Maximize Availability & Performance

By
Melanie Kacerek

HPWorld 2001 Paper #50
High Availability Track

Quest Software, Inc.
8001 Irvine Center Drive
Irvine, California 92692

949-754-8000 telephone
949-754-8999 fax
mkacerek@quest.com email

www.quest.com

Maximize Availability & Performance

==

By Melanie Kacerek
Quest Software, Inc.

Backups, extracts, reorgs, upgrades, data block corruptions, disk failures, fail-overs, tornadoes, hurricanes, and earthquakes – these are all common interruptions to today's business operations. In actuality, most outages are a result of human mistakes such as untested updates and patches, rather than natural disasters. However, for companies which are seriously pursuing continuous 24-by-7 operations, preventative measures must be taken to address both types of downtime.

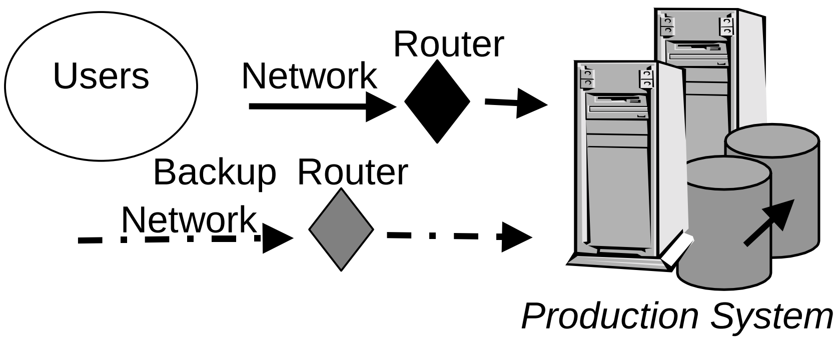

Diagram 1 Redundancy for Disaster Resilience

Redundancy is the key to availability

Redundancy is the key underlying principal to most disaster tolerance strategies. Many companies invest in secondary disks, secondary networks, secondary routers, secondary systems, and sometimes even secondary data centers in order to protect their business. However, if you carefully review the diagram above that depicts a standard schematic for disaster preparedness, you will notice that it still contains a single point of failure. That vulnerability is arguably the most critical element – the database.

Protecting the database is often overlooked or left undone because many of the early options were too costly. The early solutions to database protection were expensive to purchase, time-consuming to deploy, and labor and CPU-intensive to maintain.

Criteria for selecting a solution

Today's high availability options vary widely in effectiveness. Your organization's requirements will determine your criteria. Some points to consider when evaluating high availability solutions for your databases are:

- Costs
- Completeness
- Minimized fail-over times
- Independence
- Resilience
- Performance
- WAN support
- Daily functionality
- Flexibility
- Scalability

Costs

When considering high availability alternatives, the ease of implementation and ongoing maintenance should be considered, since they help determine the real costs of the solution. Some "simple" solutions are actually permanently labor intensive, so the initial low cost can be deceptive.

Some such solutions would be nightly copies, backups, extracts or snapshots. When these are performed manually, the labor costs should be calculated, as well as the risks associated with unautomated procedures.

Completeness

The next item to consider is completeness. If your DR copy is only as-of last night, you have lost how many hours of work? By how many employees? What if the crisis hits at monthend?

From a completeness standpoint, two subtopics should be addressed. First, does the high availability solution you are considering provide a complete copy of your database? Does it maintain all of the data and data types you need to continue production activity?

Next, does the solution minimize your risk of lost data. Review each alternative with the following questions in mind:

- Can data ever be lost?
- When can data be lost?
- How much data can be lost?

If data can be lost during normal operations such as during the transmission over a network, the value of the proposed solution is questionable. If some data is trapped on the primary system and is lost when the primary system fails, that may or may not be acceptable, depending on the type of transaction at risk and the business being conducted.

Tradeoffs

Minimizing Fail-overs

In most cases, there is a tradeoff between the risking losing some data and minimizing downtime during a fail-over. For example, a hardware mirroring solution in synchronous mode will not lose transactions when the primary system fails. However, before business can continue on the secondary system that contained the mirrored copy of the database, time must be spent opening and recovering the redundant database. In the financial industry, every single transaction is important, and a lengthy fail-over process that insures that every transaction is recorded is preferable to losing even one deposit, withdrawal, buy, or sell order.

In contrast, to many e-commerce sites, minimizing the outage experienced by Web users takes priority over losing data. In those environments, the desire is to minimize downtime first, and minimize lost data second. For such sites, having the target instance already open and available is a priority.

Independence

Some high availability clustering solutions provide quick, automated fail-overs, so that end users are virtually unaware of the problem. Many of these clustering solutions can be depicted as shown in Diagram 2. If System A fails, users and applications can be migrated to System B. However, the database is still a single point of failure. Plus, in some solutions, the success of continued operations after a fail-over is threatened because the systems share critical components.

