Designing GIS for High Availability and High Performance

Don Brady
High Performance Technical Computing
Compaq Computer Corporation
Marlboro, MA USA

don.brady@compaq.com

Abstract

Geographic Information System (GIS) application design has become Internet-centric; the enterprise has become spatially enabled; traditional databases have integrated islands of spatial and non-spatial data. These are some of the trends that are helping to promote the expansion of GIS into the information mainstream as a core technology, and more importantly into mission-critical applications. But computer hardware can fail, and if mission-critical applications cannot be kept running effectively – that is, "available", such failures are costly to an organization.

GIS applications are available only if they allow users to access the GIS server application(s) and the GIS data files. High Availability environments are designed for computing installations that require critical systems to be automatically and seamlessly restarted in the event of a hardware failure. They can ensure that data remains accessible, and that applications be kept running, even during a prolonged hardware failure.

This paper will investigate the nature and architecture of a High Availability GIS: the use of hardware and software common to most high performance computing implementations, to provide automatic failover and continuous operation in the event of system failure. It will describe how High Availability was implemented by two major GIS software platforms to minimize interruptions to applications and to keep file systems continuously available.

Introduction

A child who misses ten days of school over the course of a year attends only 94 percent of her classes. A football player who misses one week of a season plays in 94 percent of his team's games. By comparison, standalone computer systems typically can achieve about 99 percent "uptime" – about three and a half days downtime per year -- which for non-critical computing environments is generally acceptable. But mission-critical, can't-do-business-without-my-computer-system environments can tolerate no more than one-tenth that amount of downtime, or about eight hours per year, and in intervals of no more than a minute or so per instance. This is the essence of High Availability, and a primary concern of core business operations: 99.9 percent uptime, and downtime lasting for not more than a few seconds to a minute at a time.

Today one can easily argue that the Geographic Information Systems (GIS) discipline has penetrated core business operations. Every business relies on "place" information. In many situations immediate or real-time access to place information is essential, either because the spatial data itself is critical, or because the spatial information is a tightly integrated component of a broader mission-critical application. Emergency 911 systems need to identify the quickest route to a fire or an automobile accident; utility companies need to locate the source of a power failure when customers report

Proceedings of the Fourth International Conference/Exhibition on
High Performance Computing in Asia-Pacific Region
0-7695-0589-2/00 $10.00 © 2000 IEEE

an outage; the forest service needs to predict the path of a raging brush fire; a package delivery franchise needs to optimize delivery routes; a consumer products manufacturer needs to ship goods to retailers based on an area's demographic information. How common is it today for casual users of the World Wide Web to look up restaurants in a distant and unfamiliar city, and expect to find driving directions from their hotel? Or peruse the web sites of their favorite consumer products (cars, bicycles, watches, stereos) and expect to locate dealerships within a specified radius of the user's postal code? What will be the effect on those businesses if their web server hangs as a result of too much load? Or on the package delivery franchise if a disk crashes on their system? How would a hardware fault on the Emergency 911 server affect the fate of the accident victims? How disastrous could a software problem be on the system used by the forest service?

GIS has expanded into the information mainstream as a core technology. As organizations spatially enable core applications, and as the whole enterprise becomes spatially enabled, we have witnessed rapid growth in the amount of spatial data and in the number of GIS users. And as GIS integrates what used to be islands of data into large – sometimes tens of terabytes! – enterprise databases housing both spatial and tabular data, the domain of mission-critical applications expands to spatial applications.

With these changes come several major challenges: GIS data and applications are being treated as a corporate resource, just like more traditional IT implementations; and spatial applications are now commonly subjected to many of the same design principles as traditional enterprise-wide, mission-critical applications. But computers can fail, and such failures are costly to an organization if mission-critical applications cannot be remain available. A High Availability GIS solution ensures that spatial data remains available and spatial applications continue running, even during prolonged hardware failure.

The GIS environment

GIS implementations have evolved into prototypical client/server environments, minimally consisting of clients, servers, storage, and a network.

