Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Metadata Miner Assisted Integrated Information Retrieval for Argo Ocean Data

Yue Shan Chang
Dept. of Comp. Sci. & Inf. Eng.
National Taipei University
Taipei County, Taiwan
ysc@mail.ntpu.edu.tw

Hsiang-Tai Cheng
Dept. of Comp. Sci. & Inf. Eng.
National Taipei University
Taipei County, Taiwan
edisoncheng3918@gmail.com

Hsuan-Jen Lai
Institute of Comm. Eng.
National Taipei University
Taipei County, Taiwan
shan_717@msn.com

***Abstract*—Argo project is an international ocean-observatory project that has a global array of 3,000 more free-drifting profiling floats. Argo data is a large collection of data files. To retrieve Argo data from the large database is a complicated work even if the Argo database has a metadata file per day for helping user to inquire the target files. This paper proposes a Metadata Miner (MM) approach to assist user program to inquire the target files quickly. In addition, we also propose an integrated information retrieval framework that based on mediator/wrapper approach to smoothly tie the MM. According to the performance evaluation, it shows that the MM approach has a significant performance enhancement in finding the target file[1].**

***Keywords*—Metadata. Miner, Integrated Information Retrieval, Argo, Ocean data**

I. INTRODUCTION

The Argo [2] project is an international ocean-observatory project that has a global array of 3,000 more free-drifting profiling floats. These floats measure the temperature and salinity of the upper 2000 m of oceans. This allows, for the first time ever, continuous monitoring of the temperature, salinity [2], and velocity of the upper ocean. All Argo data are publicly available in nearly real-time via the Global Data Assembly Centers (GDACs)[3] in Brest, France and Monterey, California after an automated quality control (QC), and in scientifically quality controlled form, delayed mode data, via the GDACs within six months of collection. Each Argo's float will pump fluid into an external bladder and rise to the surface over about 6 hours while measuring temperature and salinity at typically 10-day intervals. Satellites determine the position of the floats when they surface, and receive the data transmitted by the floats.

All of Argo data can be mirrored to your own server from GDAC. These data are organized in a tree-hierarchy file structure, as shown in Figure 1. This is a large collection of data files. The second layer comprises of the directories of three oceans. Each directory has many subdirectories named in years. The bottom layer is a directory and consists of a metadata file and some files named by date and float ID. It is obviously that GDAC will create a directory per day. The file name may help oceanographer to identify the date that data received and the float ID. The metadata file contains the file name in the directory and their position and maximum pressure. The detail will be presented in Section 3.

While an oceanographer wants to analyze the ocean data, he can easily retrieve what he desired from these files. Even though, it is also a complicated work if we want to retrieve data for a long period and a larger geographical range.

Here we give an example that is in a SQL-like query to show what we want. For example, "SELECT TEMPERATURE, SALINITY WHERE -45<LONGITUDE<45 AND 30<LATITUDE<45 AND 20081001<DATE<20081130". In this example we would like to retrieve temperature and salinity data from those floats located in a larger geographical range and during a long period.

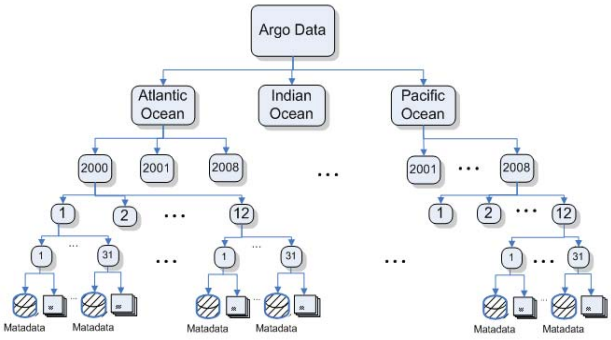

Figure 1: Argo data file hierarchy

This example we need to perform two tasks. The first task is target file discovering, we need to access the metadata file to find out the file name that meet the criteria of the retrieval. It is a complicated work because the task need to open and look into those metadata files from date 20081001 to 20081130, and find out those floats located in desired geographical range. The second task, we need to open those file met the criteria and retrieve the temperature and salinity data. Obviously, the first task is not an easy work.