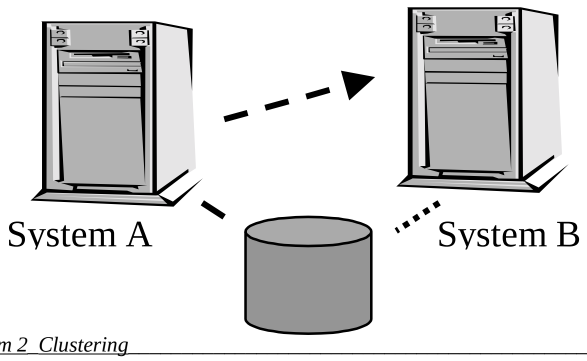

Diagram 2 Clustering

Often, clustering solutions share some components between the nodes within the cluster. These shared components can be the missing links that cause the secondary system to be incomplete. In those fail-over situations, the secondary system is then inoperable and consequently experiences a "sympathetic" failure.

Shared Nothing Disaster Resilience

To maximize availability you must maintain multiple independent, redundant databases. None of the components within the architecture may be shared between the servers or the databases on them. As a result of this design, the failure of an instance, a system, or the network does not affect the viability of the remaining systems and databases.

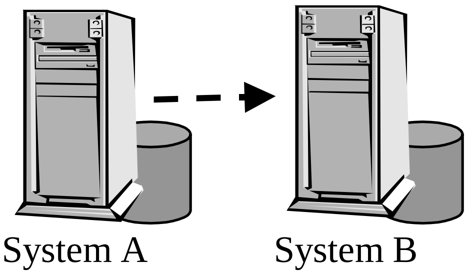

Diagram 3 Redundant systems and databases

Replication benefits

Replication provides disaster resilience in many ways. One common source of downtime is data block corruptions. Hardware mirroring will replicate these flaws. Consequently, those replicas are just as badly affected as the primary database. Log-based solutions replicate logical transactions, so they do not replicate the physical, data block corruptions. As a result, the target databases are viable fail-over options.

Another potential point of failure could be the replication queues if you select an asynchronous solution. If the queues are stored within the database, then their size is limited, and if they require more than the available space while the network is down, they could accumulate, fill the designated tablespace, and cause the production instance to fail. So, you should seek a replication solution that queues the transactional changes to the database outside the database. During the stress of a network outage, complete attention can then be focused on resolving the network problem because the replication queues do not threaten to jeopardize the production environment.

A solid replication solution is designed to solve availability issues rather than create such crises. It should be designed to automatically respond to many situations as well as their resolutions. For example, if the network fails, it should queue transactions. When the network is recovered, it should resume transporting the transactions to the target systems automatically. Similarly, if the target instance fails, a resilient solution will queue the transactions until it can regain access to that instance and resume posting.

Performance

Comparing high availability solutions must include comparing the affects of implementing the solutions. The real question is, “Does the solution adversely affect your production activity?” Trigger-based replication solutions require additional processing within your production database, chewing up precious system resources and delaying response time to your users. While this technology has improved in recent years, it still requires considerable CPU and network bandwidth.

Clustering technologies vary in their overhead requirements depending on whether the secondary systems are hot or cold. If the secondary system can be “hot,” it can be used to access the database during normal operations when the primary system is also functional. In this case, locking mechanisms must be invoked to manage the traffic. Basically, in these scenarios, the underlying database gets the traffic from multiple systems, plus the locking required to allow it. The net result is often inferior performance. A premium solution should be designed to minimize its impact on the production instance, the system, and the network.

Fast replication, or low latency, is another feature to seek. If the solution begins replicating a transaction as soon as the first operation (insert, update, delete) is recorded in the redo logs, it minimizes the delay between when a transaction is performed on the source system and when it is visible in the target instance.

Reducing the latency means that the possibility of lost data is also reduced. If the solution waits for a commit in order to begin replication or even longer for a log switch in order to transfer an archive log to the target system, the volume of data potentially trapped on the source system at the time of a primary system failure is increased.

WAN Support
If the primary business location is geographically-challenged, maintaining an alternate data center out of Mother Nature's reach makes sense. In the recent years, hurricanes have had increasingly-wide paths of destruction, causing flooding and power outages for hundreds of miles.

Another common reason for needing wide-area network support is to support bi-coastal production activity, or to enable 'follow the sun' operations. As more and more companies develop a global presence, the need for data to be available across long distances grows too.

Maximize daily performance and availability

Daily Functionality
Numerous high availability solutions provide only one functional copy of the database at a time. The other copy is inaccessible until the moment of need. A solution truly focused on availability should maintain a fully open, viable, up-to-date target instance, so daily query processing that once impeded online transactions can be relocated to the target instance. By offloading query activities, you will in turn reduce the stress on the primary production instance because you will be reducing the i/o conflicts. By reducing the i/o contention, you may reduce the possibility of an unplanned database outage while improving performance of both online transaction and report processing.

