An Integrated System for Near Real-Time 3D Visualization of NEXRAD Level II Data on TeraGrid

Vinaitheerthan Sundaram[§], Yi Ru[£*], Bedřich Beneš[*], Lan Zhao[§], Carol X. Song[§], Taezoon Park[§], Gary R. Bertoline[*], Matthew Huber[¥]
Rosen Center for Advanced Computing[§], Dept. of Computer Graphics Technology[*], Dept. of Earth and Atmospheric Sciences[¥]
Purdue University, West Lafayette, IN 47906, USA, Oracle Corporation[£], California, USA
{vsundar, yru, bbenes,lanzhao,carolxsong,tzpark,huberm,bertoline}@purdue.edu

1 Introduction

The occurrences of severe weather have caused many injuries, fatalities, and severe damage to personal and public properties. Weather forecasting and severe weather prediction help reduce such damages by providing opportune warnings so that people can protect their lives and properties, change economic operations, and plan daily activities ahead of time. Traditionally, the evaluation and analysis of actual weather conditions are done using 2D satellite images. In the past few years, the National Weather Surveillance 1988 Doppler Radar Network (WSR-88D) [1] - also known as the Next Generation Weather Radar system (NEXRAD) – became available, providing high spatial and temporal resolution 3D data on a continuous basis. Despite the availability of 3D information in the new generation of radar data, this data is most commonly displayed today as 2D images, simple 3D point clouds, or iso-surfaces. The resulting images provide limited information about the movement of the frontal systems, as viewed on web-sites such as www.weather.com, www.wunderground.com or when viewed on TV. The data is typically displayed only for single radar and the data from multiple sites are just overlapped using transparency. These results are usually enough for a simplified view but cannot accurately represent details.

A better visualization of the radar data captured by the Doppler network can significantly help weather forecasters and researchers to gain fresh insights on weather conditions and could greatly improve our knowledge and help educate future scientists. Major computational challenges exist in providing visual displays that fully utilize the 3D information in the radar data in real-time, and we have addressed some of the challenges by using TeraGrid resources. In this paper, we present an integrated solution for near real-time data delivery and 3D visualization that can be deployed as a service gateway to engage experts and non-experts alike. This system utilizes the NEXRAD data distribution already available on the TeraGrid and the TeraGrid Condor resource to speed up the processing of data from multiple radar sites, coupled with hardware accelerated graphics processing for interactive 3D visual analytics.

The paper begins with an introduction where we discuss the scientific motivation, challenges, and our contributions. We then review previous approaches, related works and existing methods in Section 3. Section 4 describes the design of the integrated system. Sections 5 to 7 describe the schemes and methods of our approach in detail. Experimental results are presented in Section 8. Section 9 concludes our work and addresses some future improvements and extensions of this work.

2 Challenges and Our Contributions

The NEXRAD Level II data stream distribution is available on the TeraGrid through the Purdue resource provider. The data are continuously received in near real time (less than 30 seconds delay) from 121 NWS NEXRAD sites and 11 DOD (Department of Defense) sites [1] using Local Data Manager (LDM) developed by Unidata.

In order to take full advantage of this data resource, it is critical for users to be able to easily access, analyze, and visualize the data in a near real-time fashion. The large volume and real-time streaming of the radar data (~50 MB/second) present major computational and data management challenges. Today, the radar data is difficult to use for an end user in several ways, including (1) Cumbersome access method: Currently the data is most commonly accessed through FTP or HTTP, which is not efficient for studying time-critical weather events. (2) Hard-to-understand data format: The raw data is stored in a radar-native format which is further compressed using a modified bzip2 algorithm. It cannot be readily used by most common visualization applications. Significant knowledge about the native radar data format and the compression algorithm and effort of programming are needed in order to convert the data into a format that is convenient for users. (3) Intensive computation: Analyzing a large amount of data over a long period of time and/or over a large geographic region requires significant computation and storage resources, which cannot be handled by a single computer.

We have developed a distributed system that addresses the above mentioned issues. This system allows users to interactively access, process, and visualize remote radar data in 3D from multiple sites

over a time period of interest through a Virtual Network Computing (VNC) client**.** The data is visualized by a hardware-accelerated, texture-based volumetric renderer on a Graphics Processing Unit (GPU). Customized transfer functions that map data values to optical properties such as color and opacity are provided to users for creating different views of the display. A key contribution of this system is that it can display and manipulate sequences of the radar data in 3D volumes from multiple sites and provide fully interactive 3D animations.

The main contributions of the paper include:

- A set of reusable data services, supported by TeraGrid resources, that provides radar data retrieval and preprocessing in near real time.
- Parallel data processing using TeraGrid Condor resources to convert the data streams from multiple sites into 3D volumes for rendering.
- A fully interactive 3D visualization tool of NEXRAD Level II data.
- A functional end-to-end integrated system that connects radar data retrieval, processing, remote rendering, and 3D interactive visualization.