The paper will present a MM approach to improve the

[1] This work was supported by National Science Council of the Taiwan, Republic of China under Grant No. NSC97-2221-E-305-004
[2] http://www.argo.ucsd.edu/
[3] http://www.pac.dfo-mpo.gc.ca/sci/OSAP/projects/argo/driver_e.htm

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

performance of first task. The approach will improve the discovering latency of the target files. In the approach, we first design and implement a MM to mining useful data from the metadata stored in each directory and to stored the mined data into a two dimension array. User application can inquire the MM to get the target file directly without looking up all the metadata files stored in each directory one-by-one. The MM can be applied to various integrated information retrieval, such as [5][6], for retrieving desired Argo data. Even though, in this paper we also propose an integrated information retrieval framework that based on mediator/wrapper approach to smoothly tie the MM in our project. We also evaluate the performance of the approach and make a comparison with a raw approach and with other integrated Argo information retrieval platforms.

The remainder of the paper is organized as follows. Section II surveys related works. Section III introduces the Argo database structure that involves the content of metadata file and Argo floats. Section IV presents the approach and algorithm of MM. Section V evaluates the performance of proposed approach and makes a comparison with a raw approach. Finally, we give conclusion in Section VI.

II. Related Work

Here we survey some related literatures.

Teng et al.[1] designed a platform architecture to deal with Argo data retrieving and analyzing via internet. Visualization based GIS and massive data management based on Oracle are also introduced in this paper.

In addition, there are other websites, such as USGODAE Monterey site[3] and Coriolis site[4], also provide easily a web interface for accessing the Argo data. USGODAE Monterey provides user interface and a set of display tools including a profile location plot for all profiles returned by the query (may be plotted with or without float ID for queries returning many profiles), Download of selected profiles (in NetCDF Multi-Profile format) as a TAR file, Plots of T-P and S-P for individual profiles, and Plots of float tracks for individual floats, etc.

Coriolis site[4] also provides user friendly interface and a set of display tools. The user interface with a subsetting tool allows selection by profile type, time and lat/long windows, measured parameter, platform type, and realtime or delayed mode QC data, etc.

Song et al. [5] presented a Grid-enabled Information Integration System based on mediator/wrapper, called *Dynamic Mediation Service Management System* (DMSMS). DMSMS searches data sources needed by users at run-time through *Data Source Registry*, and provides a GUI-based *Mediation Schema Designer* tool to help users to create mediation schemas easily and conveniently. The system also provides a mediation service over virtual data sources (mediation schemas) and a notification service to actively deal with the changes of data sources.

III. Argo Database Structure

The Argo database structure is as shown in Figure 1. This is a large collection of data files. The second layer comprises of the directories of three oceans. Each directory has many subdirectories named in years. The bottom layer forms a directory that comprises of a metadata file and some files regarding to Argo floats. These files were created according to the date of transmitting data to GDAC and float ID.

The metadata file contains the filename of floats, longitude, latitude, and maximum pressure, as shown in Figure 2. In the metadata file, a record represents a float file. For example, a record "20081001_19000922.ios, 34.135, 28.12, 255" means that the float 19000922 rise to surface and transmit sensed data back to GDAC on the date 2008/10/1 at the location longitude equals to 36.088 and latitude equals to 18.465. The 20081001_19000922.ios is the file name of float data. The metadata can help oceanographer to find the float's file at a certain position. The Figure 2 also shows partial of the content of a float file. The content of the file can help oceanographer to analyze the ocean's ecology by means of temperature, salinity and other information.

