Software Design Document for the Land Information System: Data Management

Submitted under Task Agreement GSFC-CT-2

Cooperative Agreement Notice (CAN) CAN-00OES-01

Increasing Interoperability and Performance of Grand Challenge Applications in the Earth, Space, Life, and Microgravity Sciences

Version 4.0

Revision history:

Version	*Summary of Changes*	*Date*
1.0	Initial release.	8/13/02
2.0	Updated version	1/10/03
2.1	Revision on CT's feedback	3/14/03
2.2	Updated for Milestone I	7/11/03
2.3	Revision on CT's feedback	10/7/03
3.0	Updated version for Milestone J	9/07/04
4.0	Updated version for Milestone K	2/11/05

Table of Contents

List of Figures

List of Tables

1. Introduction

The Land Information System (LIS) is designed to perform land surface simulation and data assimilation on parallelized computing platforms, at very high spatial resolutions (up to ~1km) and in near real-time. It imports a continuous flow of atmospheric forcing data and a collection of land surface parameter datasets, and produces a huge amount of land surface data to satisfy the needs of diverse users. Such an operation poses many challenges to the data handling functionality of LIS, and requires a highly reliable and efficient data management design.

This document designs the LIS data management. Specifically, the design covers the following five functional areas,

- End-to-end data flow
- Data retrieval, distribution and storage
- Data analysis capabilities, including interpolation, re-projection, sub-setting, and file format conversion
- Link to the user interface
- Interoperability through ALMA and ESMF compliance

The goal of the data management design is to have a system which will ensure smooth end-to-end data flow for LIS' distributed and near real-time operations, handle huge amount of data efficiently, provide useful data processing capabilities, be easily accessed, and can couple with other Earth system models through the standard ALMA data format and ESMF interfaces.

This document is organized as follows. Section 2 presents the global data flow design and traffic estimation at the network level. Section 3 and 4 document the design details of LIS input data and output data, respectively. Section 4 is also concerned with the connection between the output archive and the user interface. Section 5 deals with interoperability considerations of the data management components.

2. Data Flow Design and Data Volume Estimation

2.1 Overview of LIS data flow

LIS deals with three sets of globally gridded data: atmospheric forcing data, land surface parameter data, and output production data. LIS uses the atmospheric forcing data to drive the time evolution of the land surface models at each grid/tile unit, defined by the land surface parameters. The land surface simulation is performed by each land surface model's dynamical and physical equations, forced by the atmospheric data. The simulated results, a set of land surface and atmospheric related variables, constitute the LIS products, the output data.

Figure 1 shows LIS logical data flow on the LIS cluster platform. On SGI Origin platforms, the flow is the same, except that local disk operations on SGI will take the role of the IO nodes on the cluster, and the compute nodes' tasks on the cluster are performed by processors with shared-memory.

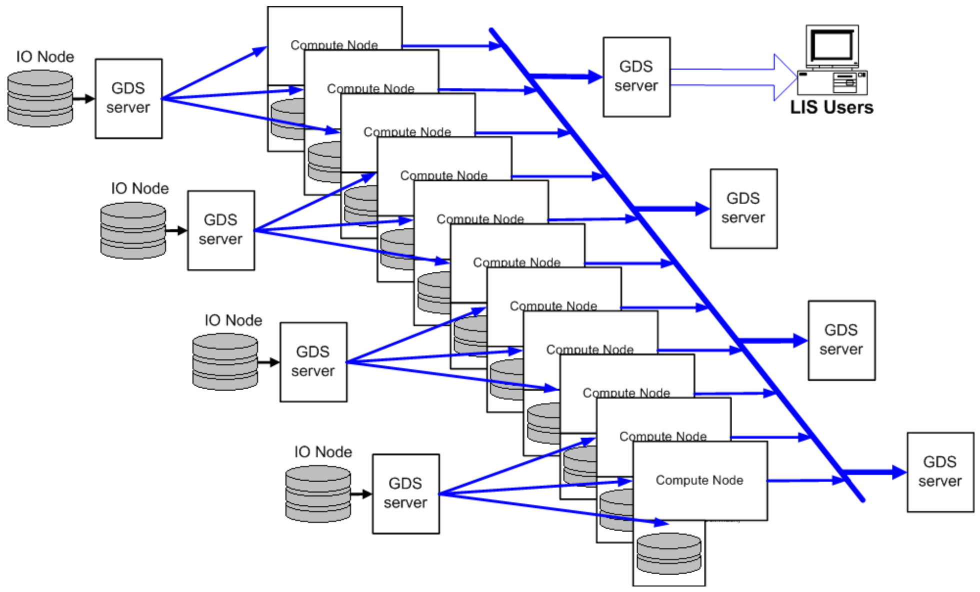

Figure 1: LIS global logical data flow and storage location on the LIS Linux cluster. The forcing data are stored on the IO nodes and served by the GDS servers running in parallel to the compute nodes, while an identical copy of the land surface parameter data are stored on each compute node, since their parameter data are mostly static. The output data are stored on the compute nodes' local disks, and we have developed a modified GDS server to retrieve data directly from these local disks on the compute nodes, with much improved performance.

As shown in Figure 1, the LIS end-to-end data flow involves two related areas: data input/output on hard disks and data traffic over the network. The design of the data management has to make sure of consistent data handling and high throughput without encountering bottlenecks in either area.

In addition, GrADS-DODS (GDS) servers play a critical role in transferring data to/from and inside LIS, so special design considerations and benchmarking have to be given to GDS. In Section 2.2 below we define the LIS internal data storage standard, and in 2.3 we estimate the disk data volume. Section 2.4 analyzes network traffic requirements, and 2.5 illustrates our design of parallel GDS servers (Figure 1) and their impact on performance. Section 2.6 shows our parallel output performance. We also implemented a peer-to-peer technology (Section 2.7) for high performance data replication on the cluster.

2.2 Data storage convention

The primary LIS data storage file format will be binary and GRIB. Binary format will be used whenever access performance is critical, and GRIB will be used to accommodate users' special requests. Both the binary and GRIB data will be saved in the sequence which allows direct access by GrADS and GrADS-DODS server without reformatting.

Data saved in binary files are ordered as big endian, to maintain backward compatibility with LDAS code. A detailed specification of LIS binary file convention is given in Appendix B.

2.3 Disk data volume estimation

LIS data are archived and served in three categories: land surface parameters, atmospheric forcing, and model output. The land surface parameters data are resolution-specific: LIS users need to download the data which match the resolution they intend to run. The atmospheric forcing data, on the other hand, are provided at their native resolution, and LIS code will perform interpolation to any one of the LIS-supported resolutions on the fly.

