RESEARCH PAPER

Seismic Data Server Application Service for SEEGRID Seismology Virtual Organization

Can Özturan
Dept. of Computer Eng.
Boğaziçi University
Istanbul Turkey
ozturaca@boun.edu.tr

Bilal Bektaş
Computational Sci. and Eng. Prog.
Boğaziçi University
Istanbul, Turkey
bilal.bektas@boun.edu.tr

Mehmet Yılmazer
Kandilli Observatory and
Earthquake Research Institute
Boğaziçi University
Istanbul, Turkey
mehmety@boun.edu.tr

Abstract—Seismic Data Server Application Service (SDSAS) is developed on the SEEGRID infrastructure in order to serve massive seismic data that are archived from national seismology centers using a high level interface that is easy to use and adapt. It serves official lists of earthquakes, stations and waveform data collected from various South Eastern European (SEE) countries. SDSAS contains three main components: (i) bash scripts to collect and to organize the seismic data by utilizing storage elements, LFC and metadata catalogue AMGA, (ii) C++ iterators that could be used by applications to access station data, earthquake data and information about seismic waveform files collected from regional SEE countries and (iii) a web services client in the form of C++ iterators to interface to comprehensive European seismology data from the ORFEUS datacenter. The client for the ORFEUS data is developed using the web services provided by the NERIES project. We provide usage examples for seismology data iterators and present various timings obtained from workflows that use SDSAS waveform objects.

Keywords- seismology; grid ;

1. Introduction

Before the evolution of grid infrastructures, the web was the main medium of delivery for seismic data. Seismic data providers would place their data on web servers from which scientists could either query or download the data that were of interest to them. This arrangement, however, had several shortcomings. Firstly, data and computational resources are basically decoupled. It is the duty of each scientist to manually download and manage all the data himself. Automation of this process may require the writing of various scripts which may be quite difficult for a user. Secondly, retrieving massive data over the web can be too slow and hence not practical. There have been efforts also to make seismic data available by web services which will help in automation, but again this requires users to learn web services programming and to link such services with their applications. Grids solve these problems by offering a platform where computational, storage resources and other miscellaneous resources are available all connected by high speed networks. A grid user can write a program, run it on the grid where it can access distributed data just like accessing local data with the help of middleware and other tools.

There are, however, some challenges that we encounter: how to make this complex stack consisting of the grid infrastructure, distributed data and various complex tools that are used to archive and index data easily accessible to novice users and legacy codes that have been around for decades. This paper presents the Seismic Data Server Application Service (SDSAS) which is developed as a joint research activities (JRA1) service in order to address such issues in SEE-GRID-SCI project [1]. SDSAS serves massive seismic data that are archived from national seismology centers using a high level interface that is easy to use and adapt. It serves official lists of earthquakes, stations and sensor information collected from various South Eastern European (SEE) countries. There are two main motivations for the development of this service. The first is to provide a high level and easy to use service that enables a researcher or a developer to quickly access seismology data without the need to learn additional tools. The second motivation is to achieve automatic mapping of high level user specifications such as dates, hours and locations to appropriate pathnames. In this way, the details of where the data files reside are hidden from the user and hence writing of location independent code is facilitated. This in turn leads to less application code changes, when data locations change.

SDSAS contains three main components: (i) scripts to collect and to organize the seismic data by utilizing storage elements, file catalog LFC [2] and metadata catalogue AMGA [3], (ii) C++ iterators that could be used by applications to access station data, earthquake data and information about seismic waveform files collected from regional SEE countries and (iii) a web services client in the form of C++ iterators to interface to comprehensive European seismology data from the ORFEUS datacenter [4]. The client for the ORFEUS data is developed using the web services provided by the NERIES project [5]. We provide usage examples for seismology data iterators and also present some timing resuts for accessing in particular the ORFEUS data.

SDS Web Interface is a separate application for presenting seismology data in AMGA tables through an interface that utilizes kml and Google Earth API. It makes use of the SDSAS services in its implementation. It also provides additional services like the ability to view earthquakes computed by earthquake finding applications.

The following sections present the architecture of SDSAS, iterators for data access, NERIES interface for accessing European seismology datacenters and the SDS web interface. The paper is concluded by a discussion

2. SDS Architecture

The following seismology data is available in SEE-GRID-SCI Seismology VO data repository:
a. Earthquake data
b. Station data
c. Waveform data

Figure 1 shows the software and data storage stack for the Seismology VO. Massive seismic data waveform files from each country are uploaded to each country's own storage server, registered to LFC and file metadata are recorded in AMGA metadata catalogue by SDSAS upload scripts. Currently LFC and AMGA servers are located at TUBITAK-ULAKBIM site in Turkey. Information about earthquakes and stations are directly kept in AMGA tables. Storing massive waveform data files in a distributed fashion offers the advantage of improving applications' data access performance by providing multiple pathways to the data files and by avoiding overloading of a centralized data server. The storage of indexes to the files on AMGA server helps to reduce the load on LFC since bulk LFC traversals are quite slow.

