IBM i OS Journaling and High Availability

A guide to understanding the foundation of software-based replication in IBM i environments

Introduction

IBMs midrange computers, from the earliest System/38 to today's Power Systems servers, have a reputation for reliability. That reliability is in great part due to the journaling features that have been present in all variations of its operating system. Every system event and configuration change is journaled (logged) so that recovery from application errors or hardware problems is easier and faster.

Journaling was initially introduced as a feature of the operating system on IBM System/38 computers and was intended to aid technicians in their system recovery efforts. In the event of a catastrophic failure, the system administrator could reload the last good backup tape and, by applying the saved journal entries that had accumulated since that backup tape was made, restore the database to the point in time when the journal entries were last saved to tape. Less catastrophic system crashes, in which there was only the loss of data in memory, were an easier recovery. At the power-up Initial Program Load (IPL) following the crash, the journal entries stored on disk would be sequentially applied to the database, recovering the lost transactions up to the point of failure.

As one would expect, the necessary recovery process for system failures was a time-consuming operation. Retrieving the backup tapes from their offsite location and first loading the system tapes and then the journal entry tapes to recover a large and complex database took several hours at best and sometimes several days. Organizations with a low tolerance for extended downtime increasingly found the tape-based recovery method inadequate.

Journaling was not originally intended to be used as an all-encompassing high availability (HA) and disaster recovery (DR) agent; yet over time, it has become the foundation upon which HA and DR solutions are based. Subsequent improvements to journaling itself, as well as tangential technologies brought forth by both IBM and third-party vendors, have fortified the recovery and resiliency capabilities of IBM i, which now encompass security, compliance, and data-integrity checking. Journaling is a powerful feature and valuable tool for all HA and DR solutions for the IBM i. This white paper will cover what you need to know about journaling, what it can do, and how it supports HA software.

> "A journal is a chronological record of changes made to a set of data. The purpose of the journal is to provide a means to reconstruct a previous version of the set of data. When a change is made to a record in a database file that is being journaled, a copy of the record is written to the journal, along with information describing the cause of the change."
>
> - Frank Soltis, former IBM chief scientist and "Father of the AS/400"

Journaling Basics

There are two journal types of interest relative to HA for Power Systems servers running the IBM i operating system (formerly i5/OS and OS/400): the security audit journal and user journals. The former tracks changes to object properties (e.g., object creation/deletion, authorities, etc.), while the latter tracks changes to records in data files, data areas, data queues, and IFS files.

Security audit journal

This journal is specifically designed for security, allowing for auditing of all data and configuration changes taking place on the server. It's defined by the operating system and can also be used in third-party HA solutions to assist in the detection of changes for the purpose of replicating and maintaining a backup copy of the total source (production) server environment that contains both user data and configurations. The security audit journal monitors and records changes for 98 object types. HA solutions vary in how many objects they monitor through the security audit journal.

User journals

These journals are fully configurable and designed to monitor four object types: file objects (database), data area objects, data queue objects, and IFS objects. Their purpose is to capture the changes since the last backup. The remainder of this paper will discuss the features and functions of user journals as provided by the IBM i OS.

The topic of journaling, as far as this paper is concerned, starts with disaster recovery. What needs to be recreated in the event of a disaster or loss of the system? How far back in time can the recovery process go to regenerate application data changes? How much time will it take to get back to normal operations? These questions are important regardless of whether you have days or minutes to recover your critical applications.

Clearly, production data must be protected by journaling, but most businesses have data associated with non-essential operations that does not need to be journaled. For example, testing, data analysis, and other "batch" operations might be restarted from scratch or from some checkpoint, and therefore would not need to be journaled. This is also the case for temporary application work files that will be recreated automatically.