Table 1, 2 and 3 listed all the data files and specifications. As described in Section 2.1, these files will supply LIS with atmospheric forcing, land surface parameters, initial and boundary conditions required for the land surface model runs, and will store the LIS output to serve users. For example, the atmospheric forcing data translate to variables such as total precipitation convective precipitation, downward shortwave and long-wave radiation, near surface air temperature, near surface specific humidity, near surface U-, V-winds and surface pressure. In addition to the forcing files, the user also specifies parameters such as the spatial and temporal resolution, the land surface model, etc. LIS also allows users to initialize state variables, either by specifying a global uniform value or taken from a restart file produced by a previous run. Please refer to the LIS source code documentation[1] for a detailed description of the input/output routines corresponding to each file. The output from the land models translates to variables such as soil moisture, surface runoffs, canopy transpiration, etc. A list of the LIS variables passed between the modules, following the ALMA standard (Sec.5.1), is presented in Appendix A.

The atmospheric forcing data, fetched from various locations on the Internet, need to be fed into the compute nodes at regular intervals. The total data volume is estimated to be 267 MB/day. We designate one of the IO nodes to fetch and pre-process the data, then the processed atmospheric forcing data are distributed to the compute nodes' parallel processes through GDS servers running in parallel on the IO nodes.

Table 1 describes the atmospheric forcing data to be used by LIS, and the estimated data volume. The forcing data, depending on their origination, have different spatial resolution and temporal intervals, and the estimation shows LIS will deal with an incoming data flux of approximately 267MB.

[1] http://lis.gsfc.nasa.gov/Source/

Table 1: Atmospheric forcing data description and volume estimation

Dataset	***Description***	***Desired resolution***	***Native format***	***Approx size***	***Update frequency***
GDAS forcing data	The Global Data Assimilation System (GDAS) is the global, operational weather forecast model of NCEP(Derber et al 1991). LDAS makes use of GDAS 0, 0.3, and, as needed, 6 (hour) forecasts, which are produced at 6 hour intervals	Native T170, ~0.7deg	GRIB	50M/day (3.2M X 4 X4)	Every 6 hours
GEOS forcing data	Obtained from GSFC's Goddard Earth Observing System Data Assimilation System (GEOS) (Pfaendtner et al. 1995) version 4.3 that supports level-4 product generation for the NASA Terra satellite (Atlas and Lucchesi 2000).	1 deg	Binary	25M/day	Every 3 hours
AGRMET SW flux data	LDAS estimates global, downward shortwave and longwave radiation fluxes using a procedure from the Air Force Weather Agency's (AFWA) Agricultural Meteorology modeling system (AGRMET). It utilizes the AFWA Real Time Nephanalysis (RTNEPH) 3-hourly cloud maps (Hamill et al. 1992), and the AFWA daily snow depth (SNODEP) maps (Kopp and Kiess 1996) to calculate surface downwelling shortwave radiation using the algorithms of Shapiro (1987)	~48km	Binary	48M/day	Every 1 hour
AGRMET LW flux data			Binary	144M/day	Every 1 hour
CMAP Precipitation	This data set consists of monthly averaged precipitation rate values (mm/day) for the time period Jan 1979-Dec 2001. The data is 2.5x2.5 gridded (144x72) and covers 88.75N to 88.75S and 0E to 357.5 E (eastward). The data range is approximately 0 to 70mm/day. The standard (without NCEP/NCAR Reanalysis data) version has some missing values.	2.5X2.5	NETCDF	9.5M/month	Every 5 days
Total data input flux				**267M/day**	

The land surface parameter data include the vegetation classification, land mask, soil properties, leaf area index (LAI), etc., with a volume of about 92 GB for a 10-year collection. There data will also be served from the IO nodes by the GDS servers to the compute nodes. However, since these data will not be updated frequently, we will put a copy of selected files of these data on each compute node's local disk (such as LAI climatology), to reduce network traffic. All the data are saved in binary format (see Appendix B). Table 2 describes the land surface parameter data and gives the volume estimation.

Table 2: Land surface parameter data description and volume estimation

UMD Vegetation classification map	This file lists the frequency with which of each of the 14 vegetation types occurs in each of the 0.25 degree LDAS grid boxes. See http://ldas.gsfc.nasa.gov/GLDAS/VEG/GLDASveg.shtml for a detailed description.	1km X 1km	Binary	2G	Static
UMD Land mask	This ascii file contains the LDAS unified land/sea mask.See http://ldas.gsfc.nasa.gov/GLDAS/VEG/GLDASveg.shtml for a detailed description.	1km X 1km	Binary	2G	Static
Soil color map	The soil parameter maps used in LIS were derived from the global soils dataset of Reynolds et al. (1999). That dataset includes the percentages of sand, silt, and clay, among other fields, and is based on the United Nations Food and Agriculture Organization (FAO) Soil Map of the World linked to a global database of over 1300 soil pedons. The LDAS soil color map was interpolated from a 2 x 2.5 degree global map produced by NCAR0.01.bin	1km X 1km	Binary	2G	Static
Sand fraction file		1km X 1km	Binary	6G	Static
Clay fraction map		1km X 1km	Binary	6G	Static
MODIS leaf area index (LAI)	MODIS-based LAI data from collection 3 and 4, derived at Boston University.	1km X 1km	Binary	0.5G	Monthly
MODIS-derived LAI climatology		1km X 1km	Binary	6G	Static
AVHRR leaf area index (LAI)	This was generated using three information sources: (1) an 8km resolution time series of LAI, which was derived by scientists at Boston University (Myneni et al. 1997) from AVHRR measurements of normalized difference Avegetation index (NDVI) and other satellite observations.(2) A climatology based on the 8km time series and (3) the 1km UMD vegetation type classification.	1km X 1km	Binary	0.5G	Monthly
AVHRR-derived LAI climatology		1km X 1km	Binary	6G	Static
Slope	Derived from GTOPO32 DEM data. GTOPO30 is a global digital elevation model (DEM) with a horizontal grid spacing of 30 arc seconds (approximately 1 kilometer). GTOPO30 was derived from several raster and vector sources of topographic information by USGS.	1km X 1km	Binary	2G	Static
c file size for 10-year period				**92G**	

The output data, produced by the distributed compute nodes in parallel, will be collected, assembled, and stored on the IO nodes. Users can access the output data, as well as atmospheric forcing and land surface parameter data, via three flexible channels: regular web servers, from which users can directly download data files; GrADS-DODS (GDS) servers, which enable then to access user-defined subset of LIS data; and Live Access Servers (LAS), which can dynamically generate images and data files of various formats based on users' selection of the data subset.

Since it is not feasible to store the output in a single file (200 GB/day, see below), we want to distribute the data across all the IO nodes. To keep the huge output data volume manageable, we designed a storage scheme that will distribute the land surface variables in the output data across the IO nodes. Since there are 30-60 variables in the output data, with some of them having multiple levels, we can let each IO node store the global data of only 4~8 of the output variables. So on average, the I/O traffic is segregated and each IO node is only taking 1/8 of the total data traffic, and the subsequent operations by the GrADS-DODS server and the LAS server are greatly simplified. Table 3 gives the output data volume estimate for various spatial resolutions for CLM runs. For the finest target resolution (1km by 1km), the output data flux is estimated to be 200GB/day. The data output volume for the other two models, NOAH and VIC, is slightly less than CLM.

