HPC Design for Estimating High Resolution Weather Conditions

Dorren R. Schmitt[1]
The Weather Channel
Atlanta, GA

And

John V. Matthews
The Weather Channel
Atlanta, GA

Introduction[1]

The Weather Channel (TWC), based in Atlanta, Georgia, designed and deployed a system called High Resolution Aggregate Data (HiRAD) that produces near real-time reports of current conditions within the Conterminous United States (CONUS). Within this domain, HiRAD estimates values for 61 surface variables at high resolution in time and space. The HiRAD system was described in detail in Neilley and Rose (2006).

This paper describes the operational infrastructure used to execute HiRAD on a high-resolution 2.5km grid and how the hardware and software specifications were measured. This paper describes the methods of integrating commercially available software in a non-traditional manner to provide the basic distributive infrastructure to allow HiRAD to run in near real time.

HiRAD uses a parallel high performance compute cluster designed to distribute the work of computing the current condition variables across 124 CPUs. This was accomplished through employing scheduling job software (PBSPro and MOAB), maximizing Network File Systems (NFS), and the scaling of the HiRAD software. The system was designed to maximize the server hardware for deriving the observation data and executing additional jobs to render and distribute the data to external systems for display and product creation.

The end result is 1.9 million observations across CONUS in less than six minutes. The HiRAD system synthesizes the raw data into 61 fields and distributes all but 18 for each of the 1.9 million points.

This paper is divided into four sections. Design Considerations describes the requirements driving the design. Design Implementation describes the design choices. Scheduling of Jobs describes how PBS and MOAB were employed to run HiRAD. Design Expansion describes the extensibility of the design for a growing application.

Design Consideration

In addition to computing observations in less than six minutes, HiRAD required four nines of availability. This availability requirement impacted many choices in the design as robust solutions were needed.

[1] Corresponding and presenting author. Dorren R. Schmitt, TWC, 300 Interstate North Pkwy, Atlanta, GA 30339, USA; email: dschmitt@weather.com

HiRAD is an application that divides CONUS into irregular polygons (called tiles). The tiles are designed to most efficiently use the available data to derive all necessary calculations for the 61 fields. Figure 1 displays how CONUS was subdivided into tiles.

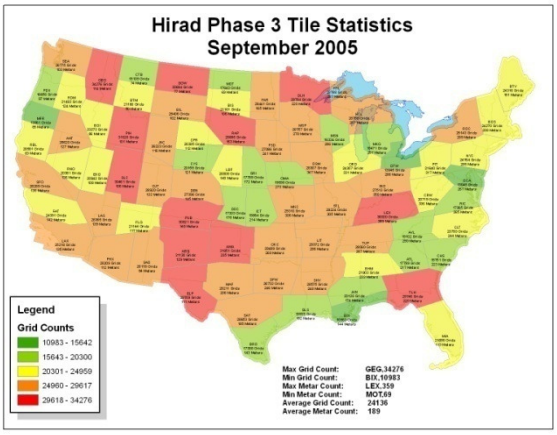


Figure 1 Neilley and Rose (2006)

The first consideration was how to design the distribution of the tiles. The software used for this needed to serve these present needs but be extensible for future growth. The software needed to be able to provide automated deployment but also have capabilities of allowing for interstitial deployment either for testing or for over-coming operational issues.

Although similar to a Beowulf cluster, the parallel computing cluster was not designed for processing to be explicitly shared. Each tile runs as an independent "application". This design feature of the software greatly impacted the software choices.

The second design consideration was the computer hardware. The obvious objective was to minimize the number of nodes. But, at the same time, there was the need to ensure expandability and fault tolerance without significant future capital purchases. Considerations included types of CPUs, RAM, network interoperability, storage requirements, and data distribution.

Another key aspect of the design was the operating system. The operating system needed to be one that allowed flexibility for doing parallel processing and provided opportunities to be able to choose from a variety of scheduling software. This was integral both for determining the clustering software but also for application writing.

With an application that was still being developed, there was a certain level of needing to take an educated swag at these requirements. So, a key feature of designing the core infrastructure was to enable it to be expanded at minimal cost to accommodate application requirements that were not yet well defined.

The next design consideration was the ingestion of the necessary input data. The amount of data and the number of data sources would be important factors in establishing how ingestion would be optimized. It was also clear in the beginning, that numerous protocols would be necessary (i.e. LDM, FTP, sockets connections).