The basic journaling process (see Figure 1), sometimes referred to as "local journaling," can be described as follows:

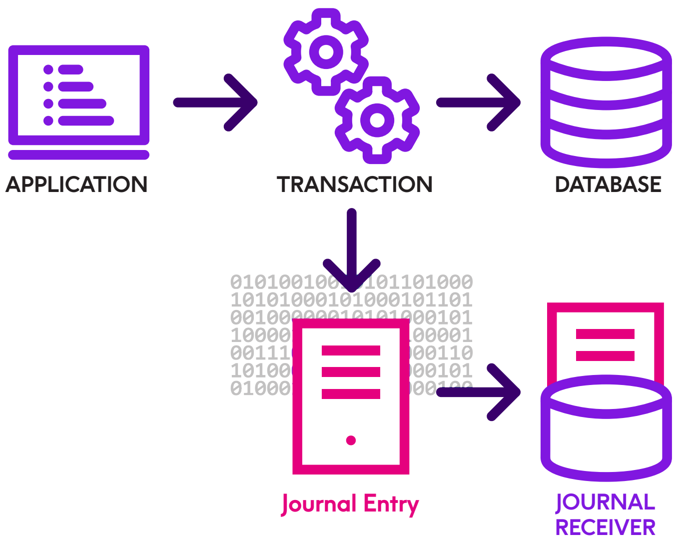
1. An application processes transactions that update records in a database file.

2. The Operating System (OS) System Licensed Internal Code (SLIC) intercepts the transactions, records the changes that have taken place, and creates a record (or "journal entry") of the actual changed data. Depending on how an application is built, including auditing level, there will likely be many entries for each transaction. Other information written in the journal entry includes the date and time of the transaction, user identification, initiating program, job identification, relative record number, library of the file being journaled, and whether the journal entry was generated by a trigger.

3. The newly created journal entry is written to a storage area called a "journal receiver," where it is available for use.

FIGURE 1: Local Journaling

Source Server



4. The database change is pinned in memory and is not released until the journal entry is written to the journal receiver on disk. In order to optimize system performance, the updated database record itself will continue to reside in main memory until forced to disk.

   Configuration of a user journal consists of deciding what libraries, objects, and files need to be journaled to ensure the system can be recovered in the event of an outage. With modern HA/DR software, the actual configuration of a journal is handled with simple clicks from a high-level GUI interface, but in its basic form, a journal receiver is first created using the Create Journal Receiver (CRTJRNRCV) command. Next, a journal is created and an associated receiver is specified using the Create Journal (CRTJRN) command. Finally, journaling is started for specific files by associating a file to a journal, accomplished with a Start Journal command such as Start Journal Physical File (STRJRNPF).

Journaling and Recovery Point Objectives

IBM i operating system storage management allocates storage to application processes and handles the retrieval and saving of all data in segments called "memory pages." Main memory is treated as a cache that is optimized for performance. As a result, data that is referenced repeatedly will be held in memory across multiple changes to the data. In fact, an object could stay in memory indefinitely. This presents a major problem for high availability and disaster recovery since the target server must be as up-to-date as possible at the point of a failure. Holding data in memory for an indeterminate amount of time makes it impossible to accurately quantify a recovery point, the time difference between the state of business operations at the point of failure and the restart point on the target server following a failure. Journaling is the solution to recover data that would otherwise be lost when the source server fails. While the database record itself remains in memory across multiple data changes, a journal of the individual changes is stored on disk in the storage subsystem in a journal receiver at the completion of each individual data change. In the event of an outage resulting in the loss of main memory, the current state of the business at the point of failure can be recreated with the sequential application of the journal entries to the copy of the database on the target server.

Local journaling captures changes on the source server and writes them to a journal receiver in storage on that server. In the event of a system crash, that journal receiver is used during power-up IPL operations to recover changed database, IFS, data area, and data queue files for objects that had not yet been saved to storage. These journal receivers can also be used by third-party HA software, which can send them to a backup server and apply them in real time, in order to be prepared for a possible failure where the target server becomes the primary or source server that runs the business operations.

> Journaling is the only solution for maintaining a zero or near-zero Recovery Point Objective (RPO) on the IBM i platform.

Due to the unknown latency between data changes made in memory and the writing of these changes to disk, journaling is the only solution for maintaining a zero or near-zero Recovery Point Objective (RPO) on the IBM i platform. This is true whether using journal-based logical replication solutions or SAN-based hardware replication solutions. All solution options must have journaling as part of the protection of the data.