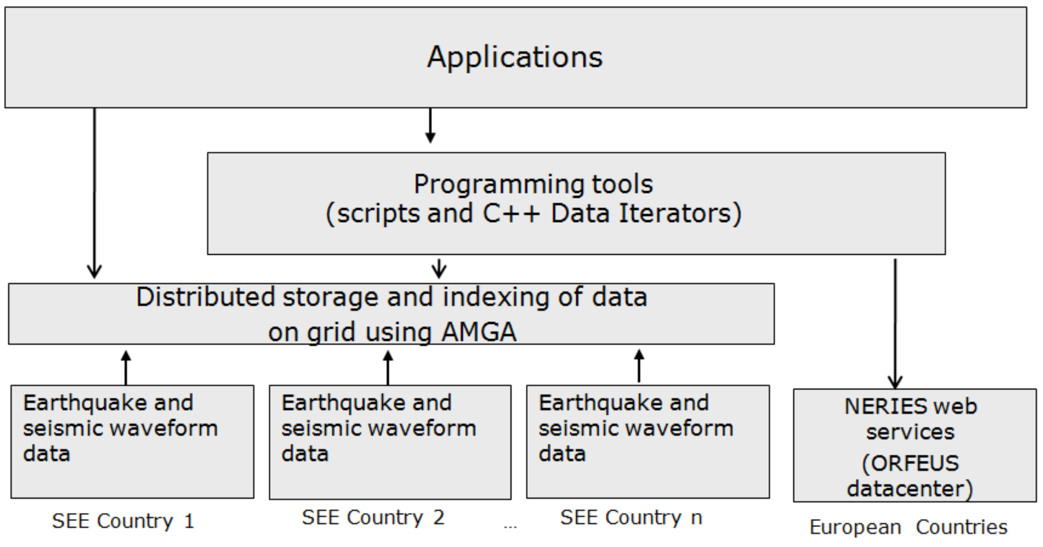

Applications
Programming tools
(scripts and C++ Data Iterators)
Distributed storage and indexing of data
on grid using AMGA
Earthquake and seismic waveform data
Earthquake and seismic waveform data
Earthquake and seismic waveform data
NERIES web services (ORFEUS datacenter)
SEE Country 1
SEE Country 2
...
SEE Country n
European Countries

Figure 1. Software / Data Stack for Seismology VO

Figure 2 depicts the architecture of the SDSAS. AMGA tables storing various information are as follows: Earthquakes table, Stations table, Detailed Station and Sensor table and Waveform File table. The first three tables (earthquakes, stations and detailed stations) are relatively small tables. However, the waveform file table is quite big (currently storing more than 7 million records). Having a single table with so many records posed performance problems – queries to this table were slow. Therefore, this table was partitioned into several monthly tables, and hence improving the performance in this way.

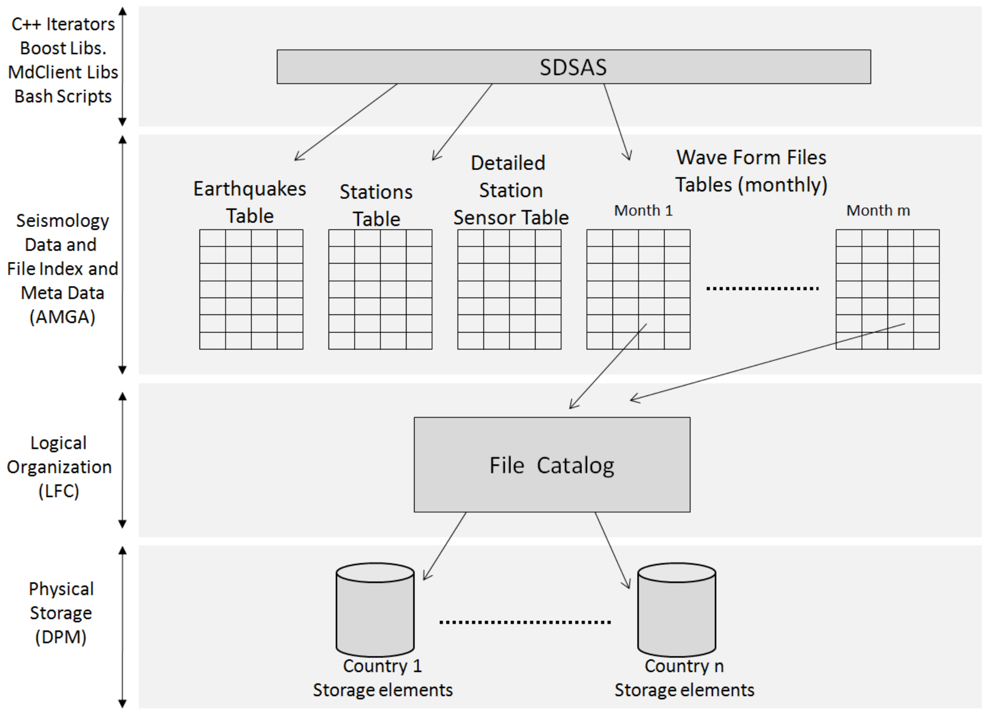

C++ Iterators
Boost Libs.
MdClient Libs
Bash Scripts
SDSAS
Seismology Data and File Index and Meta Data (AMGA)
Earthquakes Table
Stations Table
Detailed Station Sensor Table
Wave Form Files Tables (monthly)
Month 1
Month m
Logical Organization (LFC)
File Catalog
Physical Storage (DPM)
Country 1 Storage elements
Country n Storage elements

Figure 2: Architecture of the SDSAS

Seismology waveform data files are organized logically on an LFC server. The directories are located under :

```
/grid/seismo.see-grid-sci.eu/data/
```

and the organization of data under this directory is shown in Figure-3. Note that a separate directory is kept for each country. Under this directory, a waveform files directory exists in the form year/month/day. A file corresponding to a time interval is placed under the lowest level directory that contains the time interval.