Data unification was the next design consideration. The challenge was to determine the most robust and efficient method to make all of the data ingested available to all of the nodes in the cluster. And likewise, how to allow the results of the computation be available, synthesized, and then distributed. As part of data availability, the distribution method of the synthesized results was integral. The results needed to be available to other systems as soon as possible. Thus, the distribution choice was critical.

Design Implementation

Designing the HiRAD infrastructure was one that entailed several Proof of Concepts (POC). After trying several potential software packages, TWC decided to use a two application approach. The primary scheduling software was PBS Professional (PBS Pro) by Altair Engineering. The decision was made because this software provided economical and reliable job scheduling. Totally unsupported open source software, such as OpenPBS, was not an option because of the robustness of the design that needed to be executed. Additionally, another derivation of OpenPBS called TORQUE was available, but TORQUE did not have any failover capability of the scheduling head nodes.

There were also other commercial products available but either those software packages did not provide the level of robustness that was desired and/or the cost was prohibitive.

PBS Pro provided a good balance of support, significant development initiatives, and cost. The entry price point for PBS was reasonable and allowed for future growth without significant incremental costs. PBS Pro was used for the submission of the jobs to the compute nodes. One issue with PBS Pro was the difficulty of creating an automated job scheduling. PBS Pro was very good for submitting jobs and doing manual scheduling but unattended scheduling was not PBS Pro's strength.

To supplement PBS Pro, TWC also decided to implement MOAB from Cluster Resources. MOAB was a meta-scheduler that not only supported TORQUE but also PBSPro. MOAB provided the needed aspect of automated scheduling along with failover capabilities.

With this scheduling design, two head nodes were built to be the PBS and MOAB head nodes for the distribution of the jobs. Because these machines were solely dedicated to job submission, these servers are standard servers with 4 GB of RAM and two single core AMD CPUs. Because of the failover requirements of PBS, an NFS mount for the PBS software was created to allow both head nodes to share the same software install.

The second design consideration was the compute node hardware itself. The focus was to use the latest hardware available to minimize the hardware footprint and maximize the utilization. At the same time, fault tolerance needed to be built in to the design.

In considering servers, dual core AMD CPUs were chosen to minimize the number of machines. At the time of design, these were the first dual core CPUs. Additionally, it was determined that HiRAD was going to be a memory intense application. At the time of design, AMD system board and CPU design optimized memory more efficiently than Intel. Because of this, AMD chips were the choice for all systems in the design.

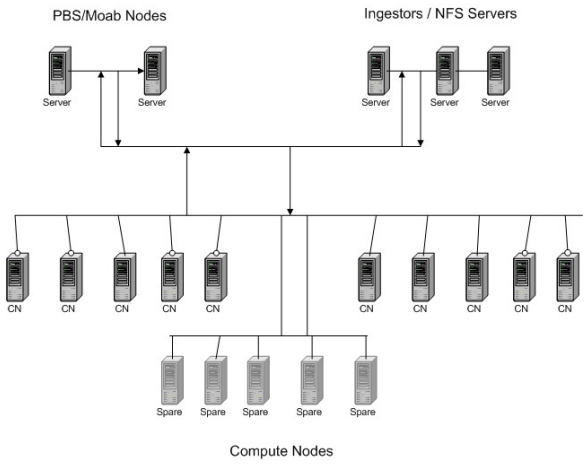
With the determination of the high memory usage, machines were purchased with 4 GB per core in the compute nodes. Thus a two socket, dual core machine would have 16 GB of RAM. Additionally, HiRAD was written on a 32-bit Linux OS and each job was being written to be single threaded. The software was designed that, a single job would not use more than 3.8 GB of RAM. This decision choice would ensure not to exceed the 4GB limit per process for a 32-bit OS. This compute node design would allow for a maximum of four simultaneous jobs to run on a single compute node most

efficiently without having the OS invoke page swapping because of limited memory resources.

To provide sufficient CPU resources for running all jobs at the same time and provide fault tolerance, 112 cores were decided for computing the tiles and running post processing jobs to render the results for distribution. At any point of time during a HiRAD run, there are at least 20 cores available in case of hardware, power, or network failure.

Figure 2 displays the hardware infrastructure for the HiRAD HPC. Two servers provided all the scheduling capability. There were three computers for the ingestion of data. One of the machines was a failover that could be used for either text or graphical data ingestion. And finally, a farm of compute nodes with spares provided the computing capability for the various tile and deployment jobs.

Figure 2
Hardware Design



The operating system early on was decided to be Linux. The next decision was to determine which distribution. The various distributions have their unique strengths and weaknesses. The decision was made to use CentOS. This is generally a stock distribution of Red Hat but without out the support and maintenance.