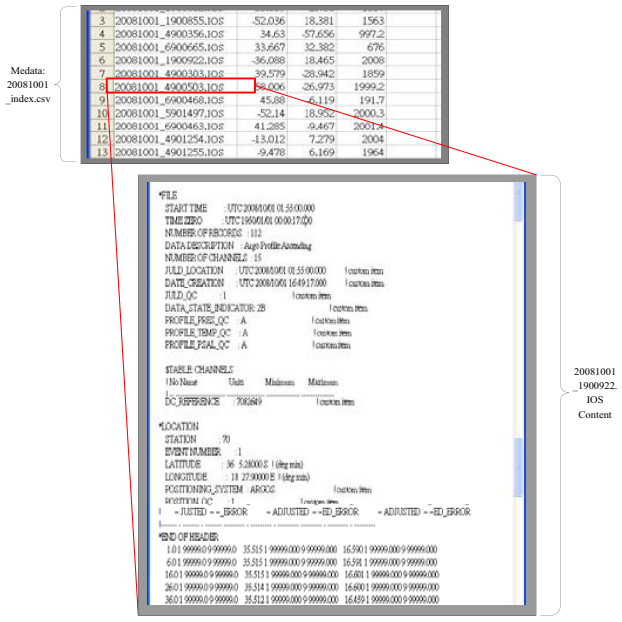

Figure 2: The content of metadata each day

Although the metadata file can help oceanographer to find out the float's file at a certain position. The oceanographer will open and look into up to 61 metadata files while the example mentioned in the Section 1 used, and compare with 61*n records to find out all of target files for analyzing if each directory has n records. This will be a tedious work. A raw approach is to implement a program to look into the metadata one-by-one for searching the target files.

IV. Metadata Miner

This paper is mainly designing a MM to classify the data in the metadata. Here, in order to demonstrate the example, we only do classification according to the position of float, such as

longitude and latitude. We believe that the approach can be applied to other attributes. The basic idea of the MM is to classify the float's file according to float's position (longitude and latitude) every day and create a month-level metadata per month, as shown in Figure 3. In other words, we create another metadata in the upper level. The pull-up metadata can improve the retrieving performance.

The metadata is constructed as a classification grid that is a two dimensions array, as shown in Figure 4. The idea is that the MM executes classification process for each metadata file every day in a month. The classification is to retrieve the float's file name according to the position stored in a record and save the filename to relevant grid. The filename saved in the same grid represents that these floats respond their data in a certain geographic range.

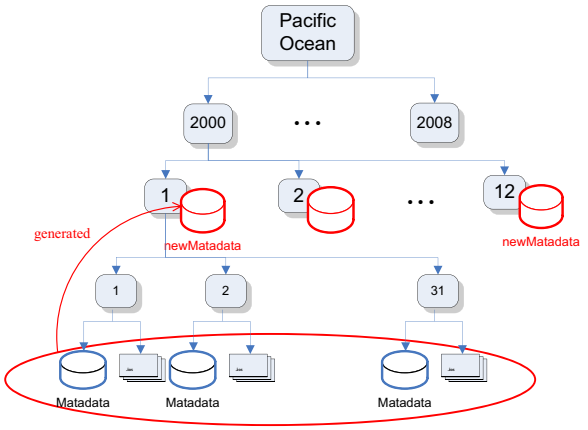

Figure 3: metadata pull-up

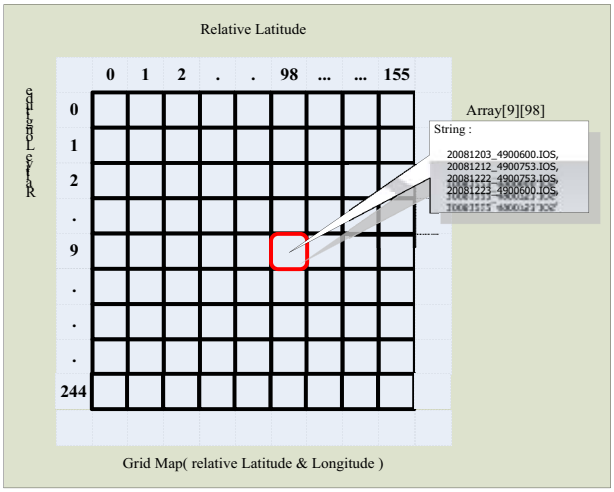

Figure 4: The classification grid of MM

Obviously, user application can get the float's filename only inquiring the month-level metadata directly. The inquiry process is simply and directly because we adopt a simple mapping scheme according to the location of floats.