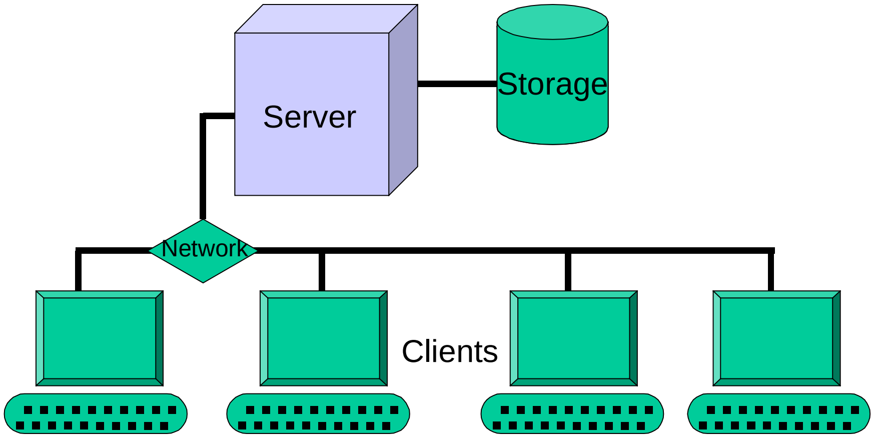

Figure 1: Standard client/server configuration

On the front end are the clients, or users, accessing applications and data. Clients' systems most commonly run the Windows NT operating system on an Intel platform. Clients can be "thin", loosely defined as using a graphical user interface (GUI) to access applications that execute on a server; or "fat", meaning that the spatial applications execute on the client platform. On the back end are data servers, providing access to what are often very large sets of both spatial and tabular data through a standard programming or user interface.

Data server hardware platforms are quite often larger servers, usually running UNIX, or perhaps Windows NT. These same hardware systems may also run the enterprise's applications, including the GIS. Increasingly though, the trend in GIS implementations is the introduction of a middle tier, creating a three-tier client/server solution. In such a configuration, the GIS and other applications would be physically relocated to the middle tier.

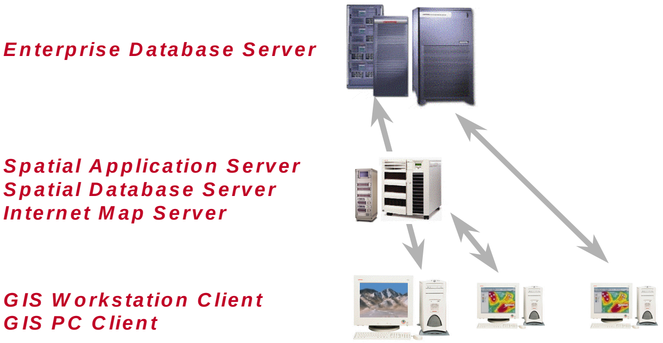

Figure 2: Three-tier GIS client/server configuration

Two important benefits of a three-tier configuration are that each system can be scaled and tuned to most efficiently provide the type of service required of it (that is, one system tuned for database access, another for web-serving or GIS applications); and functional applications can be run in a different operating environment than the database.

In either client/server model, data is stored centrally on a server system. Clients access the data by making requests over the network to a program on the server. The server program coordinates the clients' access to the data, and satisfies clients' requests by accessing the data store and responding to the clients' requests.

Risks to the environment

A GIS is "available" only if it allows users to access the spatial server applications and the spatial data files. Inherent to any application environment are various potential system failures – the result of hardware crashes, software faults, or environmental problems -- that can cause the applications and the data to not be accessible to their users. They can occur on the client systems, on the data or application servers, on the storage systems, or on the network. It is incumbent on those responsible for implementing a GIS – or any mission-critical application – to consider all potential causes of failure, assess their impact, and plan accordingly.

Consider the impact of losing an application server or data server: all clients accessing the applications and files on that server are affected. All users' activities are suspended until the server problem is diagnosed and resolved. High Availability solutions are designed for computing installations that require critical applications to be automatically and seamlessly restarted in the event of failure, ensuring that data remains accessible and that applications can be kept running, even during a prolonged failure of the second and third tiers of a client/server implementation.

[This paper does not specifically address the loss of a client, storage device, or network. A disk failure can cause loss of data, and will affect users accessing any files stored on the failed volume. Disk failure can be solved by RAID (Redundant Array of Independent Disks) technology, with mirroring and striping. While a RAID solution is an integral and essential component of a complete High Availability solution, it is not a topic unique to High Availability. Similarly, network failure can be catastrophic; clearly if a network becomes unavailable, users cannot access the application or data servers. Proper planning to minimize its occurrence and reduce its impact is necessary for any implementation that requires continuous uptime. However, treatment of redundant networks is also a separate topic, and is not treated here.]

Clustering for High Availability GIS

When considering a High Availability solution, one often thinks of expensive and custom hardware, and sophisticated, costly application design. But the more practical High Availability GIS implementation is a combination hardware and software solution requiring multiple instances of hardware (standard servers and network), and redundant and shared data (RAID). The servers work both independently and cooperatively: each is running its own set of applications or serving its own set of users, but if one server fails, the High Availability software directs that server's counterpart to take over for it.