We show visualizations of radar reflectivity over several hours spanning multiple sites in near real-time using TeraGrid resources. Moreover, our system provides all the technologies necessary to create instant or real-time visualization if the radar data stream is converted to 3D volumes for rendering offline and stored instead of being done on-demand. By providing near real-time radar data and interactive 3D visualization of radar reflectivity, our system enables meteorologists, climatologists, and hydrologists to understand, monitor, and predict severe weather in a timely manner.

3 PREVIOUS WORK

For a long time, researchers have been developing weather visualization systems to help weather forecasters better understand the data in order to provide timely and accurate weather forecasts. The Integrated Data Viewer [2] developed by Unidata, can read different data formats, including satellite imagery, gridded data, surface observations, balloon soundings, Doppler Level II and Level III radar data, and NOAA National Profiler Network data, and display them in 2D or 3D fashions. The National Climate Data Center (NCDC) also provides some visualization tools to display Level II radar data. The NOAA NCDC Java NEXRAD Viewer and Data Exporter are specific to load Level II and Level III radar data into an Open GIS [3]–compliant environment. As part of the Collaborative Radar Acquisition Field Test (CRAFT) [4] project, the Interactive Radar Analysis System (IRAS) [5] can read and display Level II radar data via the Internet in real time. In 1990, Hibbard et al. presented the first version of the Vis5D system [6] for interactive visualization of large grid datasets from numerical weather models. In Vis5D, the data are constructed in the forms of 5D rectangular grid of points. Later, Hibbard proposed a Java class component library - VisAD [7]. The major feature of VisAD is to provide geographically distributed users with interactive and collaborative visualizations

Due to the inherent limitations of these tools, they are not yet able to fully accommodate the needs of many users. Most APIs are written in Java language, which makes the data retrieving and processing slow. Most representations are limited to 2D and 3D point clouds. Many tools can only represent the dataset from one single site or simply mosaic 2D images. Hence, the visual representations generated by most tools are not accurate. Another issue is that most existing systems handle the data generated at one time step.

In order to provide accurate rendering and multi-field visualization, Riley et al. presented a new visualization system in [8, 9] to produce a realistic representation based on the particles' optical properties such as extinction and scattering. Song *et al*. proposed an integrated atmospheric visual analysis and exploration system [10] that supports physics-based rendering, illustrative rendering, and particle and glyph rendering. The rendering of clouds is based on a modified slice-based volume rendering scheme. A system presented in [11] allows for real-time acquisition, organization, and visualization of atmospheric datasets in a geospatial environment.

Ueng et al. proposed a system to specifically visualize the Doppler radar data [12] through a three-pass technique. A new technique called the Vortex Objective Radar Tracking and Circulation (VORTRAC) [13] was developed and is currently tested for analyzing hurricane strength at the National Hurricane Center (NHC) in Miami, Florida.

4 SYSTEM DESIGN

Our overall goal is to create an integrative system for 3D interactive visualization of NEXRAD Level II data in near real-time. In the process of developing this system, we also created a set of data and visualization services that are reusable by a broader user community and for use with existing systems. As shown in Figure 1, our integrated system consists of three main service components: (1) radar data access service, which allows users to retrieve and preprocess radar data in near real-time, (2) radar data processing service, which converts the data streams from multiple sites into 3D volumes for rendering, and (3) remote rendering service, which combines multiple 3D volumes and produces 3D interactive visualization. We will discuss each of the service components in detail in the following sections. A map-enabled graphical user interface is built on top of these service components, through which users can interactively select and visualize the radar data in 3D for a region of interest remotely via a VNC client. At the back end, the radar data are stored and managed by a Storage Resource Broker (SRB)-based data grid supported by the TeraGrid [14].

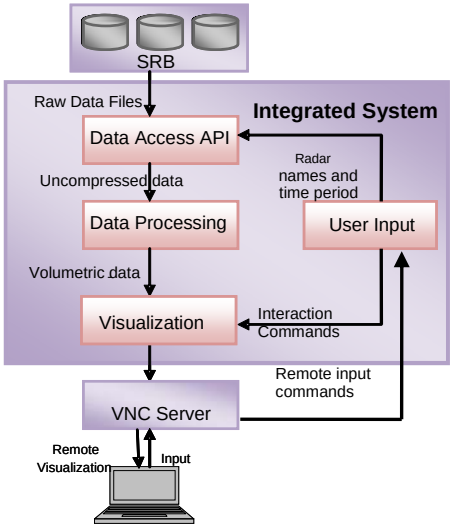

Fig. 1. An integrated system for 3D visualization of NEXRAD Level II data.

5 NEXRAD TERAGRID DATA COLLECTION AND SERVICES

The NEXRAD Level II data are collected as the radars go through a programmed set of movements, which involve a continuous rotation over 360° in azimuth and a simultaneous increase in elevation by 1° to 3° per complete sweep [1]. The spatial resolution is 1 kilometer for reflectivity and 0.25 kilometer for velocity and spectrum width in range, and 1 for all the three fields in azimuth. The radar makes up to 14 scans in elevation ranged from 0.5° to 19.5° determined by the Volume Coverage Patterns (VCPs). Figure 2 is a graphical representation showing the structure for reflectivity of Doppler data. Velocity and spectrum width components also have similar structures.

