White Paper

Satellite Imaging Ground Stations: A System Overview

Rick Reid, SGI

1.0 Architecture Overview 2
2.0 Data Capture 2
3.0 FailSafe Capture 3
4.0 Dual-Capture Method 4
5.0 Data Storage 4
6.0 Archiving and Data Retrieval 5
7.0 Compute Servers 6
8.0 Product Servers 6
9.0 Control Servers 6
10.0 Ground Station Architectures for Real-Time Processing 8
11.0 Summary 10

1.0 Architecture Overview

The high-level system architecture of a ground station is illustrated in figure 1. Sensor data is received by the data-capture function and stored in the unprocessed-data buffer (UDB). This data is then read from the (UDB) by the compute servers, processed as necessary and stored back in the processed-data buffer (PDB). Product servers then read the processed data and produce the required products for storage in the distribution-data buffer (DDB) and delivery to the system's users. Control servers provide the data structures required for control of system resources and allocation of these resources to the mission. The control servers also manage all data archive and retrieval functions and provide mission control.

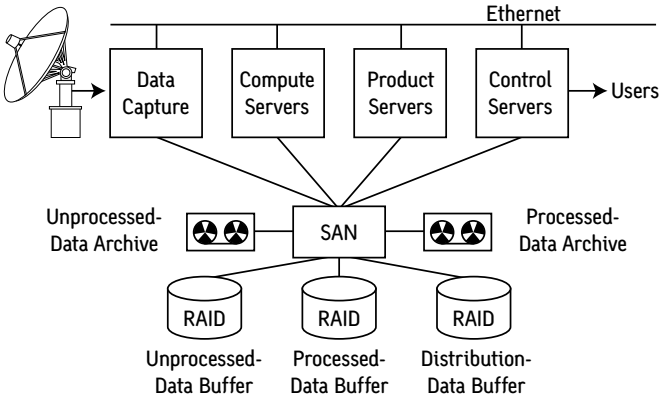

Fig. 1. Ground station architecture overview

2.0 Data Capture

Data capture is the process of receiving downlinked data from the satellite's sensors, formatting the data as required, storing it on disk, and indicating the availability of the data for processing. The degree of complexity of these subfunctions varies with the data-input bandwidth and the reliability requirements of the system. The architecture of a system capable of receiving data at 600MB per second is considerably different from the architecture of a system with a maximum acquisition capability of 5MB per second. Data paths from the I/O interface to memory must be capable of sustaining at least 1.6 times the data-receipt rate. Processing power must be sufficient to perform the necessary protocol processing, data access, and validation. As the data-capture rate increases, the amount of protocol processing and CPU data access prior to storage must be minimized. One way to do this is to format input data into the largest possible blocks and read directly into user space instead of having the operating-system buffer copy it. SGI® IRIX® is one of the few commercial operating systems that can provide this capability. At a minimum, the capture system's memory bandwidth must be capable of sustaining the sum of the input-data rate, CPU data access rate, and output-data rate. Systems with global I/O capability and multiple memory modules such as SGI® NUMAflex™ architecture are particularly well suited to high-performance data capture. Each input buffer can be placed in a different memory module so that only one of the data-capture subfunctions (input, CPU data access, and output) is active in the memory module at a time.

After data validation, the data block is usually written to storage or queued to a processing function or both, depending on the real-time requirements of the system. A proven system design is to segregate the data into files and queue the file identifiers for subsequent processing. This concept facilitates the use of multiple processes, processing partitions, or separate systems as compute servers. Data pointers or file identifiers are queued by the data-capture function and then removed from the queue by the compute servers as they complete their previous task and become available for additional work. Should a process fail during processing, the file identity is simply inserted back to the queue head where it will be processed as soon as the next available compute server completes its task. The files may also be queued in memory for those systems with more real-time requirements. As the input-data rate increases, the preallocation of files in the UDB must be considered. Normally the UDB is completely full; therefore, files must be purged to make room for the data associated with the next data-capture period. It is best to perform this function prior to the start of the real-time data capture in order to minimize the amount of time required to open the files associated with the new data.

The degree of data-capture reliability required determines the architecture of the data-capture function. There are three basic approaches: single capture, fail-safe capture used in an SGI® FailSafe™ environment, and dual capture. The single-capture approach illustrated in figure 2 is the simplest to implement, the least expensive, and the least reliable.

