“Achieving Century Uptimes”
An Informational Series on Enterprise Computing

As Seen in *The Connection*, A Connect Publication
December 2006 – Present

About the Authors:

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein, have a combined experience of over 90 years in the implementation of fault-tolerant, highly available computing systems. This experience ranges from the early days of custom redundant systems to today’s fault-tolerant offerings from HP (NonStop) and Stratus.

Gravic, Inc.
Shadowbase Products Group
17 General Warren Blvd.
Malvern, PA 19355
610-647-6250
www.ShadowbaseSoftware.com

Achieving Century Uptimes

Part 23: Fast Failover with Active/Active Systems (2 of 2)

July/August 2010

Dr. Bill Highleyman
Paul J. Holenstein
Dr. Bruce Holenstein

Active/active systems[1] provide continuous availability not because they avoid faults but because they can recover from faults so quickly that no one notices. This capability requires that failover to a backup system not only be fast but that it be reliable.

As shown in Figure 1, an active/active system comprises a common application running on multiple geographically-distributed processing nodes. The nodes have access to geographically-distributed consistent copies of the application database. The application network survives any single point of failure with no data loss.

A transaction can be directed to any processing node in the application network. Should a node in the application network fail, all that needs to be done is to route further transactions to a surviving node. Recovery can be accomplished in seconds to subseconds. Not only is recovery fast, but it is reliable since it is known that the new node to which transactions will be routed is operational. After all, it is currently processing transactions.

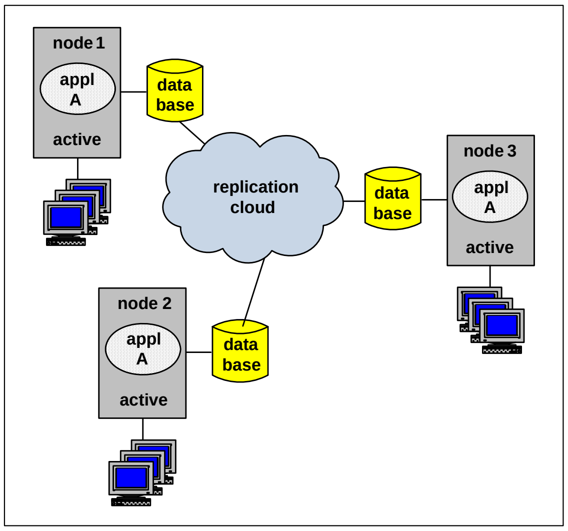

Figure 1: An Active/Active Network

The key to continuous availability with an active/active system is the ability to fail over rapidly. But how can this be done? In Part 22 of this series, we discussed client redirection and network redirection. Client redirection depends upon the client system having enough intelligence to detect a fault and to switch to a backup system. Network redirection moves this responsibility to the routers in the network.

In Part 23, we explore server redirection, in which the nodes themselves monitor faults and control network reconfiguration.

[1] What is Active/Active?, *Availability Digest*; October 2006.

Server Redirection

Fault Detection

Before users can be switched to a backup node, it must first be determined that a node has failed. With client redirection or network redirection, this is the responsibility of the client or the network, respectively.

With server redirection, fault detection is the responsibility of the nodes in the application network. Fault detection in this case is typically done with heartbeats. Heartbeats are "I'm alive" messages exchanged by the nodes. A heartbeat may be a simple ping, or it may be a more complex exchange of information. The problem with a simple ping is that the target system may be able to respond even if it is overloaded or is suffering an application failure. Heartbeats generated higher in the application stack can be significantly more meaningful. Replication traffic may serve as heartbeats with specific heartbeat messages sent only if there is a lull in replication.

It is imperative that the heartbeat network be extremely reliable. If the heartbeat network should fail between two nodes, each will assume that the other node is down and will try to seize its users – in effect, a tug-of-war. Heartbeat networks should at least be redundant networks. In many systems, the network connecting users to the nodes is used as an ultimate backup.

In order to achieve continuous availability, fault detection must be fast. Recovery from a fault cannot be initiated until it is known that there is a fault. On the other hand, it is important not to jump to conclusions and declare a fault when there is none. These are conflicting requirements. Just because one heartbeat is missed does not mean that the far node is down. The heartbeat might simply be late because of a spike in load on the far node. Therefore, it is common to wait for several missed heartbeats – typically, three – before declaring a node down. This delays the detection of real faults.

The other side of the equation is the rate at which heartbeats are sent. The faster the heartbeat rate, the higher the network load, though for a small number of nodes this may not be a significant factor. If heartbeats are sent once per second, and if three heartbeats must be missed, then fault detection time is two to three seconds. If ten heartbeats per second are sent, the fault detection time is reduced by a factor of ten.

