A HIGH PERFORMANCE LDM DATA CLUSTER

Robert C. Lipschutz, David Hagerty*, Paul Hamer, Peter Lannigan*, and Chris MacDermaid
Cooperative Institute for Research in the Atmosphere (CIRA)
Colorado State University, Fort Collins, Colorado USA, and
NOAA Earth System Research Laboratory (ESRL), Boulder, Colorado USA

*NOAA Earth System Research Laboratory (ESRL), Boulder, Colorado USA
Contract with Riverside Technology, Inc., Fort Collins, Colorado USA

1. INTRODUCTION

Within the NOAA Earth System Research Laboratory (ESRL), the Global Systems Division (GSD) develops weather information systems, weather forecast models, and other applications in support of the National Weather Service, the Federal Aviation Administration, and other agencies. Well-known GSD products include the Rapid Update Cycle (RUC) model, the Local Analysis and Prediction System (LAPS), the Meteorological Assimilation Data Ingest System (MADIS), and Science On a Sphere® (SOS). A common feature of these and other projects is that they require observational and model data provided by acquisition systems running within GSD's Central Facility. These systems handle some 800 GBytes of incoming data per day as they acquire, decode, store, and distribute the needed data sets for GSD scientists and their collaborators.

To substantially improve its data systems, GSD recently commissioned a six-host cluster. This new Linux-based system replaces a collection of aging Linux High-Availability (HA) pairs and stand-alone platforms to provide much needed scalability and throughput performance, as well as excellent reliability, resource utilization and configurability.

In this paper, we detail the implementation of the cluster architecture, describing how the combined services of Red Hat Cluster Suite, Sun Grid Engine (SGE), *fcron*, and Unidata's Local Data Manager (LDM) have enabled us to construct a high performance real-time data processing environment that serves the GSD community. Given the central role of LDM in the architecture, we particularly focus on the how LDM provides cluster services for data transport, and on the use of event notifications in LDM.

Corresponding author address: Bob Lipschutz, NOAA/ESRL/GSD, R/GSD2, 325 Broadway, Boulder, CO 80305. Robert.C.Lipschutz@noaa.gov

2. LIMITATIONS OF THE HA ARCHITECTURE

While GSD's HA pairs have operated successfully for a number of years (Lipschutz and MacDermaid, 2005), and provided robust availability through their automated fail-over mechanism, a major drawback has been the need for two machines, one of which is mostly idle, to support a single processing configuration. Thus, system expansion or refreshment necessitated acquiring and supporting two new hosts at a time, adding to systems administration effort, power and cooling burden, and cost. In addition, while several HA pairs initially may have shared common system configurations, the individual nature of the systems resulted in drift in the configurations over time that became problematic to maintain.

At the application level, needing to individually configure each host's functionality was tedious. It often forced choices for where to locate processing based on where prerequisite data existed, rather than where system resources were available. This resulted in both unbalanced processing loads and more complicated data flows. Further, limited local disk space on the HA systems hampered processing of the ever larger data sets that need to be processed. Documenting the processing for a data set and

troubleshooting data outages was complicated by the need to identify the specific platforms where processing elements were performed.

3. GOALS FOR THE CLUSTER

In planning for the cluster, the intent was to address the HA systems' limitations with these benefits:

- Scalability – processing capacity can be added one machine at a time.

- Throughput – load balancing mechanisms allow for much greater aggregate throughput than was achieved on hand-configured systems. Of course, throughput also improves using newer, multicore processors and faster networking and storage devices.

- Utilization – cluster hosts are more effectively used than HA pairs in that half the hosts are not virtually idle backups. Rather, all cluster nodes are active, and load balancing methods ensure that jobs are distributed sensibly. In addition, the cluster's shared storage arrangement provides space to all cluster applications, whereas in the HA architecture, spare local disk space on one host is unavailable to another host that might need space.