The waveform data files are stored on storage elements whose disks are managed by Disk Pool Manager (DPM) [6]. Note that the choice of AMGA, LFC and DPM on top of which we architected our SDSAS was mostly motivated by the ready availability and the support of these services on the SEEGRID grid infrastructure that we were using. There are similar systems which are mainly used on the U.S. grids that we could alternatively use to build our SDSAS on. These are : Storage Resource Broker (SRB) [7], Integrated Rule-Oriented Data System (IRODS) [8], PetaShare [9], Rich Object Archival System (ROARS) [10], and the Simple API for Grid Applications (SAGA) [11]. As an example, the Louisiana effort [12] uses some of these tools to build a regional cyber-infrastructure for e-Research for researchers and universities. One of the main reasons for introducing SDSAS, was to free the Seismology VO users from the need to learn such tools when porting their applications. Our approach also has the added advantage that if in the future, we use some of these tools (e.g. IRODS, Petashare, SAGA etc.) on which to build SDSAS, then the ported applications do not need to be modified.

```
.../albania/
            Waveformdata_Years/
                               Year/
                                    Month/
                                          Day/

            Station_PoleZeroFiles/
.../armenia/
...                   ...
...                   ...
.../turkey/           ...
```

Figure 3: Organization of directory hierarchy on LFC

3. Data Iterators

AMGA tables can be queried directly by using various AMGA interfaces (shell, perl, java, C/C++). The aim of the SDSAS C++ iterators is to provide a higher level interface to seismic data that does not necessitate the seismologist users to learn AMGA and which provides additional functionalities. SDSAS provides four classes for submitting queries and parsing of AMGA results into records. These are: earthquake, station, waveform and user defined records. Corresponding iterator classes can then be used for accessing the returned records. Table-1 gives the classes and records provided by SDSAS and and how these classes should be used in step by step fasion. Note that SDS_UserRecs object is provided in order to let users access user defined AMGA tables thorough an iterator interface that is quite similar to those of SDS_Quakes, SDS_Stations and SDS_WaveFormFiles.

Table-2: Classes and Records provided by SDSAS and their step by step usage

Step 1	Step 2	Step 3
AMGA query and results parsing classes	**Records prepared from AMGA results**	**Iterator classes for accessing records**

`SDS_Quakes`
`SDS_Stations`
`SDS_WaveFormFiles`
`SDS_UserRecs`
	`SDS_Quake`
	`SDS_Station`
	`SDS_WaveFormFile`
	`SDS_UserRec`
		`SDS_Quakes_Iterator`
		`SDS_Stations_Iterator`
		`SDS_WaveFormFiles_Iterator`
		`SDS_UserRecs_Iterator`

Figure 4 shows an example program. SDS_Init is an object that initializes some internal parameters. This object should always be created at the beginning of the program before using other SDSAS objects. SDS_Quakes object queries earthquake records during the period Aug 16th, 1999 to Aug. 17th, 1999. The returned results (times of occurrences, magnitudes and regions of earthquakes) are than printed using the array accessing mechanism. Note that when the SDS_Quakes object is created, AMGA query is automatically generated in the constructor method. The parsing of the results, the preparation of the SDS_Quake records are also done in the constructor. Another way to access the returned results can be via iterator pointer dereferencing as follows:

```
for (SDS_Quakes_Iterator i = q.begin();  i != q.end() ; i++) {
  cout <<  (*i).datetime << " "
       <<  (*i).md       <<  "  "
       <<  (*i).region   << endl;
}
```

There are also some earthquake methods that are available. These can be used to sort the earthquake records or to generate a Google Earth kml [13,14] file for showing the returned earthquake records graphically. For example, the statement q.time_sort() would sort the records in ascending order of time. The statement q.kml("example-quakes") would create a kml file called example-quakes.kml which could be loaded to Google Earth for displaying records visually resulting from the related query. The kml file is shown in Figure 5.