The multiplicity of servers, and the software that enables the cooperation among them, is known as clustering, and can be implemented at both the middle and back tiers of a three-tier client/server configuration. Rather than implement a mission-critical GIS on a hardware platform that incurred the engineering cost of fault-tolerant design, an organization can much more economically configure its GIS on a cluster of two (or more) standard, inexpensive and cooperating servers. If the expected availability of either server is 99 percent, then mathematically the likelihood of both (or all) servers being down at the same time is much less than one-tenth of a percent, exceeding the general requirement of High Availability. And since the cluster nodes are typically configured to run different applications under normal circumstances, the nodes of a cluster are not redundant in the same sense as the second and third brake lights on the rear of an automobile. Thus an additional benefit of clustering, besides providing the basis of an affordable High Availability solution, is scalability: an enterprise can add members to a cluster over time to run disparate tasks and meet its computing needs. In the event of failure on one node, any of the surviving nodes can temporarily take on the workload of the failed system. Further, the nodes of a cluster do not need to be same model server or be similarly configured.

A key component in the design of High Availability clustering is its ability to continually monitor the performance of each node in the entire cluster, detect individual system failure, and automatically and seamlessly fail over affected applications to surviving nodes. If the GIS server fails, the GIS application can be automatically started up on ("failed over" to) a cooperating server, and user processes transparently switched over with it. The simplest High Availability configuration requires two clustered servers; to be sure, any discussion pertaining to a two-node cluster also applies to clusters of more than two servers. Under normal circumstances in a two-node cluster, one server could run the GIS application (and possibly additional applications as well), and the other server could run other unrelated applications. Alternatively, both servers could run the GIS application, with each instance serving a subset, or partition, of the GIS data, or a subset of the clients.

Where is the High Availability?

High Availability can either be built in at the operating environment level, or can be engineered by the application software vendor into the GIS itself. We will now look at an example of both of these approaches: Version 5.0 of Compaq Computer Corporation's *Tru64*™

0-7695-0589-2/00 $10.00 © 2000 IEEE

UNIX operating system and *TruCluster*™ Available Server, which provides a complete and robust High Availability environment on the Compaq AlphaServer™ platform; and the effort undertaken by a software vendor to incorporate similar High Availability features at the application level on the AlphaServer platform prior to the Version 5 release of the operating environment. In both cases, the user of the GIS application gains the same functional benefits. But it is important to note that if the High Availability services are built into the operating environment, then any GIS application can easily benefit from the High Availability features without the customer or the software vendor incurring additional costs of more sophisticated application design and engineering. If High Availability is not part of the operating environment, then it is the software vendor who must engineer the functionality into the application platform.

High Availability provided by the operating environment

While UNIX systems provide a solid foundation, meeting the availability requirements of mission-critical applications demands a more comprehensive, and dependable clustering solution. And to be truly affordable, such a solution must require no unique system configurations, specialized operating system variants, or proprietary storage components: the clustering environment must use the same standard server platforms, operating environment, SCSI disks, fibre channel, disk controllers, and network adapters as other systems. (Compaq uses standard and commodity components in all facets of the *Tru64* UNIX *TruCluster* products.)

TruCluster Available Server provides multi-host access to shared disks and a generic failover mechanism, making applications and data highly available. The following figure shows a typical high-availability configuration. Here, two servers each run an independent workload of applications or network services.