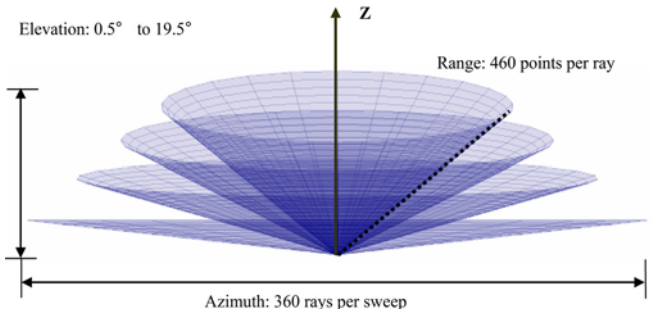

Fig. 2. 3D structure of Doppler radar data (Reflectivity).

The NEXRAD Level II radar data stream produces the raw datasets on a continuous basis. The radar data files vary in size from a few megabytes to tens of megabytes each, depending on the weather condition. The temporal resolution is 4-5 minutes in severe weather vs. 9-10 minutes in calm weather.

Due to the large size, the radar data files are compressed with a modified bzip2 algorithm before it is stored at the distributor sites. The sizes of the compressed files vary from a few hundreds of kilobytes to a few megabytes.

There are three main challenges in managing the streaming radar data and providing easy access to users: (1) The radar data is received continuously at near real-time via the LDM software and, as a result, the data management system needs to be able to efficiently detect and manage the latest data immediately after it is received. (2) The raw files, in a compressed (bzip2) format, cannot be read directly by existing off-the-shelf radar data libraries such as NASA TRMM Radar Software Library (RSL) [15]. (3) There is a lack of easy ways to programmatically access the data in a timely manner to support time-sensitive applications. Users often download data using FTP or HTTP.

We have designed a SRB-based data management solution and developed customized services to provide easy access to the data. As new data are received, they are automatically registered under special shadow directories managed by a SRB data grid on the TeraGrid. Each shadow directory corresponds to a radar station and its content is dynamically updated to reflect the latest data received. On top of the SRB middleware, a THREDDS (Thematic Realtime Environmental Distributed Data Services) [16] application server is deployed and integrated to provide a dynamically generated radar data catalog and remote data access through several existing radar data visualization tools including IDV [2].

To make the radar data available to a wider user community, we further developed a set of radar data access interfaces that retrieve the desired radar files and convert them into usable format based on the radar station name and the time period provided by the user. There are two interfaces developed using the SRB C library and RSL: a command-line interface and a set of Application Programming Interfaces (API) provided as a C++ library. Using these APIs, a user can retrieve any radar data file available in our SRB repository. The files retrieved can also be uncompressed.

Our library provides a set of flexible interfaces that allow users to access both raw and converted data. It can be linked with a third party application program. It contains a factory class that generates instances of *RadarSet* class. The *RadarSet* class connects to SRB and retrieves the data users are interested in. Once the raw datasets are obtained, they are processed using the RSL library. The library can be used to retrieve variable values such as reflectivity and radial velocity from the compressed radar data files. The retrieved data are stored in the computer's main memory for processing. Each *RadarSet* object corresponds to a radar station. It currently supports the following operations:

(1) *buildDataList* retrieves all the matching file names for a particular station and a time period.
(2) *getOldestFileName* retrieves the oldest file name available for a given station.
(3) *getRadarFile* retrieves the radar file from SRB without uncompressing.

(4) *readRadar* retrieves the radar data file from SRB, uncompresses it, and then stores the converted data to a *Radar* structure in memory using RSL.

Because of the wide area coverage and extremely high temporal and spatial resolution of NEXRAD observations, and their availability in near real-time via high speed TeraGrid network, this observing system is unique in its ability of providing initial conditions and evolution constraints for numerical weather prediction models, such as providing important constraints when assimilated into models and analyses [17]. Our TeraGrid data services allow researchers to utilize the NEXRAD data in models and tools without having to learn the details of how to find and convert the data.

6 RADAR DATA PROCESSING

6.1 3D Rectilinear Grid

To ensure an efficient volume rendering process in the rendering stage, all data is resampled in a rectilinear grid structure which contains a collection of cells arranged on a regular lattice [18]. A bounding box that contains data from all the sites was generated first. There are more data points distributed in the X-Y plane than in the Z direction (the elevation direction) in the grid. Therefore, the grid (Figure 3) is designed to have a non-uniform structure such as 256 by 256 by 128, which provides the grid with 128 layers of X-Y planes, each consisting of a 2D uniform grid of 256 by 256.