Table 3: LIS output volume estimation

Dataset	*Description*	*Desired resolution*	*Native format*	*Approx size*	*Update frequency*
CLM output data	LIS output data of ~37 variables	1km X 1km	Binary with optional GRIB	200G/day	every hour
		5km X 5km		8G/day	
		1/8 X 1/8 deg		0.9G/day	
		1/4 X 1/4 deg		0.2G/day	
Total data output flux				**210G/day**	

2.4 Network data traffic estimation

As shown in Table 3 above, the total output data volume (210GB/day) produced by the cluster is much larger than the input data volume (267MB/day), so the network traffic is dominated by the upstream traffic from the compute nodes to the IO nodes, where the output data are stored. The data will travel trough two network links: link A -- a compute node to a fast Ethernet switch port; link B -- the switch's gigabit port to an IO node. Figure 2 shows the network traffic between the two network links. Following is the worksheet for the estimation of the traffic at these two links:

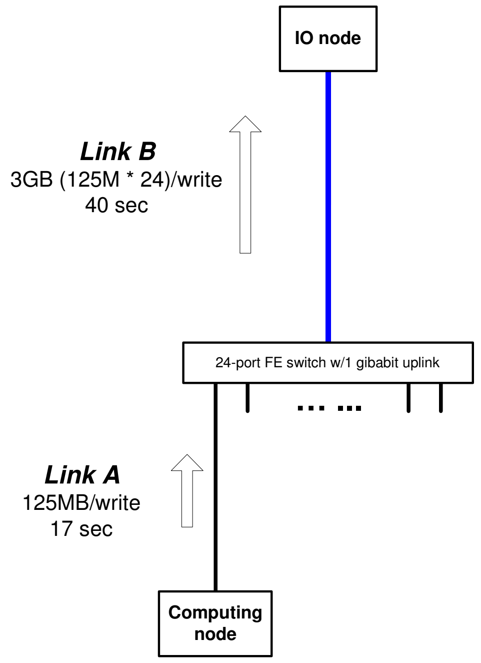

Figure 2: Network traffic estimation within the cluster. The traffic is dominated by the output dataflow from the compute nodes to the IO nodes. The output data will go through two network links: Link A, from a compute node to an Ethernet switch; Link B, from the switch's gigabit port to an IO node.

Worst case scenario assumption: all the compute nodes are writing the output data to the IO nodes at the same time; effective bandwidth is 60% of the Ethernet wire bandwidth. Link A traffic:

Data volume each compute node will produce:
`210GB/day * (1/192) ~ 1GB/day`

Frequency each compute node writes output data to an IO node:
`every 3 simulation hours`

Total writes a compute node has:
`8 per simulation day`

Data volume each write per compute node:
`1GB/day * (1/8) = 125MB/write`

Time taken for the data to travel over link A:
`125MB*8/(100M*60%)= 17 sec`

Link B traffic:
In average, the number of compute nodes each IO node receives data from:
`24`

Total data volume each IO node receives per write:
`125MB * 24 = 3GB`

Time taken for the data to travel over link B:
`3GB*8 /(1G*60%) = 40 sec`

In summary, in the worst-case scenario, it takes only 40 seconds for the 3-hour simulation data to be transferred from the compute nodes to the IO nodes. In reality, the data traffic will be much spread in time, so the network is even less likely to be a bottleneck. It is expected that the disk IO speed on the IO nodes, and the throughput of the LIS code, will limit the performance of the output aggregation.

To support this estimate, Figure 3 below shows benchmark measurements of the cluster's effective bandwidth over two kinds of network links: a compute node to an IO node, and a compute node to a compute node. The benchmark tests were performed over TCP protocol, which is the protocol most of LIS application traffics, such as GrADS-DODS, FTP, and MPI, are based upon. The measurements showed that for large data blocks (> 3000B), the effective bandwidth for both links are well above 60% (60Mbps) of the Ethernet wire bandwidth (100Mbps), so our traffic estimation is valid. For LIS' MPI jobs, we will tune the message buffer size to take advantage of the large data block performance shown in the figure.

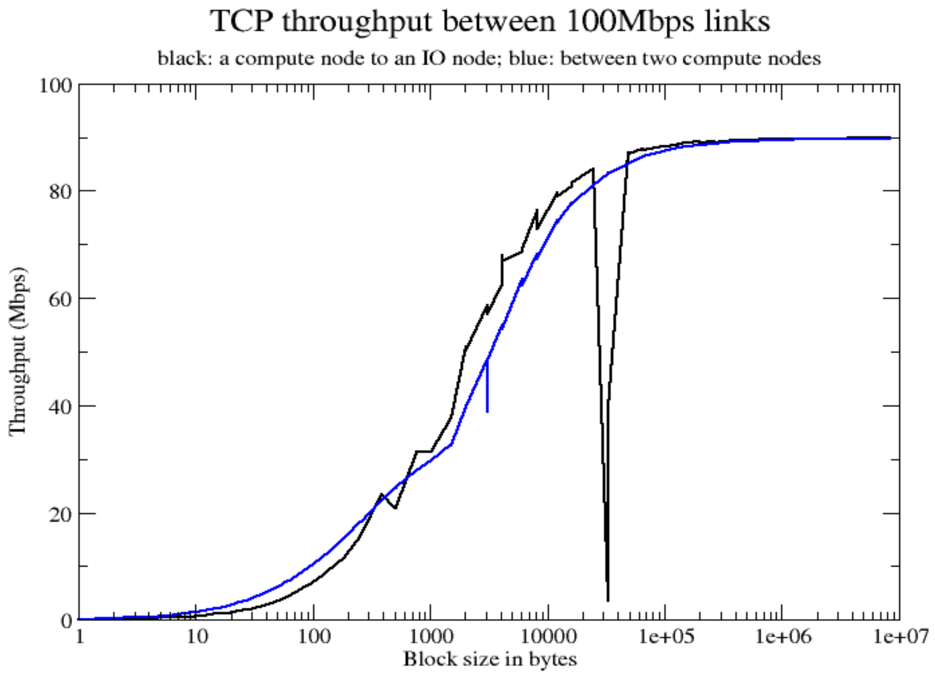

Figure 3: Network bandwidth benchmark tests. Black curve: TCP throughput between a compute node and an IO node. Blue curve: throughput between two compute nodes. The tests were done using the NetPIPE package (http://www.scl.ameslab.gov/netpipe/).