Recovery from an unplanned outage means going back to a starting journal entry in the journal receiver and applying the changes sequentially to ensure that all of the changed objects lost in memory on the failed source server are accounted for. To keep this recovery process from taking an inordinately long time, a journal task continually updates this starting point. Journaling provides functionality to periodically sweep through the entire main storage to write to disk all database, IFS, data area, and data queue records that have journal entries saved to disk but the record itself has not yet been saved. After each sweep of the memory, the next journal entry becomes the starting point that will be used for recovery of transactions in the event of an outage.

Journaling Features

Journal entries have standard structures for the different object types supported by journaling. The following features can be configured to minimize the amount of data contained within the journal entry so as to optimize bandwidth usage and performance.

Selective journaling

The administrator has control over what is journaled and thus has the ability to optimize for performance and bandwidth between source and target servers. Journaling can be configured for libraries, directories, folders, and individual objects. More importantly, it can exclude things like high-volume temporary work files to decrease journaling and HA replication performance costs. Many HA and DR products have additional ease-of-use configuration capabilities such as excluding specific objects from journaled libraries.

Journal minimal data

This key journaling feature for database objects reduces the size of what is saved in the journal. Normally, the entire record, or row, is included in the journal entry. With Journal Minimal Data, only the portion of the data that was actually changed gets saved. This is based either on a bit boundary or on a field or column boundary. The savings in storage space can be significant, but it comes at the expense of reduced ability to read a journal entry for debug purposes.

Before images

These images show the record (or row) in the database as it appears before the operation makes changes to the data. This record is not included in the journal entry by default but can be included if needed by the application. System OS-based applications such as commitment control (see below) or system-managed access-path protection (SMAPP), which journals the access paths, require the before images and will override this feature as necessary.

Commitment control

This feature treats a series of transactions as a single entity. All of the transactions to a file associated with a single process are considered to be "pending" until the entire operation is complete. For example, the ATM application for a bank might have to rollback a canceled transaction, or a reservation system might have to rollback an abandoned booking. In those cases all transactions associated with those actions need to be considered pending until the entire operation is committed. Journaling is a key element of commitment control, maintaining the before images so that the changes can be rolled back if necessary.

Journal caching

Caching is a major performance feature of journaling that optimizes disk writes. It is available under Option 42 of the IBM i operating system. Since the writing of journal entries is sequential in order to maintain the sequence of the transactions, there are time delays when a series of individual write operations are made to the disk. Between all write operations, there is at least one revolution of the disk platter before the next sequential sector on that track can be located and written. By caching a full track of sectors and bundling them into a single 128K write operation, the entire track can be written in a single revolution of the disk, eliminating the wait time between sequential disk sector writes. The performance gain can be substantial. Even with an additional caching operation when using disk adapter I/O write cache hardware, additional savings will be realized with journal caching.

The caution when using journal caching is that the journal entries are held in memory until a full 128K bundle is assembled. Because of this, there is a potential of losing up to one track of journal entries if there is a source server crash.

Journal at birth

This feature allows new objects that are added to an existing database file to be enrolled into the journal associated with that file without the HA product having to monitor for new object creation events in the security audit journal and then enroll the new objects. The savings in time and complexity contribute to the overall performance of the application. Inherent Journaling is the name for a similar function used for IFS objects.

Remote Journaling

Remote journaling is an extension of local journaling that efficiently sends the journal entries to a journal receiver on a remote server (see Figure 2). It is part of the Operating System code with access to all of the optimization features available at that level below the machine interface (MI), including bypassing some of the code layers in the normal communication interface operations.

The basic remote journal process (see Figure 2) can be described as follows:

1. An application processes transactions on the source server that typically update records in a database file.

2. The OS journal code creates a journal entry of the actual changed data and sends it to the local journal receiver in the source-server storage. It also initiates a communication task to send the same journal entry to a remote journal receiver on one or more target servers.

3. The memory page remains pinned on the source server until the local journal receiver is updated.

4. Further action with the remote journal receiver is the responsibility of high availability software, but typically that software applies the journal entries in the remote journal's receiver to the database on the target server so that the state of the data is synchronized between the source and target servers in real time.