CentOS provided a low cost OS while using a standard distribution that allows for system administrators to support the OS. CentOS also provided the greatest breath of distribution software to choose from.

The fourth design decision was the ingestion of the raw input data. This process included determining appropriate data sources both public and private. Then for each vendor of data, what were the possible methods of receiving the data and finally, what was the most reliable method of the available choices. Whenever possible, secondary sources of data were investigated in case a vendor had either a data outage or a distribution outage.

Whenever possible, the primary choice of data receipt was to use Local Data Management (LDM) software. LDM is a reliable, robust delivery process that pushes data to ensure minimum latency. After LDM, File Transfer Protocol (FTP) was the most frequently used method. TWC identifies and pulls the data from a website or FTP server.

For data internal to the company that was needed as input, cross NFS mounting of file system and socket connections were the methods of choice. It was determined that these methods would provide data most reliably and nearly instantly. Since this data was internal, security concerns from these methods were not an issue.

Given the volume of data that needed to be ingested, two dedicated machines were implemented for active data collection. The first system ingested the vast majority of data. This data included NWS text data, computer

models, and lightning and provided the core of the information. This machine would run LDM, have FTP open and several other services to provide the flexibility to receive data in the most timely but secure method. All other systems would not have nearly as many services open for security purposes.

The second ingest machine was used to ingest and save graphical radar data. This machine did not have the numerous sockets open, but NFS was employed to ensure when the images were available from another internal system, that HiRAD received the data as soon as the images were created.

To maintain a robust system, a spare failover system was provided for redundancy. The design of the ingestion of input data was such that this third machine could run either the text or graphical data ingestion software but could not run both at the same time. The risk of this design would be the loss of both systems at the same time. This risk was determined to be low and acceptable.

The fifth design decision was how to distribute the input data to all the compute nodes for use by the HiRAD software. Given the number of machines that comprise the compute cluster, there were three possible choices.

The first choice was to sync the input data from the ingest server to all of the compute nodes several times an hour. This seemed to be inefficient. This method was prone to latency. There could also be inconsistencies if one or more compute nodes failed to get the new data before a HiRAD run needed to take place. There was also the issue of needing to purchase machines with large local hard drives. If the application grew, the compute nodes could run out of disk space. Then, there were issues with how would the compute nodes distribute the results. This choice in data distribution was eliminated quickly given the limitations.

The next choice was a global file system (GFS). When HiRAD was being developed in 2005, the choices of global file systems were not sufficiently robust to meet the reliability requirements. Global file systems were just emerging in the early 2000s as an alternative to NFS. Clusters had a file system, Red Hat had one, and HP developed a distributed file system. When these file systems were tested, it was determined that to provided the reliability necessary for this application, these file systems had not matured sufficiently. Many of the file systems actually stated at that time, they were not designed for production use. After sufficient research and POCs, using a global file system was not supportable for this application.

Finally, HiRAD could use a network file system (NFS). NFS has been a protocol of sharing directories or volumes since the 1990s. NFS was developed by SUN Microsystems and was now on the third version. This was a mature technology, well known by system administrators, and commonly used in information technology. NFS was determined to be the most robust choice to distribute the data.

Given NFS was chosen, the next consideration would be where it would be serviced out and the file system used needed to be decided. For robust journaling, Veritas File System (VFS) as opposed to Ext2 or Ext3 was determined to be the file system. Veritas was chosen for the robust journaling and proven history of being easily moved from one system to another. Additionally, since the text ingest server receives most of the data, it was determined that machine was the most appropriate one to

mount the file system locally and run the NFS services to export the file systems.

The text ingest server mounted a Veritas File System Volume from the Storage Area Network (SAN). This SAN mount has sufficient space for both the input data, but was also where the output data from HiRAD compute nodes would be written.

Each compute node uses NFS3 to mount the volume exported by the text ingest server. All compute nodes have the ability to both read and write to this volume. The compute nodes read in the data from the volume, places the data in memory, runs the computations, and then writes out the results to the same volume. Additionally, all log files and other necessary files needed to be shared with all nodes in the cluster are on the SAN volume.

The other advantage of the SAN volume is the ability to grow the volume. Through the past several years, the HiRAD application has matured with how data is distributed, what data is used as input data, and what data is produced. This in turn has required more storage. Unlike local storage, SAN storage has been able to be grown several times to allow for the expansion of HiRAD. Additionally, with faster CPUs and newer machines, moving this data would be simple. Instead of needing to copy the data from one local machine to another when new ingest servers were installed; all that needed to be performed is simply allow the new servers to see the storage on the SAN and mount the storage on the new servers. Instead of a day of copying, syncing, and re-syncing with the potential data loss and outage risk, the implementation of a new server would take minutes. This results in a significant reduction of risk and personnel time.