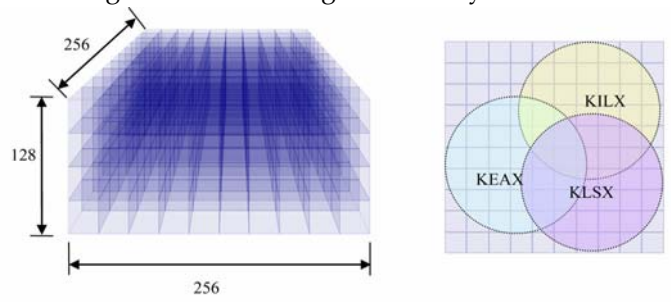

Fig. 3. 3D 256x256x128 grid structure (left) and bounding box.

6.2 Integrating Data from Multiple Sites

The completion of the 3D grid is followed by resampling of radar reflectivity data values into it. For each site, the RSL library stores the data in the local spherical coordinates (azimuth, elevation and range) with the origin at the location of each radar station. To integrate the data from the multiple sites, we use a global coordinate system that aligns the data. Since radars at different location are operated at different tempos, the data from one site are collected at a time and rate different from another site. We perform interpolations between two time stamps to reduce the temporal aliasing.

When the reflectivity values are re-sampled at each grid cell into the pre-defined 3D array, the volume is constructed. It is easy to conceive that, in partially overlapping regions, values obtained from more than one site are likely to be sampled to the same cell, which means some cells have duplicate values, and some cells only have a single value. To eliminate the potential error resulting from the redundant data sampling, we calculate the sample average. Since the distribution of the sample values may still be sparsely populated in the volume space, vertical interpolation is applied to fill the gaps. Due to space limitations, we omit many details related to the rendering portion of this system (available in [19]). A more detailed view of the data flow in the data processing and rendering components is shown in Figure 4.

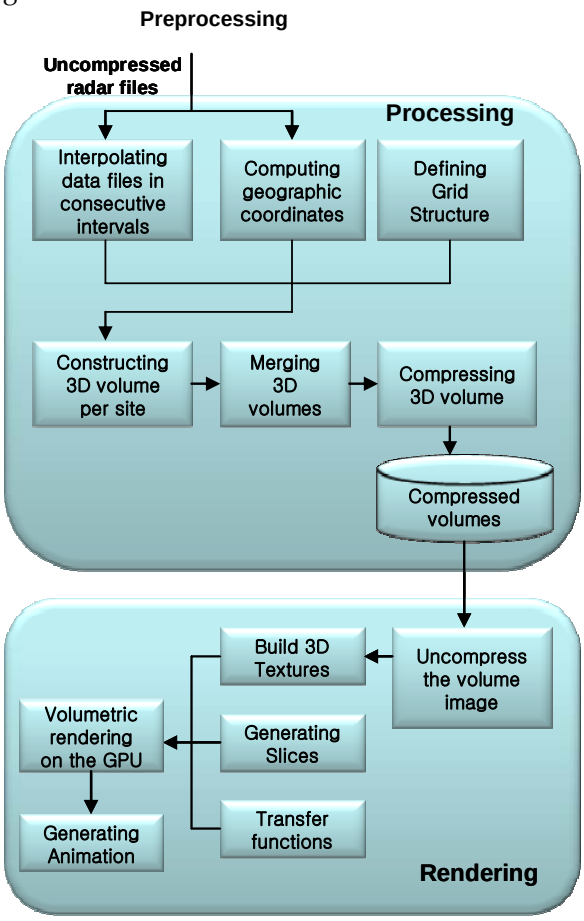

Fig. 4. Data flow in NEXRAD data processing and rendering.

6.3 Parallel Data Processing on TeraGrid

Our first prototype used a sequential implementation of radar data fetching and processing techniques as described in 6.1 and 6.2 as a proof of concept. There are apparent limitations of the sequential implementation and the performance can be greatly improved by parallelization.

As shown in Figure 4, the volume for individual sites can be constructed independent of the other sites and all individual volumes can be merged at the end. Moreover, for the same site, each interval is independent of the other and, hence, volumes corresponding to an interval are independent of the other intervals. Therefore, each site can construct a partial 3D volume for each interval. These partial 3D volumes can be merged at the end to form a full 3D volume for each interval. We also note that reading data for each site from SRB over a network

can be easily parallelized as they are independent of each other and can be uncompressed in parallel as well. Our 2nd version of the implementation takes advantage of such parallelism.

We create two types of jobs for each visualization request: processing jobs and merging jobs. The processing job retrieves the data from SRB, processes them and produces partial 3D volumes one for each interval. The merging job combines the 3D volumes from all sites and creates the full 3D volume for each interval. The granularity of parallelization can be changed based on the processing power available and the size of input data. We used one processing job for each station and one merging job for all stations after the processing jobs complete. The ordering of job execution is controlled using DAGMan [20]. The jobs are submitted via Condor to the TeraGrid Lear resource which consists of 512 dual-CPU Dell PowerEdge 1425 compute nodes. Each node has two 64-bit 3.2 GHz Xeon CPUs and 4 GB of RAM. This parallelization resulted in a large speedup from running the processing jobs in serial. Detailed performance data is provided in Section 8.