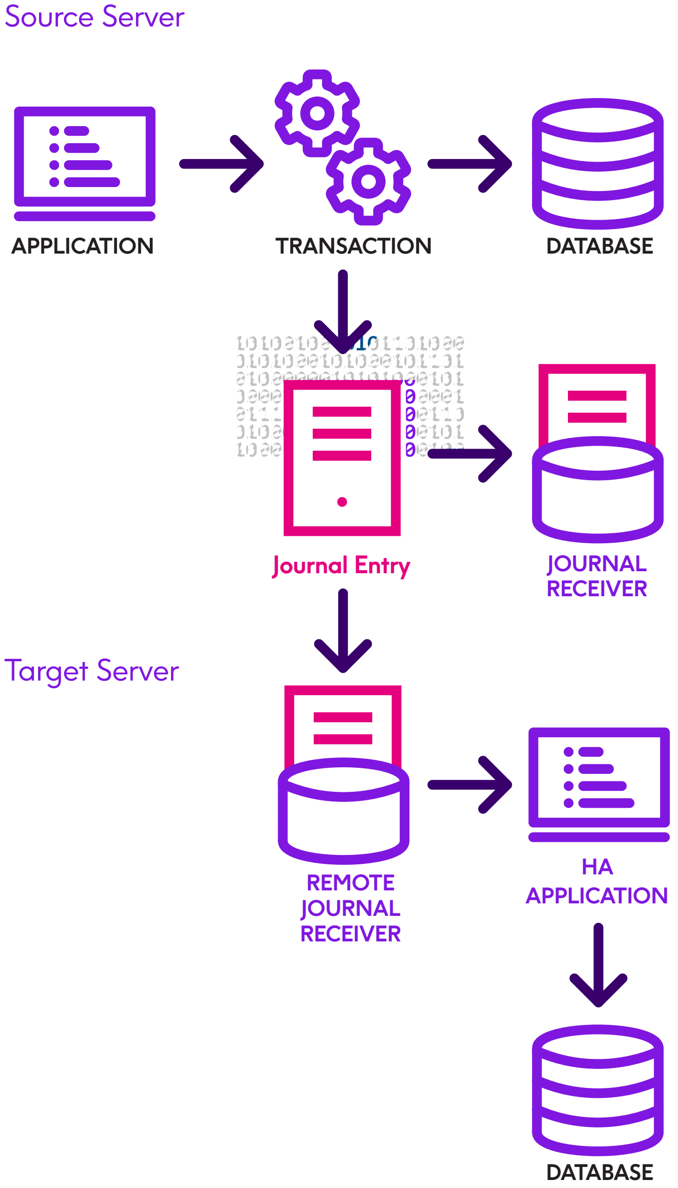
Remote journaling is designed for high availability, focusing on the efficient transfer of journal entries to a target server where they will be safe from any planned or unplanned loss of the source server. For data protected with a user journal, it provides an efficient mechanism for rapid transfer of the data changes to the target server.

Remote journaling provides solid processes for ensuring that the security and integrity of the data transfer is maintained for the HA/DR environment. This includes the following functionality:

Auditing of data transfer

Auditing keeps track of the journal entries sent to the target server. If acknowledgement from the target is not received, this function handles the resend process, guaranteeing that journaled information reaches the backup.

FIGURE 2: Remote Journaling



Broadcast mode

This functionality sends the journal entries to multiple target servers and handles the complexity associated with multiple connections. The theoretical limit of 255 connections is more than enough for any environment. This is one way that the HA application can maintain multiple target servers and keep track of the proper direction of replication to the various servers following a switch of the production environment to one of those servers.

TCP/IP (IPv4 and IPv6)

TCP/IP is the preferred supported communication transfer mechanism used to connect servers, and remote journaling supports the required configurations. This includes configuration of secure sockets, which provides the encryption needed to meet today's security requirements.

Synchronous and Asynchronous Remote Journaling

Remote journaling transmits journal entries either asynchronously or synchronously.