2.5 Input performance optimization with parallel GrADS-DODS servers

GDS plays a critical role in the data management and performance of LIS. All the atmospheric forcing data have to be staged timely to the compute nodes by the GDS servers. Some land surface parameters are potentially to be staged by GDS servers too. In addition, the huge amount of output data are served to the users and to data analysis/visualization applications (*e.g.*, LAS) by GDS servers. Therefore it is crucial to have quantitative measurement of a GDS server's performance so bottlenecks can be identified and optimization can be performed.

As shown in Figure 1, we designed a parallel GDS architecture to solve the input bottleneck problem. The input data were mirrored on all the IO nodes. For input data serving, we take advantage of GDS server's dynamic sub-setting capability to serve only the subset of the input data to each compute node from request-response transactions, thus reducing the total data traffic. To further boost input data serving throughput, we used multiple GDS servers running in parallel on the IO nodes, and implemented a dynamic load balancing scheme to optimize the collective performance of the GDS servers.

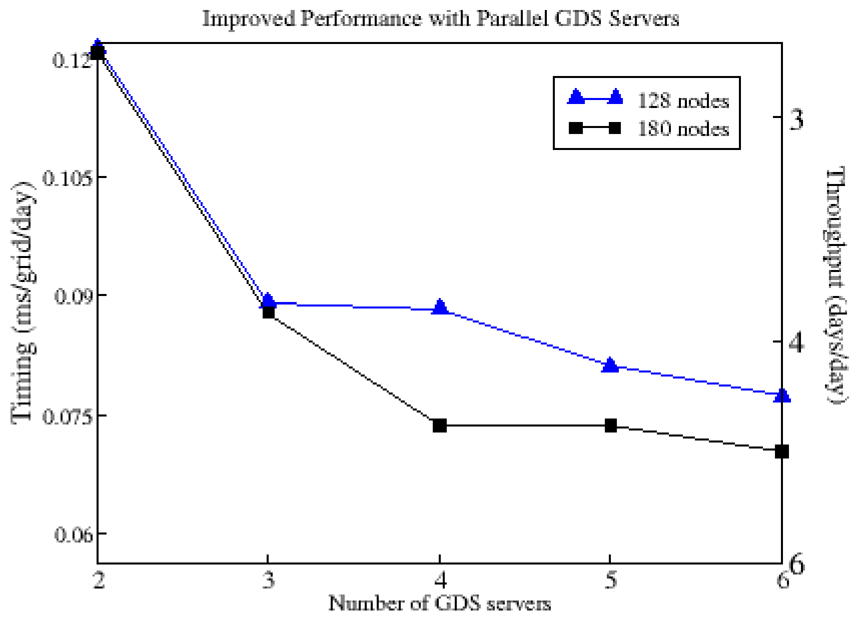

Figure 4: Normalized timing (left y-axis) and simulation throughput (right y-axis) as functions of the number of GDS servers used, for LIS with NOAH LSM and GEOS base forcing. The blue and black curves are for runs with 128 and 180 compute nodes, respectively.

Figure 4 shows the performance improvement as the number of GDS servers was increased. For 128 compute nodes and with 2 GDS servers, NOAH/GEOS ran at 0.122 ms/grid/day, while with 6 GDS servers, it ran at approximately 0.078 ms/grid/day, with a throughput improvement from 2.8 days/day to 4.4 days/day. However, the most significant performance improvement took place when the number of GDS servers was increased from 2 to 3. More GDS servers helped, but not as much. On the other hand, for 180 compute nodes, 4 GDS servers provided the most performance improvement.

2.6 Parallel output and distributed storage

As we are using the large number of CPUs for the highly parallel runs for LIS, and after the input bottleneck was eliminated by using parallel GDS servers, the output bottleneck poses a much more severe challenge if we follow the conventional approach to first aggregate the output data to a central storage system and then serve the users from there. We completely eliminated the need for central data storage by modifying the GDS server system, to expand its capability to serve data directly off the local disks of every compute nodes, as shown in Figure 1. Therefore, each compute node directly writes its output data to its local disk, and then the modified GDS system can treat the collection of the compute node disks as a big disk system, and serve the output data to users from there the same way as from a single local disk. By doing this, the output bottleneck for LIS runs is completely eliminated. Otherwise, it is estimated that it would take 4 to 10 days to finish a 1 day LIS simulation, if the conventional data aggregation and serving were still used.

Figure 5 shows the performance improvement from this innovative storage design. This storage system can retrieve a subset of data more than 10 times faster than from the conventional central storage. Thus GDS systems with this storage system as the backend will serve users with much a much higher throughput.

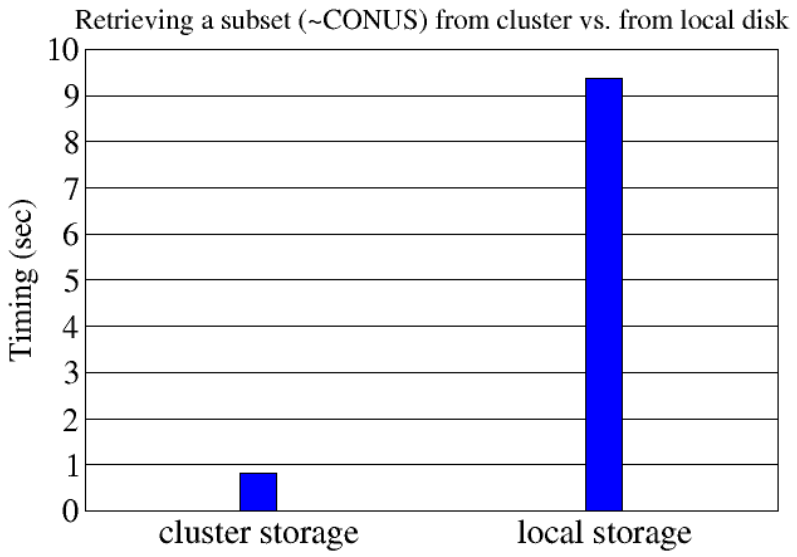

Figure 5: Performance comparison between subsetting from localized central storage (right) and parallelized, distributed cluster storage (left). For this example, the subset is approximately the US continent.

2.7 Data replication with peer-to-peer technology

Although it is trivial to copy small files from one computer to a few other computers over the network via conventional approaches such as NFS, FTP or HTTP based file sharing, it becomes a formidable task to copy a large amount of data from one computer to hundreds of other computers by this means, as with the case for data replication on the LIS Linux cluster. The traditional client—server paradigm, such as NFS, won't scale when tens, or hundreds of clients are copying data from the server, as either the server's bandwidth will be quickly saturated or the server gets overloaded with degraded performance. In addition, the chance of data corruption greatly increases as the data volume and network traffic are increased. Worse still, if a node crashes in the middle of data replication, it has to start over again from the beginning when it comes back online.