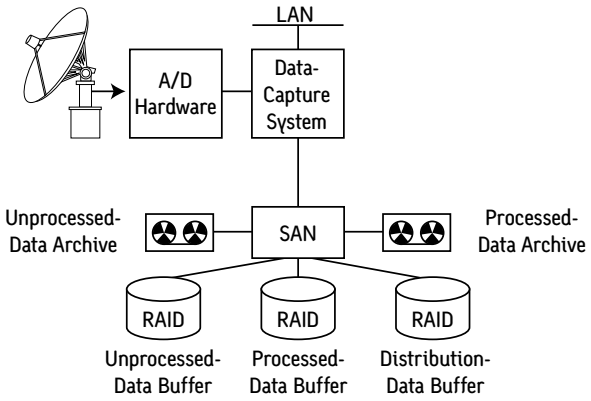

Fig. 2. Single-capture architecture

With this approach, there is a single system to receive the data and write it to storage. If this system fails, the input data is lost and continues to be lost until the system is repaired. The total cost of reacquiring the data must be compared with the cost of the additional hardware, software, and maintenance associated with more-reliable approaches in order to determine if the single-capture approach is best for your ground station.

3.0 FailSafe Capture

The FailSafe capture approach is a more reliable (but more expensive) approach. With this approach, there are two data-capture systems; one system is online and the other system is in standby mode. There are three degrees of standby: cold, warm, and hot.
Cold standby means the system is not involved in the data-capture function at all but simply has the necessary capabilities and interfaces to perform the data-capture function should the online system fail. This approach is illustrated in figure 3.

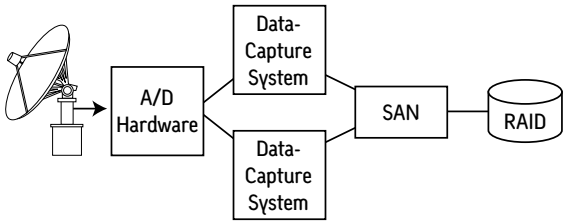

Fig. 3. Cold-standby Failsafe capture architecture

If the online system fails, the processes running in the standby system are terminated and the data-capture process is started (either manually or automatically by the control server). During the time required to load the data-capture function and make it operational on the standby system, all input data is lost. The advantage of this approach is that the system is back online prior to repair of the first system. Another disadvantage, of course, is the cost of the second system, although this cost is not totally allocated to data capture because the system can be used for other processing functions when both systems are operational. Warm and hot standby share the same architecture, which is illustrated in figure 4.

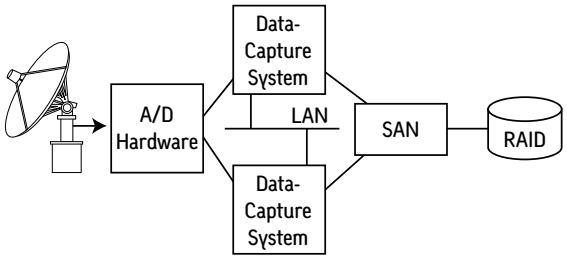

Fig. 4. Warm- and hot-standby Failsafe capture architecture

The major difference between the systems illustrated in figure 4 and figure 3 is the use of a LAN between the two systems. This LAN provides the FailSafe heartbeat. Each system pings the other system on a set time interval. If one system fails to respond for two sequential pings, it is reset by the pinging system and that system assumes the online function. The process is called switchover or failover, and for most operating systems—certainly any that are UNIX® OS based—it requires a minimum of 30 seconds.

A warm standby system does not receive the data from the satellite sensor; therefore, input data is lost during the switchover process. A hot standby system, on the other hand, does receive the sensor data. Usually the data is duplicated in the analog to digital hardware and sent to both systems. Each receives and validates the data, but only the online system stores the data in the UDB. Upon completion of the write to storage by the online data-capture function, a message is sent to the standby system indicating that the data block is safe on disk and can now be dumped by the hot standby system. Should the online data-capture system fail, the standby system must have sufficient buffers to hold the data blocks being received until it decides to take over the online function.