Commanding Switchover

Once a node fault is detected, the clients using that node must be redirected to surviving nodes by the surviving nodes themselves. This can be done in several ways:

<u>Commanding the Clients</u>: A node can send supervisory messages over the user network to those clients that it wishes to acquire. Upon receipt of such a message, a client will redirect future transactions to the requesting node.

This technique requires that the clients have the intelligence to process such requests and to switch their IP addresses. Furthermore, if there is no broadcast or multicast capability to communicate with the clients, individual messages will have to be sent to the clients by the acquiring node. This could delay the takeover process.

Commanding the Network: If the client systems do not have the intelligence to process a switchover request, they know only that they are to route their transactions to a specific IP address. This can be a virtual IP address that the routers in the network can map into a physical IP address. User redirection is then accomplished via the routers.

Using this technique, a node that wishes to acquire users from a failed node sends directives to the appropriate routers to reroute the set of IP addresses being used by those clients. The routers will then route further transactions from those clients to the requesting node.

This technique depends upon the routers having the intelligence to accept such directives from the network. It solves the problem associated with the client-commanding technique since clients do not have to have rerouting intelligence.

Gratuitous ARP: Another technique for rerouting client traffic destined for a virtual IP address is to use a gratuitous ARP. Before describing the use of a gratuitous ARP, let us first review ARP, the Address Resolution Protocol used in IP networks to resolve an IP address to a physical device interface (a MAC, or Media Access Control, interface). Routing tables in the routers convert the destination IP address of an incoming message to a MAC address on the same subnet and route the message to that MAC address. The MAC address could be that of another router, or it could be a network interface card (NIC) on an endpoint – either a node or a client system.

Routing tables are self-discovering. Not only do routers periodically exchange their routing tables with their neighboring routers, but they also monitor traffic to determine new IP/MAC pairs. Every message contains the IP/MAC address pair of the source and of the destination. These address pairs are used by the routers to update their routing tables.

If a sending node does not know the MAC address of the destination to which it wants to send a message, it sends an ARP request asking for the MAC address of the desired IP address. The device (endpoint or router) that is handling that IP address responds with the MAC address to use.

A gratuitous ARP is an ARP request in which a node asks for its own IP address. Though it may not get a response, the ARP message, as does any message datagram, contains the IP address and the MAC address of the sender, which updates the routing tables of all routers on the subnet. Thus, all routers on the subnet map the IP address of the sender to the sender's own MAC address. This is often used by a node to advertise its presence on the network.

A gratuitous ARP can be used by a node to seize a virtual IP address. By sending an ARP request in which its sending IP address is the virtual address that it wants to seize and the sending MAC address is its own, all further traffic directed to that virtual IP address will be routed to the requesting node. The gratuitous ARP has, in effect, rerouted all traffic from the clients that are using that virtual IP address to the requesting node.

This technique requires that all clients and nodes be on the same subnet, though that subnet can be stretched over WANs to accommodate geographically-separated nodes. If the clients and nodes are not all on the same subnet, rerouting will not be totally effective until the routing table changes have propagated throughout the network, which could take many minutes and lead to a slow failover.

DNS Redirection

One additional technique for user redirection is to update the DNS (Domain Name Server). In this case, the client knows the node with which it is communicating by its URL (Uniform Resource Locator) rather than by its IP address. The URL is an easy-to-remember name but is meaningless to an IP network. It must be converted to an IP address, a task accomplished by the DNS server.

Every network has a DNS server, whether it be local on the network or provided by the ISP. When a URL must be converted to an IP address, it is sent to the local DNS server (which, in fact, is a leaf in a global DNS tree) to obtain the IP address. If it is desired to route users using that URL to a different node, all that needs to be done is to update the DNS server with a new IP address for the URL.

This solution is simple but has one major problem. DNS entries are typically cached, and it may take several minutes for the cache to be updated with the address change. Therefore, continuous availability cannot generally be achieved with DNS redirection.

Interconnecting-Network Reliability

The nodes in an active/active network are interconnected by communication links or a communication fabric. The links are used to replicate database changes between the nodes and to exchange heartbeats. The loss of the replication network can be disastrous to an active/active network. Therefore, this network must be highly reliable. Ideally, each node pair should be interconnected by redundant independent links. The loss of an interconnect can lead to a tug-of-war and split-brain operation.

Tug-of-War

If the heartbeats between two nodes are lost, each will think that the other node is down. Each will attempt to acquire the clients connected to the other node. In the best case, this will happen once; and all that results is that the client/node relationships have been reversed. In the worst case, the clients will be ping-ponged back and forth between the nodes.