- Maintainability – cluster hosts share a common system configuration, substantially simplifying ongoing system administration. Host reboots and failovers for patching also are easier than with the HA arrangement. On the application side, multiple configurations are replaced by a single, common set of configuration files. In addition to reducing the number of files to maintain, this also eliminates duplicate configuration items across hosts.

- Supportability – application data and log files reside on the shared storage, eliminating the need to know which specific host ran a job.

- Energy efficiency – higher efficiency in CPU utilization means less heat generation, and so lower cooling requirements compared to the HA computing infrastructure.

In addition to these general goals, we strongly desired to retain as much as possible of our long-established application software, thus minimizing code and configuration changes. We further recognized a continuing need, well served by the HA pairs, to be able to split onto separate platforms such high volume data streams as those from the NWS Satellite Broadcast Network (SBN - NOAAPORT) and the WSR-88D Level-II radars, and thereby use independent processing resources to minimize resource contention.

4. SOFTWARE COMPONENTS

To achieve our goals, we implemented the cluster with these software components:

- the **CentOS** Enterprise Linux operating system,

- the **Red Hat Cluster Suite** for cluster-wide application services and failovers,

- **Sun Grid Engine (SGE)** for job activation and load balancing,

- **fcron** for cluster-wide time-based job triggering,

- Unidata's **Local Data Manager (LDM)** for data transport and event-based job triggering, and

- **Open-E** storage software for controlling the shared (via NFS) Data Storage Server (DSS) RAID disk system.

Within this framework, GSD's Object Data System (ODS) applications (Hamer, 2005) are responsible for such data processing tasks as converting GOES Variable (GVAR) satellite data, Gridded Binary (GRIB) model data, WSR-88D Level-II radar data, and a variety of point observation data types into the netCDF formats

needed by GSD user applications. Only minimal changes to the existing ODS software were needed to accommodate the cluster's architecture. In addition, two new scripts were developed to provide common methods for submitting jobs to the SGE job queue from LDM (*pqact*) and *fcron*, while several old scripts were extended to include a cluster-wide locking mechanism to avoid concurrent instances of some jobs. Modifications to LDM scripts were also needed, as described in Section 5.4.

5. LDM ON THE CLUSTER

LDM has long served as the key middleware in GSD for real-time data transport and event-driven data processing services. In preserving the same LDM data processing capabilities used on the HA systems, the cluster architecture compartmentalizes data streams that deserve to be separated, but also shares across hosts the data and processing that can be accommodated on any host. To accomplish this, we differentiate data flow services from event handling based on data arrival “notification” messages.

5.1 *LDM and ODS*

In a manner similar to *pqact*, ODS client applications process desired data messages by attaching directly to the LDM data queue. For example, the *LdmGrib2Flat* client captures GRIB records in the NOAAPORT stream. The client writes the data into forecast-hour GRIB files that it opens in dynamically determined directories using the center, sub-center, model, and grid parameters found within the GRIB. Likewise, the ODS *LdmPoint2Tar* client writes selected point data messages into hourly tar files by data type, and *LdmNexrad2TarGZ* streams Level-II WSR-88D data into gzipped tar files by volume scan time for each radar station.

On completion of a GRIB file, *LdmGrib2Flat* emits a uniquely identified data arrival notification message indicating the location on disk of the newly available file (Fig. 1). For each GRIB file that requires subsequent processing, the associated notification identifier is specified in a *pqact* configuration, along with the command to be run. Given a desired notification, *pqact* pipes the notification message to a script, *qsubOnTextNty.pl* that extracts the GRIB file's path and then submits the specified command to the SGE job queue for execution. For instance, the ODS *Grib2NetCDF* application may be run to convert a GFS model GRIB file to netCDF format.

Point data tar files can be subsequently read by the *Point2NetCDF* program to decode the raw data and create netCDF files, for example containing METAR or maritime observations. These jobs, typically run on a schedule, are submitted to the SGE queue by *fcron* using the script *qsubCmd.pl*.

