USGS
science for a changing world

A Distributed System for Fast Reprojections of Geospatial Data for *The National Map*

Brian G. Maddox
Mark Schisler
Mary Beth Danuser

Open-File Report 2007-

U.S. Department of the Interior

U.S. Geological Survey

CONTENTS

ILLUSTRATIONS

KEY WORDS

Fast reprojection geospatial data

ABSTRACT

One of the issues with the concept of Web Map Services is in working with data from multiple cooperators. These data may be in different projections that the service does not support and might need to be converted to a common coordinate system. Alternatively, having a Web Map Service try to project data could be a slow operation, resulting in a user having to wait for the data to be transferred to their application. A solution to this is to use distributed processing and mathematical interpolations to project the data between coordinate systems. Alternatively, a new method of wide-area distributed processing can be used to ensure that there is enough computational power for such a system.

INTRODUCTION

With the introduction of *The National Map*, data holdings of the United States Geological Survey (USGS) are transitioning from the traditional model of hosting data in national standards formats to using cooperator data from different states in different formats via Web Map Sevice (WMS) systems. The differing formats include file types, coordinate systems, photometrics, and other specifics.

This may create a problem if an area of interest involves an area that is covered by multiple data providers in different formats. This forces the data user to reproject the data, or the data cooperator to reproject the data before sending it to the user. In the former case, it can inconvenience the users as they must manually reproject the data before they can use it. In the latter, it can cause the cooperator's server to become overloaded from the processing required to reproject data for multiple clients.

A distributed system is an obvious solution for providing reprojection services for geospatial data from WMS. Such a system would have the capability to scale to fit processing loads, and would allow the USGS to provide data in almost any requested projection.

BACKGROUND

A question one might have with data reprojections is “Why should I worry about it?” There are several reasons that it's important to have some form of reprojection capability. First, to be a part of *The National Map*, a cooperator really only needs to guarantee the Geographic Projection of the data. This allows multiple cooperator data to be displayed together, but does not necessarily fulfill high-end Geospatial Information System (GIS) requirements. These users will want data in their preferred coordinate system and in their preferred file format.

Cooperators are neither required to offer multiple projections nor are they required to perform the reprojections with any specified method. Methods to reproject may range from a simple, but inaccurate, one-dimensional interpolation to a rigorous point for point cartographic transform. Free-form reprojections can result in data that do not fit together because of differences in the mathematical accuracy of the method chosen. Having a single service providing reprojection capability can ensure that the mathematical transforms are rigorous in accuracy and that the data will fit together by obtaining them in native format from the cooperators and then reprojecting them before they reache the end user.

Reprojection can also be a numerical and data-intensive activity. Some cooperators might not have the capability to offer reprojections due to a lack of computational

resources. Others might be able to reproject, but they might not be able to handle large numbers of requests as the popularity of *The National Map* increases. Providing a scalable and powerful reprojection service encourages cooperators to provide data as it removes the burden from them.

Having data in a common projection system has more uses other than simple screen display. Hazards response activities, for example, would require different data to be in a common coordinate system that might not be available from all cooperators. Having to download the data and reproject it on their own might not be possible due to time constraints and lack of computational resources in the field. Home users might want a full-sized dataset in a common system if they live in an area that is covered by multiple providers in multiple systems. In this case, reprojecting full data requires more processing power than a simple screen display.

Work in reprojecting large amounts of geospatial data actually began on a National Mapping Discipline Prospectus project. Called "A Parallel Approach to Computing for the Geographic Sciences", this project explored how distributed processing could be used to further the mission of the U.S. Geological Survey. The National Geospatial Technical Operations Center III component of this project involved developing a system that would use distributed processing to convert very large geospatial datasets between various map projections. For more information about this research, please consult USGS Open File reports 2001-465 and 2003-117.

An idea came out of this research to use Mosix to automatically distribute the processing load between multiple computers. Mosix is a series of patches to the Linux kernel that allow automatic and transparent process migration from heavily loaded computers to those that have processing capability to spare. All of the machines in a Mosix cluster work together to ensure that the process load is spread equally amongst themselves. More information can be found on the Mosix website at http://www.mosix.org.