The last key design element of HiRAD is to determine how to distribute the results. Calculating the data does no good if the results cannot be distributed quickly and efficiently to all internal systems that have use of the data. Once again, NFS, sockets, secure copy, and FTP are the key methods.

The complication of cross mounting file systems from one application system to another is possible but not advisable. The biggest issue is if the exporting system has issues then it could produce issues for the mounting system. This was an unacceptable risk so NFS was eliminated as a viable option.

The second possible method was a socket connection. Although this is a viable method of distributing data, some of the applications the data would go to were third-party systems with no capability to have custom development to integrate this mechanism of data ingestion. The decision was made to at least initially use a single method. Since all internal customers could not use socket connections, this method was discounted.

The third viable choice was secure copy (SCP). This is certainly a secure method. The disadvantage is that SCP is a common protocol for Linux and UNIX but not as common for Windows. Once again, because of the variety of systems that need to have the data distributed to them, SCP was an option but deemed not to be the preferred method.

The preferred method to distribute data to the internal customers was FTP. The technology was a push technology. In other words, when the data was ready, the data was sent to the internal customers. The data was also sent as part of the job. This provided no latency.

Thus the infrastructure of HiRAD consisted of multi-core servers running CentOS and PBS Professional. Data were ingested on two separate server but all data were shared via NFS to all machines in the HiRAD cluster. Finally data were distributed to customers' systems via FTP.

Scheduling Jobs

Two software packages PBS Professional and MOAB were used to schedule the jobs and created all the data distribution.

MOAB was used for two purposes. The primary purpose was easy deployment of scheduled runs. HiRAD runs multiple times in an hour, every hour of the day, every day of the year. TWC needed a simple, configurable means that could failover with PBS Professional to run the master script that created the entirety of the HiRAD run.

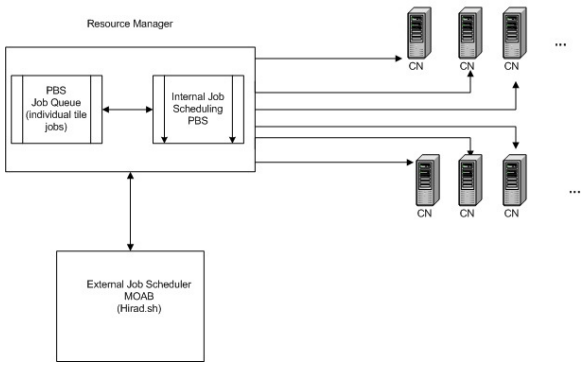
MOAB provided a simple configuration file that allowed for the definition of a scheduled deployment similar to cron. The deployment can be configured to re-fire multiple times in an increment determined by the customer. For example, for a user hirad the script HiRAD would run five minutes after the hour, and be retriggered every 20 minutes. Thus, HiRAD would run every hour at 5, 25, and 45 minutes after the hour.

The advantage of doing this over cron was the ability to fail over to the secondary PBS Professional head node if something occurs to the hardware or to the PBS Professional software. There could be some possible ways of using cron, but the risks for using cron was determined to be not worth the difference of using MOAB.

Another advantage of MOAB, albeit not often used, was the graphical capabilities. The user interface (UI) of MOAB was very good. MOAB allowed for easy interaction for deploying jobs through the UI as well as being able to see the compute node usage and progress of the jobs.

Figure3 provides a graphical view of the inter-relationship between PBS and MOAB. MOAB is the external scheduler that triggers a master script. PBS has all the core responsibility of deploying and managing the jobs.

Figure 3
PBS/MOAB Scheduling



Configuring PBSPro to deploy jobs required several deployment decisions. First it was decided, for simplicity, to deploy using the principle of "First in First Out" (FIFO). This is a common deployment scheme. Jobs are run in the order they are submitted. Since all jobs were essentially being submitted simultaneously and there were sufficient resources for the jobs to run at once, the method was not a critical decision. The second decision was how to distribute the jobs across the nodes. For this, the decision was to not load balance across the nodes but to deploy as many jobs on a single node as possible. Deploying the maximum jobs to a given machine allowed for spare idle machines and was determined to be the most robust method for this application. Additionally, all compute

nodes were configured as time-shared nodes. This was the most appropriate mode since multiple jobs would run on each machine. Each job was locked to a single CPU.

