HA/DR As Key Components of a Business Continuity Solution: An FAQ

High Availability / Disaster Recovery Frequently Asked Questions

May 2024

Contents

Business Continuity 5

Business Continuity – System availability - Very short scheduled maintenance windows 6

Rocket HA/DR solutions provide capabilities for short maintenance windows 6

Overview and Architecture Questions 6

What is HA/DR? 6

How are effective HA/DR solutions designed? (The Stack) 6

What parametrics are commonly utilized to compare HA/DR solutions? 7

Recovery Point Objective (RPO) 7

Recovery Time Objective (RTO) 7

Different RPO and RTO for each HA/DR solution 7

What is U2 Replication? 7

How is U2 Replication Architected? 8

Publisher / Multiple Subscriber Model 8

Transmits only changed records – Highly Efficient Architecture 8

Each Subscribing System has its own definition of files and/or accounts to replicate 9

Each Subscribing System queue may be started and stopped by itself 9

Replication protocol level changes ease upgrading and testing 9

Subscriber systems can be on a different O/S 9

Data warehouse subscribing systems on another operating system 9

Moving to another operating system 10

U2 Replication's standard architecture provides sustainability by your staff 10

U2 Replication is a Highly Flexible Architecture that provides 12 robust solutions 10

1) Failover (Subscribing) Server for Disaster Recovery 10

2) Perform Maintenance (like resizing) with limited downtime on Hardware, OS and U2 DB 10

3) Use the Failover (Subscribing) Server as a Report Server 11

4) Perform Backups from the Failover (Subscribing) Server 11

5) Build Data Warehouses on Other Operating Systems and Other Databases 11

6) Build Test Data 11

7) Build Anonymous Test Data 12

8) Enable SOX (Government) Requirements 12

9) Move to Another Operating System 12
10) Data Warehouse Consolidation 12
11) Testing a new release 12
12) Upgrade to a new release with little to no downtime 12
Planning and Implementation Questions 12
What are the steps to implement U2 Replication for HA/DR? 12
How long is the implementation project of a Single Publisher to a Single Subscriber? ("Core Project") 13
Does it take as long to implement additional items (like building a Test and Development Server)? 14
Should our company attempt to learn and implement U2 Replication ourselves? 14
We do not recommend you learn and implement this yourself. 14
Please include Rocket Customer Solutions Engineering in your plan for success. 14
Not a place for On-the-Job Training 14
Rely upon our experience, plans and methodologies 14
If you go it alone, please purchase a "Validation Package" 15
How can we cost justify implementing an HA/DR solution? 15
Do you have any recommendations on how to sell this internally in my company? 15
Best Practices 16
Do we want to replicate the whole system? 16
Usually, specific files are more efficient 16
Account Based replication is easier to maintain 16
Make sure the solution you select is supportable 16
Ensure your solution has multiple layers of support 16
Document the solution – Use the "Recovery Blueprint" to educate 17
Use MVX: Performance to Monitor your replication processes 17
How frequently should you test failover/recovery? 17
Periodic testing should occur annually at a minimum 17
Use MVX: Performance to Monitor your replication processes 18
How thorough does the disaster drill need to be? 18
Technical Questions 18
What is the server overhead associated with U2 Replication? 18
Where is the overhead noticed in hardware usage? 19
Memory: 19

Disk Subsystem / SAN: 19
CPU: 19
HW Sizing Verified / HW Contingency / Formal Benchmark 19
How much bandwidth will be required between Publisher and a Remote Subscriber (WAN)? 19
What is the network overhead associated with U2 Replication? 19
Where to place the transaction logs for U2 Replication? 20
When is the best time to implement U2 Replication? 20
Is there anything we can do to reduce the overhead? 20
Yes, Perform an Application Optimization Project 20
Use U2 Audit Logging to identify Unnecessary Writes 21
Perform a HealthCheck 21

Business Continuity

Business Continuity encompasses a loosely defined set of planning, preparatory and related activities intended to ensure that an organization's critical business functions will either continue to operate despite serious incidents or disasters or will be recovered within a reasonably short period. As such, Business Continuity includes three key elements:

1. **Resiliency:** designing and engineering critical business functions and the supporting infrastructure to withstand most disruptions. Redundancy and spare capacity are vital to building a resilient infrastructure.
2. **Recovery:** plans for recovering and/or restoring critical and less critical business functions. High Availability and Disaster Recover (HA/DR) of your computing structure falls in this category.
3. **Contingency:** a plan for coping effectively including preparations if resiliency and recovery plans are insufficient.

Your organization will have to define the business impact for each alternative, or component, which is researched and implemented. Organizations should be quite formal in defining the Service Level Agreements (SLAs) for responsiveness to an incident or disaster. Within your organization, individual workgroups may have quite varying requirements for recovery and the duration that they can be down.