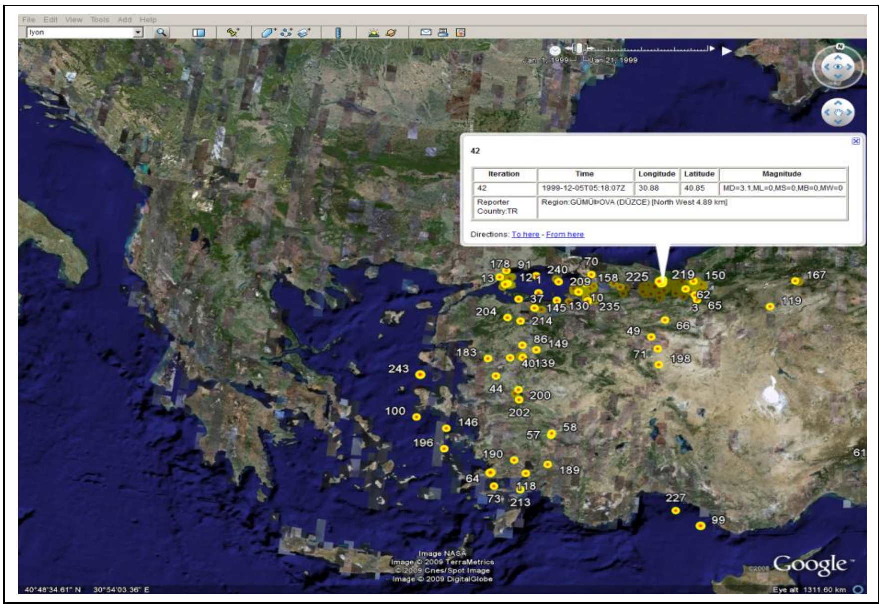
```
#include "sds.h"
#include <iostream>

using namespace sds ;

main(int argc,char *argv[])
{
    SDS_Init sdsinit ;
    SDS_Date_Range dr( SDS_Date(1999,Aug,16), SDS_Date(1999,Aug,17) );
    SDS_Quakes q(Turkey,dr) ;


    for (int i = 0 ;  i < q.size() ; i++)     {
         cout <<  q[i].datetime   << "  "
              <<  q[i].md         << "  "
              <<  q[i].region     << endl;
    }
    q.time_sort() ;
    q.kml("example-quakes") ;
}
```

Figure 4: Example illustrating Earthquake data iterator usage



42

Iteration	Time	Longitude	Latitude	Magnitude
42	1999-12-05T05:18:07Z	30.88	40.85	MD=3.1,ML=0,MS=0,MB=0,MW=0
Reporter Country:TR	Region:GÜMÜÞOVA (DÜZCE) [North West 4.89 km]			

Directions: To here - From here

Figure 5: Google Earth visualization of the kml file generated by SDSAS

Figure 6 shows another example on how waveform file records can be accessed in a given date range. First a date range, a dr object is created. Then an SDS_WaveFormFiles object is created with the following parameters : a date range, country from which the file is to be retrieved, station code (in this case a particular station called ENEZ), sensors and the file formats. The returned records are accessed by the SDS_WaveFormFiles_Iterator object and several record fields are printed. The number of records is then printed. Finally a kml file is generated. The kml file generated shows the locations of stations from which the waveform data was obtained. Values of the various fields of each record are also given when the placemark is pressed.

```
#include "sds.h"
#include <iostream>

using namespace sds ;

main()
{
    SDS_Init sdsinit ;
    SDS_Date_Range dr( SDS_Date(2007,Aug,16), SDS_Date(2007,Aug,17) ) ;
    SDS_WaveFormFiles w(dr,Turkey,"ENEZ",SDS_AllSensors,SDS_AllFormats) ;
    SDS_WaveFormFiles_Iterator wend = w.end() ;

    for (SDS_WaveFormFiles_Iterator i = w.begin();  i != wend ; i++) {
         cout <<  (*i).name << " "
              <<  (*i).path <<  "  "
              <<  (*i).format <<  "  "
              <<  (*i).size <<  "  "
              <<  (*i).country <<  "  "
              <<  (*i).organization <<  "  "
              <<  (*i).station <<  "  "
              <<  (*i).sensor <<  "  "
              <<  (*i).startdatetime <<  "  "
              <<  (*i).enddatetime <<  endl ;
    }

    cout << w.size() << endl ;
    w.kml("example-waveform") ;
}
```

Figure 6: Example showing accessing of waveform files

The kml file example-waveform.kml generated by this example is shown in Figure 7. Note that the placemark when clicked shows the values of the fields in each record. When applications want to access the waveform file, they can simply open the file whose LFC path is given by the path field.

We have presented two full blown examples showing how to access earthquake and waveform file records. Data can be accessed in various other forms. In the next subsection (3.1), we give further examples on usage.

3.1 Further examples of Data Iterators

Table-3 gives additional examples of data iterators. The examples given have the following interpretations:

(i) Constructs an earthquake query object for the date period 16/Aug/1999-17/Aug/1999 for earthquakes with MD magnitude greater than 4.5 from all the countries.

