Using Grid Computing to Enable Distributed Radar Data Retrieval and Processing *

Diego Arias and Wilson Rivera
Parallel and Distributed Computing Laboratory
Electrical and Computer Engineering Department
University of Puerto Rico at Mayagüez, Puerto Rico 00680, USA

Abstract

We describe a tool that implements a set of services to manipulate and store data from a radar network in a transparent way to end users. A major requirement of this system is data availability and reliability. Consequently, we have implemented a redundancy schema based on the Information Dispersal Algorithm (IDA). Preliminary results show that the IDA based replication provides better reliability and less storage spending than traditional replication.

1. Introduction

The National Science Foundation Engineering Research Center for Collaborative Adaptive Sensing of the Atmosphere (CASA) is focused on developing Distributed Collaborative Adaptive Sensing (DCAS) [1] as a systems technology to improve our ability to monitor the earth's lower atmosphere. Current approaches to sampling the first three kilometers of atmosphere are physically limited in their ability to provide the required resolution and coverage. For example, radar technology is currently limited by the focus on long range sensing by single instruments. Requiring radar to view distances up to 240km, as in the case of NEXRAD [2], introduces the problem of the earth's curvature [3]. As the range increases away from the radar, the earth's surface curves away under the radar beam. This causes the volume of atmosphere being observed to be located at an increasing height above the earth's surface. The radar is unable to observe the atmosphere close to the earth's surface where people live. DCAS aims to radically alter the radar paradigm. Rather than relying on single radar to provide long range (hundreds of kilometers) coverage, DCAS proposes to mosaic the output of lower power shorter range (tens of kilometers) radars.

It must be acknowledged that reducing the range would require an increase in the number of radars to cover the same land area. By directly comparing areas, reducing the maximum required range from 240 km to 30 km would require approximately 64 short range radars to cover the area of the single long range radar. While this may appear to detract from the DCAS argument an analogy may be made with the field of computing. Recent years have shown the utility in using many commodity computers networked to form a larger system in the place of a single more expensive, larger system. The act of networking many inexpensive radars to cover the same area as a single high power radar introduces new capabilities into the system, such as fault tolerance and adaptability of the network sensing strategy, which the larger systems are currently not capable of.

A parallel development in the technology landscape is grid computing [4], which involves coordination, storage and networking of resources across dynamic and geographically dispersed organizations in a transparent way for users. The Open Grid Services Architecture (OGSA) [5], based upon standard Internet protocols, is becoming a standard platform for grid services and application development. The integration of grid computing and radar network technologies enables the complementary strengths of these technologies to be realized in an integrated platform. However, it poses several challenges such as the need to comply with emerging APIs for grid and Web services, the coordination of communication, and the requirement of a more data-centric

* This material is based upon work supported by the National Science Foundation under Grants No. 0313747 and No. 0424546. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

infrastructure focused on distributed services. In this paper, we look at the technical problems of integrating radar data into grid architectures and present a grid service based infrastructure to transport, manipulate and store data from different radars, while preserving data integrity.

The organization of this paper is as follows. In section 2, the grid integration approach of the radar network to a grid infrastructure is discussed. In section 3, the problem of radar availability and reliability is discussed and solutions to perform data replication over the grid infrastructure are presented. Experimental results are presented in Section 4. Finally, conclusions are listed in section 5.

2. Grid Implementation

The PDClab Grid Testbed, deployed at the University of Puerto Rico-Mayaguez, is an experimental grid designed to address research issues, such as the effective integration of sensor and radar networks to grid infrastructures. The PDClab grid test-bed components run CentOS 4.2 and the Globus Toolkit 4.0.1[1].

The resources available include:

- An IBM xSeries Linux cluster with 64 nodes, dual-processor at 1.2GHz, 53GB of memory and 1TB of storage.
- Eight (8) IA-64 Itanium servers, dual processor at 900 MHz, each with 8GB of memory and 140GB of SCSI Ultra 320 storage
- Two (2) IA-32 Pentium IV servers, dual processor at 3 GHz, each with 1GB of memory and 120GB of ATA-100 storage
- One (1) IA-32 Pentium III server, dual processor at 1.2 GHz with 2GB of memory and 40Gb of SCSI Ultra 160 storage

For the design and development of a grid-service based system to access and manipulate radar data, the initial approach considered is a grid based system which includes a Grid Portal Interface developed using Gridsphere[2] based portlet framework, a distributed storage system to radar data management, and Grid services implementing distributed algorithms. The Grid Portal Interface provides transparent access to end-users. This interface allows to end users the manipulation of both, processed and raw radar data, as well as visualization of weather information, such as reflectivity in order to estimate rainfall rate over the west area of Puerto Rico.