7 RADAR DATA RENDERING

Texture-based volume rendering has been used to display the high-quality visualization of the weather data. In object-order volume rendering, a stack of 2D parallel polygonal slices are usually used to represent 3D discrete scalar field. Once the 3D texture is stored in GPU, it is mapped into these semi-transparent slices to texture them. In our implementation, the 3D volume was represented as a unit cube referred to as proxy geometry, and was divided into viewport-aligned slices in the equal-distance fashion, the back-to-front order parallel to the image plane. After the stored 3D texture was applied to the proxy geometry, the scalar reflectivity value is mapped onto the slices by performing 3D texture lookup functions. However, the optical properties of each sample, such as the color and opacity still remain undecided. Having considered the fact that the reflectivity is a scalar field, we chose a 1D look-up color table as the transfer function and stored it as a 1D texture. By taking the 3D texture samples - the reflectivity data - as the look-up color table indices, and mapping them into the display attributes, the color and density of the volumetric object were identified. The look-up color table is similar to the one used in the NWS web page [21] and is widely used for radar reflectivity data.

To support the interactive transfer function, we created an interface that acts as the transfer function window, allowing users to create and modify transfer functions to their interest. The created or modified transfer functions are then applied to map the data onto the appropriate color and opacity.

Figure 5 shows the developed transfer function window for red, green, blue (RGB) and alpha channels representing the NWS look-up color table [21].

The shading and animation was performed using the NVIDIA CG shading programming language, which is very suitable for volumetric rendering the scene as well as accelerating the rendering speed.

8 RESULTS

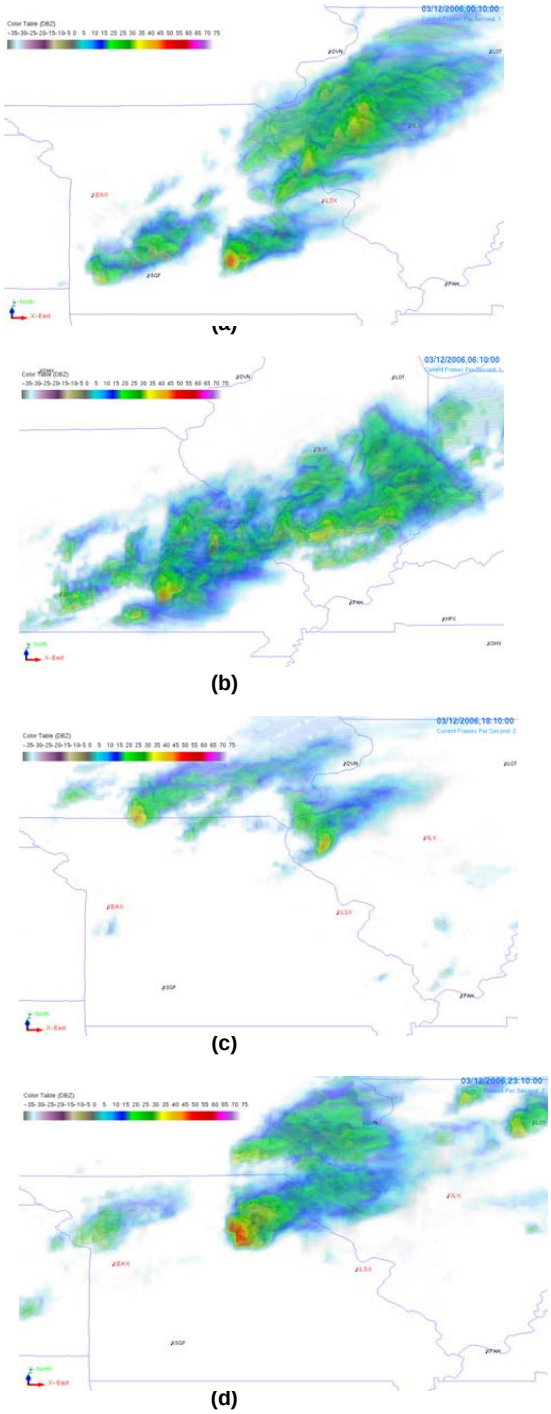

Fig. 5. Images rendered from different timestamps (a): 00:10:00; (b): 06:10:00; (c): 18:10:00; (d): 23:10:00

We have implemented a NEXRAD II weather data visualization interface using our data processing and rendering techniques. The application was written in C/C++ language and built on a number of libraries

including RSL, CG, OpenGL, GLUT, and GLUI. In this section, some rendering results are first presented to verify our methodology, and a number of experiments are described to show the costs and performance of computation and rendering in the system.

8.1 Visualization

The datasets used in our system for testing contain the data from scanning 24-hour supercell storms on March 12, 2006, in the Midwest region of the United States. All simulations in 7.1 and 7.2 have been carried out on a Windows desktop equipped with a 3.20GHz Pentium 4 processor, 2.0 GB of main memory, and an NVDIA Quadro FX 3500 graphics card with 256 MB of video memory. An exception is section 7.2 where the result is obtained from different computer hardware. The resulting image resolution is set to 796x532 for all the experiments.