The associated adverse impact of an incident or disaster to your organization often drives the allocated budget for implementing an alternative. Included in this portion of the impact is the reputation (or public image or identity) of the organization's ability to recover. We cannot emphasize enough how important it is to quantify the adverse impact of an incident or disaster to your organization. By quantifying the impact, your organization can prioritize and provide budget for those areas that are most impacted.

Business Continuity is a continual life cycle process that should be reviewed constantly and at a minimum, annually. You need to implement a (proposed) life cycle loop of 1) analysis of impact, 2) proposed solutions, 3) design of solutions, 4) solution implementation, and 5) testing & acceptance review. Different or additional steps may be included by your organization.

Any Business Continuity plan includes both manual and automated practices that your organization will perform during a disaster. Some key manual practices to include:

- Defining the manual process that you will use to take orders while your computer system is down.
- Determining how to communicate within your organization.
- Outlining actions that your staff will perform to get back up including an estimate of when your computer system will be back up.

For the computing solutions at your organization, High Availability and Disaster Recover (HA/DR) are key foundation components which should be implemented. Accuracy is increased through automation. As much that can be automated, should be automated, to increase accuracy and to speed recovery.

Business Continuity – System availability - Very short scheduled maintenance windows

Another key aspect of business continuity is having the production servers available for as much time as possible. Performing maintenance is inevitable for:

- Hardware or firmware
- Operating systems
- Application software changes and
- Database reorganization (called file resizing)

Rocket HA/DR solutions provide capabilities for short maintenance windows

Rocket HA/DR solutions provide the ability to move from server to server very quickly. As you run on one server, the other server is available for you to perform maintenance. This key architecture component means that your downtime is only the time it takes to orchestrate and move from server to server – commonly almost instantaneously.

Now you can meet your Service Level Agreements (SLAs) with much greater ease. These solutions also provide for increased quality of the operating environment, thereby making your computing solution even more stable and resilient.

Overview and Architecture Questions

What is HA/DR?

High Availability (HA) is a complete solution that can recover quickly and accurately from a system failure. Usually, the intent is to recover to the same server or to have two servers that are clustered together to recover quickly for High Availability.

Disaster Recovery (DR) is the ability to recover accurately to a second server, usually at another location. A DR server may be across the room, across the campus, across the city, or across the nation. This way, for example, should the primary data center burn to the ground, the applications and data would be located at another facility, thereby preserving the permanence of the organization due to the ability to recover at the second location.

Recovery Point Objective of zero data loss is possible with High Availability. Usually there is some minimal data loss with Disaster Recovery implementations to a remote location.

How are effective HA/DR solutions designed? (The Stack)

Effective HA/DR solutions are built upon the hardware, operating system, virtualization software and Rocket® U2 Replication. This is commonly called “The Stack” of the solution. Each component has been tested upon each other component.

Rocket U2 databases will utilize whatever capabilities are presented by the hardware, operating system, or virtualization software. For example, if the hardware has redundant capabilities and this is presented through the operating system as a single resource, Rocket U2 Replication can capitalize on that

capability. Dual power supplies are an example of this. The same is true for a SAN disk subsystem or virtualization capabilities.

This “Stack” of resources is the foundation of what can be architected and implemented utilizing Rocket U2 Replication. The “Stack” is not complete unless U2 Replication is incorporated into the “Stack.“ Without U2 Replication, the solution is incomplete, potentially resulting in significant loss of data or broken data files.

What parametrics are commonly utilized to compare HA/DR solutions?

Two standards are often used to compare HA/DR Solutions.

Recovery Point Objective (RPO)

The Recovery Point Objective defines the amount of potential data loss. An RPO0 (RPO Zero) is defined as no data loss. An RPO0 is possible with some HA solutions but can be very expensive. There will always be some data loss in Disaster Recovery (DR) to another system. Having as little data loss as possible is always everyone’s objective. However, this becomes more expensive as you move closer to no data loss.

Recovery Time Objective (RTO)

The Recovery Time Objective is the length of time it takes for you to recover in either your High Availability or Disaster Recovery scenario. The Recovery Time Objective will be different for each HA and DR solution.

The amount of automation is the greatest variable in determining the recovery time. In other words, by utilizing hardware, operating system, and automated failover software, you can design servers that will failover automatically, saving valuable time. We have implemented systems whereby the system will failover automatically and recover within three (3) to five (5) minutes to a remote location using automated failover software (not included within U2 Replication).

What we like about automatic failover systems is that you establish very formal rules for when you will fail over to another system. What we don’t like is that if the rules are not set up correctly or are not well thought out, you may find yourself failing over to a remote system when you are not expecting it.