The advantage of the warm- and hot-standby architectures is that less data will be lost if the online data-capture system fails than in a single capture or cold-standby approach. The disadvantages are the additional cost and software complexity.

It should be noted that the architectures in figures 2–4 likely contain single points of failure along the data path somewhere between the RF hardware and the storage. If the design requirement is that no data

shall ever be lost, then the dual-capture method is the approach recommended. This approach is illustrated in figure 5.

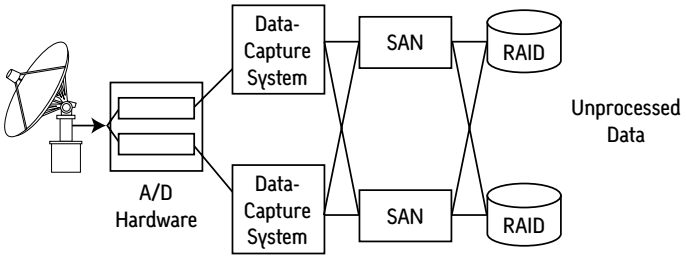

Fig. 5. Dual Capture Architecture

4.0 Dual-Capture Method

The dual-capture approach eliminates all single points of failure. Satellite sensor data is received over dual analog to digital hardware, sent to separate capture systems, and written to separate RAIDs via separate SAN switches using different RAID controllers. Both data-capture systems receive, process, and store the data using completely independent hardware. One of the redundant data files created in the UDB is deleted by the control server upon completion of status processing from each of the data-capture systems. Undoubtedly this is the most reliable data-capture approach, but it is also the most costly.

5.0 Data Storage

The data storage architecture of a ground station is also highly dependent upon the data-capture rate and the degree of reliability required. If data can afford to be lost, JBOD storage may be the best solution (and most likely has the lowest cost). If the requirement is to never drop a bit, then dual capture and dual RAID-based volumes are the answer.

When the total of the data-capture write rate plus the processing and archiving read rates exceed the sustainable rate of a single RAID controller, multiple RAID volumes must be utilized for the unprocessed-data buffer. For example, a UDB capable of sustaining 1GB per second alternating reads and writes requires at least eight Fibre Channel interfaces, four RAID controllers (assuming each 2Gb-per-second Fibre Channel interface can sustain 125MB per second and each controller can sustain 250MB per second), and four RAIDs (logical units) per controller. It also requires a volume manager and filesystem capable of creating and efficiently using logical volumes of this magnitude. Not all SAN filesystems are capable of efficient high-speed, multivolume operations. SGI IRIX has demonstrated 2GB-per-second sustained bandwidth deployed in a multivolume UDB ground station architecture.

As the storage data rate increases, several other factors must be also be considered in the design of the system architecture. The filesystem's ability to issue I/O directly from user space becomes critical. The latency required for one or more copies of the data by the filesystem not only wastes precious CPU and memory resources but makes it more difficult to keep the RAID queues sufficiently loaded to prevent idle disk time.

The size of the data blocks is also important, especially as a higher percentage of peak performance is required from the storage architecture. The bigger the data block, the more time that is spent by the drives actually reading or writing instead of shifting the heads or simply being idle. Another factor, which must be considered in determining the correct block size, is the number of disk drives in the buffer architecture. In the example described above where the UDB must sustain 1GB per second, if 4+1 RAID 3 were used for each RAID, data would be stripped across 64 user drives and 16 parity drives. A 64MB data block would therefore result in 1MB of data per drive. A drive capable of sustaining 25MB per second would therefore require about 40 milliseconds to read or write the data and would be capable of storing multiple data blocks in its cache. In addition, it is best to not mix data block sizes in a volume. If all data blocks in the volume are the same size, fragmentation is minimized as files are repeatedly created and deleted. It is preferable to provide a separate volume for support data and other files that have small data blocks.