The key extension to ODS for the cluster was to enable specification of separate LDM queues for data services and notifications. Thus, the ODS clients process data from an ingest queue and insert notification messages into a local notification queue.

Figure 1. A sample ODS data arrival notification message.

5.2 LDM Data Services

To independently handle various incoming and outgoing data streams, a number of LDM instances are configured under Cluster Suite to run as relocatable cluster services. These services are normally set to run on different hosts, but two or more can also run concurrently on a

single host, if necessary in a failover situation. The specific hosts on which each service may run is determined within the cluster by defining Failover Domains, configured based on a desire to reasonably balance data volume across the cluster.

After considering the known data flows targeted for the cluster, we established the following five LDM services:

ldm-noaaport – ingest service for all NOAAPORT data

ldm-radar – ingest service for WSR-88D Level-II and WSI NOWrad radar data

ldm-other – ingest service for GOES GVAR and miscellaneous data types

ldm-outbound – distribution service for sending cluster data to other GSD hosts

ldm-fdr – Facility Data Repository service for saving data to GSD's Mass Store System.

A non-LDM sixth service, **MainIP**, establishes the cluster's canonical IP address and runs the single *fcron* daemon instance.

Each LDM service establishes a unique IP address in addition to its own queue, *pid*, and log files, enabling multiple *rpc.ldmd* services to coexist on a host. Thus, an external LDM may request data from the *ldm-outbound* service without knowing the particular host it happens to be on.

As an LDM ingest service receives data, its associated *pqact* and ODS client processes are responsible for writing the data to the shared storage. On successful completion of data files, the various data writing methods then emit notification messages to signal the events. And, rather than inserting the notifications back into their own service queues, the writers instead put the notification messages into a local "Notify LDM" queue.

5.3 The Notify LDM and SGE

The Notify LDM (*ldm-notify*) operates independently on all the cluster hosts to initiate processing on receipt of data arrival notification messages. These Notify LDM instances start up at host boot time using the local host IP address and identical *ldmd.conf* and *pqact.conf* configuration files on each host. While processing for many files simply entails copying from the shared storage to GSD's public file server, some files require additional format conversion, image remapping, or other post-processing. After each processing step, a new notification message is inserted into the Notify queue to provide a trigger for yet another step.

The key to the cluster's load-balancing architecture is this: rather than simply running jobs on the host on which a notification was emitted, jobs are submitted to SGE, which finds the least loaded host for the job. Since data files on the shared storage are visible to each host, and Notify LDM is configured identically on all hosts, processing is thus dynamically distributed across the cluster members. If a member leaves the cluster, say for OS patching, the remaining hosts pick up the work that would have been assigned by SGE to the missing host.

5.4 LDM Scripting and Configuration

To manage the data service and Notify LDM instances in a shared environment, several accommodations were needed.

First, we established each instance's home path on shared storage as */usr/local/ldm/ldm-<instance>*, where *instance* is *noaaport*, *radar*, *other*, *outbound*, *fdr*, or *notify*. Since the data services only run on one host at a time, the *ldmd.pid* file for each service instance lives in its instance home. Similarly, the *ldmadmin-pl.conf*, *ldmd.conf*, and *pqact.conf* files reside in *./etc* directories under their respective home paths and can be managed from any cluster host; log files reside under *./log*, and are visible from any host, as well. In contrast, to run independently on all hosts, the Notify LDMs require their *ldmd.pid*,

pqact.conf, and *ldmd.log* files to reside locally on each host. Thus, the Notify *ldmd.pid* files are written to the local path */var/run/ldm,* while the Notify *pqact.conf* files reside under */var/ldm/etc* to accommodate unique *pqact* 'state' files on each host, and *ldmd.log* files are written into local */var/ldm/log* directories.