The deployment of jobs used internal scheduler and job queuing of PBSPro. The method HiRAD used for exploiting the power of PBSPro was the HiRAD launch script. By using a single script to deploy all jobs, HiRAD runs were easily deployed on the fly interstitially, if necessary.

The HiRAD script started off by running a pre-processing job to establish the list of current available data to be used in a run and format the data as required. An additional function of pre-processing was to determine if sufficient data were present to merit a run. This was a precaution in case of data outages by data providers to ensure that any results provided would have sufficient reliability. This single job was deployed on a compute node and was considered a blocking job. This means that the execution script waits until this job exits before submitting another job.

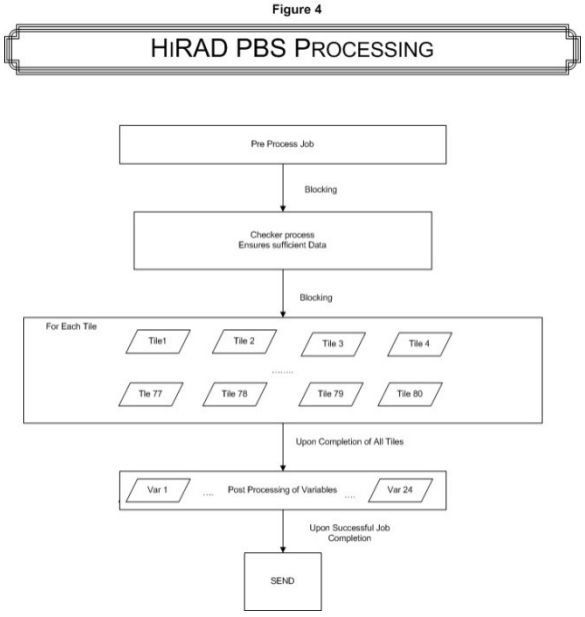
After the pre-processing job completes, the polygon jobs and the post-processing jobs were queued by PBSPro. The post-processing jobs were submitted using PBS such that they depend upon the completion of all the polygon jobs.

The next step was the execution of the polygon tiles themselves. These jobs run on the compute nodes. There are sufficient numbers of CPUs that all jobs run at the same time. Each polygon requires one entire CPU and nearly 4 GB of RAM. That quantity of RAM is necessary because data and libraries are loaded into memory to reduce the computational time as much as possible. Because of the varying sizes of the tiles, the tiles finish at different rates.

Once all the tiles jobs have successfully exited, then the post processing job executes. This job renders the data into files to satisfy the needs of customers. As this post processing job completes, the resulting output files are FTP'd to the appropriate customer. In total, there are 110 jobs that are run during each HiRAD run.

Figure 4 displays the process in which HiRAD is deployed and executed. There are two blocking jobs to ensure data integrity. There are also other job dependencies built in to ensure data integrity.

Figure 4



Design Expansion

Over the past several years there have been numerous changes in the design of HiRAD. Additional input data has been incorporated, additional variables have been produced, additional tiles have been incorporated, and different formats for the output data files have been created.

The strength of the initial design was the flexibility and ability to grow. Outside of

changing some of the dependencies of jobs and expanding the number of jobs, the numbers of compute nodes, CPUs, RAM requirements, third-party software such as LDM, PBS Professional, and MOAB have remained constant.

Given current technology, one area that is being considered to change is the method of using NFS. TWC is currently investigating eliminating the use of NFS on the ingest server and moving this capability to a device dedicated to doing that work. Network Attached Storage (NAS) is being seriously considered to provide enhanced robust storage capability and provide additional growth capabilities for the ingest server.

Summary

HiRAD is a parallel high performance compute cluster designed to distribute the work of computing the current condition variables employing scheduling software and NFS. The system was designed to maximize the server hardware for deriving the observation data and then submitting additional jobs to synthesize the data for graphical systems and distribute the data to the other systems for displaying the data.

Designing of the HPC comprised several key aspects to build a robust system to handle the demands of producing results three times an hour, 24 hours a day, 365 days a year. The first design requirement was to produce results as quickly and reliably as possible. Additionally, the HPC needed to allow for expansion of domain, variables, and data. And finally, the HPC needed to ensure that if hardware failure occurred that there was sufficient computing power to automatically recover from a realistic set of failure. The HPC was designed with the latest technology available in 2005. And with the exception of faster CPUs and more cores per chip available today, this design is the most efficient utilization of computer hardware.

References

Neilley, P. and B. L. Rose, 2006: A real-time system to estimate weather conditions at high resolution. *Proceedings of the 22nd AMS Meeting on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology, Atlanta, GA*.