Mosix could be used to provide scalable processing capabilities for systems performing large amounts of numeric and data processing. A method first outlined in USGS Open File Report 2004-1091 has been developed to utilize all of the machines in an organization to form a large Mosix-based supercomputer. In this method, each computer would run Linux with a Mosix-enabled kernel, while other applications could be run through translation layer or another operating system could be run on top of Mosix via an emulator. This idea is similar to Grid computing, but differs in that applications do not need to be specifically written to take advantage of Mosix. Mosix also allows features such as automatically migrating processes when a machine is shutting down, or allowing a machine to leave or join a Mosix cluster on demand. This project was developed with the intention of being run on a Mosix cluster to provide scalable high-end computational resources for near real-time map reprojections.

METHODS AND TESTING

There were two parallel software development projects undertaken for this project to provide a proof-of-concept. The first was to modify the Prospectus-developed software so that it would more easily run under Mosix. The original software was designed to use a message passing library and followed a master/slave processing pattern. The master process dispatched work to the slaves and received work back from them. All of the processes using the message passing system had to be manually started on each machine. For this to run under Mosix, the system had to be rebuilt so that it would use the standard Unix fork() system call so that Mosix could automatically migrate the slave processes to balance the load. The other part of the project was to develop a front-end WMS to accept connections from outside clients. Originally, this design was to act as a virtual WMS by pretending it had all of the data listed in *The National Map* Catalog in any projection.

WEB SERVICE FRONT-END

Currently, the front end server is multi-threaded and uses the ADAPTIVE Communication Environment (ACE). Two different types of requests that can be made using this server are MapReprojection and Capabilities requests. In the MapReprojection request, the client will use SOAP HTTP POST binding to send the request, and will receive a map projection. A capabilities request is a request for what capabilities the server has, and is done using a HTTP GET request. An XML file containing the information about what capabilities the server supports is returned.

In a MapReprojection request, a client to the front end will send several URL's that they have obtained through the National Map Catalog. It will also send a statement of what data they want projected, the projection system information (such as what type of projection and the parameters of the bounding box), and the format of the output file. The requests from the client will include portions of multiple datasets that the client wants merged. The server will then pass these on to the proper data providers, and download the data into a common directory between the front-end and back-end servers. The front-end then opens a socket to the back-end and sends the projection parameters, including the bounding box, the projection information required for output, the file type, and where the files are located. During this time, the front-end must keep all of the connections open between itself and the client and back-end server. After the back end server has processed the data and writes it to an output file, a signal is sent to the front-end, letting that server know that it has completed the request. The front-end server then takes that information and sends back to the client a new bounding box and reprojected output file.

Originally, the front-end system was planned to be a virtual WMS that would be listed in *The National Map* Catalog as having all of the data in the Catalog in multiple projection

systems. The initial version of the front-end would actually take an incoming request from a client and query the Catalog to get the data to pass on to the reprojection engine.

There were several problems found with this arrangement. The first is that there is not a good way to handle a virtual WMS according to specifications. Normally, a WMS is set up to provide data that are stored in a database by the cooperator. The virtual WMS would not actually hold any data. It instead would provide the data on-demand by downloading it and having the reprojection engine convert it to the client's specifications in near real-time. This way, someone wishing to use the engine would directly interact with the virtual WMS instead of the individual data cooperators. The case could occur, however, where a client comes to the virtual WMS and asks for data that are not available. This could be due to a problem on the cooperator's servers, or a change that has not yet been propagated back to the Catalog.

During normal operation, a client would query the Catalog and then contact the individual data cooperators to get the required data. If a cooperator's server happened to be down, or the data happened to be missing, that server could send an error message to the client so it would know that specific piece of data was unavailable. However, the operation of the virtual WMS would be to allow a client to obtain a single data file from multiple cooperators in a common projection system. If, for example, one out of three cooperator's servers happened to be down, there is not an effective means to tell the client that one-third of the data are missing. The ideal way would be to try to contact the client and negotiate if a partial dataset is acceptable or if it has to be an all-or-nothing approach. With the standards currently in place, it would always be the all-or-nothing method.

