Cloud Computing – Where ISR Data Will Go for Exploitation

Albert Reuther, Jeremy Kepner, Peter Michaleas, William Smith
{reuther, kepner, pmichaleas, william.smith}@ll.mit.edu
MIT Lincoln Laboratory, Lexington, MA 02420

May 21, 2009

Introduction

The asymmetric confrontations that coalition forces have faced in the past twenty years has demanded a greater reliance on sophisticated intelligence technologies and data products in order to gain greater situational awareness of adversarial activities. Highly sophisticated SAR image processing, cyber-warfare, and persistent surveillance platforms are producing higher data rates and larger data products that are increasingly becoming more difficult to move from place to place for data and metadata exploitation. Also the relationships between these data and metadata products are becoming even more difficult to correlate, synchronize, and register. The key to solving these technological logjams is to move the data as infrequently as possible and do all of the processing where the data is being stored, including extracting the metadata of the data. Further, this metadata should be collected in a distributed data store for scalability in metadata ingestion and exploitation. Several cloud computing technologies deliver the infrastructure required for this type of processing.

Defining Cloud Computing for ISR

The term cloud computing has been making many headlines recently in the technical press. As with much technical buzzword hype that is approaching the "peak of inflated expectations" on the Gartner "hype cycle" [1], there is a technical basis amidst all of the hype. With certain technical areas that come under the umbrella of cloud computing, the technical basis shows a great amount of promise for exploiting ISR data and metadata products. But before we assess the potential impact, we must be more specific about two different types of cloud computing: utility computing vs. data intensive computing. Table 1 compares these two types of cloud computing.

Table 1: Comparison of cloud computing technologies

	Data Intensive Computing	Utility Computing
Goal	Compute architecture for large scale data analysis	Compute services for outsourcing IT
Challenges of Scale	Billions of records/day, trillions of stored records, petabytes of storage	Concurrent, independent users operating across millions of records and terabytes of data
Key Technologies	Google File System, MapReduce, BigTable, Hadoop, Sector/Sphere	Service virtualization, IT as a Service, Platform as a Service (PaaS), Software as a Service (SaaS)

Data Intensive Computing

The three main components of data intensive cloud computing are: a distributed file system (e.g., Google File System [2], Hadoop Distributed File System (HDFS) [3], and Sector [4]), a computation paradigm (e.g., MapReduce [5]), and distributed database-like hash stores (e.g., Google BigTable [6] and Hadoop HBase [3]). Among the distributed file systems, there is a further dichotomy: block-based storage versus file-based storage. Block-based file systems like Google File System and Hadoop Distributed File System partition larger files into blocks (>64MB by default) and disseminate the blocks across data servers. Conversely, file-based file systems store complete files on data servers, regardless of size. All of these file systems are intended for write-once, read-many access, and they can replicate the files they contain; the default replication rate is three, the original file and two replicas. While the file systems usually support the deletion of files, it is often slow since they are usually not designed for deletion.

Each of these distributed file systems are architected in the following manner: they have one master computer node that keeps track of where all of the files and replicas are stored and one or more nodes that store the data. If a client application wants to access a file, it must first contact the master node to determine the location(s) of the file, and then it must contact the data nodes to operate on the file.

The MapReduce parallel computing framework enables the execution of codes, in which each of the processes does not need to communicate with any other. Hadoop also has a MapReduce capability, while the MapReduce environment for Sector is called Sphere. To launch a MapReduce job, a list of files is generated against which the code is to execute. One Map job is dispatched for each file-block or file that was specified. The resulting output of the Map jobs can then be collected and combined by the Reduce jobs.

This work is sponsored by the United States Air Force under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions and recommendations are those of the author and are not necessarily endorsed by the United States Government.

Finally the distributed database-like hash stores save data objects using alphanumeric row and column indices and a time stamp. The data objects can be anything though it is designed to store smaller pieces of data; that is, it is not intended for storing images or videos. If multiple entries are made with identical row and column indices, the most recent entries are kept, while older entries are discarded. It is also possible to ask for a the data object for a specific row and column index pair that is exactly or near a particular timestamp, provided that it has not been discarded. The two aforementioned database-like hash stores are all built using a block-based distributed file system for storing data.

For enterprise-wide ISR data processing and exploitation, an effective way to architect the data intensive computing is to use a file-based distributed file system for storing the ISR data files, which include SAR images, E/O images, video, and presentation files. Then MapReduce jobs are used to extract metadata from the ISR data files and to store the metadata instances in the database-like hash stores. ISR analysts then use software tools to search the metadata instances and the ISR data files for information and relationships; the analysts can also retrieve ISR data files directly from the file-based distributed file system.

LLGrid as a Cloud Testbed

Two of our efforts in data intensive cloud computing are the building of a data intensive cloud testbed on LLGrid and the development of D4M, a computational model that is more capable than MapReduce.

With the aforementioned data ingest method in mind, we have built a data intensive cloud computing testbed using the TX-2500 LLGrid cluster at MIT Lincoln Laboratory [7], of which each node includes two 3.2 GHz single-core Intel Xeon CPUs, 8 GB of RAM, a 1.42 TB local RAID, and two gigabit Ethernet interfaces. We have installed Sector, Sphere, and the three components of the Hadoop suite. We have tested some ingestion of data into both Sector and HDFS, and we have tested the MapReduce, Sphere and Hbase capabilities. Table 2 shows the sizes of the distributed file systems that we have created on the compute nodes of TX-2500. The replication factors were set to the package defaults.

	Hadoop DFS	**Sector**
Number of nodes used	350	350
File system size	298.9 TB	452.7 TB
Replication factor	3	2

D4M – Dynamic Distributed Dimensional Data Model

The MapReduce computational framework enables the execution of embarrassingly parallel jobs. More complex parallelism can be addressed with MapReduce by iterating multiple times through the map and reduce phases, but this turns out to be quite inefficient. D4M enables the efficient execution of classic parallel jobs on a distributed cloud file system by launching the processes of a parallel job in a data location aware manner. Given a list of files in the distributed cloud file system, D4M determines the nodes on which to launch the processes of the parallel job so that all file operations are local to each node; i.e., none of the data to be processed must be transferred across nodes since the transfer of the files can incur high latencies. So far, we have demonstrated D4M by launching pMatlab/gridMatlab jobs onto a distributed cloud file system. This concept can be extended to a variety of other programming environments including Java, parallel VSIPL++, PVL, etc.

We plan to present performance metrics of the D4M, Sector/Sphere, and Hadoop technologies as a function of processor and data scale.

References

[1] Hype cycle, http://en.wikipedia.org/wiki/Hype_cycle.

[2] S. Ghemawat, H. Gobioff, and S.T. Leung, "The Google File System," 19th ACM Symposium on Operating Systems Principles, Lake George, NY, Oct. 2003.

[3] Apache Hadoop Project, http://apache.hadoop.org/.

[4] Y. Gu and R. Grossman, "Exploring Data Parallelism and Locality in Wide Area Networks," Workshop on Many-task Computing on Grids and Supercomputers (MTAGS), co-located with SC08, Austin, TX. Nov. 2008.

[5] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, Dec. 2004.

[6] F. Chang, J. Dean, S. Ghemawat, et.al., "BigTable: A Distributed Storage System for Structured Data," OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, Nov. 2006.

[7] A. Reuther, B. Arcand, T. Currie, A. Funk, J. Kepner, M. Hubbell, A. McCabe, P. Michaleas, "TX-2500 – An Interactive, On-Demand Rapid-Prototyping HPC System," HPEC 2007, Lexington, MA, Sep. 2009.