsible for processing all values for a particular key and producing the final key/value results. To perform gridding in this method, the standard gridding algorithm was modified to run in a Map and a Reduce stage. Additionally, communication between the Map and Reduce stages has to be performed via key/value pairs. Communication is accomplished by assigning each grid cell a unique number. In a 100 km resolution grid there are 360 grid cells in both latitudinal and longitudinal direction. In this example we use a Mercator grid projection for gridding. Each grid cell is then given a number starting at 0 in the upper left corner at the north pole of the array grid cells and is incremented for each neighboring cell first along a

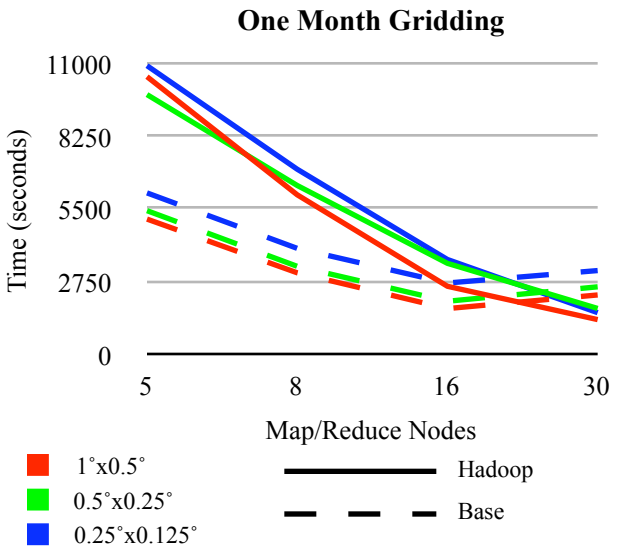

Figure 5: One month gridding of reduce nodes increased as map nodes are increased. We can see that the speedup is larger for Hadoop as the number of nodes is creased compared to the standard algorithm.

latitudinal line and then incremented by 360 for the next latitudinal line.

With a unique grid cell layout, the Map function can communicate with the Reduce function by using the unique grid cell IDs as the keys in the key/value output it produces. The Map function reads the input radiances and determines which grid cell those radiances lie. It keeps a sum of all radiances falling into each grid cell as well as a count. Once all the data is read, the Map function produces a key/value output where the key is the unique grid cell ID and the value is the summed radiances as well as the count. The analogy with document word searching is that the occurrence of a grid cell location for a particular radiance observation within a particular granule corresponds to the occurrence of a given word in a given document. At this stage, the Map function is repeatedly called for all possible input files available locally on the compute node. This Map function is run in parallel on all compute nodes and each function is assigned locally available data. As key/value pairs are produced, Hadoop sorts the keys generated by the Map function and groups all values for a particular key together. Once all Map functions have finished executing, Hadoop assigns a randomly chosen node to execute the Reduce function grouped by key, all values for each invocation of the Reduce function belong to the same grid cell. The Reduce function adds all radiance sums and counts together and divides the final sum by the final count. Finally, the Reduce function produces the final key/value output where the key is the same as in the input key and the value is the result of the last division. These output results are the same as those produced by the standard algorithm. The results can be compared visually as show in Figure 3. The differences between the two algorithms shown in Figure 4 are essentially identical.

4. Performance

The standard and Hadoop algorithms were installed on 30 IBM JS20 blades. Each blade consist of 512MB of RAM and 40GB Hard drive with approximately 25GB free. Each blade had a 2.2Ghz dual core PowerPC CPU. For the standard implementation the input data was located on an NFS server with a 1GBit connection. For gridding we used the AIRS Visible instrument on the Aqua satellite. We performed gridding at three different resultions (1˚x0.5˚, 0.5˚x0.25˚, and 0.25˚x0.125˚). Gridding was done for a one Month interval and was performed with varying cluster sizes.

Figure 5 shows the results of the runtimes for the standard and the Hadoop algorithm at the different resolutions run at cluster sizes ranging from five nodes to thirty nodes. From the results we can see that Hadoop is initially slower at fewer nodes. This is believed to be due to the additional overheard of Hadoop. At approximately 30 nodes, the MapReduce algorithm has gained a significant speedup to be faster than the standard algorithm run on 30 nodes. A second observation is that the Hadoop algorithm runtime seems to be less dependent on the grid size.

For example, at 30 nodes there is very little difference between the runtime for the three different grid sizes gridded. With the standard algorithm, however, the more finer grained resolution grid sizes require consistently higher compute time.