Raw data from radars are sent to a data server via wireless communication. GridFTP[3] is used to improve data transport from the data server to the PDCLab Grid Testbed (see figure 1). Data exchange between server and the Grid testbed is authenticated using Grid Security Infrastructure (GSI). Preliminary tests to transport data using the *globus-url-copy* client and the *gsiftp* protocol, have been successful.

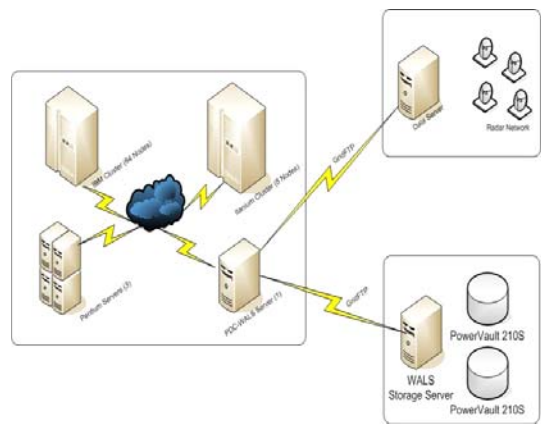


Figure 1:Grid Testbed and Radar Integration

Data files are dispersed using the Information Dispersal Algorithm, explained in the next section, with a redundancy level of 100%. At the same time these file blocks are sent to the grid testbed using gsiftp protocol, in a 1:1 distribution, meaning a block of file per node. Original files are erased from the server to safe storage resources and the data remains distributed in the grid. A log file is preserved in the server to register the scans per day. Relevant information about the distributed files is also preserved in the server. When an end-user enters the portal, a single selection form is displayed to allow the user choose the interest date. After selection of the data set, the client can

[1] http://www.globos.org
[2] http://www.gridsphere.org

[3] http://www.globus.org/toolkit/docs/4.0/data/gridftp

request for specific scans, at this moment the data still remains in the grid.

3. Radar Data Availability and Reliability

Implementation of redundancy schemas is a common strategy to enhance reliability in data storing [7]. Two different redundancy strategies have been implemented and analyzed: A simple replication schema [8] and the Information Dispersal Algorithm (IDA) [9].

3.1 Information Dispersal Algorithm

The information dispersal algorithm (IDA) was proposed as a fault-tolerance technique to be used in secure and reliable storage systems. In the basic approach, a file F is striped into n blocks of size $|F|/m$, where |F| is the size of the file and m is the number of blocks required to recovery the file F. A set of secret keys are used to disperse the file, providing confidentiality to the information. Since $m \leq n$, the redundancy level given by (n/m-1)%, can be selected to be smaller than replication technique. The storage spending is $|F|*(n/m)$. An important feature of this technique is that any m blocks will reconstruct the file and labels are not necessary for each block. Additionally IDA tolerates up to r failures, where $r = n - m$. Hence, IDA guarantees a higher availability.

Let F= $b_1, b_2, b_3, \ldots$ be a file, where b_i is an integer taken from a certain range $[0 \ldots (2^B -1)]$. If b_i is two bytes long, as in the actual implementation, then $0 \leq b_i \leq 65535$. Let p be a prime number greater than b_i. Each b_i is an element of the finite field Z_p where all arithmetic operations are done in mod p. Since $p > (2^B -1)$, this implies an excess of one bit per byte when integers greater than $(2^B -1)$ are obtained, this requires a storage space increment. In order to avoid the space waste, all b_i values are represented as polynomials with binary coefficients ($b_B x^B + b_{(B-1)} x^{(B-1)} + \ldots + b_1 x + b_0$) and use a larger degree irreducible polynomial $p(x)$ instead of the prime p [10]. The polynomial must suffice ($p(x) \in Z_2[x]$) in such a way that all operations can be done in the finite field E=GF(2^B). GF refers to "Galois Field".