A. *Mapping method*

The subsection presents the simple mapping scheme used in the MM. First of all, the system needs to decide the granularity of a grid, which will affect the size of grid map. The smaller granularity will obtain larger grid size. In the Figure 4, the granularity of the two dimension array is set to 1 degree (longitude) by 1 degree (latitude). In addition, we need to compute the range of longitude and latitude in a month. It is used for getting the bias of the two dimension array. The steps of the scheme are as follows:

1) MM will scan each metadata file of a month and compute the range of longitude and latitude. We first define some terms used in the step.

$long^{i}_{\max}$: *The maximum longitude of the month i.*

$long^{i}_{\min}$: *The minimum longitude of the month i.*

$lat^{i}_{\max}$: *The maximum latitude of the month i.*

$lat^{i}_{\min}$: *The minimum latitude of the month i.*

These variables can be obtained by as following formulas:

$$long^{i}_{\max} = \max\left\{long^{i,1}_{\max}, long^{i,2}_{\max}, \cdots, long^{i,last\_day}_{\max}\right\}$$

$$long^{i}_{\min} = \max\left\{long^{i,1}_{\min}, long^{i,2}_{\min}, \cdots, long^{i,last\_day}_{\min}\right\}$$

$$lat^{i}_{\max} = \max\left\{lat^{i,1}_{\max}, lat^{i,2}_{\max}, \cdots, lat^{i,last\_day}_{\max}\right\}$$

$$lat^{i}_{\min} = \max\left\{lat^{i,1}_{\min}, lat^{i,2}_{\min}, \cdots, lat^{i,last\_day}_{\min}\right\}$$

where :

$long^{i,j}_{\max}$: *The maximum longitude of the day j in the month i.*

$long^{i,j}_{\min}$: *The minimum longitude of the day j in the month i.*

$lat^{i,j}_{\max}$: *The maximum latitude of the day j in the month i.*

$lat^{i,j}_{\min}$: *The minimum latitude of the day j in the month i.*

2) According to the terms defined above, MM can compute the bias of longitude and latitude. The bias can be used for computing the size of the two dimension (grid map) , as shown in Figure 4, and obtaining the shift value in the grid map. The bias of longitude and latitude can be defined as follows:

$$longBias_i = long^{i}_{\min} - 1$$

$$latBias_i = lat^{i}_{\min} - 1$$

3) The bias of longitude and latitude can be used for declaring the two dimension array of grid map. The array size is as follow:

$$gridMap[long_{\max}^{month} - longBias][lat_{\max}^{month} - latBias]$$

4) The final is that the MM search all of metadata file of the month, retrieve each record in metadata file, and then put the filename of a float into the related grid. Next subsection will introduce the mining algorithm in detail.

In the Figure 4, it is possible that multiple filenames are put into a grid. Here we cascade all filename as a string. The two dimension array can be saved as a file for futher searching.

B. Miner Algorithm

The Argo data files are increasingly. The first time MM will search all of metadata files in the Argo database. While the up layer metadata created, the MM can only search that the new added metadata file. The original Mining algorithm is shown in the Figure 5.

```
function Miner(){
    For(int ocean=0,ocean<3,ocean++){
        For(int year=2000;year<nowYear;year++){
            For(int month=1;month<12;month++){
                For(int day=1;day<monthday[month];day++){
                    Read metadata each day into String as
                    fileNameString;
                    count++;
                }

                maxLong = Max(all Longitude data this month);
                maxLat = Max(all Latitude data this month);
                biasLong = Min(all Longitude data this month);
                biasLat = Min(all Latitude data this month);

                Create a gridMap[maxLong-biasLong][maxLat-biasLat];

                For(int i=1 to count)
                {
                    gridMap[Longitude- biasLong][Latitude-biasLat]
                    =fileName;
                }

                Write gridMap array into a file;

            }
        }
    }
}
```