The use of a SAN and a shared filesystem are very beneficial in a ground station architecture. SGI® SAN performance is very close to 100% of direct storage performance when large files are usually only being accessed by a single process and few if any files are modified after creation. Data therefore can flow between storage and the servers in the system at full storage rates, and network bandwidth limitations are eliminated. However, as the necessary storage buffer bandwidth increases, so must the number of Fibre Channel interfaces per server, and the number SAN switch ports increase. In a system with an unprocessed-data buffer that sustains 1GB per second and two capture servers, 24 SAN ports would be required just for the capture servers and the RAID controllers. If there were four compute servers (not unrealistic for a system processing 1GB per second), each with eight FC interfaces, another 32 ports would be required (56 ports total) just to access the unprocessed data. This does not include the ports required for archive recorders and the other buffers in the system. The SAN can easily expand to require multiple 64-port SAN switches. Most compute servers do

not need read access at the rate required for data capture. Therefore, the number of Fibre Channel interfaces per compute server can be reduced. This presents a problem when a capture server is writing to the UDB in parallel over multiple Fibre Channel interfaces and the compute servers are reading from the same volume over far fewer interfaces. The duration of the read operation will be limited by the sustainable bandwidth of the Fibre Channel interfaces to the compute server. For example, if the capture server has eight Fibre Channel interfaces and the compute server has two Fibre Channel interfaces, the compute server read operation will take approximately four times as long. The longer duration for each data-block read delays completion of the write operations from the capture server because it ties up the necessary drives. This results in longer write completion times and requires more memory buffers in the capture servers to compensate for the delays. This problem can be compounded even more when several compute servers all issue read operations at the same time. The capture server's write operation may then have to wait for completion of all the queued reads.

The number of SAN switch ports can be reduced by combining multiple compute servers into a single system. The global I/O capability of IRIX and SGI® Origin® 3000 series servers allows the I/O interfaces to be shared by the compute servers. This allows multiple compute servers to share the full number of Fibre Channel interfaces to the SAN and eliminates the problem of slower read rates.

6.0 Archiving and Data Retrieval

Both the unprocessed data and the data products produced by the system are usually archived as part of the normal processing inherent in a ground station. The unprocessed data is retained so that it can be reprocessed in the future to reproduce the original product or to make additional products by using different parameters or algorithms. Products are archived so they can be readily available for retrieval and distribution to customers without having to be completely reprocessed from scratch. This approach results in two archives, usually called the unprocessed-data archive (UDA) and the processed data archive (PDA).

Several factors affect the design of the UDA and PDA. The most significant factor is the data-capture rate and the size of the unprocessed-data buffer. If the UDB can store 24 hours of unprocessed data, there must be a sufficient number of recorders in the UDA to record this same amount of data in 24 hours: archiving a day's worth of data in a day.

If the UDB can only store one satellite's orbit of unprocessed data, the UDA must be able to archive the peak amount of data received per orbit within one orbit's time. The smaller the UDB capacity, the more UDA recorders required. And, conversely, the larger the UDB, the fewer UDA recorders required.

Another factor that must be considered in the design of the unprocessed-data archive is the data retrieval rate. The amount of data retrieved from the UDA to the UDB must be considered in the capacity-requirement calculations of the UDB as well as in calculating the number of UDA recorders required. The number of retrievals must also be considered in the recorder-quantity calculations. Most likely, retrievals will have a much shorter time requirement than other UDA data storage operations. Customer requests are not usually averaged over some period but instead are given a higher priority than archive storage functions. Consequently, recorders are allocated to retrieval operations as required (up to a certain threshold) thereby reducing archive storage operations. It is recommended to always leave some recorders dedicated to UDB storage operations.

The size of the processed-data archive is a function of the capacity of the distribution-data buffer, the size of the products produced, and retrieval rate required for the PDA. Like the UDA, the buffer capacity determines how much time there is to archive the data. Again, if the DDB can hold a day's worth of product files, then the number of recorders is determined by the speed of the recorder, the duty cycle of the recorder, and the time allocated to archive the buffer. This formula is illustrated in figure 6 below.

$$\frac{\text{Buffer capacity in MB}}{\text{(Recorder rate in MB/sec)}\ \text{(Seconds allocated for buffer storage)}\ \text{(Duty cycle \%)}} = \text{Number of recorders required}$$

Fig. 6. Archive recorder algorithm

A recorder duty cycle of around 50% will usually compensate for media access, media load, tape positioning, repositioning of the media after the file transfer, and media unload times. However, if the sum of these times is greater than the file transfer time for the average file size and the particular hardware selected, lowering the percentage is recommended.