Another issue with the virtual WMS is the load it would place on the Catalog. There could be twice as many queries for each client request generated by the virtual WMS to deal with the problems previously mentioned. These multiple queries would ensure the existence of the data and their accessibility. Combined with the normal load on the Catalog, this could impact performance as it tries to respond to the usual daily requests.

To address these problems, the decision was made to turn the front-end server into a standard web service (WS). A WMS is a specific form of a WS, so clients that can communicate with a WMS should also be able to speak to a WS with minor modifications. By moving to a standard WS, the server could define it's own interface and be able to easily handle the problems that were associated with being a virtual WMS. For example, it can give clients the option of an all or nothing approach, for example. It could also deal with the Catalog load problem by requiring that clients pass in the URLs that they receive from the Catalog. This way, the WS can contact the cooperators without having to requery the Catalog.

DISTRIBUTED REPROJECTION BACK-END

Design Challenges

The Mosix fast-reprojections project was assembled from various pieces of code, making it very difficult to adapt to meet growing requirements, and indeed even to meet the minimal ones for which it was originally designed. It primarily was comprised of fragments of code from the Message Passing Interface (MPI) reprojections project, and a 1-1 projector written around five years ago.

Some time was spent learning the inner-functionality of the code and its relationships. Once this had been done, several major bugs were fixed which allowed the project to function for 1-1 image reprojections in an automatic load balancing environment. Shortly thereafter, it was requested that the project work for n-1 image reprojections. At that time, it was judged that the best way to approach the old code base in the future would be by rewriting it in a more object-oriented fashion, thus allowing it to be more adaptable.

The greatest weakness of the original project was its complexity in not only its own code base, which was arbitrarily modularized and written in a very ad-hoc fashion, but also in its use of several large polymorphic libraries that were complex in their own rights. The original code was implemented for a research product, not for production use. To ease development for posterity, a Facade Pattern (Gama et. al., 185) was implemented to make a common interface for an abstraction called a ProjImage. The ProjImage encapsulated two large systems: the USGS ImageLib system of classes, and the USGS ProjLib system. Polymorphism was then used to separate Mosix reprojection code from non-Mosix reprojection code by creating a Mosix projector object and a regular projector.

To accommodate n-1 projections, the Composite pattern (Gama et. al., 163) was used in the ProjImage system. A ProjImageList inherits from the same interface as the ProjImage, making it a simple matter to use the same projector object to perform n-1, 1-n, or n-m projections.

Finally, to insulate the user of the library from future changes to the construction of particular ProjImages, a ProjImageFactory (Gama et. al., 87) was written.

Capabilities

The web service, for which the back-end was designed, was to be equipped by the back-end with the ability to reproject anywhere from one to several images from a Geographic projection to a common projection and a common image. The back-end's capabilities surpass this as it now is able to reproject multiple images in multiple different projections to the same image. It takes input from a parameter file and is able to accept input image data from RGB or Grey-scale images in PNG, JPEG, GEOTIFF, or DOQ image formats,

and can output images in all of the aforementioned formats as well.

Currently, the code that performs the image reprojections is being reintegrated with more modularized, re-written code that takes care of the division of reprojection “work” between different “jobs”. After this integration is complete, the program will be able to function quickly in an automatic-load balancing environment, such as Mosix, because its reprojection “work” will be able to be divided and shared between multiple machines.

Algorithmic Details

Images sent to the back-end service are sent along with parameter files that contain all pertinent data concerning the image's projection, file name, and its Geographic bounding box. These data are then used to construct a ProjImageIn object. A ProjImageIn can be used alone with the primary Projector object, or a group can be assembled by the ProjImageFactory into a list of ProjImages. This ProjImageInList can then be forwarded to the Projector in the same manner as a ProjImageIn would be.