LIS needs to pre-stage several Gigabytes of data to every compute node from one node, as these data are static parameters and shared by all the compute nodes, and it would be too expensive to let compute nodes to access the data during runtime over the network. To efficiently replicate these datasets within the cluster without incurring the complexity of client--server approaches, we introduced one of the P2P technologies, BitTorrent (BT), to serve as the data replication channel with the cluster.

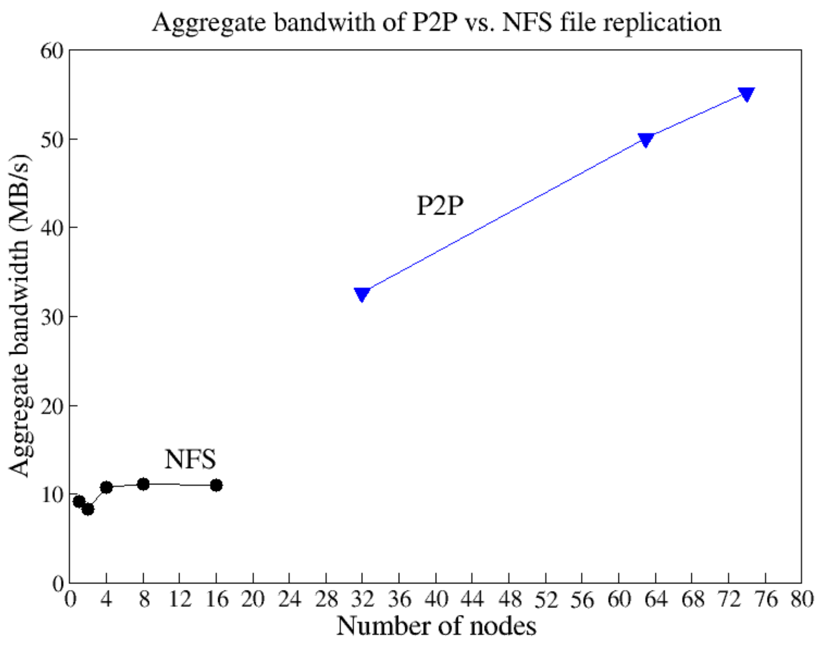

Figure 6: Comparison of conventional NFS-based and P2P-based data replication performance. The average aggregate bandwidth is defined as the total amount of data transfer divided by the total time taken to finish the transfer. A file containing 2GB random data was used as the test dataset. The tests were performed with all the nodes connected with fast Ethernet links.

To get data from one computer (the ``seed") to many others (``peers"), BT first splits the data on the seed into small chunks. Then each peer get a random chunk from the seed. Then the peers can exchange with one another of the chunks and eventually every peer will get all the chunks, thus the whole dataset. Thus the data transfer is mostly taking place between the peers. There is a meta-data server (``tracker") which keeps track which peer has which chunks, and also records the message digest of each chunk. With the message digest, each peer can guarantee the integrity of each chunk, eliminating the possibility of data corruption. Finally, each peer can resume previous transfer without starting over after a crash.

Figure 6 shows the test results on LIS cluster with the conventional NFS based and P2P-based data replication. NFS quickly reached the physical limit of the server's bandwidth with less then 10 clients, and saturated at a bandwidth of 11 MB/s. With BitTorrent, however, the aggregated bandwidth scaled nearly linearly as the number of nodes was increased, and for 74 nodes, it has an aggregated bandwidth 5 times higher than NFS server's. Due to the guaranteed data integrity, no verification was needed after the data were replicated to all the nodes.

3. Input Data

3.1 Atmospheric forcing data

The atmospheric forcing data are centered around two independent atmospheric simulation/assimilation models, the Global Data Assimilation System (GDAS) by NCEP, and the Goddard Earth Observing System Data Assimilation System (GEOS) by GSFC. In addition, a selected set of real-time, operational atmospheric observations, including radiation and precipitation, are incorporated, whenever available, into either the GDAS or GEOS forcing, to reduce any possible model biases.

Figure 4 and 5 illustrate the process to acquire the GDAS and GEOS forcing data, respectively. The process to fetch, store, pre-process and distribute these two sets of forcing data, as well as the other observational data, is similar. The original forcing data are first downloaded from the Internet source sites to LIS local disks, automatically by cron jobs at the regular intervals specific to each dataset. The local copies are then processed to select only the data to be used by LIS, and converted to the desired format. These processed data are then imported by LIS and distributed to the land surface models via GDS servers.

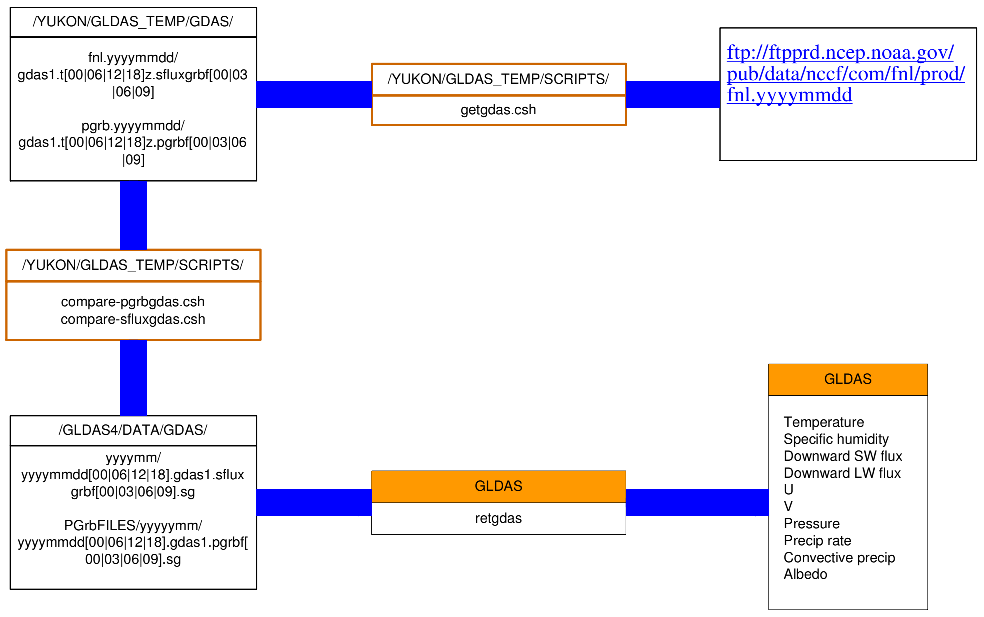
/YUKON/GLDAS_TEMP/GDAS/
fnl.yyyymmdd/
gdas1.t[00|06|12|18]z.sfluxgrbf[00|03|06|09]
pgrb.yyyymmdd/
gdas1.t[00|06|12|18]z.pgrbf[00|03|06|09]
/YUKON/GLDAS_TEMP/SCRIPTS/
getgdas.csh
ftp://ftpprd.ncep.noaa.gov/pub/data/nccf/com/fnl/prod/fnl.yyyymmdd
/YUKON/GLDAS_TEMP/SCRIPTS/
compare-pgrbgdas.csh
compare-sfluxgdas.csh
/GLDAS4/DATA/GDAS/
yyyymm/
yyyymmdd[00|06|12|18].gdas1.sfluxgrbf[00|03|06|09].sg
PGrbFILES/yyyyymm/
yyyymmdd[00|06|12|18].gdas1.pgrbf[00|03|06|09].sg
GLDAS
retgdas
GLDAS
Temperature
Specific humidity
Downward SW flux
Downward LW flux
U
V
Pressure
Precip rate
Convective precip
Albedo

Figure 7: Acquisition of GDAS forcing data by LIS. The forcing data is fetched periodically from NCEP's operational archive, temporarily stored on the local disks, processed by various programs, and finally fed to the land surface models with ten variables.