Products should always first be retrieved from the distribution-data buffer because it provides the fastest possible access to the data. If the desired product/file is not available on the DDB, the next location queried should be the PDA. If found, the file would then be

retrieved from the PDA to the DDB. Failure to locate the desired file in either the DDB or PDA means reprocessing is required. The order of retrieval from this point would be the processed-data buffer, the unprocessed-data buffer, and lastly, the unprocessed-data archive.

7.0 Compute Servers

Most compute servers do not need read access at the rate required for data capture. For example, if the processing algorithm requires around 1,000 operations per input sample, 1GB-per-second processing would require a system of over 1 TFLOPS. The amount of time allocated for processing is often determined by the production requirements of priority products. The compute server is sized to perform the processing required for the highest priority product within the allocated time. Processing-time allocations can also be determined by the frequency of vehicle contact. If the vehicle has a 90-minute orbit and only a single contact per orbit, then processing-time allocation is the time between the ends of the vehicle contact periods: basically, completing an orbit's worth of processing per orbit.

The maximum benefit is obtained from the compute servers when the processing algorithms are kept as busy as possible. This is accomplished by always having the n+1 data file, or at least multiple data blocks of the total file, buffered in memory for processing when the algorithms complete processing on the nth data file. This requires that the I/O read and write functions be performed by an application driver in parallel with algorithm processing. The application driver communicates with the control servers to obtain the path identities of the files to be processed and created by the compute server. It opens both the input and output files, allocates buffers, and reads the data file (and all support data) into the global memory buffers. The global buffer addresses are then passed to the algorithm processes for processing when they are available. The algorithm-process CPUs access their individual portions of the global memory buffers, process the data in their local memories, write the processed data back to the global output memory buffers, and signal the application driver that the output data is ready for storage. The application driver then writes the data file or data block back to the appropriate storage buffer. While this process is going on for the nth file or data block, the application driver is reading the n+1 file or data block into a second set of global memory buffers. The design goal is to always have a set of buffers available for the algorithm-process CPUs so that these processors never go idle. Optimizing the algorithm processes and assuring that file I/O times do not exceed file processing times will result in maximum utilization of the compute servers.

8.0 Product Servers

Product servers, like compute servers, process files. They read data files from the processed-data buffer, turn them into products, and write these products to the distribution-data buffer. These products are then queued and distributed to users by the control servers as requested. Product servers also read files from the DDB to combine them with recently processed files or to redistribute existing products.

Ground stations process raw satellite data to create varying levels of geospatial products, which meet a broad range of needs. Examples of these product levels are:

- Level 1: Radiometrically corrected—oriented to sensor patch, corrected for transmission errors, adjusted for brightness/contrast
- Level 2: Standard geometrically corrected—corrected for systematic distortions, no ground control points or terrain elevation required
- Level 3: Precision geometrically corrected—ground-control points improve product accuracy; rectified to a constant elevation
- Level 4: Orthorectified—constant elevation and corrected for terrain relief
- Level 5: Digital terrain data—precision terrain information and stereoscopic imagery pairs
- Level 6: Pan-sharpened—multispectral data sharpened with black-and-white data of same area
- Level 7: Mosaics—digitally assembled images to create large contiguous areas; applied to levels 2, 3, 4, and 6

Product levels 1 and 2 are usually provided by the processing functions of the compute servers. Product levels 3 through 7 are produced in the product servers.

9.0 Control Servers

Control servers provide three major functions in a ground station. These functions are resource management, process management, and mission management. All of these functions require databases, interprocess communications, and the support of a high-availability system architecture.

Resource management maintains the state of the computer systems and networks in the ground station. The functional state of each capture server, control server,

compute server, product server, disk drive, RAID controller, network path, and SAN switch and port is maintained in a database. The possible states for each piece of equipment are illustrated in figure 7.