If you do not purchase automated failover software (which is very expensive), many customers will use a manual failover process, where a person must initiate the failover. For most MultiValue customers, HA solutions are automated, whereby DR solutions are commonly manual failover.

Different RPO and RTO for each HA/DR solution

Each U2 Replication HA/DR solution described has a different Recovery Point Objective (RPO) and Recovery Time Objective (RTO). Single publisher to a single subscriber has a different RPO and RTO than a cluster replicating to a single subscriber. Ask us and we will explain what can be built to ensure that you meet your Service Level Agreements (SLA).

What is U2 Replication?

U2 Replication is a feature shared by both Rocket® UniData and Rocket® UniVerse that automatically publishes critical data to one or more subscribing U2 systems for high availability, offloaded reporting, data warehousing and more.

How is U2 Replication Architected?

Publisher / Multiple Subscriber Model

U2 Replication is based upon a publisher / subscriber model. There can be multiple subscribers (more on that later).

When a U2 application writes a record, the record that is written is captured while it is in the write buffer, written to the transaction log, and then transmitted to each subscribing server via a socket connection.

Transmits only changed records – Highly Efficient Architecture

U2 Replication only captures and transmits the written or changed records. This is an extremely efficient architecture. Indexes, records that are generated by triggers, and other items are not transmitted. The benefits of only transmitting the written or changed records are:

a) This radically reduces the amount of data transmission from the publisher to the subscribing server(s).
b) Indexes and triggers are applied on each subscribing server. This shifts the workload from the publishing server to the subscribing server, saving valuable computing resources on the publishing (production) server.
c) Each record is rehashed on the subscribing server into its correct location within the file. Therefore, all file structure changes are not transmitted from the publishing to the subscribing servers. All file expansion or contraction occurs on each subscribing server which ensures file integrity.
d) Please remember that MV records are already naturally compressed since each field is variable length. Forty (40) to sixty (60) percent of a typical first normal form database is spaces and zeroes, which MV records naturally compress. Therefore, MV databases are already efficient with data transmission.
e) In recent releases of the U2 databases, compression algorithms are used to reduce the record size immediately before transmission from the publisher to the subscriber(s). With this addition, the amount of data transmitted is even smaller.
f) Field level replication allows you to configure replication so only the fields being updated are replicated. For example, when updating just a field in a very large record, this can not only greatly reduce the amount of data being replicated but will also improve the speed and efficiency of replication.

U2 Replication is a highly efficient architecture which pushes the workload from the production server onto the subscribing servers, with considerably less network traffic.

Conversely, going to the other extreme, disk subsystem hardware-based replication replicates everything – the record written, each index, each record updated from a trigger, and everything necessary to manage the disk subsystem (File Allocation Table, Track and Sector information, etc.) Thus, the amount of network traffic is also significantly higher for disk subsystem hardware-based replication, making it the most inefficient model used.

Each Subscribing System has its own definition of files and/or accounts to replicate

Each subscribing system can have different uses. Therefore, each subscribing server has its own definition. With different uses, example definitions are:

- A failover and/or reporting subscriber would replicate all accounts/ files necessary for recovery.
- A subscriber used for building a data warehouse of sales information may only replicate a few files that specifically contain sales information.
- A subscriber that is used for building test data may replicate all the accounts / files to test the application system in total, or only the few files required to test a portion of the application.

By having separate definitions for each subscriber, each subscriber can be optimized on its own, without impacting any other subscriber.

Each Subscribing System queue may be started and stopped by itself

Each subscribing system queue may be started and stopped by itself, expanding the possibilities for use by your organization. One organization we work with has extensive month-end reporting requirements that take multiple days to run and complete. They replicate all data from the publishing server to a subscribing server used for month-end reporting. Then, at month-end, replication to the month-end reporting subscriber queue is stopped. Month-end reporting is performed on the month-end reporting server, which usually takes many days. While the replication is stopped, all transactions are retained on the publishing server. Then, after the month-end reporting has completed, replication to that subscriber is resumed. The month-end reporting subscribing server is caught up after a period, usually less than two (2) to six (6) hours, depending on volume.

The ability to start and stop queues individually also has advantages when building different data warehouses. Different business cycles can be managed by realigning portions of the operating business data store on the production server.

Replication protocol level changes ease upgrading and testing

This new feature allows a subscriber to receive logs from an older Replication protocol level. Highlighted in UniVerse 11.4.1, this feature allows you to upgrade and test a new release on the subscriber while still running an older release on the publisher. This is another valuable feature for upgrading a production system with little to no downtime. Once you've satisfactorily tested the new release on the subscriber, you can failover to the subscriber and begin running your production on the new release.