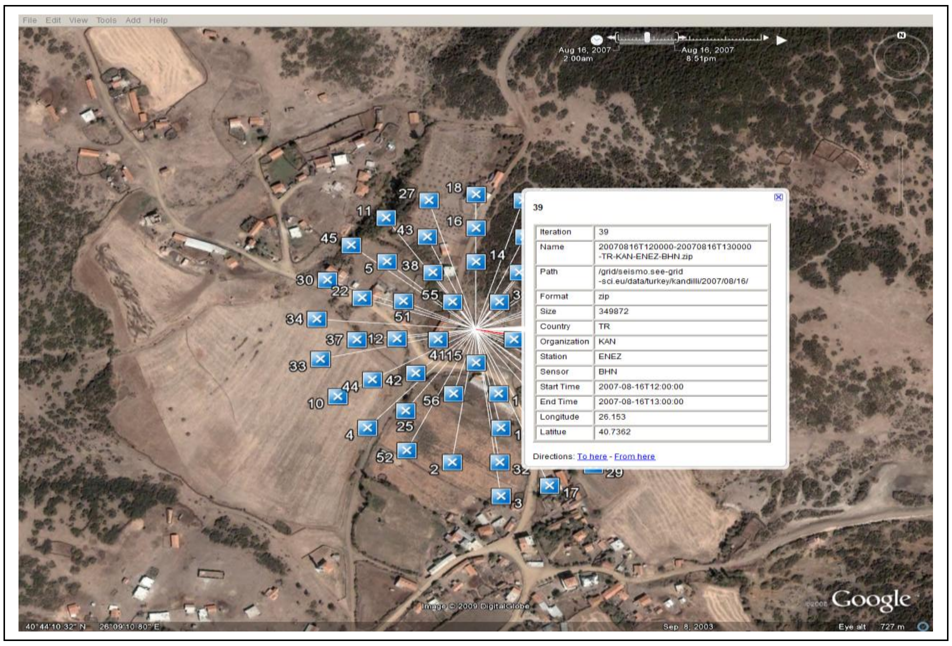
(ii) Similar to (i), but this time date range is constructed from strings argv[1] and argv[2] given as command line arguments. ISO format is assumed for the date strings, e.g. 19990816.

(iii) Constructs an earthquake query object when given a direct AMGA condition string.

(iv) Constructs a station query object with default behaviour. Default behaviour is retrieval of all station records.

(v) Constructs a station query object given a direct AMGA condition string. In this case, it is required that stations with installation time in the specified date range be returned.

(vi) This example first constructs a starting time from the first command line argument argv[1]. ISO format is assumed for the time (i.e. yymmddThhmmss format). Then some number minutes (given in the second command line argument argv[2]) is added to the starting time using the Boost [15] datetime libraries' minutes() function and + time addition operator. A time interval is constructed from these two times. Then a waveform file query object is constructed to return all files containing all or part of the specified interval


File Edit View Tools Add Help
Aug 16, 2007 2:00am
Aug 16, 2007 8:51pm
Image © 2009 DigitalGlobe
Google
40°44'10.32" N 26°09'10.80" E
Sep 8, 2003
Eye alt 727 m

39

Iteration	39
Name	20070816T120000-20070816T130000 -TR-KAN-ENEZ-BHN.zip
Path	/grid/seismo.see-grid -sci.eu/data/turkey/kandilli/2007/08/16/
Format	zip
Size	349872
Country	TR
Organization	KAN
Station	ENEZ
Sensor	BHN
Start Time	2007-08-16T12:00:00
End Time	2007-08-16T13:00:00
Longitude	26.153
Latitue	40.7362

Directions: To here - From here

Figure 7: Kml file showing generated waveform file records of Figure 5

Table 3: Further data iterator examples

Example	Example Statements

(i)	`SDS_Date_Range dr( SDS_Date(1999,Aug,16), SDS_Date(1999,Aug,17) );`
	`SDS_Quakes q(SDS_AllCountries,dr,4.5) ;`

(ii)	`SDS_Date_Range dr( SDS_Str2Date(argv[1]), SDS_Str2Date(argv[2]) );`
	`SDS_Quakes q(SDS_AllCountries,dr,4.5) ;`

(iii)	`SDS_Quakes q("(DateTime >= 20011011) and (DateTime <= 20011018)\`
	`                and (MD <=5)" );`

(iv)	`SDS_Stations s ;`

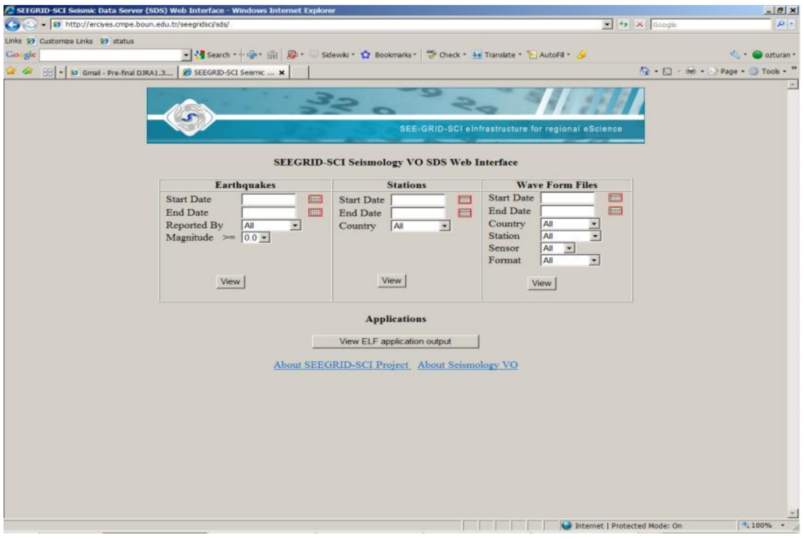
(v)	`SDS_Stations s("(Installation_DateTime >= 19990101) and \`
	`                (Installation_DateTime <= 20100101) " ) ;`