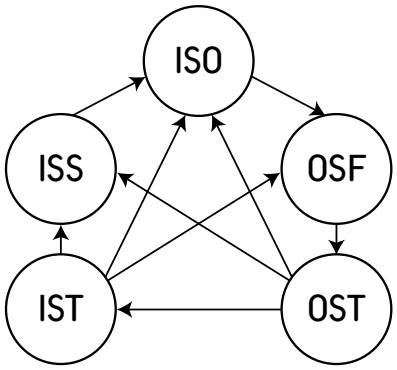

Fig. 7. Equipment state diagram

The in-service-online (ISO) state indicates the unit is operational and available to perform its function in the online system architecture. Failure of a unit moves the equipment from the ISO state to the out-of-service-failed (OSF) state. Once a unit enters OSF, it must be serviced and repaired. After repairs are complete, the unit is taken to the out-of-service-test (OST) state where vendor diagnostics, etc., can be executed, but it is not certified as operational. Upon completion of diagnostics, the normal transition from OST is to in-service-test (IST), where the required software is loaded and the unit reestablishes communications with the control servers. However, under emergency conditions in which this unit is required to continue ground operations, the unit can move directly from the OST to the ISO state: caution is recommended. Following establishment with the control servers and software synchronization, the unit moves to the in-service-standby (ISS) state or the ISO state, whichever is appropriate. Defining states for all units in the system allows for quick identification of the configuration and for easy determination of available resources for operations.

Resource management also includes buffer and archive management. Files in the UDB, PDB, and DDB are not deleted until the disk space is required for newer data. Running the buffers in this “full” manner provides the fastest possible access to the data for retrievals and for restarting processes after a failure. If a compute or product server fails, its output files are purged and its input files are simply queued to the next available server. The control server database maintains the identity and capacity of the oldest files marked for possible deletion in each of the buffers. Files are marked for deletion after they have been processed and archived as necessary. Prior to the start of a data-capture period, buffer management calculates the storage capacity required in each of the buffers for all data to be received and created as a result of the upcoming data-capture period. This amount of storage is purged from each of the buffers. Files are then allocated for each file to be generated. Preallocation of files minimizes the real-time delay associated with dynamic deletion and creation of data files. The new file identities are maintained in the database and distributed as part of the queue entries sent to the archive management, application driver, and product server processes.

When there are multiple volumes in a single buffer, load leveling must be considered. Load leveling maintains a similar fill level in each of the volumes in a single buffer. Several factors should be considered when determining which files should be deleted and created on which volumes. The files should be as evenly distributed as possible across the volumes. For example, if most of the files created in a data-capture period were to be allocated to a single UDB volume, then that volume would not only receive most of the write activity while other volumes were less busy, but would also receive most of the read activity as the data was being processed.

Buffer management must also consider file size when trying to evenly distribute the files. Not all files received from a satellite are of the same size—some are very small and some are very large. Allocation of multiple large files to one volume and multiple small files to another volume results in an imbalance of volume capacity as files age. The final factor considered when load leveling buffer volumes is the processing priority of the data. If all the high-priority files are written to a single volume, all compute servers will immediately try to access that volume as the files are queued for processing.

Archive management is another subfunction of resource management running on the control servers. It involves several tasks depending upon the complexity of the archive requirements. The main tasks include copying the required data files to the archives, retrieving files back from the archive, managing the archive content database, communicating with the robotics control software, deleting files and compacting tapes, and, possibly, managing a “shelf” archive.

The efficiency of copying files to the archive depends on the organization of the tapes and the number of retrievals expected. The most efficient method for organizing an archive with low retrieval rates is to mount blank tapes, calculate which files will best fit on the available tapes, and initiate parallel copy operations beginning with the oldest files. This minimizes the number of tape mounts, streams files to tape, and fills the tapes as much as possible. It also makes aging

the archive by date very easy—full tapes are simply either overwritten or removed from the online archive and stored offline. However, if frequent retrieval of files involves multiple files associated with specific criteria, such as geographic location, it would be best to archive the data according to the desired criteria. This results in more tape mounts during tape storage operations but far fewer mounts and therefore faster retrievals where time is more important. For this reason, the organization of the unprocessed-data archive may not be the same as the organization of the processed-data archive.