To control the LDM instances, we extended the standard *ldmadmin* perl script. The modified *ldmadmin* looks for a shell environment variable, LDMINSTANCE that determines the path for the associated *ldmadmin-pl.conf*, which in turn sets the various instance-specific LDM parameters; without a valid LDMINSTANCE, *ldmadmin* exits with an error message. We further facilitated the use of instances by developing a new script, *ldmadmin-wrapper*, to which we linked *ldmadmin-<instance>* files. In addition to ensuring that the instance environment variable is correctly set, this cluster-aware wrapper also prevents inappropriate commands from being run. In particular, because the data service instances may only be started and stopped by the Cluster Resource Manager to ensure cluster consistency, *ldmadmin-wrapper* restricts which options may be run from the command line. Thus, '*ldmadmin-noaaport start*' will exit with a message listing the valid options that may be run (e.g., *pqactHUP*); '*ldmadmin-noaaport watch*' on the NOAAPORT ingest service host is valid and will display incoming message information, while on a different host it will simply indicate on which host that service is located and then exit.

We also needed to accommodate starting and stopping the LDM instances by automated methods. Specifically, the Cluster Resource Manager is responsible for data service activation, while the Notify LDM is activated at host boot time. To provide that function, we implemented an LDM resource script, */etc/init.d/ldm*, again linked by instance versions (e.g., ldm-noaaport), to cleanly start and stop LDM instances with the correct environment. Whereas a typical LDM resource script will check for a viable LDM queue and repair it if necessary, the cluster's version extends that concept by mounting a new *tmpfs* partition at startup and creating a new in-memory queue for the instance; similarly, stopping a service will unmount the partition. Using *tmpfs* for the queue ensures the highest possible throughput, and guarantees that an LDM queue is not corrupted on failover or reboot.

Finally, another *ldmadmin* modification forces each LDM instance to write to a specific log file, thereby overriding LDM's default *syslogd* logging method. This strategy avoids the problems that would arise from trying to handle cluster-wide logging through the host-based *syslogd* facility. It also eliminates needing to build and maintain separate instances of LDM to support independent logging. Because LDM does not presently have a method for closing physical log files (short of restarting), we also extended LDM's *newlog* utility to “zero out” the currently open log file after rotating the previous logs down by one version number.

6. LOGICAL FLOW

While there are numerous scenarios for processing the many types of data handled by the cluster, the following steps describe a typical flow, as depicted in Figure 2:

- On system startup, ldm-notify is started on all cluster hosts.

- The ldm-noaaport service is started by Cluster Manager on dc01 and begins receiving NOAAPORT data from its upstream source; meanwhile, the ldm-outbound service starts on dc03 and accepts connection requests from external clients.

- The *LdmGrib2Flat* client on dc01 reads from the ldm-noaaport queue and writes GRIB data into shared directories under /data/grib/noaaport.

- On completion of a GRIB data file, *LdmGrib2Flat* inserts an *LdmGrib2Flat_NOAAPORT* notification into the local ldm-notify queue.

- *pqact* on dc01 identifies the notification

message as one that requires a *Grib2NetCDF* action, and submits the configured job into the SGE queue.

- SGE finds the least loaded host in its list of available cluster hosts, and runs the *Grib2NetCDF* job on that host, say dc02.

- *Grib2NetCDF* on dc02 reads the specified GRIB file and creates a NetCDF file in the configured directory under /tmp_data/grids. When finished, *Grib2NetCDF* inserts a notification message into the local ldm-notify queue on dc02.

- *pqact* on dc02 identifies the notification message as one that requires a *moveTmpDataFiles* action to copy the file from the /tmp_data/grids location to the desired public server location, and so submits the configured job to SGE.

- *moveTmpDataFiles* is started by SGE on the least-loaded node, and on completion inserts a new notification message in the ldm-notify queue on that node.

- If necessary, another job may then be activated on the cluster, for example to send the netCDF file into the ldm-outbound queue for delivery to an external client.