(vi)	`string ts =  argv[1] ;`
	`SDS_Time    t1(from_iso_string(ts)) ;`
	`SDS_Time_Range tr(t1,t1 + minutes(atoi(argv[2])) ) ;`
	`SDS_WaveFormFiles w(tr,SDS_AllCountries,SDS_AllStations,`
	`                    SDS_AllSensors,SDS_AllFormats) ;`

3.1 SDS Web Interface

A web interface to the SDS seismology data has also been developed. Using the web interface, AMGA tables can be queried to graphically display results in Google Earth. The web programs are basically CGI programs written in Perl language. The Perl scripts then invoke executables written in C++ using the SDSAS iterator objects. Figure 8 shows the screendump of the SDS web interface. The web interface can be accessed at the address:

http://hasandagi.cmpe.boun.edu.tr/seegridsci/sds


SEEGRID-SCI Seismic Data Server (SDS) Web Interface - Windows Internet Explorer
http://erciyes.cmpe.boun.edu.tr/seegridsci/sds/
SEE-GRID-SCI eInfrastructure for regional eScience
View
View ELF application output
About SEEGRID-SCI Project
About Seismology VO
Internet | Protected Mode: On
100%

SEEGRID-SCI Seismology VO SDS Web Interface

Earthquakes	Stations	Wave Form Files
Start Date	Start Date	Start Date
End Date	End Date	End Date
Reported By: All	Country: All	Country: All
Magnitude >= 0.0		Station: All
		Sensor: All
		Format: All

Applications

Figure 8. SDS Web Interface

4. NERIES Client Interface

As part of the NERIES project, web services were developed in order to provide an interface to ORFEUS archives of station and waveform data. NERIES provides the following web services [16] related to continuous waveform data:

- getInventory method providing information about available networks and stations,
- dataRequest method for submitting a request for waveform data,
- checkStatus method for checking the status of requested data,
- dataRetrieve method for retrieving the ftp address of requested data
- purgeData method for post-cleanup on the server after the download is completed.

A high level client interface to these web services was developed so that a users can access NERIES data without the need to learn web services and xml. The interface has been developed in the form of iterators presented in the previous section. LibCurl [17] library was used for communicating with the soap server and Libxml2 [18] library for parsing xml responses returned from the server. LibCurl and libxml2 libraries are readily available as installed software on linux distributions. Table 4 shows the NERIES client classes, records and iterators available in the client.

Table 4: NERIES client classes, records and iterators

NERIES query and results parsing classes	**Records prepared from NERIES results**	**Iterator classes for accessing records**
`NRS_Stations` `NRS_WaveFormFiles`	`NRS_Station` `NRS_WaveFormFile`	`NRS_Stations_Iterator` `NRS_WaveFormFiles_Iterator`

Figure 8 presents example client code that utilizes an object called NRS_WaveFormFiles for downloading waveform data when given a time period, a network and a station in the network. The example in fact implements a waveform downloader application. The downloader can be invoked on the command line, for example, as follows:

```
>./nrsdownload 20080410T235959  60 HL ARG
HL-ARG--2008-04-10T23:59:59-2008-04-11T00:59:59.mseed
```

Here, the downloader program called nrsdownload is given command line arguments of 20080410T235959 which is the starting time, 60 minutes of duration time, HL network and the ARG station in this network. The program initializes the NRS_Init object before creating any objects. Then using the Boost datetime library objects ptime and time_period, it constructs a time duration starting at the given time (i.e. given by argv[1] argument). Then the NRS_WaveFormFiles object w is created which in its constructor calls the NERIES web service dataRequest to have the data prepared, and then repeatedly as needed the checkStatus service to see if the data has been prepared. If the data has been prepared, then an ftp address is returned as a result of the dataRetrieve call. Data file in miniseed format is ftped and saved under the name:

```
HL-ARG--2008-04-10T23:59:59-2008-04-11T00:59:59.mseed
```

The filename includes the network name, station name and the time period for the data. Finally, when ftp is completed, purgeData web service is called to do cleaning up on the server. All of these operations are carried out in the constructor of NRS_WaveFormFiles objects. The filename(s) are returned after download(s) are completed. Filenames can be accessed by creating an NRS_WaveFormFiles_Iterator object.

```
#include "nrs.h"
#include <iostream>

using namespace nrs ;

main(int argc,char *argv[])
{
  NRS_Init nrsinit ;

  string ts =  argv[1] ;
  ptime t1(from_iso_string(ts)) ;
  time_period tp(t1,t1 + minutes(atoi(argv[2])) ) ;

  NRS_WaveFormFiles w(tp,argv[3],argv[4])
  NRS_WaveFormFiles_Iterator i = w.begin() ;
  cout << (*i).filename << endl ;
}
```