Process management is the second major function of the control server. It involves control over and initiation of the data capture, data processing, product processing, and product distribution processes. The process manager receives information from mission management about the data to be captured during the next satellite contact period and the products to be produced. This information is stored in the control server database. The resource manager is then tasked to delete files until sufficient space is available and to allocate the required buffer files for the next contact period. When the resource manager completes its buffer management function, the information defining capture times, sizes, and associated file-path identities is formatted into a capture plan and transmitted to the capture servers. As each file is captured and written to the UDB, the status of the file is transmitted to the process manager and stored in the control server database. If dual capture is being used, the status from each capture server for this input file is compared and the file with the best status is selected for further processing. The other file is marked for possible deletion. Once the selected file is identified, the process manager analyzes the input queues of the compute servers and adds the file's path identity and other processing support data to the correct server's work queue. The correct server can be determined by a number of different criteria, ranging from which is the least busy to which has specific algorithms loaded for this particular file. The specific and often dynamic capabilities of the compute servers are maintained in the control server database. Status is transmitted back to the process manager each time the file completes a step in the series of events it must go through, from capture to final product archiving. The process manager journals these status messages for failover recovery, processes them, and queues the newly created files to the next appropriate process for each step in the series. If a compute server or product server should fail during the series, the process manager simply queues the input file to another server capable of performing the failed step.

Mission management is the process of receiving the user's collection or retrieval requests, determining how and when those requests can best be satisfied, tasking the satellite, and then providing the details of the collection and the products required to the process manager.

Users request an area to be collected by identifying the area geographically via latitude/longitude coordinates or by specifying the name of an area in the mission management's database. They may also specify the type and quality of the collection desired. Graphics capability is a significant help to the user in producing requests. Mosaic images of the desired area can be overlaid with graphics showing specific vehicle orbits, imaging capabilities, and known targets from the database. This type of visualization greatly reduces the time required to generate a valid imaging request.

Mission management receives and verifies these requests, determines how they can best be satisfied, and assigns them to a specific orbit of a specific collector. Once all nominations for a specific orbit of a specific vehicle are determined (by reaching either a cutoff time or the vehicle's capacity), the necessary vehicle activities associated with the requests are simulated to ensure the vehicle's capabilities are not exceeded. Upon completion of the simulation, a vehicle command list is prepared and verified. Another simulation is usually used for this verification to make sure that the vehicle's capabilities are not exceeded. The details of each user's request are acknowledged back to the user at this time as well as being transmitted to the process manager for capture and processing scheduling. Once the command list has been verified, it is ready to be uplinked to the vehicle. Orbit tasking is usually prepared several revolutions prior to actual usage. Also, command lists are usually stored on the vehicle for more than one orbit should the ground station be unable to communicate with the vehicle for some period of time. This eliminates the possibility of a wasted orbit due to failed communications.

10.0 Ground Station Architectures for Real-Time Processing

Ground stations that require real-time processing of the data usually have a different system architecture than the architecture illustrated in figure 1. The unprocessed data may or may not be stored on disk but is routed directly to the compute servers for processing. Usually, not all of the processed data is saved, just those segments or files of interest. As figure 8 illustrates, there are still capture servers, compute servers, product servers, control servers, and archives in a real-time ground station, most with functions similar to those described previously.

The main difference between the figure 8 architecture and the architecture shown in figure 1 is the routing of unprocessed data from the data capture function directly to the compute servers. If the unprocessed data must be saved, it is also routed simultaneously to a file server for storage in the unprocessed-data buffer. Specific attention to the design of the network connecting the components of the architecture is required for real-time systems. If the input data rate is very high, 1GB per second for example, each link in the network must be able to sustain at least this rate. Therefore, in the example, each link must have a peak bandwidth of around 1.6GB per second. The SGI Origin 3000 series processor interconnect is well suited to move large data blocks between processes or partitions because of its 3.2GB per second per link bandwidth.

Processing data in real time is easier to do if it is possible to segment the data into chunks or granules. Data is received for a certain period or until a natural separation point is determined, and the resulting unprocessed data granule is transmitted over the SGI® NUMAlink™ interconnect and queued to a compute server process or partition. The next granule is received, transmitted, queued to another compute server, and so on for each of compute servers in the architecture. This concept allows the real-time processing load to be distributed among the number of compute servers it takes to handle the granule input rate such that an idle compute server is always available.

For example, if data is coming in at 100MB per second and the granule size is 10 seconds (or 1GB) of data, and there are 10 compute servers, each server would have 100 seconds to process a granule and be ready for its next granule. The output from the compute servers would be transmitted and queued to a product server in the same manner, as would the products produced from the granule. Data granules can also be transmitted to a file server after each step in the process and written to storage as necessary. All transfers are memory and all granule queues are memory based. The goal of this architecture is to keep the compute and product servers as small and as busy as possible because they are the most expensive resource in the architecture.