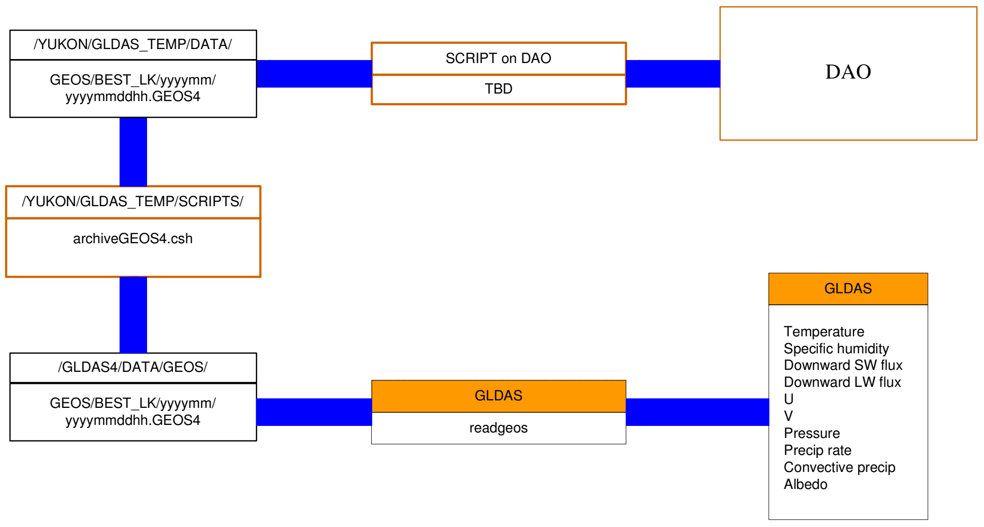

Figure 8: Acquisition of GEOS forcing data by LIS. The forcing data is pushed periodically from DAO's operational archive to the local disks, temporarily stored there, processed by various programs, and finally fed to the land surface models with ten variables.

3.2 Land surface parameter data

The land surface parameters, as listed in Table 2, are collected from a diverse set of sources. Most of these parameters are not time-dependent, so the acquisition and processing of them are a one-time operation. However, as newer land parameter data are constantly made available, the process of obtaining these parameters, and the programs used for this purpose, will be modified from time to time. Figure 6 shows the process for the current land surface parameter data collection. The source data are in disparate formats and resolutions, so a unique program is developed to handle each data source. Eventually all the data formats will follow the specification in Appendix B.

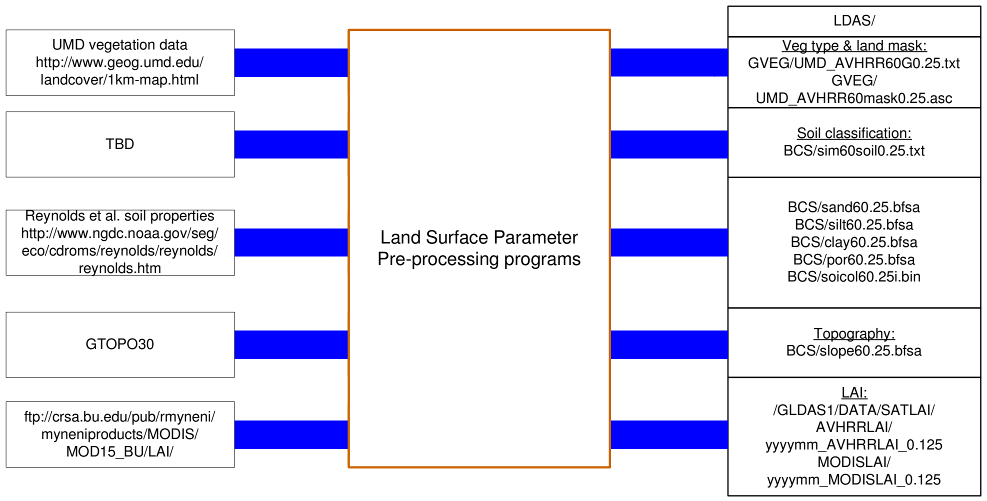
UMD vegetation data
http://www.geog.umd.edu/landcover/1km-map.html
TBD
Reynolds et al. soil properties
http://www.ngdc.noaa.gov/seg/eco/cdroms/reynolds/reynolds/reynolds.htm
GTOPO30
ftp://crsa.bu.edu/pub/rmyneni/myneniproducts/MODIS/MOD15_BU/LAI/
Land Surface Parameter Pre-processing programs
LDAS/
Veg type & land mask:
GVEG/UMD_AVHRR60G0.25.txt
GVEG/UMD_AVHRR60mask0.25.asc
Soil classification:
BCS/sim60soil0.25.txt
BCS/sand60.25.bfsa
BCS/silt60.25.bfsa
BCS/clay60.25.bfsa
BCS/por60.25.bfsa
BCS/soicol60.25i.bin
Topography:
BCS/slope60.25.bfsa
LAI:
/GLDAS1/DATA/SATLAI/
AVHRRLAI/
yyyymm_AVHRRLAI_0.125
MODISLAI/
yyyymm_MODISLAI_0.125

Figure 9: Acquisition of land surface parameters. This process features a diverse collection of data sources.

3.3 Getting input data from GDS server

It is straightforward for a compute node to retrieve a sub-set of the input data from the GDS server. The current versions of GrADS has the DODS-client capability built in, and can be driven by batch script to perform data retrieval for a compute node. The LIS code will invoke a GrADS batch script which lets GrADS retrieve the desired sub-set from the IO nodes and save the data on the compute node's local hard disk. The LIS code will subsequently read from the local disk for the retrieved data. A sample GrADS batch script is shown below.

```
#!/bin/bash
# This script gets a variable from GDS server's 1km
# prototype data repository in land space.
# Each land space has 3378 segments, with each seg having 42577
# points, making 143,825,106 land points
# usage: gdsgetvar.sh varname seg_id tstep
varname=$1
seg_id=$2
tstep=$3
datadir='/data'
urlroot='http://x6:9095/dods/proto/1km'

/data1/grads/grads -bl <<EOB
sdfopen $urlroot/$varname
set t 1
set x 1 42577
set y $seg_id
set fwrite -be $datadir/${varname}-${seg_id}-${tstep}
set gxout fwrite
d var
```

```
quit
EOB
```

4. Output Data

4.1 Data compression

Due to the high data volume LIS will generate at the target spatial resolution (~1km), efficient data storage is critical. LIS modeling is performed on the global scale, but the code is concerned with grid points over land masses only, while the grid points over oceans play no role. We will take advantage of this characteristics, and for the output data, we will save only the data over the land grid points, while keeping a land/sea mask handy for mapping the land points back onto the global grid on the fly. With this data compression scheme, the data volume will be reduced by 70~80%. With this reduction, we will not likely employ further compression schemes due to the resource consumption in the compression/de-compression processes.