Synchronous remote journaling

This method holds off saving the journal entry to disk on the source server until acknowledgement of receipt of the journal entry by the target server is acknowledged. Because of this, synchronous remote journaling is more than a communication transfer function. If the journal entry is not received by the target server, the change is not recorded by the source server in the local journal receiver. This provides the potential for an RPO of zero loss of data in the event of a system failure.

Although an RPO of zero that can be achieved with synchronous remote journaling is compelling in HA and DR solutions, for most companies the effect on system performance often makes it impractical, particularly if the target server is located greater than 25 km from the source server. Servers are now capable of speeds that outrun the physical transfer capabilities across today's communication technologies, resulting in distance-determined latency times that can leave the business operations on the source server waiting.

Asynchronous remote journaling

This method does not add delays to the writing of the journal entry on the source server. The remote journal communication transfer is initiated at the same time with the assumption that it will be successful. Verification is after the fact. Most companies that use remote journaling in their HA solution find that the performance advantages of asynchronous operations are worth the tradeoff as the potential loss of in-flight data can be handled by check-pointing techniques such as commitment control.

Remote Journaling Features

Remote Journaling has many features that are designed to optimize bandwidth, performance, security, and accuracy.

Catch-up mode

When a communication connection between source and target servers is dropped for any reason, remote journaling will go inactive and an indication will be sent. Local journaling will continue on the source server, and the unsent journal entries will be buffered until the communication connection has been restored. When communication is restored and remote journaling is activated, it will come up in "catch-up mode." All of the unsent journal entries will be bundled together into the maximum packet size to optimize bandwidth usage until the journal entry backlog is eliminated. This provides a great performance boost, and speeds up the process of getting the target synchronized with the source.

Remote journal filtering

The remote journal can be configured to filter out journal entries that do not need to be sent to the target, as well as reducing the content of the journal entry. This reduces bandwidth requirements. Remote journal filtering is available under Option 42 of the IBM i operating system. Three criteria can be used to filter entries sent to the remote system:

- Filtering of before images—This type of filtering eliminates the before image from the journal entry that is sent to the target server and reduces bandwidth requirements related to journaling. While before images can be removed from what is saved in the journal entry, they are necessary when implementing commitment control. However, rollback is something that occurs on the source server, not the target server. Thus, journal filtering allows for keeping the before images in the journal entry for commitment control on the source server while not sending them to the target server.

- Filtering by object—Object filtering is a major tool for additional bandwidth reduction. Not only does it allow the administrator to filter out user-related data objects such as temporary files, but there are also files that are journaled by the OS on the source server for debug or commitment control purposes that are not needed for HA and DR on the target server. These are filtered out by default when remote journal filtering is activated and can result in significant bandwidth savings.

- Filtering by program name—This filtering is designed for active-active solutions where all servers are both source and target servers. HA solutions that provide this feature need to know that the journal entry coming back from the target server did not originate from the source server by the HA application.

The criteria related to the three filtering functions are specified when activating a remote journal. Different remote journals or individual local journals can have different filter criteria. It should be noted that remote journal filtering can be specified only for asynchronous remote journal connections.

Journal caching

This feature of local journaling also works with remote journaling to decrease bandwidth requirements over communication links. The journal on the source server will bundle the sequential journal entries into 128K blocks and send it to the communication interface as a single operation. The entire set of bundled journal entries will not be written to the source server storage subsystem until acknowledgement is received from the target server.

Journal minimal data

This feature of local journaling also works with remote journaling to reduce the size of the journal entry and reduce remote journal bandwidth requirements.

Remote journal validity checking

This feature ensures that the communication process is not introducing errors into the data being transmitted when operating over unstable networks. The TCP standard communication interface error-detection logic will retransmit packets when it detects single-bit errors in the transmission. There is a finite risk of double-bit errors going undetected by this logic. Remote journal validity checking can be activated to provide a data check that encompasses the entire journal entry transfer operation; otherwise, these errors introduced by the network will go undetected when the journal entry is saved in the copy of the user database. In this case detection of the errors relies solely on auditing and repair functions within the HA software.

Secure remote journal

This feature provides encryption of the journal entry during transfer of the journal entry over the communication link. The journal entry is un-encrypted when placed in the remote journal receiver on the target server.