Figure 9: NERIES client example: nrsdownload program for downloading ORFEUS data

Figure 10 presents another NERIES client example. This one looks quite similar to the first one. It, however, does not input a station name on the command line. It uses NRS_Stations object to retrieve all station records corresponding to a network that is passed to it. This example then downloads waveform files from *all* the stations in the given network. At the end of the program it also calls the kml method of the NRS_Stations object to generate a kml file showing stations locations and station record values when viewed in Google Earth.

```
#include "nrs.h"
#include <iostream>
using namespace nrs ;

main(int argc,char *argv[])
{
    NRS_Init nrsinit ;
    string ts =  argv[1] ;
    ptime t1(from_iso_string(ts)) ;
    time_period tp(t1,t1 + minutes(atoi(argv[2])) ) ;

    NRS_Stations s(tp,argv[3]) ;
    NRS_Stations_Iterator send = s.end() ;

    for (NRS_Stations_Iterator i = s.begin();  i != send ; i++) {
       cout << (*i).network().code << " " <<  (*i).code << endl ;
       NRS_WaveFormFiles w(tp,(*i).network().code,(*i).code) ;
    }

    cout << s.size() << endl ;
    s.kml("stats") ;
}
```

Figure 10: A more sophisticated NERIES client example

5. Discussion and Conclusions

A great challenge we faced when building SEEGRID Seismology Virtual Organization was how to organize massive seismic data of the order of terabytes and contained in millions of files, make them searchable and easily accessible to users without requiring them to write complex queries. Our solution was to provide seismology specific iterators that would allow the user to iterate spatially as well temporally in a program by specifying high level constructs such as stations, countries and date ranges. Such specifications are quite natural to users and allow them to concentrate on their problems rather than requiring them to learn new complex tools.

We were able show the effectiveness of our approach when we gave two Seismology Virtual Organization trainings events (one in January 2008 and one in December 2009) to a group of seismologists who had no or very little programming experience. We first gave a lecture on AMGA and showed them how to make queries to AMGA in C++ example programs . This proved to be too difficult to many trainees. Most of them experienced problems because they had to deviate from their problem domains and deal with issues such as the use of new AMGA APIs, and parsing of the results returned from AMGA which were extremely difficult tasks for them. We then gave a tutorial lecture on seismic data server application service and presented them with example C++ programs that utilize iterators. The trainees quickly grabbed iterators because they did not worry about parsing of results and could concentrate on dates, date ranges and stations which were natural to them. We believe that our seismology specific iterators concept has been quite successful and can be adopted by other grid virtual organizations that want to make their data easily accessible to their users.

Table 5 shows various statistics on the waveform data that have been collected from various country networks. As far as performance issues are concerned, we note the following:

- Slow Internet especially in the South Eastern European countries made it extremely difficult to transfer more than 7 million files to storage elements on the grid in short span of time. Therefore, data files were transferred on hard disks to storage centers.
- File registration facility LFC that provided file cataloguing service turned out to be extremely slow on bulk operations. Registrations of more than 7 million files would require weeks and months on LFC. Therefore, system administrators at TR-GRID which maintain major storage facilities of the Seismology Virtual Organization had to write special scripts to do fast bulk LFC registrations.
- AMGA metadata catalogue showed extremely poor performance when inserted more than 7 million file records. To resolve this, AMGA tables were partitioned into monthly tables which

improved performance considerably. Querying of a single monthly table for few tens of records takes about 1 second. Querying of a single monthly table to have all the full records returned takes about 8 seconds. Querying of all the tables to have all the full records returned takes about 8 minutes. Note that when SDSAS is presented with a date range, it only queries the tables that contain the date ranges.

- We have also timed downloading of 1 hour waveform data file (about 500-650K byte file) from the ORFEUS datacenter using our interface. Timings ranged from 12 to 16 seconds on a user interface machine located in Ankara, Turkey.

Table 5: Various statistics on waveform data collected

	Currently in Seismology VO
Size of Continuous Waveform Data	2.63 TB (since 2005)
No. of networks	5
No. of stations	92
Number of files	7.8 million

Note that the scalability of SDSAS depends on the scalability of the services it is built on, namely LFC and AMGA. Our experiences with LFC has been that bulk file registrations are extremely slow. With AMGA, large tables caused problems, which we overcame by monthly partitioning of waveform data file tables. On the SDSAS iterators side that runs on the client nodes, parsing of retrieved records is quite fast. Under normal usage conditions, as we stated above, it takes about in general 1 second to make a query to a monthly AMGA waveform file table. To further test the response time of AMGA , we have also implemented a workflow where *n* nodes independently make a query into the AMGA server by creating SDSAS **`a SDS_WaveFormFiles`** object to retrieve waveform file records from all stations. Note that we have no control over the grid workflow scheduler. Hence, execution of all nodes may not be carried out concurrently. But considering the fact that this is indeed the way (i.e. by the use of workflows) that almost all of the Seismology VO applications have been gridified, our scalability experiment does mimic our real life usage on the SEEGRID. Table 6 shows the timing results that we have obtained. Note that we have two columns, showing the timings obtained when a query (i) into the same month has been made and (ii) different months have been made by each node. Since different months are implemented as different tables, it is expected and confirmed by our timings that AMGA will have different response times for these two cases.