Figure 5: Mining Algorithm

V. DATA RETRIEVING & PERFORMANCE MEASUREMENT

The grid map can improve the performance of Argo data retrieving. This section will show the new Argo data algorithm and give an example to depict the use of the algorithm and then evaluate the performance of the retrieving process.

A. Data Retrieving Algorithm

Data retrieving for a long period and a larger geographical range is a complicated work. This section will show two data retrieving algorithms to retrieve Argo data. The one is based on the original Argo database and its metadata file named *RawQuery* algorithm, the other is based on our proposed Metadata Miner approach named *MMQuery* algorithm. The Figure 6 lists the pseudo code of the *RawQuery* algorithm. The algorithm will inquire each metadata file of Argo database to compare and get the target file.

```
Function RawQuery(String queryString){
    Record the range of longitude,latitude;
    Named high_longitude,low_longitude,
            high_latitude,low_longitude;
    Record beyond "where" string;
    Named start_date,end_date;
    Record beyond "where" string as conditionString;
    For(start_date : end_date){
        Open day_metadata;
        Readline metadata;
        If(low_latitude<latitude<highlatitude &

                low_longitude<longitude<highlongitude ){
            Open fileString+=filename;
            file_count++;
    }}
    For(1:file_count++){
        If(open fileString != null){
            Open file;
                if(match the conditionString){
                    Print result;
        }}}}
```

Figure 6: The RawQuery algorithm

Figure 7 lists the *MMQuery* algorithm. The algorithm does not inquire all of metadata file instead of inquiring the new generated metadata (grid map) with a desired geographical range. The algorithm can easily compute the longitude bias and latitude bias, and then map the location of grid map according to the two biases. Finally, the MM can get directly the target file list from the grid map. The map function is as follows:

$$gridMap[long - longBias][lat - latBias]$$

```
Function MMQuery(String queryString){
    Record the range of longitude,latitude;
    Named high_longitude,low_longitude,
            high_latitude,low_laitude;
    Record beyond "where" string;
    Named start_date,end_date;
    Record beyond "where" string as conditionString;
    For(start_date : end_date){
        Open month_arraymetadata;
        For(low_longitude:high_longitude){
            For(low_laitude:high_latitude){
                openfile+= month_gridMap[longtitude-
                longBias][latitude-latBias];
        } }
        while(openfile != null ){
            open openfile;
}}}
```

Figure 7: MMQuery algorithm

Here shows an example to demonstrate the operations of these two algorithms. For example, an oceanographer wants to collect the temperature and pressure data. The query string may be as follow:

"SELECT temp, salinity FROM -20<longitude<20,-10< latitude <10 WHERE 20080101<date<20090131"

The RawQuery algorithm first needs to look for the metadata that stored in the directory from 2008/01/01 to 2009/01/02 every ocean. Figure 8 shows the operations of

RawQuery algorithm for one ocean. It is obvious that there are approximately 400 metadata files in one ocean need to look for. When the algorithm opens one metadata, it will compare the *Longitude* and *Latitude* for fitting the criteria of query string. If it is satisfied, the algorithm will keep the file name, and finally respond to the user. The algorithm will execute about 389(days)x3(ocean)xN(records) comparisons, where N is the number of records in a metadata file.