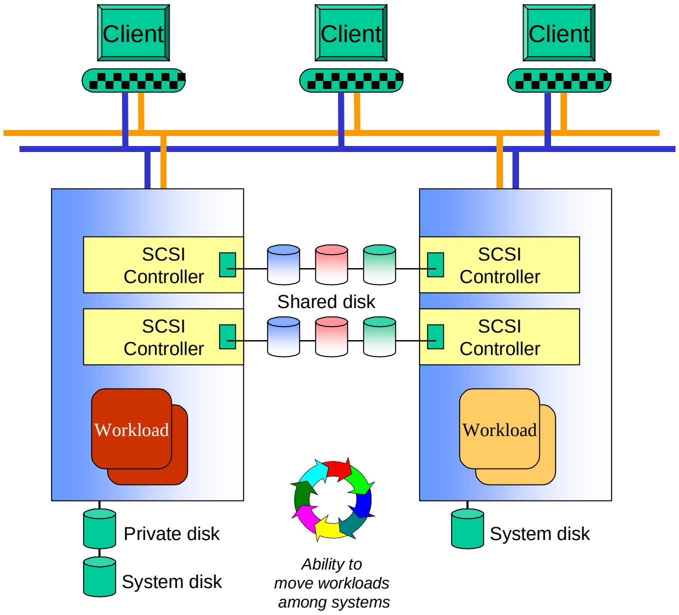

Figure 3: Highly available server configuration

Each system monitors the health of the others by watching for "heartbeat" signals sent over both network and SCSI channels. This dual monitoring system ensures reliable failure detection, while differentiating among network, I/O, and host failures. If one of the systems stops signaling, High Availability services detect the problem and automatically initiates a failover of applications to the remaining systems. As failover is initiated, the recovering system takes over the failed system's network identity and storage devices. This involves either the Advanced File System (AdvFS) built into *Tru64* UNIX or a standard database management system (DBMS). AdvFS is a journaled, local file system that provides higher availability, and greater flexibility and recovery than traditional UNIX file systems. The recovery takes just a few seconds for AdvFS; DBMS recovery time will vary, depending on the DBMS and the size of the database involved. *TruCluster* Available Server configurations generally accomplish failover within 15 to 30 seconds.

During the time that the failed server is out of commission, the surviving server may suffer a performance degradation due to its increased workload of the applications that had been running on the failed server. It is imperative that resource planning exercises take this into account when implementing a GIS. In a many-server configuration, High Availability services can be configured to set multiple specific nodes as failover targets, thereby spreading out the load of the failed server among multiple surviving servers.

Some organizations that cannot tolerate even these temporary performance impacts on cluster nodes plan for the eventuality of system failure by configuring one server in the cluster as a "hot standby": a node that is a regular member of the cluster, but under normal circumstances running no applications. The High Availability services are configured to automatically direct failover to the hot standby when a cluster node fails, resulting in steady performance on all other surviving nodes. If the cost of an idle standby system is also unacceptable to the organization, then a reasonable compromise could configure the target failover node as a development machine or with lower-priority tasks that can be temporarily suspended in the event of a primary system failure.

When the failed server is repaired and brought back online, it will resume sending "heartbeats", which will signal its restored availability and be detected by the other members of the cluster. "Failback" can now be initiated automatically or manually: those applications and users that failed over to other nodes can be returned, re-achieving a load balancing across the entire cluster. The failback will invoke the same seamless and automatic steps as the initial failover, but in reverse order from the surviving servers to the resurrected server.

User-transparent application failover can also be imposed manually, and is a common practice under two circumstances:

1. Load balancing: if one node in a cluster becomes over-burdened, one or more of its applications can be failed over to other nodes;
2. Planned upgrades: to perform a hardware or software upgrade on a node, all applications running on that node can be failed over to other members of the cluster to allow it to be temporarily taken out of service.

High Availability as an integral component of the operating environment benefits both the enterprise (*all* applications running on the cluster can potentially take advantage of all the High Availability features) and the application software vendor (no incremental cost and complexity of engineering High Availability into the application). Prior to the release of V5.0 of *Tru64* UNIX and *TruCluster* Available Server, one GIS vendor successfully implemented High Availability into their application software.

High Availability provided by the GIS platform

In this model, the clustered server systems are both connected to a shared SCSI bus which connects RAID storage volumes that are visible to both servers: both servers can share access to all volumes. Each server may also have its own system disk (not on the shared SCSI bus).

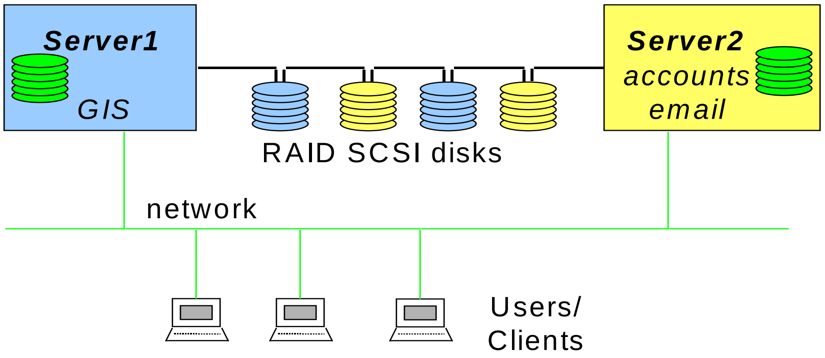

Figure 4: Simple High Availability configuration

As with any High Availability implementation, detection of node failure is critical. This was accomplished using the same model as *TruCluster* Available Server, with regular "pinging" of each of the cluster nodes. If the node running the GIS application does not respond to a ping, then the GIS application is not available, and the failover process will begin. All applications and data served by the failed system are momentarily unavailable, and all its users and clients momentarily idle. The High Availability services seamlessly take a few essential actions in order to restore normal service to the affected users:

1. The surviving server takes over the storage that was accessed by the failed server;
2. a) The GIS application that was running on the failed server is started on the surviving server;
   b) Client requests to the failed server are re-directed to the GIS application instance started up on the surviving server.