To construct the output image, a forward projection is first done on the outer-most geographic bounds of the input image(s). These outermost bounds are used, in conjunction with another parameter file, the input photometric (RGB, or Grey scale), and the samples per pixel of the input imagery to create a ProjImageOut. The Projector does a reverse projection to get the pixels for the ProjImageOut, or the destination image. In other words, it iterates over the blank pixels of the new image, translating each pixel into a geographic coordinate. It then passes these coordinates to the ProjImageInList or ProjImageIn to see if there are input data to fill this blank pixel. If there are, a pixel is returned and its data are copied to the output image. If not, a sentinel (or “fake”) value is returned, and that pixel is left blank.

The Mosix Projector currently uses the USGS MathLib's interpolation capabilities to estimate the location of reprojected coordinates by forming a mesh over each reprojection transformation. This is done as a technique for shortening runtime. In performing an n to 1 projection, the number of meshes varies linearly with the number of input projections. In the case that the input images are not all in a Geographic projection, and there are multiple input images, one will have n + 1 meshes used to perform a reprojection, where n is the number of input projections. In all other scenarios, n meshes will be used. The additional mesh is used in the former case to reproject all input image extents to an intermediate projection, Geographic, before being projected to the output projection. These intermediate extent coordinates are then saved. The common intermediate projection acts as a universal coordinate system. It becomes clear that this “universal coordinate system” was added for efficiency, when one considers that it would otherwise be necessary, in the worst case, for every pixel coordinate in the destination image to be reprojected to every source projection to tell from which source image the destination image's pixel should be drawn when doing the reverse projection to construct

the body of the output image.

Problems

One problem has been noticed with the process described. The “extents” of the input images, when projected to a Geographic coordinate system and successively to the output projection, seem to be garbled a bit. This might be due to a difference between the mapping of source projection coordinates from the Geographic projection to the destination projection and a source to destination mapping, or because of accentuated error in the interpolation that the projector uses to estimate the extent's coordinates. This garbling, whatever the cause, results in a final miscalculation of the needed size of the destination image, which in turn causes the resulting image to lose source data. As this functionality will probably not be used in the final release because all source images will be in a Geographic projection, it has not been judged as an immediate issue. Several possible solutions to this problem are being explored so that this code may be reused in future projects.

Load Balancing and Means of Output

As mentioned earlier, the Mosix Projector is used to perform automatic load-balancing by dividing the work that needs to be done. In the case of this project, the “work” that is divided is the actual reprojections themselves. The Fast-Projections project divides the output image into multiple horizontal chunks, or groups of rows of pixels. Then each “chunk” is given to a Projector, whose exclusive duty it is to reproject that portion of the image. These multiple projectors are then submitted as individual processes in the operating system environment, where they can be distributed among other computers in the cluster. A data “receiving” process is kept running on the machine that started the reprojection. What is sent back from the Projectors to this process is the reprojected portions of the output image. Because these pieces can come back from the migrated Projectors fairly quickly, code has been integrated into the project to allow for a light-weight process, or a thread, whose dedicated purpose is to write output data to the disk. This will theoretically allow the computer to more easily balance the duties of receiving input and writing output as the tasks will be coordinated by the operating system.

The resulting image will usually arrive back to the image writer non-sequentially. To speed up the sequential writing of images to the disk, it had been suggested that jobs be divided up into progressively larger chunks. This would make it more likely that the image would arrive back to the data writer in a more sequential fashion. This functionality has yet to be integrated into the project.

The next two images illustrate outputs from the current experimental version of the distributed projector.

Figure 1: Two tiled UTM images of Des Moines, Iowa reprojected to a common image. A demonstration of the clipped-boundary problem. Notice the missing data on the right boundary of the image.