As the activity driven by the data services proceeds, many time-based ingest actions, particularly ftp fetch jobs, are initiated by the *fcron* service. Like *pqact* jobs, the *fcron* jobs are also submitted to SGE, and so distributed evenly across the cluster. In addition to acquisition tasks, *fcron* jobs also perform many housekeeping tasks such as file purging and monitoring. Job activity on the cluster is monitored by the SGE *qstat* utility (Fig. 3).

7. CONCLUDING REMARKS

The cluster system now operating in GSD's Central Facility establishes a reliable, capable, and scalable new architecture for meeting substantial and ever-growing data ingest and processing requirements. While the learning curve traversed to successfully implement this system was relatively steep and, at times, arduous, the resulting system has been well worth the effort.

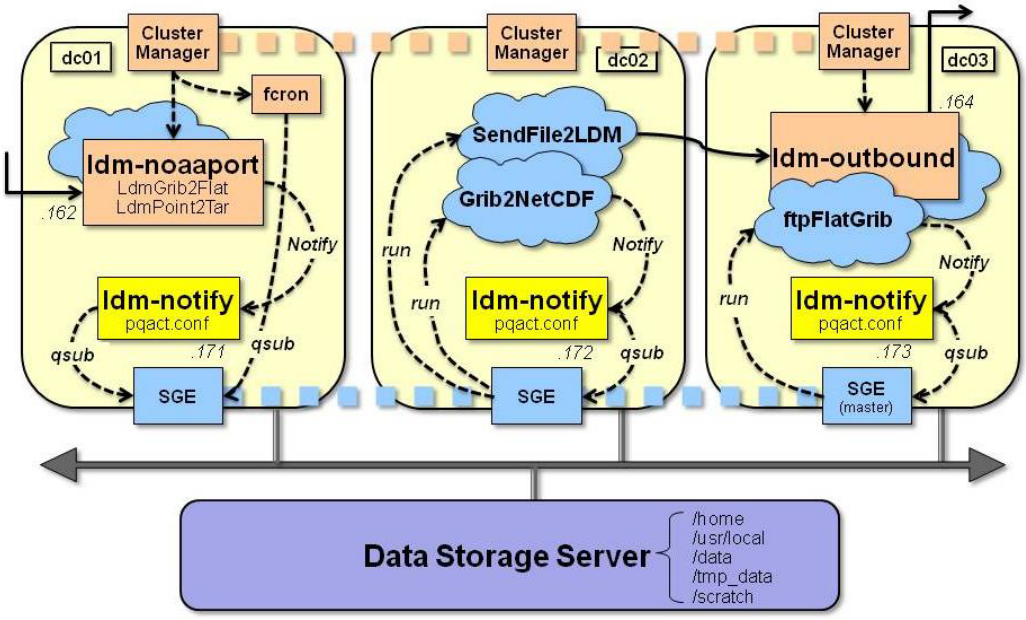


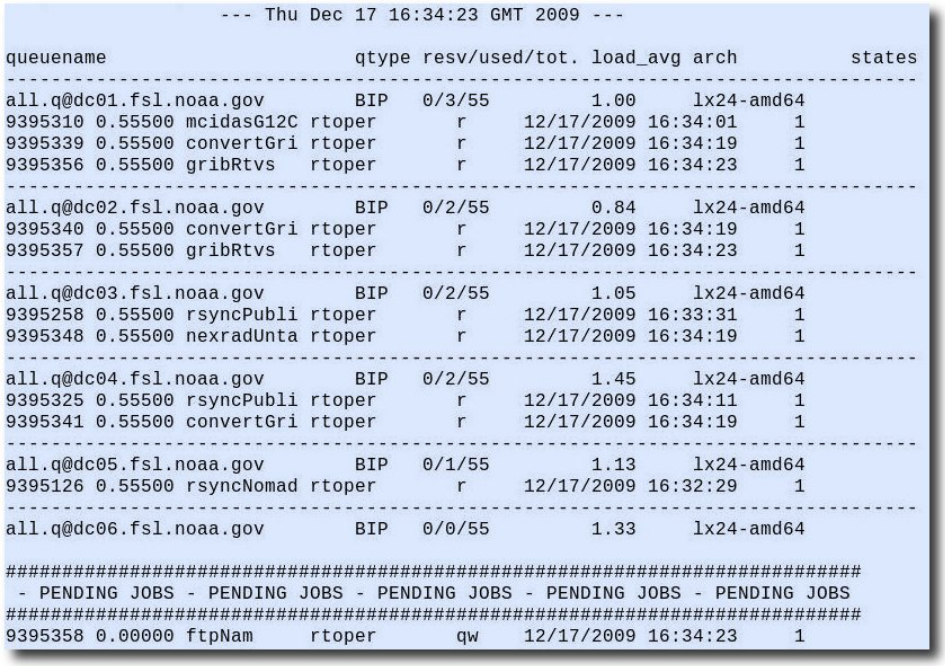
Figure 2. Cluster logical flow.