4.2 Connection to the User Interface

LIS output data will be archived locally on the IO nodes' RAID disks. We will keep as much historical data as permitted by the storage capacity. However, the most recent data will be given the top storage and retrieval priority, and historical data will be removed if needed to make room for newest datasets.

To ensure flexible and easy access to LIS' huge amount of output data, we will deploy two web-based data supplying engines, the GrADS-DODS server and the Live Access Server (LAS), in addition to the conventional web server. Figure 9 shows the architecture of these two engines, and their interface with the output data archive. Together, they will provide users with the capabilities to perform data searching, data analysis, subsetting, visualization, format conversion and re-projection.

The GrADS-DODS server is an implementation of the DODS (Distributed Oceanographic Data System) protocol, with GrADS as the server backend to take advantage of GrADS' existing capability of data retrieving, subsetting and analysis. The GrADS-DODS server will provide data access via DODS-protocol to DODS clients. The GrADS-DODS server uses a typical client-server architecture to communicate with the DODS clients. The communication protocol between a client and a server is HTTP. A GrADS-DODS server has the following components: Java servlets contained in the Tomcat servlet container, to handle the client requests and server replies via HTTP protocol; DODS server APIs, to parse the DODS requests and package output data; interface code, to translate the DODS requests into GrADS calls; and finally, GrADS running in batch mode, to actually process the requests, and perform data-retrieving, sub-setting and processing on the server side.

The LAS server provides a web interface for users to search a data catalog, to visualize the data interactively, and to download the data in various formats. LAS uses perl scripts to retrieve the metadata from the LIS output files, and save the metadata in a SQL database system, MySQL. The metadata catalog will be presented as a selectable and searchable web page. A user can pick a dataset, or a spatial and temporal subset of it, and interactively generate images on the fly using the graphics engine Ferret via the standard CGI mechanism (see http://ferret.pmel.noaa.gov/Ferret/ for more technical details). An ongoing effort is to integrate the popular GrADS into LAS as another graphics engine, so the users can have the option to choose the image engine they prefer.

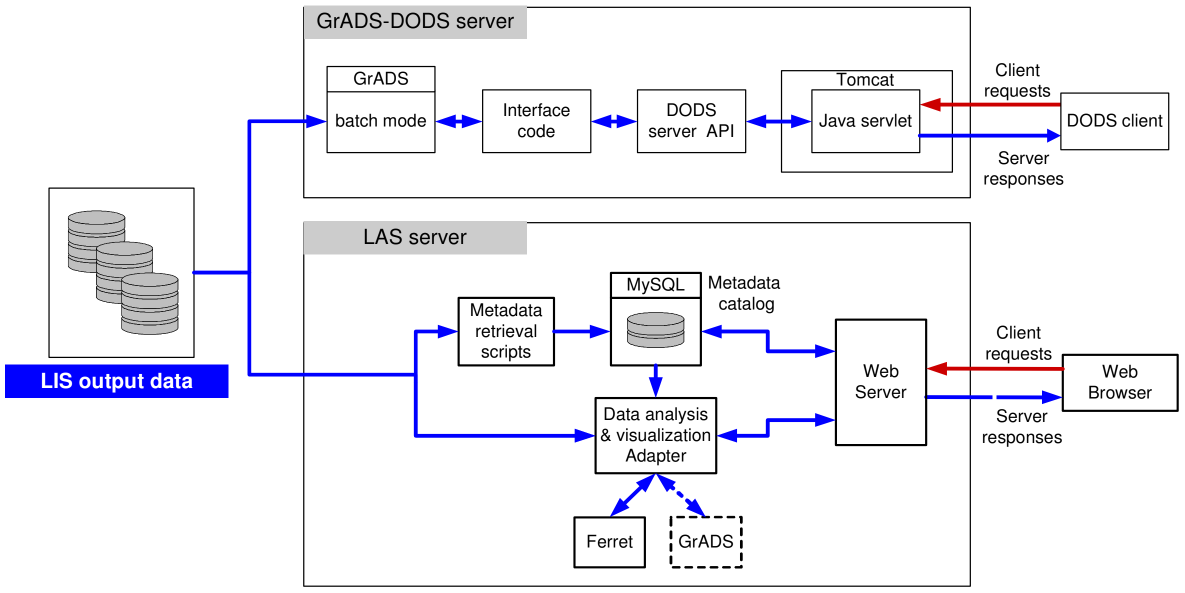

Figure 10: Architecture of GrADS-DODS (GDS) server and Live Access Server (LAS), and their connections to the LIS output data. GDS and LAS were developed by COLA (http://grads.iges.org/grads/gds/) and NOAA (http://ferret.wrc.noaa.gov/Ferret/LAS/ferret_LAS.html), respectively.

5. ALMA and ESMF Compliance

5.1 ALMA Standard Compliance

The ALMA standard specifies a unified scheme for the names, units and sign conventions of land surface and atmospheric forcing variables, as well as the numerical format for the data files holding there variables. Compliance with ALMA standard will facilitate data exchange between different land surface and atmospheric models. LIS is committed to follow the ALMA standard, and the data flow will follow ALMA's specification. Appendix A lists the ALMA variables used by LIS. The *Interoperability Design Document* gives more detailed description for the ALMA compliance design and implementation.

5.2 ESMF Compliance

Based on component technologies, ESMF defines the architecture and interfaces of numerical models for Earth system simulations. It will also provide a standardized collection of utilities to make the implementation of the component interfaces easier. This framework will standardize the coupling between various Earth system models, and facilitate reuse and interoperability of the modeling code.

Guided by the ESMF architecture specification, we designed the software architecture of LIS data management as shown in Figure 8 below. The land surface parameter data are encapsulated inside the land models as part of their intrinsic data structure. LIS get the atmospheric forcing data, on the other hand, in a way as if it gets the data from a ESMF-compliant atmospheric model. We will wrap the data supplying mechanism for the forcing data with the ESMF atmospheric model component, and supply the forcing data with the standard interfaces. LIS will invoke the standard ESMF mechanism to acquire the forcing data via the ESMF coupler component. LIS will also provide ESMF interfaces to supply its output data (Export_state) to other ESMF-compliant models as their input.