Additionally, we can see in Figure 5 that Hadoop's runtime is significantly slower at 5 nodes than the regular method. We believe this difference can be explained by the data size and computation method. We chose to grid Hadoop at the granule level. As each granule contains only 5 minutes of sensor data, it is processed quickly by our map function. As a side effect the map function is started and stopped in quick succession. Comparatively, the standard gridding algorithm is started and stopped between processing of an entire day. Hadoop's map function on average required an execution time of 20 seconds while the standard gridding approach had an execution time of 12 minutes. This gap in runtime is what, we believe, is the major factor responsible for the big differences in total runtime. As the number of compute nodes is increased, more map functions are started and stopped simultaneously, resulting in less accumulated overhead. Future, experiments will verify this assertion by increasing Hadoop's processing resolution to an entire day.

5. Conclusions

We have shown that the Hadoop Map-Reduce parallel programming paradigm can be used to process scientific satellite data and that it can improve the processing time of gridding algorithms. Although gridding times at a one month interval were close to those of the base algorithm, we have shown that, as resolutions are increased and as the averaging period becomes longer, say seasonal or yearly, the Hadoop algorithm showed increased savings of computer run times. The code created for the Hadoop algorithm is also simpler than that of the base algorithm. This is due to the fact that the Hadoop algorithm does not require any parallelization code. The standard algorithm requires additional code to synchronize among many nodes and deal with synchronization issues. In addition, since this system will be used operationally, the Hadoop file system has backup capabilities built in as well as automatic management rollover in case of processor failure.

Currently, we have made Hadoop available as a service such that any approved BlueGrit user of the MC2 facility can invoke this parallel programming paradigm. It can also be used for deriving higher level data products from arbitrary remote sensing systems. Future work in this project includes processing additional instruments and measuring their speed improvements. Instruments such as AVHRR and MODIS will be included in the Hadoop gridding algorithm and compared to standard methods of gridding.

6. Acknowledgements

We would like to express our appreciation to the Systems Technology Group of IBM for their generous gift that provided us with the computer resources to conduct these studies and their support of this project. This work was also supported under a NASA ACCESS grant.

7. References

[1] J. Dean, S. Ghemaway. MapReduce: Simplified Data Processing on Large Clusters. In OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004. Pages 137-150.

[2] J. Dean, S. Ghemaway, MapReduce: Simplified Data processing on Large Clusters. Communications of the ACM. January, 2008.

[3] The Apache Software Foundation. Hadoop, http://hadoop.apache.org/core/

[4] Multicore Computational Center, Compute Cluster, http://mc2.umbc.edu

[5] H. H. Aumann, M. T. Chahine, C. Gautier, M. D. Goldberg, E. Kalnay, L. M. McMillin, H. Revercomb, et.al., AIRS/AMSU/HSB on the Aqua mission: Design, science objectives, data products, and processing systems. IEEE Transactions on Geoscience and Remote Sensing, 2003.

[6] M. T. Chahine, T. S. Pagano, H. H. Aumann, R. Atlas, C. Barnet, J. Blaisdell, L. Chen, M. Divakarla, E. J. Fetzer, M. Goldberg, C. Gautier, S. Granger, S. Hannon, F. W. Irion, R. Kakar, E. Kalnay, B. H. Lambrigtsen, S.Y. Lee, J. Le Marshall, W. W. McMillan, L. McMillin, E. T. Olsen, H. Revercomb, P. Rosenkranz, W. L. Smith, D. Staelin, L. L. Strow, J. Susskind, D. Tobin, W. Wolf and L. Zhou., AIRS: Improving Weather Forecasting and Providing New Data on Greenhouse Gases. Bulletin of the American Meteorological Society: 2006 Vol. 87, No. 7, pp. 911-926.

[7] M. Halem, D. Chapman, P. Nguyen, C.Tilmes, Y. Yelena, N. Most, K. Stewart, "Service Oriented Atmospheric Radiances (SOAR): Services for Gridding and Analysis of Multi-Sensor Satellite Radiance Data for Climate Studies" , Trans. Of Geosciences and Remote Sensing Journal. 2009 Vol.47,No. 1 Jan. P.114

[8] N. Golpayegani, M. Halem, P. Nguyen. Cloud Computing Infusion for Generating ESDRs of Visible Spectra Radiances. AGU Fall Meeting 2008.