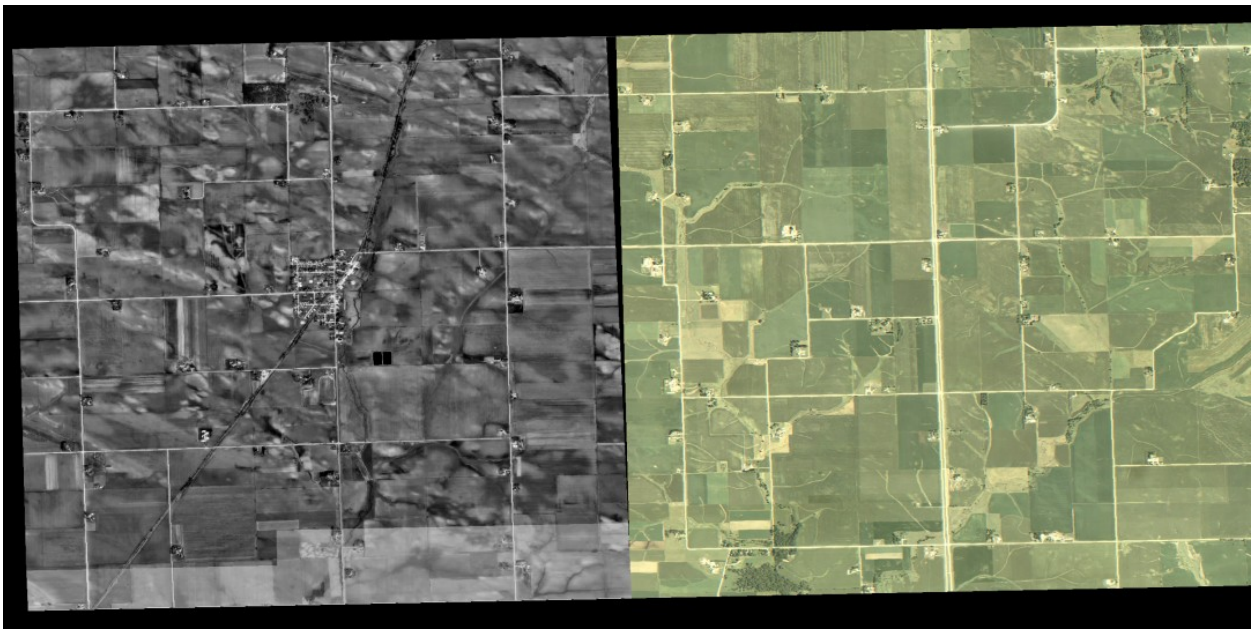

Figure 2: A demonstration of the output capabilities of the project. Here black and white and RGB data have been combined into a common image.

DISCUSSION

This project is in many ways an extension of a previous research Prospectus project, and demonstrates how research can be turned into operational capability to further the mission of the USGS. The original research project went through several iterations: a single-threaded test system; a Parallel Virtual Machine-based application; a MPI-based application; and finally to a Mosix/MPI application. This project then took that code and is turning it into a production-level system, removing the MPI portions and replacing it with standard Unix fork() calls to create the clients so that Mosix can automatically balance the load. It also is refactoring the code so that it follows standard software development methodologies; something that is not important to research, but is important to production capabilities.

This work opens up additional functionality for *The National Map*. The USGS will be able to provide near real-time reprojections of data sources for users. This is important for two reasons. The first is that it gives functionality to make things easier for developers writing client applications and for users who want to see various images on screen. The other is that with the USGS providing this service, it allows for a controlled and standard method for performing the reprojections. Instead of relying on different cooperators to provide different methods for doing reprojections, the USGS can provide a standard method to accurately combine and reproject data.

The second capability is to provide data in multiple file formats. These formats are

standard and widely used in Internet applications. This gives software developers and users the ability to deal with standard file formats instead of dealing with multiple individual file formats.

This system is also scalable to any number of machines that can be made available for processing. Mosix allows machines to join and leave a processing cluster at will. This makes it easy, for example, to allow machines to join the processing cluster in the evening when employees go home and leave in the morning when they come to work. Additional machines can also be added on demand as the processing load increases. This allows the USGS to provide fast and accurate reprojections and allow *The National Map* to be able to easily provide the quilt characteristics of "weaving together different datasets from different cooperators."