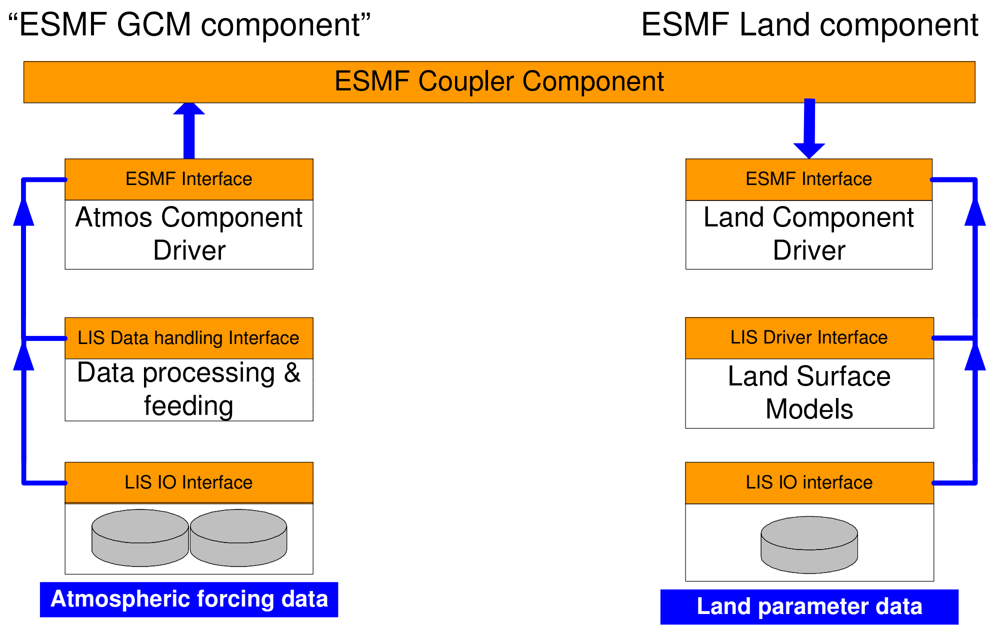

Figure 11: ESMF-compliant software architecture for LIS data management. The land surface parameter data are encapsulated inside the land surface models. The atmospheric forcing data are wrapped with a ESMF-compliant atmospheric component, and the data are supplied to LIS via the ESMF coupler component.

Appendix A

List of LIS variables passed between the modules (Following ALMA Convention)

```
nswrs Net Surface Shortwave Radiation (W/m2)
nlwrs Net Surface Longwave Radiation (W/m2)
lhtfl Latent Heat Flux (W/m2)
shtfl Sensible Heat Flux (W/m2)
gflux Ground Heat Flux (W/m2)
snohf Snow Phase Change Heat Flux (W/m2)
dswrf Downward Surface Shortwave Radiation (W/m2)
dlwrf Downward Surface Longwave Radiation (W/m2)
asnow Snowfall (kg/m2)
arain Rainfall (kg/m2)
evp Total Evaporation (kg/m2)
ssrun Surface Runoff (kg/m2)
bgrun Subsurface Runoff (kg/m2)
snom Snowmelt (kg/m2)
snowt Snow Temperature (K)
vegt Canopy Temperature (K)
baret Bare Soil Surface Temperature (K)
avsft Average Surface Temperature (K)
radt Effective Radiative Surface Temperature (K)
albdo Surface Albedo, All Wavelengths (%)
weasd Snowpack Water Equivalent (kg/m2)
cwat Plant Canopy Surface Water Storage (kg/m2)
soilmc Total Column Soil Moisture (kg/m2)
soilmr Root Zone Soil Moisture (kg/m2)
soilmt1 Top 1-meter Soil Moisture (kg/m2)
mstavc Total Soil Column Wetness (%)
mstavr Root Zone Wetness (%)
evcw Canopy Surface Water Evaporation (W/m2)
trans Canopy Transpiration (W/m2)
evbs Bare Soil Evaporation (W/m2)
sbsno Snow Evaporation (W/m2)
pevpr Potential Evaporation (W/m2)
acond Aerodynamic Conductance (m/s)
lai Leaf Area Index
snod Snow Depth (m)
snoc Snow Cover (%)
salbd Snow Albedo (%)
tmp2m Two Meter Temperature (K)
humid Two Meter Humidity (kg/kg)
uwind Ten Meter U Wind (m/s)
vwind Ten Meter V Wind (m/s)
sfcprs Surface Pressure (mb)
soilt Soil Temperature (K)
soilm Soil Moisture (kg/m2)
lsoil Liquid Soil Moisture (kg/m2)
```

Appendix B
LIS Binary Data File Convention

1. Introduction
   The majority of LIS data is saved in Fortran binary files, with various formats. This note defines the official LIS file scheme, to facilitate unified and consistent access to LIS data by LIS code, user programs and GDS client-server system.

2. Byte order
   LIS data, by default, are saved in binary files as big endian numbers.

3. Storage organization
   For a specific spatial resolution, the spatial grid space has NC columns and NR rows. In addition, a vectorized land space will often be used, with NL land points.
   The minimum storage unit is a 2-D array of NC X NR, or a 1-D array of NL elements. Two dimensional grid space data and 1-D land space data are always saved in separate files.

4. Missing/undefined values

Data type	Missing/Undefined value
character*1	CHAR(255)
integer*1	-128
integer*4	-9999
real*4	TBD

5. File name extension convention and access code samples
   A LIS binary file name extention has 4 fields. The first field is one or more numeric characters, indicating the total number of storage units the file has. The second field is the lower-case character "g" or "l", indicating grid space or land space, respectively. The third field is the lower-case character "s" or "d", indicating sequential or direct access. The last field, has 2 character width, with the first character indicating the number of bytes each number in the file takes, and the second character, as "c", "i", or "r", indicating the type of data as character, integer or real, respectively.

   Example1:
    datafile1.2gs4r

   Sample Fortran code segment to read this file:

```
Real*4 v1(NC, NR), v2(NC, NR)
Open(12, file="datafile1.2gs4r", form="unformatted")
 read(12)v1
 read(12)v2
Close(12)
```

   Example2:
    datafile2.15gd4i

Sample Fortran code segment to read this file:

```
  Integer*4 v1(NC, NR), v10(NC, NR)
  Open(12, file="datafile2.15gd4i", form="unformatted", &
   access="direct", recl=NC*NR*4)
   read(12, rec=1)v1
   read(12, rec=10)v10
  Close(12)
```

Example3:
 soilcolor.1ls1c

Sample Fortran code segment to read this file:

```
  Character*1 color(NL)
  Open(12, file="soilcolor.1ls1c", form="unformatted")
   read(12)color
  Close(12)
```