Table 6: Timings obtained for making

	Query into the same month	Query into different months
Number of nodes (n)	Avg. Time (in secs)	Avg. Time (in secs)
1	0.55	0.72
4	1.14	0.98
8	1.98	1.45
16	3.39	1.02
32	3.16	1.18
64	2.12	1.02

We observe that the timings increase in the case of 8, 16 and 32-node workflows and then decreases in case of 64-nodes workflow. One would expect that in the case of 64 nodes, we should have the greatest time if all accesses were concurrent. However, most probably, the scheduler (which we do not have control over), break large workflows into multiple smaller parts and schedule these in different times. Hence the actual number of concurrent accesses to AMGA server is fewer than those of the other cases.

SDSAS software and documentation is available at [19]. SDSAS has also been used successfully by four gridified applications in the SEEGRID Seismology VO. These applications are : Seismic Risk

Assessment Application (SRA) [20], Fault Plane Solution (FPS) application [21], Earthquake Location Finding (ELF) [22] application and Massively Parallel Seismic Data Wavelet Processing Application (MDSSP-WA) [23]. All these applications are available via [19].

In general, we believe that SDSAS satisfied its requirements: (i) easiness to use and (ii) performance under the usage patterns of about 35 Seismology VO users in the SEEGRID. We expect the number of Seismology VO users to be small. However, even if this number increases to hundreds, the AMGA servers we use can be replicated easily since our seismology data is read-only. In the future, if IRODS is made available and supported by the personnel on our infrastructure, we plan to develop an IRODS based SDSAS.

Acknowledgements:
This research is supported by the FP7SEEGRID-SCI project under the grant number 211338. We also thank the TUBITAK-ULAKBIM personnel for their help in uploading and registering millions of seismology data files on the grid and for making available TR-GRID.

References

[1] SEE-GRID eInfrastructure for regional eScience, http://www.see-grid-sci.eu.
[2] C. Munro, B. Koblitz, N. Santos, A. Khan, Measurement of the LCG2 and Glite File Catalogue’s Performance, IEEE Transactions on Nuclear Science, Vol. 53, No. 4, pp. 2228-2232, Aug. 2006.
[3] AMGA: Arda Metadata Catalogue Project, http://amga.web.cern.ch/amga.
[4] Observatories and Research Facilities for European Seismology (ORFEUS), http:// www.orfeus-eu.org.
[5] NERIES project, http://www.neries-eu.org/.
[6] DPM: Disk Pool Manager, http://www.gridpp.ac.uk/wiki/Disk_Pool_Manager.
[7] SRB: Storage Resource Broker, http://en.wikipedia.org/wiki/Storage_Resource_Broker
[8] IRODS:Integrated Rule-Oriented Data System, http://www.irods.org.
[9] M. Balman, I. Suslu, and T. Kosar, Distributed Data Management with PetaShare In Proceedings of ACM SIGAPP 15th Mardi Gras Conference, Baton Rouge, LA, January 2008
[10] Hoang Bui, Peter Bui, Patrick Flynn and Douglas Thain, ROARS: A Scalable Repository for Data Intensive Scientific Computing, The Third International Workshop on Data Intensive Distributed Computing at ACM HPDC 2010, June, 2010.
[11] SAGA: Simple API for Grid Applications, http://saga.cct.lsu.edu/.
[12] Katz, D.S. and Allen, G. and Cortez, R. and Cruz-Neira, C. and Gottumukkala, R. and Greenwood, Z.D. and Guice, L. and Jha, S. and Kolluru, R. and Kosar, T. and others, Louisiana: A Model for Advancing Regional e-Research through Cyberinfrastructure, Phil. Trans. R. Soc. June 2009 vol. 367 no. 1897 2459-2469.
[13] Google Earth, http://earth.google.com/.
[14] OGC KML, http://www.opengeospatial.org/standards/kml/
[15] Boost Date Time Library, http://www.boost.org/doc/libs/1_39_0/doc/html/date_time.html.
[16] NERIES Web services, http://neriesdataportalblog.freeflux.net/webservices/.
[17] Curl Library, http://curl.haxx.se/.
[18] Libxml2 library, http://xmlsoft.org/.
[19] Seismology VO Applications, http://wiki.egee-see.org/index.php/SG_Seismology_VO.
[20] E. Ülgen, A. Akkaya, Ç. Kocair. C. Şener, Seismic Risk Assessment A Grid-based approach for the South-East European Region, See-Grid-Sci User Forum, Istanbul, Dec. 8-9, 2009.
[21] M.Yılmazer, B. Bektaş, M. Kozlovszky, Gridification of Fault Plain Solution (FPS) Application, See-Grid-Sci User Forum, Istanbul, Dec. 8-9, 2009
[22] M. Yılmazer., B.Bektaş, C.Özturan, R. Arıkan, M. S. Geden., Parallelization of Earthquake Location Finding (ELF) Application, See-Grid-Sci User Forum, Istanbul, Dec. 8-9, 2009.
[23] L. Jordanovski, B. Jakimovski, A. Misev, Massively Parallel Seismic Data Wavelet Processing Using Advanced Grid Workflows, ICT Innovations, Springer Lecture Notes, Ohrid, Sept. 28-29 2009.