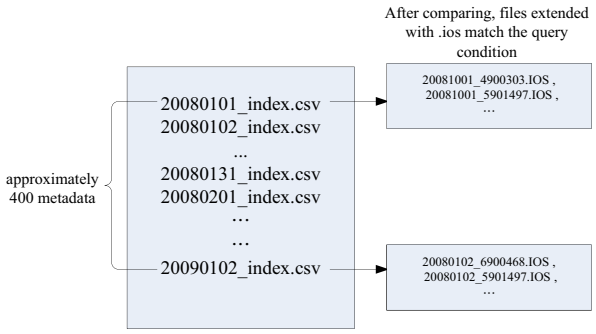

Figure 8: The operation of RawQuery algorithm for one ocean

The MMQuery algorithm only needs to access the grid map of each month that created by MM, as shown in Figure 9. The MMQuery only access the certain range in the grid map and get the cascaded filename string. Here the MM only inquires 13 new created monthly metadata to retrieve all of target filename. The operation is obviously simple and easy. If the grid map can be pulled up to yearly level directory, I believe that the performance of query process will be further improved.

B. Performance Measurement

This subsection examines the performance evaluation. We implemented the two algorithms: RawQuery and MMQuery respectively. These two algorithms were run on a notebook that with an Intel Core 2 Due 1.4GHz CPU and 1GB RAM.

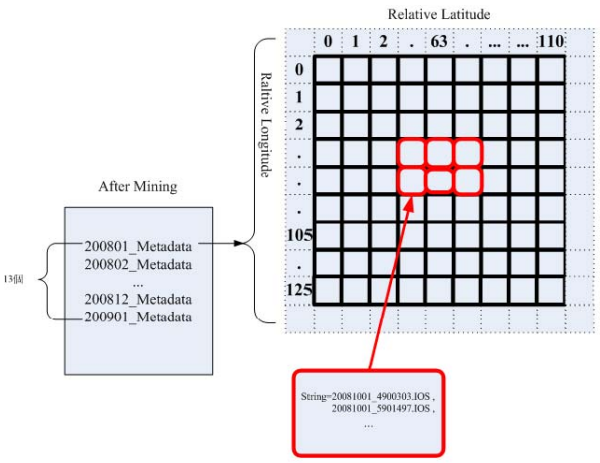

Figure 9: The operation of MMQuery algorithm for one ocean one month.

Figure 10 shows the result of measurement. The MMQuery algorithm takes only about 50ms for each round of measurement. The time only slightly increases with the number of days increased. The reason is that the MMQuery only takes a bit of time to compute the range of grid map and to retrieve the target filename from cascaded file string of grid map. The latency of RawQuery algorithm is obviously increasing with the number of days increased. The reason is that the more number of days will be inquired, the more metadata file will be accessed. Since the time complexity of the RawQuery is O(n) and of the MMQuery is O(1). The times of latency on RawQuery over MMQuery is about 3~30 while the number of days is 10~60. It shows that the MM approach has a significant performance enhancement in finding the target file.

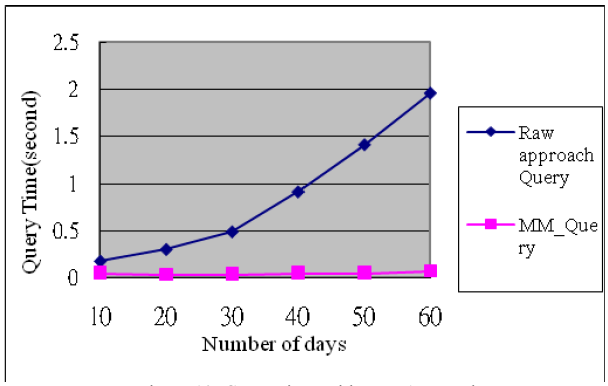

Figure 10: Comparison with Raw Approach.

In addition, we also, based on mediator/wrapper architecture, implement an integrated information retrieval platform that tie a MM to demonstrate the flexibility, as shown in Figure 11. The system allows user inquiring the target file by entering SQL syntax as shown previous subsection. This is a progressive prototype that only can show the target file name in the current state. But it is enough to compare the query performance with other system, such as USGODAE.