Journal receiver read authority

The administrator can determine the authority level required to read the journal receiver, reducing the exposure of unauthorized access to data involved in the journaling process, whether local journaling or remote journaling.

How HA/DR Applications Build Upon Journaling to Provide a Complete Solution

The fundamental role of an HA solution is twofold:

1. To provide quick recovery of operations in any unplanned downtime scenario
2. To provide continuity of operations during planned downtime events

What journaling provides is a solid platform on which an HA solution can be built— whether hardware-based or software-based. It offers protection for transaction integrity in an IBM i environment where memory is used as performance cache.

It is essential that your software-based HA solution manage and maintain the following:

System synchronization

Synchronization ensures that all transactions to the source server database have been recorded to the copy of the database on the target server. If the data and objects on a target server are not identical to those on the source server, the ability to switch the business operations to the target server when required may be compromised. It is this switch confidence that is key to any successful HA solution.

Replicating non-journaled objects

Journaling handles the four basic object types that make up the vast majority of user data. However, there are other objects that may be important to the user's environment and are critical for maintaining the business when operations are switched to the target server. An example would-be changes to objects such as program objects or user profiles, which are not reflected in a journal and must be handled independently to properly synchronize source and backup servers. Failure to do so may jeopardize the success of a planned or unplanned switch. HA software products typically provide object-level replication services that handle some number of these vital non-journaled objects.

If the target server is to be complete and ready to run critical business applications when called upon to do so, the HA solution must also replicate application objects (programs, user profiles, authorization lists, configuration objects, spooled files, etc.). In this case, in addition to applying the journaled changes to the copy of the database on the target server, work must be performed on the source server to capture and send the occasional but necessary changes of these other objects to the target server.

Data integrity

The ability to verify that the contents of the copy of the database are the same as the contents of the primary database at any given time is critical to ensuring effective HA.

IBM i OS journaling on its own does not provide source and target synchronization nor integrity validation. This functionality is provided only by advanced HA software. Without it, switching the production environment to the target (the act of substituting the target for the source server, also referred to as "role swap") cannot reliably occur.

Switching

One of the more critical features of any HA solution is the ability to quickly utilize your target server as the production environment, whether during a test or during an actual downtime event. This process is referred to as a switch or role swap. There are many factors to a successful switch, which include HA software functionality as well as strong internal processes and regular testing.

Housekeeping

It is the responsibility of an HA solution to remove journal entries that are no longer needed from the target server once they are applied to the database. Entries prematurely designated for removal would result in data-integrity or synchronization problems. Some HA products provide journal management tools that coordinate with their journal-apply processes so that journal entries are deleted only after the HA software is finished with them. These products can coordinate the deletion of both remote and local journal receiver entries.

In Summary

The foundation of every software-based high availability solution is journaling— both local and remote. In fact, even hardware-based HA solutions require local journaling to maintain system integrity. Integrated within the IBM i operating system, it is the critical plumbing of local and remote journaling that makes it possible to meet aggressive objectives for rapid recovery time (RTO), a complete recovery point (RPO), and low impact on network bandwidth. Software vendors have built HA solutions around these powerful journaling capabilities, with the top vendors providing a suite of capabilities necessary for a complete, reliable HA solution.

Precisely offers several journal-based high availability solutions that scale to handle the workloads of SMBs to the largest enterprises. Built-in audits with self-healing and automated procedures assure confident switching. And a variety of easy-of-use features enable the solutions to be managed in ust minutes a day.

Armed with the knowledge of journaling, do your homework to understand each component of an HA solution. Fully recognize your business requirements and apply the appropriate technology, methods, and skills to achieve a solution that addresses your specific availability needs and provides a satisfactory ROI. Precisely is here to help with all your software and service needs for IBM i high availability.

precisely

About Precisely

Precisely is the global leader in data integrity, providing accuracy and consistency in data for 12,000 customers in more than 100 countries, including 90 percent of the Fortune 100. Precisely's data integration, data quality, location intelligence, and data enrichment products power better business decisions to create better outcomes.

Learn more at precisely.com.