The transfer of storage access is possible because of the shared SCSI bus: all servers connected to the bus can access the shared devices, though software assures that at any point in time only one node has control of it. The failover of applications and their users is a bit more complex. Traditionally a client system is aware of a number of physical server systems available on a network, but conceptually, the client sees the network as providing a set of application services, such as *e-mail*, *accounts-receivable* and *GIS*. The user clients do not need to know on which hardware systems these application services are running; the clients locate and access the services by using a logical network address assigned to each *service* (e-mail, accounts-receivable, GIS).

The logical network address for each service is in turn mapped to the network address of the physical server (the hardware) on which the service is running. In the event of Server1 failing, its GIS application will be physically relocated to Server2, and the GIS service's logical network address will be re-mapped from Server1 to Server2. From the client's perspective, the client is accessing the logical address of the GIS service, which does not change regardless of which physical server the GIS application runs on.

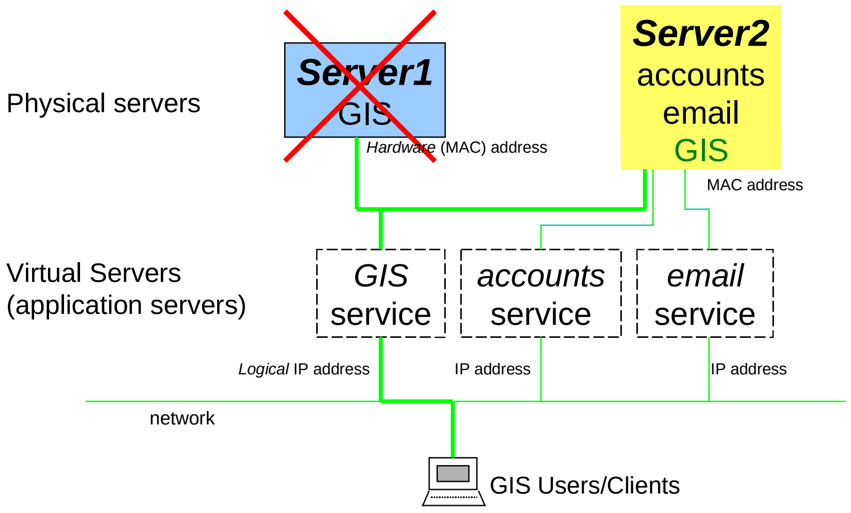

Figure 5: Automatic failover for High Availability

For clarity, suppose two applications (services) are running on one server. Each application would have a unique logical network address, which the clients would use to access either of the applications. The server would have a unique physical network address. Both logical addresses of the two applications would map to the same physical address of the server. If the server failed, the High Availability services would fail over both applications to another server, which would have its own unique physical network address. The High Availability services would re-map both applications' unique logical network addresses to the one physical network address of the surviving server. The failover is transparent to the users and the client applications because they are connecting to the server applications via the server applications' logical addresses, which have not been altered despite the hardware failures.

These actions are taken seamlessly, automatically, and transparent to the users by the High Availability engineered into the GIS software. Only after these events are carried out will the clients be able to resume normal operations. While this is occurring, a message may appear on the users' monitors informing them of a temporary problem that is being resolved. The time required to carry out the automatic failover will vary depending on a number of factors; 15-30 seconds is a realistic expectation.

Summary

In conclusion, we have looked at two functionally similar High Availability solutions for GIS environments, implemented on Tru64 UNIX by the operating environment, and on an earlier version of Tru64 UNIX and on Windows NT by an application vendor. Clusters of standard systems, rather than specialized and expensive fault-tolerant hardware, are the platform on which the environment runs. Service failover is transparent to a client user with a High Availability GIS solution. The user may experience at most a minute's disruption of service during the failover period, but once failover has completed the user can continue work without having to re-establish connection with the GIS application server: this is done transparently. In most cases users will not even be aware that a server has failed. Application failover can also be invoked manually to facilitate load balancing or rolling software upgrades. All applications installed on the cluster can easily avail themselves of these features when High Availability is implemented by the operating system.

The heterogeneous nodes of the cluster work both cooperatively and independently. They take on the load of a failed member, but also run their own workloads and can be added dynamically to the cluster to scale the environment as the customer's needs grow.