Figure 12 shows the performance comparison between our system and USGODAE. The result shows that our system is superior to USGODAE in terms of query time about 4~6times. According to these performance evaluations, it has shown that the MM approach has a significant performance enhancement in finding the target file.

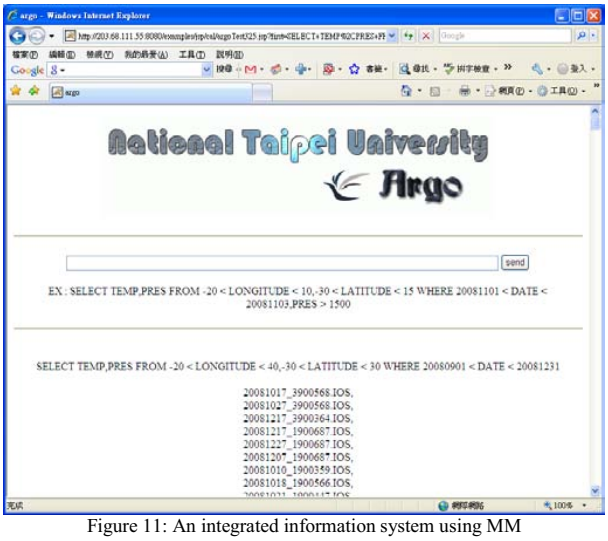

Figure 11: An integrated information system using MM

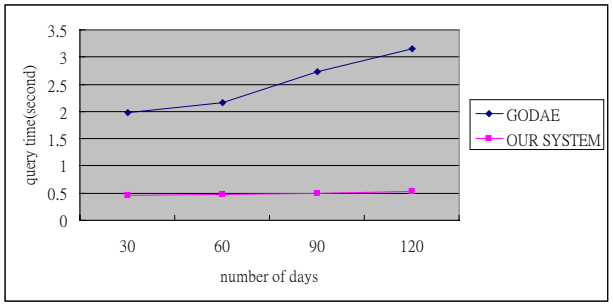

Figure 12: Performance comparison with GODAE

VI. Conclusion and Future Work

In this paper we have proposed a Metadata Miner (MM) approach to assist user program quickly to inquire the target files of Argo database. The MM can mine useful data from the metadata stored in each directory and store the mined data into a two dimension array. User application can inquire the MM to get the target file directly without looking up all the metadata files stored in each directory one-by-one. The algorithm is useful for retrieving data from Argo floats. In addition, we also propose an integrated information retrieval framework that based on mediator/wrapper approach to smoothly tie the MM. According to the performance evaluation, it shows that the MM approach has a significant performance enhancement in finding the target file.

In the future, we will implement the algorithm into the grid platform and environment, such as the Globus toolkit, to practically take the advantage of grid computing to the Argo project.

References

[1] J. Teng, Z. Liu, M. Sun, C. Sun,J. Xu "Development of Online Argo Data Service Platform Based on GIS," IEEE International Conference on Geoscience and Remote Sensing Symposium, 2006. IGARSS 2006. July 31 2006-Aug. 4 2006, pp. 1316 – 1318.

[2] Y.-P. Huang, L.-J. Kao and F.E. Sandnes, "Efficient mining of salinity and temperature association rules from ARGO data," Expert Systems with Applications, vol. 35, no. 1-2, , Aug. 2008, pp.59-68.

[3] http://www.usgodae.org/cgi-bin/argo_select.pl

[4] http://www.coriolis.eu.org/cdc/argo_rfc.htm

[5] J. Song, S. Yoo, C.-S. Park, D.-H. Choi. Y.-J. Lee, "The Design of a Grid-enabled Information Integration System Based on Mediator/Wrapper Architectures," 2006 International conference on Grid Computing & Application, Las Vegas, USA, pp.114-120.

[6] A. Wohrer, P. Brezany, A. M. Tjoa, "Novel Mediator architecture for Grid information systems," Future Generation Computer Systems, Vol. 21(2005), pp.107-114.