In order to disperse F, a set of n vectors $a_1, a_2, a_3, \ldots, a_n \in E$ must be chosen, each of length m, such that every subset of m different vectors is linearly independent. These vectors are the keys that will be used to disperse every block of the file.

Let A_{nxm} be a matrix whose i^{th} row is a_i. The file is divided into sequences of length m ($b_1, b_2, b_3, \ldots, b_m$) and the dispersal operation is achieved mapping each sequence b_j into a new sequence of n elements using A_{nxm}.

$$A_{nxm} \cdot \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Each resulting element c_i is stored in a separated block of file.

In order to reconstruct the file, m blocks are required ($s_1, s_2, s_3, \ldots, s_m$) and the recovery operation is performed as follows: let B_{mxm} be a matrix whose rows are ($a_{s1}, a_{s2}, a_{s3}, \ldots, a_{sm}$). To recover the first m elements of F, the first element from each different block is needed. The whole file is obtained mapping sequences of m elements from each block into sequences of m elements using the inverse of B_{mxm}.

$$B_{mxm}^{-1} \cdot \begin{bmatrix} c_1 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Note that the inverse of the B_{mxm} matrix is guaranteed to exist since the rows of matrix A are mutually independent, which implies that any submatrix (in this case B_{mxm}) is not singular and thus invertible by deleting m rows of A_{nxm},.

An A_{nxm} matrix with the properties above mentioned is the Vandermonde matrix. The i^{th} row of this matrix is defined as

$$i^0, i^1, i^2, i^3, \ldots, i^{n-1}.$$

By definition, this matrix has the property that any submatrix formed by deleting m rows of it, is invertible. Additionally any matrix derived from this matrix by a sequence of elementary matrix transformations, will maintain this property [11].

Finally, an irreducible polynomial must be chosen. For the current implementation the polynomial p(x) of degree B over GF(2^B), when B = 16 is

$$p(x) = x^{16} + x^{12} + x^3 + x + 1.$$

The implementation of the IDA involves several operations over finite fields. In this case over $GF(2^{16})$. IDA is implemented as follows:

(1) Create the dispersal matrix A *n*x*m* which must obey the properties described before.

(2) Divide the file F into sequences of *m* elements, where each element is 2 bytes of size. Note that |F| must be divisible by *m*, therefore, padding must be added. In order to disperse the file, each sequence is multiplied by the matrix A to obtain the new sequences. The first block will have the 1^{st} element from the each new sequence. The second block will have the 2^{nd} element from that sequence and so on.

(3) A unique tag for each block must be established before these are written as separated files. This tag corresponds to the i^{th} row of the matrix A. this tag is necessary to choose the correct recovery B matrix.

(4) After the tagged files are ready, they must be distributed in *n* nodes or according to the established data distribution strategy. The two first bytes of each file are used to identify the correspondent row. Thus a maximum of 2^{16} blocks are permitted. The complete path of these files will be registered in a log file.

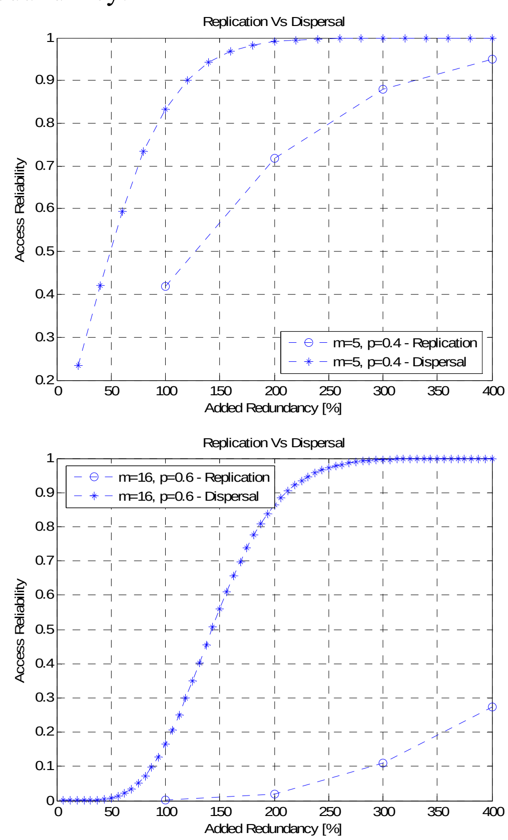
(5) In order to recover the file F, the existence of at least *m* blocks must be verified; this condition is necessary and sufficient to achieve the recovery operation. The two first bytes of each file are read to identify the row of the matrix A. The algorithm chooses the first *m* files and creates the recovery matrix B with the rows found. Then the inverse of the B matrix is calculated.