Subscriber systems can be on a different O/S

A subscribing server can be a different operating system. There are two primary uses:

- Building a data warehouse on another operating system or database
- Moving to another operating system

Data warehouse subscribing systems on another operating system

Subscribing servers can be running on a different operating system to the publisher and with the use of Rocket U2 External Database Access (EDA) the data can reside in another database such as SQL Server or Oracle. An example of this is when the publisher is running on AIX and the subscriber on Windows and the subscribed files are EDA mapped to SQL server.

Moving to another operating system

U2 Replication provides the ability to move your application and data to another operating system. For example, if you are currently on IBM AIX or HP HP-UX, you now have an easy way to move to Red Hat Linux.

It is common for MultiValue applications to call out to the operating system to perform certain functions. These need to be researched so changes can be made to the application for these system calls to work correctly.

Once you get the publisher and subscriber set up, your staff can verify that everything is working properly on the subscribing system, which is on the new operating system.

Then, in a coordinated move, you can shut down the current production server on the old operating system and come up on the new server virtually instantaneously with no data loss.

U2 Replication's standard architecture provides sustainability by your staff

U2 Replication provides a standard architecture which can be maintained and supported by your computing operations staff. Since the architecture is the same for all subscribers, your staff can grasp and understand what the system is doing, using a standard method for maintenance and supportability, although each subscriber may have different uses.

We have all seen some architectures that are so complex that only highly paid, skilled database administrators can maintain the architecture. Maintaining a standard architecture provides the ability for numerous people to be cross trained to monitor and take corrective action.

U2 Replication is a Highly Flexible Architecture that provides 12 robust solutions

1) Failover (Subscribing) Server for Disaster Recovery

The most common use for U2 Replication is to build and maintain a data set for Disaster Recovery.

2) Perform Maintenance (like resizing) with limited downtime on Hardware, OS and U2 DB

You are able to move from the production server to the failover server with virtually no downtime via XAdmin. Since you can move almost seamlessly from server to server, your maintenance downtime windows go to minutes instead of hours.

Steps are:

1) Quiesce the production server
2) Reverse the roles of the production and failover servers using XAdmin
3) Run off your failover server (as if it was production)
4) Perform the work on the former production server, such as file resizing (AKA table maintenance)
5) Quiesce the server running and reverse the roles when the maintenance is completed on the production server

U2 Replication can communicate with database releases around the same release level. This allows your organization to perform the necessary hardware, operating system and U2 database maintenance (file resizing) you require.

Now you have a method to perform hardware firmware updates, change out necessary hardware, etc. via a managed procedure.

3) Use the Failover (Subscribing) Server as a Report Server

Subscribing servers are read only. The second most common usage is to push the workload of running reports onto the Failover Server. Reporting commonly takes ten (10) times more resources than a data entry or web-based user. Pushing the workload of reporting to a subscribing server preserves vital production server resources.

One unique capability is that you can define new files where data will be written. These subscriber files are not written back to the publishing server but are instead used to gather data from multiple files to create a temporary file that is used for reporting. This file is only utilized for the duration of the time it takes to generate the report. Essentially, there is no useful information except during the running of the report.

4) Perform Backups from the Failover (Subscribing) Server

Since the failover server contains a full set of the production data, backups can be performed on the Failover (Subscribing) Server. By using the resources of the Failover (Subscribing) Server, you preserve the production server resources.

Each subscriber can be started and stopped as necessary. So, if the U2 database needs to be quiesced in order to perform a backup, starting and stopping a subscriber provides a method to perform a flash copy while the database is stopped.

Rocket still recommends that you back up the production server once a week or at least once a month to ensure continuity.

5) Build Data Warehouses on Other Operating Systems and Other Databases

Since you can have multiple subscribers, it is common for additional subscribers to be utilized to build data warehouses. Combine the capability to replicate to systems that run other operating systems with Rocket U2 External Database Access (EDA) to flatten out the file to a first normal form database, then load the data into a data warehouse.

Stated another way, the production server could be on UNIX, and the subscribing server could be on Windows, with the target database on the subscriber being SQL Server or Oracle.

How is this possible? As U2 Replication transmits only the changed data from the Publishing Server to the Subscribing Server, the complexities of the big-endian systems (store the most significant byte of a word in the smallest address and the least significant byte is stored in the largest address), and little-endian systems (in contrast, store the least significant byte in the smallest address) are handled during data transmission.

6) Build Test Data

Separate subscribers can be used to build different test data suites.

For some test data, you might want to replicate the whole data set for system integration testing. But in other cases, maybe you only need a subset of the data just to test a section of code or a portion of an application.

Since you can replicate all the data in the production environment, you can restrict your application developers from accessing it. Stated another way, since you can split off a copy of your data at any time, you are able to restrict usage on your production data server.