Figure 5 includes four rendered images at different timestamps selected from the animation. The images are rendered when the view direction is tilted approximately 20 degrees relative to Z axis. The complete 24-hours animation can be downloaded from our website. From the animation, we could see the weather changes on that particular day.

8.2 Performance of Combining Different Resolutions and Sampling Rates

In this experiment, we measure the frame rate - the number of frames per second - under various combinations of the resolution and the sampling rate to evaluate the performance.

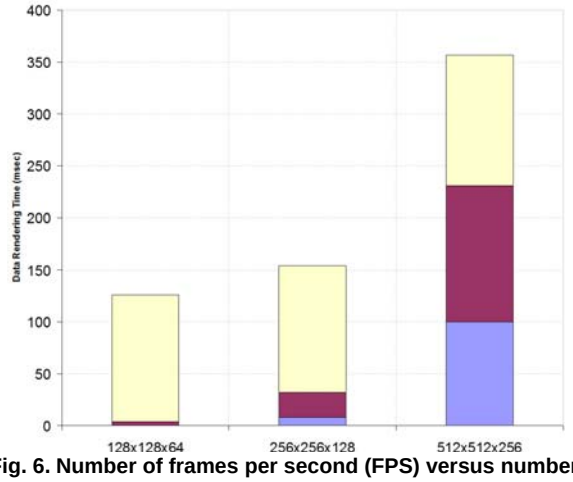

Fig. 6. Number of frames per second (FPS) versus number of slices.

Table 1. FPS Measured with Different Resolution and Number of Slices

Resolution	128×128×64	256×256×128	512×512×256
64 slices	20	18	4
128 slices	15	12	4
256 slices	8	8	3.5
512 slices	5	5	2.5
1024 slices	3	3	1.5

The total processing time is made up of three individual steps: t_1 - reading the volumetric dataset from the hard disk with decompression, t_2 - the time of reconstructing and updating the 3D texture in GPU, and t_3 - the actual rendering time for one frame. The resulting data are shown in Table 1, and its graphical representation is shown in Figure 6. The three colors blue, red and yellow in Figure 6 correspond to t_1, t_2 and t_3 respectively.

The plot indicates two important features: 1) When the resolution increased, the frame rate drops accordingly for most test cases. The trend is due to the fact that the higher resolution requires longer texture reading and updating time t_1, t_2, thereby decreasing the number of frames processed per unit time. Another observation is that, the impact of resolution on frame rates is more dramatic at a lower sampling rate. Comparatively, at a higher sampling rate, the change of the frame rate is less dependent on the volume resolution. The reason is that at the low sampling rate, the proportion of the time spent on the actual rendering t_3 over the total rendering time is less than the proportion at a high sampling rate. An increase of resolution which primarily increases t_1, t_2, therefore exerts a greater influence on the frame rate.

2) With an increase in the number of slices, the frame rate declines accordingly. This is because the sampling rate dictates the actual rendering time t_3, with a higher sampling rate leading to the reduced frame rate. It is also noticed that, the effect of sampling rate on the frame rate is more substantial when a lower resolution is used. With the highest resolution [512x512x25], the frame rate is almost independent of the sampling rate. The reason of this observation is that with the lower resolution, the time cost of texture reading and updating texture t_1, t_2 is relatively insignificant as compared to on the actual rendering time t_3. Since t_3 is dictated by the sampling rate, the majority of the computation resource is recruited for the actual rendering at a higher sampling rate. At low resolution, the magnitude of t_1, t_2 is too small to effectively dilute the effect of sampling rate, making it the single most important rate-controlling parameter for the visualization process.

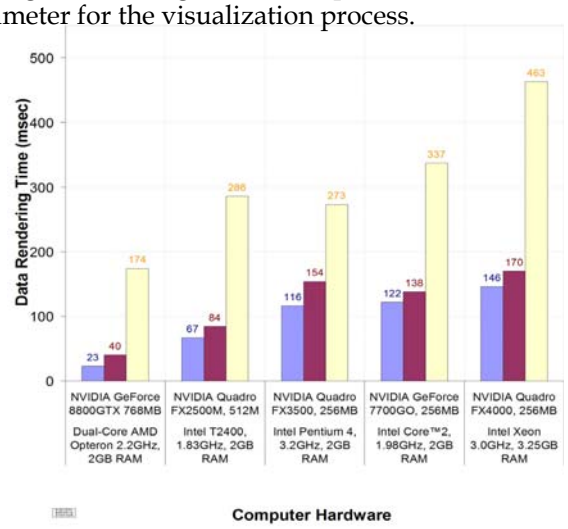

Fig. 7. Rendering time on different machines

We also tested the program on five different machines. The results are shown in Figure 7. With better hardware such as CPU and GPU, the rendering speed is significantly improved. For example, the computer with a Dual-Core AMD Opteron 2.2GHz processor with 2GB RAM and an NVIDIA GeForce 8800GTX graphics card with 768MB memory can render approximately 45 frames per second with the resolution of 128x128x64 and 256 slices.