The design of this system allows it to work on a wide range of systems. It can run on a single computer by running one slave process. This allows it to be run in situations where multiple machines cannot be allocated to the system. As more machines are added, more slave processes can be run, speeding the runtime of the overall system. This design decision was made to be as accommodating as possible. However, the project can also make use of the wide-area processing model as discussed in USGS Open File Report 2004-1091. In this model, an entire organization's computers can be used for high-end processing tasks.

As can be seen from the sample images, this project is rapidly nearing completion. It is already capable of quickly reprojecting single or multiple images. Some work is left in interpolating data values in order to fill in any gaps that might appear when data are reprojected. These images also show that intelligent interpolations can be used instead of a point-by-point transformation. This allows processing to be sped up even more as interpolations are much faster mathematically then the actual full equations to do a point-by-point reprojection.

FUTURE WORK

One of the items necessary for future work is to finish the interpolations for pixel values. This interpolation is necessary as reprojections can sometimes modify the data so that gaps may appear in the output unless value interpolations are performed. This can make the difference between having an output image with black speckles throughout or having an image where each pixel is filled with a value. Lines that can sometimes form when multiple images are reprojected to a common system can also be filled in with value interpolation. In this case, a simple bicubic interpolation can be used to find the output pixel value.

The download service also needs to be finished at some point in the future. Currently, the only actual GIS file format it supports is GeoTIFF. Support could be added later for formats such as SDTS, ArcGIS, and so on. This can be plugged in fairly easily thanks to the new modular architecture that has been implemented.

Integration of the network code, which was previously worked on by another student, with the new code base is currently in progress. Because of the new object model, communication of the "projector" and image data between different processes has become a bit more difficult to handle. The significance of this obstacle should not be overlooked.

CONCLUSION

This project is the final step in going from research to production. The research component proved that the theory of distributed processing of datasets would be more efficient and provided the accuracy to support interpolation of the datasets. This step then took the research and refactored it to a production-ready system that can be used to further the capability of the USGS *The National Map*.

This system allows data from multiple providers to be converted in near-real time to different coordinate systems. This makes it easier for cooperators to provide data as they do not have to implement reprojections on their own. It also ensures that the reprojected data are accurate as it offers the reprojection in a controlled environment, allowing the USGS to ensure that the data meet national standards.

Allowing data to be downloaded in a variety of formats ensures that there is no vendor lock-in and ensures that the data are as user-friendly as possible. Different needs can require different file formats, from heavy GIS-formats to lightweight image-only ones. This can allow *The National Map* to meet a wide variety of needs.

This system can also be used on a wide-range of computational environments, from single computers to a processing cluster. It is optimized to run under Mosix where each machine running Mosix acts as part of a larger computational system. This can then be used in the concept of a wide-area processing network, ensuring that the processing capability can scale to meet current and future needs.

The system's use of good object-oriented design practices makes it more easily extensible so that it can adapt to changing program requirements, and/or even be adapted for uses which were not necessarily foreseen. The back-end server's ProjImage system of classes could provide a good C++ API for anyone who was in need of a wide

range of image reprojection functionality, without the want or bother of the fine-tuned systems which were previously available for performing such operations (e.g., the USGS ImageLib and ProjLib).

REFERENCES

Barak, Amnon, Professor. *Mosix Cluster and Grid management*. <http://www.mosix.org>. Hebrew University, Jerusalem. 2005

Crane, Mike et.al. *A Parallel-Processing Approach to Computing for the Geographic Sciences: Applications and Systems Enhancements*. USGS Open File Report 01-465. United States Geological Survey. 2001.

Gamma et. al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading. 1995.

Maddox, Brian G. *Distributed Processing of Large Datasets: A Preliminary Study* USGS Open File Report 03-117. United States Geological Survey. 2003

Maddox, Brian G. *Using Mosix for Wide-Area Computational Resources*. USGS Open File Report 04-1091. United States Geological Survey. 2004.