7) Build Anonymous Test Data

After you've used the concepts to build test data, Rocket U2 Professional Services can provide you (for a nominal fee) with routines that will create Anonymous Test Data. These routines can mask fields (such as phone number and social security numbers) and obfuscate fields (such as last names), as well as addresses. This way, when you turn the data over to your programmers or testers, the data is anonymous, thereby meeting government requirements.

8) Enable SOX (Government) Requirements

By restricting access to your production environments, and making your data anonymous, your organization can attain SOX (Government) Requirements.

9) Move to Another Operating System

Since U2 Replication supplies a method to replicate to another operating system, you can take as long as you wish to move the data and applications, verify the application works on the new operating system, and then schedule a cut over to the new operating system with just a moment of down time.

U2 Replication provides you with a way to move from AIX or HP-UX to RedHat Linux, or from UNIX to Windows.

10) Data Warehouse Consolidation

Since you can communicate with different operating systems, U2 Replication combined with U2 External Database Access (EDA) provides a method to combine data from multiple databases and operating systems onto a single data store. U2 has always been "Glue" in many operating environments, and once again Rocket provides extremely robust solutions to meet your requirements.

11) Testing a new release

If you're upgrading, please note you have the option of testing a new release on a subscriber system while still running the existing release on the production server.

12) Upgrade to a new release with little to no downtime

If you're upgrading to a new release, use the failover features to failover to the subscriber running the newer release.

Planning and Implementation Questions

What are the steps to implement U2 Replication for HA/DR?

Fundamentally, setting up U2 Replication is an education, implementation and verification project. We must train a lot of people and verify that everything was set up correctly.

1) Planning is the first step. Defining the Service Level Agreements (SLAs) to be attained and the architecture to attain those SLAs drives the hardware, operating system, network, and U2 Replication purchases.
2) Implementation involves getting the publishing system set up to replicate to the subscribing system. Your staff needs to define what should be replicated.
3) Writing the "Recovery Blueprint" document is ongoing from the beginning, and is used in education, testing and verification.
4) Educating your staff is the next key step.
5) Testing and verification is conducted during and after education. Your staff's most important step is in the testing and verification process. It is your responsibility to "Trust, but Verify" the implementation. We can test the failover process again and again until we are sure it works correctly.
6) Assistance with "Go Live" and "Post Go Live" is the last portion. Please remember that Rocket provides a dedicated resource (when we perform the project) to monitor the system for the first two days after "Go Live" so we can immediately address any issues that arise. Why? During "Go Live" is when the art of estimating the size of the transaction logs is tested by the actual workload produced by the system. This HA/DR project is not a "Time to Learn" or "On the Job Training." When your production system fails over to another system, you need to ensure that it will work and not encounter "Another Learning Moment." This is why you need as many eyes looking at the implementation as possible, reviewing it for accuracy.

Rocket Customer Solutions Engineering utilizes Total Quality Management (TQM), where everything we learn on a project goes back into the Project Plans, Methodologies, "Recovery Blueprint" Document template, and the MV products themselves. When we learn something that would adversely impact any installation, we might contact you in the future and explain to you that the "Recovery Blueprint" Document may need to be changed, or your staff educated on what we've learned.

How long is the implementation project of a Single Publisher to a Single Subscriber? ("Core Project")

Customer Solutions Engineering has developed standard project plans, tool kits and methodologies to make implementing a Single Publisher to a Single Subscriber a three-week project. This is sometimes called the "Core Project".

1) Planning can be and is accomplished during conference calls before the project, usually taking four (4) to eight (8) hours.
2) The initial setup of the publisher and getting the subscriber installed usually takes about forty (40) to sixty (60) hours.
3) Writing the "Recovery Blueprint" document, educating your staff, and performing verification takes another forty (40) hours.
4) "Go Live" and "Post Go Live" assistance is about another forty (40) hours.
5) Minimal Project Coordination is also included and is estimated at twelve (12) hours.

Calendar time, it takes about six (6) to eight (8) weeks, which usually depends on new hardware arriving, and scheduling the end user's staff for education and testing. Scheduling the down time for testing and getting on people's calendars to perform testing can be exacerbating and can add to the timeline.

Does it take as long to implement additional items (like building a Test and Development Server)?

The largest volume of work is in getting the "Core Project"–a single publisher going to a single subscriber–defined, set up, documented, tested and verified, with "Go-Live" and "Post Go-Live" assistance.

If done at the same time as the implementation of the "Core Project":

- Building one Test and Development server only adds one (1) day to the project.
- Making a single Subscribing Server into a report server only adds three (3) days.