8.3 Parallel Processing

We designed this experiment to illustrate the performance gain achievable by utilizing parallel processing of data from multiple sites using TeraGrid resources as discussed in Section 6.3. The metric used to compare sequential and parallel solutions is the total processing time, which is defined as the total time taken to convert the raw radar data to render-able 3D volumes. We do not include the visualization time because the render-able 3D volumes generated are visualized using the same GPU for both solutions and therefore, takes the same amount of time. Moreover, the visualization time is negligible compared to the processing time for multiple sites.

For this experiment, we fixed the time period of interest to be from 11am to 4pm on March 21, 2008 and the WSR88D radar sites to be a set of 100 sites from the continental USA. We varied the number of radar stations from 1 to 100 and measured the total processing time. The uncompressed radar data processed varied from 200MB to 27GB for 1 to 100 sites. We repeated the experiment multiple times and the reported numbers are the averages of the total runs. The results from this experiment are shown in Figure 8.

In Figure 8, the total processing time for the parallel solution is the sum of Condor job scheduling time and the actual data processing time on worker nodes. The Condor job scheduler runs periodically (about every five minutes) and schedules as many jobs in the queue as possible. In our parallel solution, the DAGMan [20] first submits all the processing jobs at once and then runs the merging job after the processing jobs are completed. It is shown that although the job scheduling time contributes to a major overhead in the parallel solution, its impact on performance becomes less significant when the number of radar sites increases. In the case of a small number of radar sites, the parallelization overhead dominates the total processing time of the parallel solution and therefore, the sequential solution outperforms (761%, 11% faster than parallel for 1 and 10 sites respectively). As the number of radar sites increases, the parallel solution clearly outperforms the sequential solution, with a speedup factor of 77.5%, 249%, and 508.7% for 30, 50, and 100 sites respectively. The reason is that as the number of radar sites increases, the amount of processing increases significantly and so does the total processing times required in the sequential case (0.78 to 46.87 minutes or 5909% increase for 1 to 100 sites). However, in the parallel solution, the actual processing time only increases slightly with the number of radar sites (1.14 to 4.75 minutes or 31.7% increase for 1 to 100 sites). The slight increase in actual processing time as the number of sites increases is attributed to the increase in the time taken to merge the partial 3D volumes from different sites. The total processing time of the parallel solution remains more or less the same (6.72 to 8.92 minutes for 1 to 100 sites) compared to about 60 times increase in the total processing time of the sequential solution (0.78 to 46.87 minutes for 1 to 100 sites).

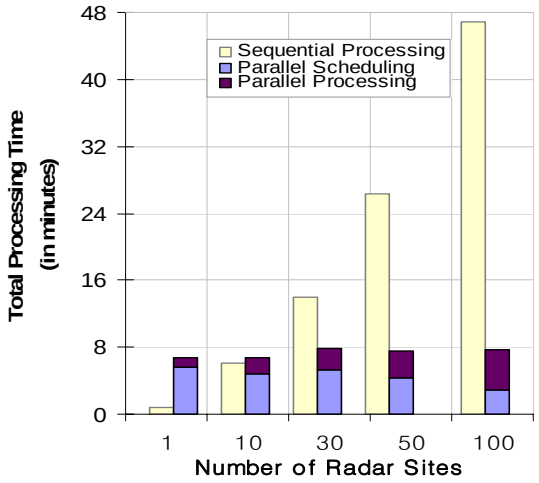

Fig 8. Comparison of total processing time for sequential and parallel data processing solutions when the number of WSR88D sites is varied.

8.4 Discussion

The experiment in 8.3 clearly demonstrates the benefits of the parallelization of the data processing algorithm using TeraGrid computation resources. It allowed us to process the radar data from a large number of radar stations. Similarly, leveraging the distributed computation resources of the large Condor pool, the parallel solution can scale very well to support visualization of radar data over a longer period of time, as well as simultaneous requests from multiple users to visualize data from different regions or in different time periods. It is noted that although the parallel solution significantly improves the processing time for a large number of radar sites, the time it takes to prepare the data still needs to be reduced in an interactive environment. Our next step is to develop a preprocessing and caching mechanism based on the parallel solution which will further improve the response time for end users.

We noted several scalability issues during our experiment. Because of the large volume of uncompressed radar data, the program quickly runs out of disk space when uncompressing radar data from a large number of radar sites. To overcome this problem, we modified the program to create partial volumes for each site and remove the large uncompressed files as soon as a site is processed. This helped us to scale it to 100 stations. When testing for all stations across the country for a 5-hour time period, we encountered a few

bad radar files which had a negative number of sweeps and cannot be handled by the RSL library. Our solution is to identify the bad data based on the file size and skip them in the experiment. In addition, the setting on the maximum number of connections of the San Diego Supercomputer Center (SDSC) SRB MCAT database had to be increased (with the help of the SDSC SRB team) to be able to support a large number of simultaneous access to the SRB server.