Currently, nearly 200 *pqact* entries are configured to handle specific data arrival events, and *fcron* manages some 230 different time-based jobs. Driven by the data flow and the work to be performed, we are presently logging over 160,000 jobs submitted to the cluster's SGE queue each day. We expect these numbers to grow as new requirements are added by our GSD data consumers.

8. REFERENCES

Hamer, P. (2005), FSL Central Facility data systems concepts, presented at 21st Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, American Meteorological Society, San Diego, CA, 2005.

Lipschutz, R. and C. MacDermaid (2005), Recent advances in the FSL Central Facility data systems, presented at 21st Int. Conf. on Interactive Information and Processing Systems (IIPS) for Meteorology, Oceanography, and Hydrology, American Meteorological Society, San Diego, CA, 2005.



```
                  --- Thu Dec 17 16:34:23 GMT 2009 ---

queuename                      qtype resv/used/tot. load_avg arch          states
---------------------------------------------------------------------------------
all.q@dc01.fsl.noaa.gov        BIP   0/3/55         1.00     lx24-amd64
9395310 0.55500 mcidasG12C rtoper       r     12/17/2009 16:34:01     1
9395339 0.55500 convertGri rtoper       r     12/17/2009 16:34:19     1
9395356 0.55500 gribRtvs   rtoper       r     12/17/2009 16:34:23     1
---------------------------------------------------------------------------------
all.q@dc02.fsl.noaa.gov        BIP   0/2/55         0.84     lx24-amd64
9395340 0.55500 convertGri rtoper       r     12/17/2009 16:34:19     1
9395357 0.55500 gribRtvs   rtoper       r     12/17/2009 16:34:23     1
---------------------------------------------------------------------------------
all.q@dc03.fsl.noaa.gov        BIP   0/2/55         1.05     lx24-amd64
9395258 0.55500 rsyncPubli rtoper       r     12/17/2009 16:33:31     1
9395348 0.55500 nexradUnta rtoper       r     12/17/2009 16:34:19     1
---------------------------------------------------------------------------------
all.q@dc04.fsl.noaa.gov        BIP   0/2/55         1.45     lx24-amd64
9395325 0.55500 rsyncPubli rtoper       r     12/17/2009 16:34:11     1
9395341 0.55500 convertGri rtoper       r     12/17/2009 16:34:19     1
---------------------------------------------------------------------------------
all.q@dc05.fsl.noaa.gov        BIP   0/1/55         1.13     lx24-amd64
9395126 0.55500 rsyncNomad rtoper       r     12/17/2009 16:32:29     1
---------------------------------------------------------------------------------
all.q@dc06.fsl.noaa.gov        BIP   0/0/55         1.33     lx24-amd64

############################################################################
 - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS
############################################################################
9395358 0.00000 ftpNam     rtoper       qw    12/17/2009 16:34:23     1
```

Figure 3. SGE 'qstat' output showing jobs distributed across the six cluster hosts based on system load. Job names (column 3) are descriptive of the work being done.