If additional items are added some time after the "Core Project", it will take a longer period of time, for the implementation will have to be defined, implemented, documented, tested and verified again.

Should our company attempt to learn and implement U2 Replication ourselves?

We do not recommend you learn and implement this yourself.

Please include Rocket Customer Solutions Engineering in your plan for success.

Designing, building, implementing and maintaining your Disaster Recovery solution is very complex, including server hardware, firmware, storage, and networking. Configuring and implementing replication to meet your Disaster Recovery Service Level Agreements (SLAs) and requirements, while considering the performance and current SLAs in your production environment requires planning and experience. U2 Replication is not a product where you should "Teach Yourself." Disaster Recovery is not a place for "On the Job Training." Inadvertent omissions and mistakes during implementation may have severe consequences for your data and your organization.

Please include Rocket Customer Solutions Engineering in your plan for success.

There are five products Rocket Software does not recommend that you implement by yourself without assistance from Rocket Customer Solutions Engineering:

1) Automatic Data Encryption (ADE)
2) Recoverable File System (RFS)
3) U2 Audit Logging
4) U2 Replication
5) U2 Web DE

Not a place for On-the-Job Training

We highly recommend you rely upon the experience gained by the many installations that Rocket Customer Solutions Engineering has performed.

Rely upon our experience, plans and methodologies

Rocket Customer Solutions Engineering offers standard packages of services to assist you in implementing U2 Replication which includes methodologies, project plans, implementation assistance,

knowledge transfer to your staff, post "Go Live" assistance, and the "Recovery Blueprint" Document which defines how your system was implemented, including recovery procedures.

The "Recovery Blueprint" Document is distributed to your staff, your Application Provider (if applicable) and the Rocket U2 Support organization. These multiple layers of support will assist you in getting your system back up should a failure occur. They all need to be informed as to what was implemented.

Restating:

Please include Rocket Customer Solutions Engineering in your plan for success.

If you go it alone, please purchase a "Validation Package"

Should you still wish to attempt this yourself (not recommended), Rocket Customer Solutions Engineering offers a "Validation Package" where we review what you have implemented and make recommendations for an effective implementation.

Please remember that a "Validation Package" project creates follow-on work. It is common for Rocket to discover items that need to be addressed before Go-Live. So please plan ahead and schedule your "Validation Package" with enough lead time for you to perform the corrective actions.

In about 10% of the cases, the implementation was performed incorrectly, and must be completely redone. We state again that implementing HA/DR is not a place for "On the Job Training."

How can we cost justify implementing an HA/DR solution?

HA/DR solutions are just like insurance. No one wants to pay for it and we all really hope we never have to use it. But we all know we must buy insurance to protect ourselves and our livelihood. Your due diligence is in making sure your organization purchases the right HA/DR solution to fit your needs without spending too little or too much. We have learned that if you take the time to educate everyone, the answer as to what solution "makes sense" will become very evident.

Do you have any recommendations on how to sell this internally in my company?

We recommend:

1) You treat this HA/DR project like an educational process. As you educate everyone, the logical answer eventually makes sense concerning what to implement, what to spend, etc.
2) Fill out the "Cost of Being Down" spreadsheet completely.
3) Go through the "Cost of Being Down" spreadsheet.
4) Remember this is like insurance. Your due diligence is in making sure your organization purchases the right HA/DR solution to fit your needs without spending too little or too much.
5) We highly recommend that this be a company decision and not an IT decision / recommendation.
    a. This is a company problem, not just an IT problem. If your company loses money or damages its reputation because of going down, it's a company problem.
    b. If at all possible, get the VP of Sales involved, especially in the discussions around the questions of "Lost Sales" from being down, and the "Damage to the Corporate Image" of being down. In numerous cases, the VP of Sales has drawn the rest of the

organization into realizing that the organization really does need to be bullet proof with a complete HA/DR solution.

c. If the IT department makes the decision by themselves (in a vacuum), when difficulty arises, the rest of the organization will point to IT to fix it rather than realizing that there are other things that must happen organizationally to achieve a true HA/DR solution.

6) Ensure you have enough funding and a contingency amount. HA/DR is a project that may not be successful if you don't have enough funding.

Best Practices

Do we want to replicate the whole system?

Usually, specific files are more efficient

You have the option of either replicating whole accounts or just specific files within an account. While setting up to replicate a whole account is easy, you may be replicating a lot more information than you need to.

***Please note**: Account-based replication is recommended regardless of company size. We use account-based replication for even our largest sites, but supplement with file-based replication for individual files or groups of files.*

Often, MV applications use temporary files to gather data and generate reports. These temporary files are only used when the report is running. Should the system fail, these temporary files are not required for recovery. Since it is inefficient to replicate temporary data, most customers select specific files to replicate from the publishing server to the subscribing server.