A tug-of-war will not happen with either client redirection or network redirection because the failure condition is not being determined by the nodes.

Split-Brain

If the replication network between two nodes fails, replication stops. If no action is taken, both nodes will continue in operation, updating their databases based on transactions received from their clients; but the nodal databases will not see the updates being made to the other node. The databases will diverge, and transactions will be executed against stale data. When the replication link is restored, there may be a raft of data collisions that will have to be resolved.

Split-brain operation is a problem with all forms of user redirection – client, network, and server. If database divergence or data collisions are not acceptable, one of the nodes must be shut down and all users switched to the other node until the replication network is returned to service. If shut-down is to be automated, a tug-of-war between the nodes must be avoided.

Reconfiguration in the Presence of a Network Failure

If tug-of-war and split-brain operations are to be avoided, one of the nodes must be shut down. There must be an adjudicator to determine which node that is. There are several typical solutions to this problem.

<u>Cluster Lock</u>: Failover can be automated using a cluster lock. The cluster lock must be on some third independent system. A node cannot force a switchover unless it has acquired the cluster lock. Should the heartbeat or replication network fail, a node must acquire the cluster lock before it can seize the clients using the other node. Only one node will be successful and will end up handling the traffic from all clients. The other node will be idle and will be effectively down.

<u>Quorum System</u>: A third system, as shown in Figure 2, can be provided to be the adjudicator. This is commonly used in cluster architectures to reconfigure a cluster following a failure. The quorum system receives the heartbeats from all nodes. The heartbeats contain several health parameters, one of which is the health of the replication network.

The quorum system can deduce the network health from the heartbeat messages and will decide when a node is to be shut down to avoid split-brain operation or a tug-of-war.

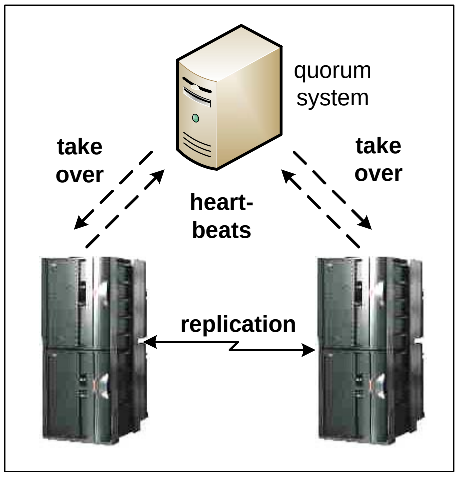

Figure 2: Quorum Monitoring

<u>Manual Intervention</u>: Perhaps the simplest solution is to shift the decision-making to the system operators. When a heartbeat or replication network failure is detected, all nodes pause and wait for manual commands to continue. The system operators diagnose the problem and determine what remedial action should be taken. This has the effect of delaying failover for several minutes and does not provide continuous availability following an interconnect failover. However, if the interconnect network is redundant and independent, this situation should hardly ever happen.

Summary

When a failure occurs in a distributed system, several factors influence failover time: detecting the failure, determining its cause, deciding upon the course of action, obtaining approval, rebuilding the database if necessary, restarting the applications, reconfiguring the network if necessary, and testing the system before restoring it to service. As a consequence, failover can take several hours.

Active/active systems can eliminate all but the first factor – detecting the failure. Therefore, failover times can be reduced to seconds or less, potentially achieving continuous availability.

As we look at the fast-failover options – client redirection, network redirection, and server redirection – we can see that there are proactive and reactive failover strategies. Client redirection is reactive. A failover is not initiated until the client detects a fault. Network and server redirection is proactive – a fault is detected and failover is completed before most of the clients even know that there has been a fault.

The preferred strategy from an availability viewpoint depends upon the application parameters. For instance, let us assume that our system serves 2,700 users averaging one transaction every three minutes – 15 transactions per second. We have the choice of using server redirection with a failover time (determined primarily by the heartbeat interval) of five seconds, or client redirection, in which a user is delayed by 0.5 seconds as it fails over.

With server redirection, 75 users (fifteen transactions over five seconds) will be delayed by five seconds each for a total user downtime of 375 user seconds. With client redirection, 2,700 clients will be delayed by a half second each for a total downtime of 1,350 user-seconds. However, the 75 delayed users under server redirection will probably be aware of the five seconds of downtime; whereas the 2,700 clients under client redirection may not even notice the half-second downtime.

There are many options for user redirection to achieve fast failover. The correct choice depends upon where the failover intelligence can be placed and on the failover parameters desired.