An added benefit of this architecture is the possibility of reducing maintenance costs while at the same time increasing the processing power available and the system availability. The addition of two more compute server partitions than are necessary for real-time pro-

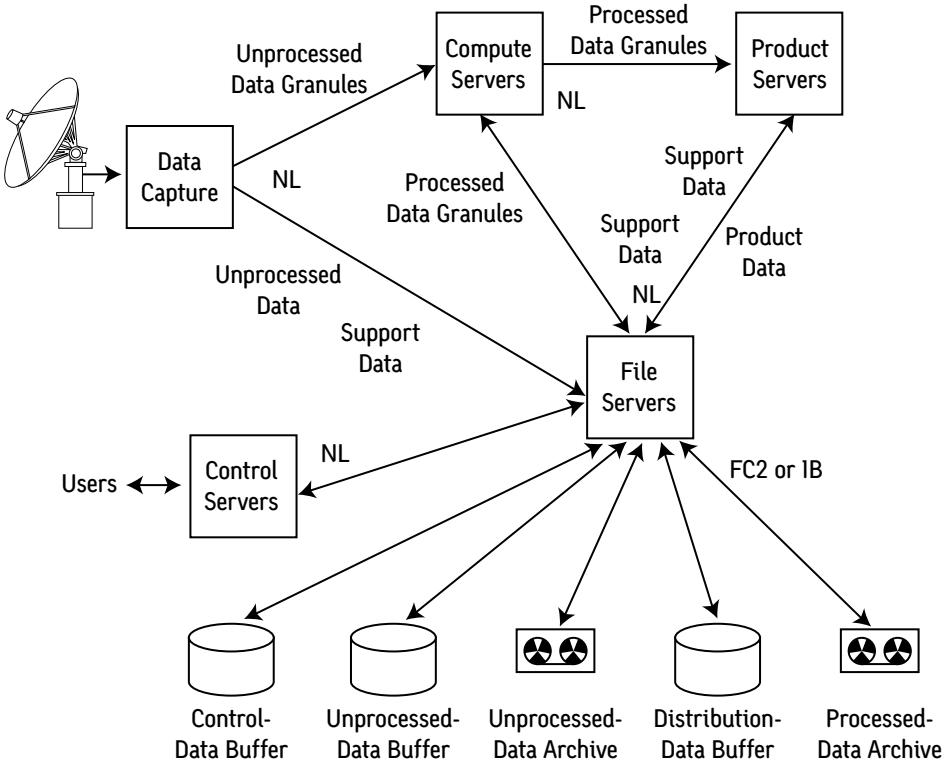

Fig. 8. Real-time ground station processing system architecture

cessing means that two partitions can fail and the system can still continue processing in real time. Having these two extra partitions allows the maintenance policy to be reduced to 48-hour response because of the low probability of three partitions failing within 48 hours.

In addition, the need for on-site spares is reduced considerably because replacement parts can be shipped via overnight delivery. The maintenance-cost savings easily pays for the extra partitions in the first few years of operations. The processing power available on-site is increased because the two additional partitions are used as normal compute or product servers, thereby increasing the peak processing capability of the ground station. In addition, the availability of a system with two additional servers will exceed 0.9999.

11.0 Summary

The complexity of a satellite ground station architecture is determined by the input-data capture rate, the amount of processing required to produce the end products, and the degree of automatic operations desired in a multisystem architecture. A high input-data rate affects the selection of the capture server itself and the design of the data storage buffers and the network or SAN connecting the system components. The input-data-capture rate and the amount of processing required per input sample determines the size and number of compute and product servers required. The more systems required the more complex the ground station. The degree of automation of the processing steps and desired recovery from failures increases the complexity of the control server code and those applications that communicate with it. However, it greatly increases the performance and operational stability of the ground station.

Corporate Office
1600 Amphitheatre Pkwy.
Mountain View, CA 94043
(650) 960-1980
www.sgi.com

North America 1(800) 800-7441
Latin America (52) 5267-1387
Europe (44) 118.925.75.00
Japan (81) 3.5488.1811
Asia Pacific (65) 771.0290

3285 (5/02)

JI3845