Account Based replication is easier to maintain

Smaller sites commonly chose to implement account level replication as it is easier to maintain. The sacrifice is that more data will be written to the transaction logs and transmitted.

The Customer Solutions Engineering consultant will discuss the alternatives with you and you will choose what to implement.

Make sure the solution you select is supportable

Do not implement something that is too complex for your organization. You will not be the only one supporting your organization's selection.

Ensure your solution has multiple layers of support

Your organization, the company that wrote your software, and the Rocket MV Support organization will all be supporting your implementation of HA/DR.

Chose a strong "Stack"

- Hardware
- Operating system
- Automatic failover software (optional)

- MV database

Select good hardware and software vendors that you know will support you. The partners you select will be invaluable in times of crisis. Chose the wrong partners and you will regret your decision in times of need.

You know that Rocket MV Support is standing behind you, make sure the other providers of the "Stack" will supply the same excellent level of support.

Document the solution – Use the "Recovery Blueprint" to educate

Documentation leads to permanence of your organization.

The documentation will also be used to educate others–everyone in the "Stack" that is providing support to you.

This is specifically why Rocket Customer Solutions Engineering places so much emphasis on the "Recovery Blueprint" document.

Use MVX: Performance to Monitor your replication processes

MVX: Performance is available with active maintenance and provides monitoring of your publishers and subscribers. Notifications are provided when:

- Replication is suspended
- The subscriber lag time exceeds three minutes
- The publisher is generating updates faster than updates can be applied to the subscriber
- The subscriber updates are being blocked
- The replication log file usage is higher than the <repLogUseTriggerThreshold>
- Replication is down
- Replication is disabled by any groups
- Replication is up after a disabled or suspended state
- There is a stalled LSN (logical sequence number)
- Easy to read tables and charts are provided in MVX: Performance so that delays or issues can be easily identified, and no math is required to determine the amount of time it will take for the subscriber to catch up. The MVX: Performance Task Manager provides step-by-step options to resolve issues.

How frequently should you test failover/recovery?

Periodic testing should occur annually at a minimum.

Failover testing should also occur when:

- There is a significant change to the hardware environment or "stack"
    - For example, changing over to a new network provider, or new disk subsystem
- There is a significant change to the volume of data being processed
    - If your company grows through acquisition, you should ensure your system can handle the growth easily

Use MVX: Performance to Monitor your replication processes

MVX: Performance is available with active maintenance and provides monitoring of your publishers and subscribers. Notifications are provided when:

- Replication is suspended
- The subscriber lag time exceeds three minutes
- The publisher is generating updates faster than updates can be applied to the subscriber
- The subscriber updates are being blocked
- The replication log file usage is higher than the <repLogUseTriggerThreshold>
- Replication is down
- Replication is disabled by any groups
- Replication is up after a disabled or suspended state
- There is a stalled LSN (logical sequence number)

Easy to read tables and charts are provided in MVX: Performance so that delays or issues can be easily identified, and no math is required to determine the amount of time it will take for the subscriber to catch up. The MVX: Performance Task Manager provides step-by-step options to resolve issues.

How thorough does the disaster drill need to be?

Testing must include significant integration points of the database server. If your server cannot communicate with other systems with which it must integrate, your staff would have to write many programs to return the data to a synchronized state. These integration points include:

- Application servers (in the MVC model)
- Enterprise service bus (ESB) architectures
- MQ Series or other guaranteed communication software
- Mission Critical Report Servers and data warehouses
- Direct attached devices (barcode readers, warehouse guns, blood analyzers, beer analyzers, etc.)

Some integration applications allow for data to be transmitted without being committed. This is a very handy feature that saves huge amounts of time during testing.

Technical Questions

What is the server overhead associated with U2 Replication?

U2 Replication doubles the number of writes – one to the transaction logs and one to the database. The additional overhead is dependent on the number of writes the application performs.

When configured properly, U2 Replication has minimal impact on publisher performance. For example, if you needed to increase the price of all your products by a percentage, it would have one write for the product and one for the transaction log. If you had to process 1 million records in your product file, you would have 1 million associated writes going to the transaction log.

Where is the overhead noticed in hardware usage?

Memory:

The fork of the write occurs in the write buffers of the U2 databases. So, additional memory is required for the configuration of the U2 databases. With memory costs decreasing over the years, most systems already have additional memory which can be allocated to the U2 databases when implementing U2 Replication.

Disk Subsystem / SAN:

Since the writes are doubling the amount of data written for replicated files, you will need to allocate additional disk space for the transaction logs. We highly recommend disk space be added on different disk drives than where the production data set is allocated.