Another benefit of maintaining a fully open, viable instance is that the time required to resume production activity after the primary system is minimized because you do not need to open and recover the secondary database.

Identify exception situations that threaten your business continuity

Once you have maximized availability for daily purposes, you should focus on the exceptions. Many HA discussions focus on disasters, or unplanned downtime. However, the truth is that in most shops, planned downtime, a.k.a. maintenance, accounts for much, much more time. The sad part is that most businesses simply accept planned downtime as a necessary evil or in contrast, they forbid preventative maintenance if it involves an outage. Either way, most businesses have stopped looking for a way to perform maintenance without downtime.

Create a maintenance window

Maintenance minutes can cost companies incredibly. If the outage is visible to customers, it could reduce repeat business, adversely affect the company's reputation, and ultimately reduce the bottom line. However, maintenance cannot be eliminated. Systems and databases will demand their maintenance, whether it is planned or unplanned. So, you need to create a maintenance window that is invisible to your customers. With a log-based replication solution, you can do that, and with that window you can perform O/S upgrades, database upgrades, and database reorganizations among other maintenance chores.

The steps

The steps:

1. Create a copy of the production database

2. Record production changes in replication queues

3. Perform maintenance on replica database

4. Check your work – test the resulting database

5. Resume replication

6. Reverse roles of primary & secondary systems

Assuming you have implemented a replication solution, the first step to creating an invisible maintenance window is easy: create a copy of the production database. This step is easy, because you have a viable replica on your target system. However, setting up replication the first time is another consideration: how do you make a copy of your production database and begin replication without interrupting your production activity? Look for a replication solution that allows you to initiate it using a hot backup. With that option, creating a copy of the production database is really easy.

The replication solution needs to be able to queue production activity while the target copy of the database is unavailable, and the solution must be able to resume replication to an altered target database in order for this process to work. This limits your choices, but it is critical in order to be able to create an invisible maintenance window.

Plan the maintenance task

Once you have the replication underway, determine the steps necessary to perform the maintenance chore – whether it is an upgrade or a reorganization. If it is an upgrade, also determine the tests you need to perform after the upgrade in order to bet your business on it. What must the revised database be able to do? How can you determine that you completed the maintenance task successfully? And finally, how much time will the maintenance task (and the validation checks) require?

Determine space requirements

The time required to perform the change plus the time required to test the results determine the space requirements for the replication queues. Make sure you that you have plenty of space to cover queuing production activity in the worst case scenario. Of course, if you are performing a reorganization, make sure you have sufficient space for that, too.

Ready, get set, go!

When you are ready, stop replication and start queuing. Make a backup of the target database – just in case! With that done, you may now perform the task on the target database or system – upgrade the database, upgrade the operating system, or reorganize the database.

Test

Next, test the results.

- If your tests fail, you have the option to restore the backup and re-attempt the task, or you may restore the backup, resume replication, and plan to repeat the task after you have identified the cause of the failure.
- If your tests were successful, double-check to see if you are ready to resume replication or if you need to make any additional changes specifically for replication. (Do you need to upgrade the replication software on the target system to accommodate a new O/S version?)

Resume replication

Once you are satisfied, resume replication and get the upgraded target database caught up with the activity from the production system by allowing the queued transactions to be applied. Remember, up until this point, all the maintenance has been "invisible," being performed without interrupting production activity.

Direct activity to target database

When the target database has caught up, you can redirect activity to the freshly maintained database so that production can begin benefiting from your efforts. Redirection can be done through any number of ways – you can manually switch

IP addresses or routers, or you can use traffic management software to do so.

Congratulations! You have now performed major maintenance invisibly. Your production environment's availability was never compromised. Activity was allowed to continue on the primary system throughout the maintenance process.

Benefits

Benefits of this approach are numerous. They include:

- Reduced stress
- Time to test
- More rested DBAs

In reality, the benefits are incredible. Suddenly, you have the opportunity to revert to a proactive approach to your systems, because your maintenance window has been restored. You have the option to plan and perform the maintenance when you are rested - this process does not demand an "all-nighter" or precious weekend time. So, your DBAs and your systems can be less stressed.

If the maintenance is performed without time pressures (because production isn't waiting to resume but instead is allowed to continue without interruption), human errors due to pressure are less likely. Plus, the results of any human errors can be detected and corrected by testing before production is directed to the modified system. So, even if a mistake is made, you have the opportunity to catch it and fix it again, invisibly! The bottom line: quality results every time.

Conclusion

With a solid replication solution, you can enjoy daily performance improvements and increased availability by offloading reports and queries to the secondary systems, as well as invisible maintenance windows to further guarantee availability and improve performance by performing preventative measures as necessary.

Melanie Kacerek is the director product management at Quest Software, where she has focused on high availability solutions for nine years. Prior to working at Quest, Ms. Kacerek implemented a retail management solution for numerous companies throughout the United States of America.