The current implementation is a prototype system that is used to evaluate the requirements of handling large volumes of radar data and developing effective visual analytics techniques. It supports 3D end-to-end interactive visualization of radar data from multiple stations over a long period of time. In the future, we will investigate how to extend the system in several directions including (1) support of multiple interactive users, (2) visualization at a higher resolution, and (3) faster response time using caching and preprocessing.

9 Conclusions

In this paper, we presented an integrated system that allows users to interactively access, analyze, and remotely visualize the NEXRAD Level II data stream, which provides high resolution radar data that is vital to understanding, monitoring, and predicting severe weather and flooding events. We also ran performance tests to determine the effectiveness of our solution on the display of near real-time Doppler 3D radar data.

Acknowledgment

This research is sponsored in part by the National Science Foundation under TeraGrid Resource Partners grant OCI-0503992. The Level II Doppler radar is provided in real-time to Purdue's Atmospheric Measurement and Prediction Consortium in partnership with National Weather Service utilizing software developed and maintained by Unidata.

References

[1] M. Huber and J. Trapp, "A review of nexrad level ii: Data, distribution, and applications," Journal of Terrestrial Observation, in press.

[2] Unidata, "Integrated data viewer," 2007. [Online]. Available: http://www.unidata.ucar.edu/software/idv/

[3] The Open Geospatial Consortium Inc, "Open GIS," 2007. Available: http://www.opengeospatial.org/.

[4] K. E. Kelleher, K. K. Droegemeier, J. J. Levit, C. Sinclair, D. E. Jahn, S. D. Hill, L. Mueller, G. Qualley, T. D. Crum, S. D. Smith, S. A. D. Greco, S. Lakshmivarahan, L. Miller, M. Ramamurthy, B. Domenico, and D. W. Fulker, "Project craft: A real-time delivery system for nexrad level ii data via the internet," Bulletin of the American Meteorological Society, vol. 88, no. 7, pp. 1045–1057, July 2007.

[5] D. L. Priegnitz, "IRAS: Software to Display and Analyze WSR-88D Radar Data", American Meteorological Society, 1995, pp. 197–199.

[6] B. Hibbard and D. Santek, "The vis-5d system for easy interactive visualization," VIS '90: Proceedings of the 1st conference on Visualization '90, Los Alamitos, CA, USA: IEEE Computer Society Press, 1990, pp. 28–35.

[7] B. Hibbard, "Visad: connecting people to computations and people to people," SIGGRAPH Comput. Graph., vol. 32, no. 3, pp. 10–12, 1998.

[8] K. Riley, D. Ebert, C. Hansen, and J. Levit, "Visually accurate multifield weather visualization," VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03). Washington, DC, USA: IEEE Computer Society, 2003, p. 37.

[9] K. Riley, D. S. Ebert, M. Kraus, J. Tessendorf, and C. Hansen, "Efficient rendering of atmospheric phenomena," Rendering Techniques 2004: 15th Eurographics Workshop on Rendering, June 2004, pp. 375–386.

[10] Y. Song, J. Ye, S. Lasher-Trapp, and M. Baldwin, "An atmospheric visual analysis and exploration system," IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 5, pp. 1157–1164, 2006.

[11] T. Jiang, T. Wasilewski, N. Faust, W. Ribarsky, B. Hannigan, and M. Parry, "Acquisition and display of real-time atmospheric data on terrain," Eurographics-IEEE Visualization Symposium, Eurographics, 2001, pp. 15–24.

[12] S.-K. Ueng and S.-C. Wang, "Interpolation and visualization for advected scalar fields," IEEE Visualization, 2005, p. 78.

[13] P. R. Harasti, W.-C. Lee, and M. M. Bell, "Real-time implementation of vortrac at the national hurricane center," 33rd Conference on Radar Meteorology, 2006, p. P11A.6.

[14] C. Baru, R. Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker," Proc. CASCON'98 Conference, 1998.

[15] TRMM Radar Software Library (RSL), http://trmm-fc.gsfc.nasa.gov/trmm_gv/software/rsl/index.html

[16] B. Domenico, J. Caron, E. Davis, R. Kambic, S. Nativi, "2002: Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL," Journal of Digital Information, 2.

[17] Alberoni, P.P., Mezzasalma, P., Costa, S., Paccagnella T., Patruno P. and Cesari, D., 2000: Doppler Radar Wind Data Assimilation in Mesoscale Analysis. Phys. Chem. Earth (B), 25, No. 10-12, 1263-1266.

[18] W. Schroeder, K. Martin, and B. Lorensen, The Visualization Toolkit, Third Edition. Kitware Inc., 2003.

[19] Yi Ru, Volumetric Visualization of NEXRAD Level II Doppler Weather Data from Multiple Sites, M.S. Thesis, Dec. 2007.

[20] Tannenbaum, T., D. Wright, K. Miller, and M. Livny, "Condor - A Distributed Job Scheduler", in Thomas Sterling, editor, Beowulf Cluster Computing with Linux, The MIT Press, 2002. ISBN: 0-262-69274-0.

[21] N. Oceanic and A. A. N. W. Service, "Noaa nws," 2007. [Online]. Available: http://www.nws.noaa.gov.