If your system has a SAN disk subsystem, you will need to allocate additional disk space for the transaction logs on different LUNs away from the production data set. As stated earlier, we do not recommend the transaction logs be placed on the same disk drives as the production data set.

CPU:

Normally, U2 database servers have additional CPU available. Why? U2 database servers are usually disk-constrained versus being CPU constrained. Usually, there is CPU available to perform the computing required for U2 Replication.

HW Sizing Verified / HW Contingency / Formal Benchmark

Hardware (Memory, Disk and CPU) required to run U2 Replication will be verified during the verification phase of the project (after implementation, before “Go Live”). Normally the cost of performing HW Contingency is offset by the cost of NOT performing a formal benchmark.

In other words, if you MUST verify the sizing of the HW in advance of the project, your only option is to perform a formal benchmark at a benchmarking facility. Formal Benchmarks are expensive, commonly costing over $100,000 USD to conduct. Rather than spending the money on a benchmark, most organizations choose to use the funds that would have been spent on a formal benchmark to purchase the additional memory, disk drives or CPUs that are identified as required through conducting the verification process.

By not performing the Formal Benchmark to ensure that you have enough hardware, you can repurpose the funding for the Formal Benchmark as contingency should you be required to buy new hardware to support U2 Replication.

How much bandwidth will be required between Publisher and a Remote Subscriber (WAN)?

The answer is included in the answer to the next question.

What is the network overhead associated with U2 Replication?

This requires some background information.

U2 database records are naturally compressed because they use a variable length record. On a first normal form database, 40 to 60% of the database is spaces and zeroes. Because of the compressed records and the fact that the capacity of networks has increased greatly in recent years, smaller and medium sized shops commonly already have enough bandwidth to carry the extra data written by the publisher to the subscribers for U2 Replication. Only large sites with hundreds or thousands of users need to increase their capacity.

More recently, the replication process allows data to be compressed during the network transfer phase, in a process called data link compression. When data link compression is turned on, the publisher compresses the data packets before sending them out. The subscriber then decompresses the packets after they are received. This reduces the amount of data being transferred, mitigating the overhead brought out by slow network connections. While newer versions of U2 Replication will optionally compress the data record, this does increase the load on the CPU to compress the record but will reduce the network traffic overhead. Choosing to compress the data record would be something to consider during the implementation / validation stages of your project.

Also consider upgrading to a version of U2 that offers Field-Level Update, where the entire record is no longer transmitted during updates, only the individual field that was modified is replicated - significantly improving performance.

Where to place the transaction logs for U2 Replication?

Rocket recommends that the transaction logs be placed on separate disks, when possible, away from the production data set. If the transaction logs are placed on the same disk drives as where the production data set resides, there will be competition for the disk writes.

When is the best time to implement U2 Replication?

NOW. It is always better to implement U2 Replication immediately to gain HA/DR capabilities, versus remaining exposed with no alternatives implemented.

Alternatively, consider adding U2 Replication when you install new hardware or perform a hardware upgrade. If you are planning to move to new hardware, or perform a hardware upgrade, or an operating system upgrade, it makes sense to implement U2 Replication at the same time.

Is there anything we can do to reduce the overhead?

Yes, Perform an Application Optimization Project

There are many advantages of having Customer Solutions Engineering perform an Application Optimization Project, including reducing the number of writes that the application performs. If the application performs fewer writes, there will be fewer transactions written to the transaction log and fewer transactions transmitted from the publishing server to the subscribing servers (less network overhead).

One interesting thing we learned from performing so many application optimization projects is that because most programmers wrote applications using programming standards, it is common to see the same problem throughout the application. Once we discover an area that can be optimized, we look horizontally through the entire application for the exact same issue. It is common for us to find it fifty (50) or maybe up to two hundred (200) times in the application. After we optimize all the occurrences

within the application, go through acceptance testing, and deploy to production, it is common for the application to require fewer resources to accomplish the same workload.

Use U2 Audit Logging to identify Unnecessary Writes

While U2 Audit Logging was developed primarily to track what is being read and written on a system for compliance purposes (HIPAA and SOX compliance most commonly), U2 Audit Logging traps every single write. This lets you quantify the number of writes occurring from the application, so you can see when a record is stuck in a loop and written far too many times.

U2 Audit Logging can be turned on and off without shutting down the U2 database. Therefore, if a programmer is curious as to whether or not there is a problem, he/she can temporarily enable U2 Audit Logging for a few files, capture the output, and then disable U2 Audit Logging against the same few files.

Perform a HealthCheck

If files are not sized correctly and are in overflow, it is common for the U2 database to perform multiple reads / writes to accomplish the write of the record. By ensuring the U2 database is optimized, and files are sized correctly, applications will use fewer resources. Learn more about U2 HealthChecks.