(6) Reconstruct the first sequence of *m* elements from the original file multiplying the matrix B^{-1} by the sequence formed by all the first elements from each file found. Similarly, the second sequence from the original file is obtained transforming the sequence consisting of the all second elements from each file and so on.

(7) Finally, padding must be removed, if necessary, to obtain the original size of the file.

4 Experimental Results

A set of experiments have been carried out with the aim to compare these algorithms and this way, determine the advantages and the disadvantages of each one. For our performance analysis we consider the total number of blocks after applying redundancy (TB), the size of each block (BS) and the added redundancy (AR) as parameters and measure the access reliability (R). In each case the storage spending (SS) required to perform redundancy.



a)

b)

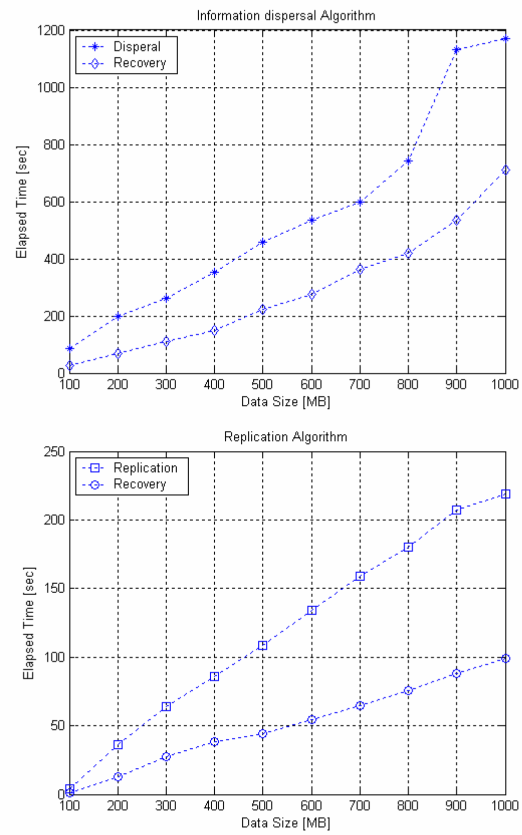
Figure 3: Reliability vs Added Redundancy comparison. a) *m*=5, *p*=0.4, b) *m*=10, *p*=0.6

Information dispersal algorithm shows a better access reliability than replication algorithm. As a reference point, for an access reliability $R = 0.9$ when the probability of failure is $p = 0.4$, $m = 5$, the added redundancy for IDA is $AR = 120$ %, while in the replication approach the added redundancy must be approximately AR ≈ 300 % (Figure 3(a)). Note that, for replication algorithm, AR increment is every 100%, because the redundancy is performed using multiplication with integer numbers. Figure 3(b) shows the behavior of the algorithms when the probability p = 0.6 and m = 16. The reliability of replication approach is quite deficient if the probability failure increments.

Note that, as shown in Figure 3, the reliability for IDA is improved when m is incremented compensating a higher probability of failure. However, the reliability for replication is downgraded if the number of blocks is incremented and is worse still if p is higher. In contrast, a higher number of m involves a even higher number of total blocks (TB) and a reduction in the block size (BS). A small BS can be desirable to obtain weightless blocks to send them over a loaded network. In turn a higher TB involves a higher number of nodes, if the node-block relationship is 1:1. Redundancy is an important feature to be taking into account when radar data must be manipulated, because the size of this data is usually large. Therefore, a proper redundancy must be selected to avoid storage overhead.

Experiments involving time measurements vs. data size, take as reference data size from National Climatic Data Center. This data is a Level II base data [12], available in compressed tape archive format. It contains data per day from specific NEXRAD Level II radar. The compressed data for a day is about 150 MB, while uncompressed is about 2.3GB. Note that this is the data mount for 24 hours of continuous scan. For rain fall measurements and precipitation estimation, the primary implementation of DCAS network requires less than 8 hours of continuous scan. Considering all the exposed before and the limited transmission due to wireless communication as mentioned, testing is achieved with data size range from 100MB to 1GB.

In order to improve elapsed time measurements a comparison point is established. Suppose a minimum access reliability of 90 %. If p=0.4, is required to provide data availability in the DCAS network, an access reliability R ≥ 0.9 can be obtained with m=8, r=5 (AR = 400% and R=0.921) for replication algorithm. Similarly, for IDA if R ≥ 0.9, m= 8, r=10 (AR= 125 % and R=0.942). Even though the added redundancy is lower for IDA than Replication Algorithm, the elapsed time required to complete dispersal and recovery operations in IDA is significantly higher than the replication approach. Figure 4 shows a comparison between dispersal and recovery operations for both algorithms with several data sizes.



a)

b)

Figure 4: Data Size vs. Elapsed Time comparison. a)IDA, b)Replication

As is shown in figure 4(b), the replication algorithm is a lot faster than IDA in both replication and recovery operations. Note that, when a file of size 1GB is required to be distributed, IDA takes long about 20 minutes and replication algorithm only takes 3.5 minutes.

5. Conclusions

Implementations of two redundancy schemas to perform radar data management are presented. The reliability was the metric selected since it is an important parameter in the DCAS systems. At this stage, Information dispersal algorithm shows a better data reliability than replication algorithm with less storage spending. However, this desirable behavior has a computational cost which implies higher response times than replication technique. Enhance time execution of the schemas is a current effort, which is focused in the replacement of the standard by high performance libraries.

Radar system integration with grid computing technologies has been discussed as well. Preliminary results demonstrate the feasibility of such interaction, when independent and non grid based applications can be integrated to the grid infrastructure with minimum requirements. The tested applications were data management related, especially data movement. A large amount of data was transported using GridFTP protocol with GSI support, and the integrity of the data was preserved successfully.

References

1. McLaughlin, D., J. Brotzge, V. Chandresakar, K. Droegemeier, J. Kurose, B. Philips, M. Preston, and S. Sekelsky, *2004: Distributed Collaborative Adaptive Sensing for Hazardous Weather Detection, Tracking, and Predicting.* International Conference on Computational Science 2004.
2. National Research Council, Committee on Weather Radar Technology Beyond NEXRAD, *Weather Radar Technology Beyond NEXRAD*, National Academy Press, Washingtion D.C., 2002.
3. Doviak, Richard J. and Zrnić, Dušan S., *Doppler Radar and Weather Observations*, 2nd ed., Academic Press, San Diego, 1993.
4. Foster I. and Kesselman C., *The grid: blueprint for a future computing infrastructure*, Morgan Kaufmann Publishers, 1998.
5. Foster I., Kesselman C., Nick J., and Tuecke S., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, June 2002.
6. Díaz, P. L., Aquino, Z., Figueroa-Alamo, C., García, R., and Sánchez, A. V., “Water Resources Data Puerto Rico and The U.S. Virgin Islands Water Year 2001”, USGS, Water-Data Report PR-01-1.
7. Wylie, J. J., Bigrigg, M.W., Strunk, J. D., Ganger, G. R., Kiliççöte, H., and Khosla, P. K., “Survivable information storage system”, *IEEE Computer*, 33(8):61–68, August 2000.
8. Santos, J. R., Muntz, R. R., and Ribeiro-Neto, “Comparing random data allocation and data striping in multimedia servers”, *SIGMETRICS Perform. Eval. Rev.* 28, 1 (Jun. 2000), 44-55.
9. Rabin, M. O., “Efficient dispersal of information for security, load balancing, and fault tolerance*”, Journal ACM.* 36, 2 (Apr. 1989), 335-348.
10. Bestavros, A., “SETH: A VLSI chip for the real-time information dispersal and retrieval for security and fault-tolerance”, In *Proceedings of ICPP'90, The 1990 International Conference on Parallel Processing*, Chicago, Illinois, August 1990.
11. Plank, J. S., “A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems”, *Software-Practice and Experience (SPE)*, 27(9):995.1012, Sept. 1997. Correction in James S. Plank and Ying Ding, Technical Report UT-CS-03-504, U. Tennessee, 2003.
12. National Climatic Data Center, “Data documentation for DSI-6500 NEXRAD Level II”, *National Climatic Data Center*, Asheville N.C., April 11, 2005.