

The Benefits of Remote Journaling in iSeries High Availability Solutions

A White Paper



Executive Summary

Two kinds of data transport engines exist in iSeries high availability solutions: proprietary harvest-and-send, and the remote journaling capabilities of OS/400. Recent information from IBM clearly points out that when remote journaling is used in high availability solutions to transmit changes to mirrored files, significant performance, reliability and data protection benefits are realized for most shops:

- *Reduced Overhead* – Remote journaling moves most of the necessary processing to the target system significantly reducing system overhead on the source system.
- *Shorter Time to Wire* – Since remote journaling runs entirely at the level of the operating system, the time needed to place data on the communication wire is substantially reduced.
- *Less Data Vulnerability* – Because of the reduced overhead, and because of how quickly remote journaling puts data on the communication wire, there is less exposure to data loss in the event of a system failure on the source.

Remote journaling is a function of the OS/400 operating system that works in conjunction with the normal journaling functions of OS/400 to essentially duplicate journal entries created on one iSeries machine and transmit them to another connected iSeries machine. Because of this capability, remote journaling provides numerous data protection and recovery benefits to iSeries shops, particularly when its capabilities are integrated in iSeries high availability (HA) solutions.

Before examining how remote journaling benefits HA, let's first take a closer look at the components of journaling and remote journaling to give you a better understanding of the process.

The Journaling Process

When you enable journaling for an object, a process is initiated in the operating system that makes a record of everything that happens to the object. The journaling process consists of two components: the 'journal' and the 'journal receiver.' When any change occurs to the object that the journal is 'watching,' the journal writes everything about this change in a very compact record called a 'journal entry.' Each journal

entry is stored in an object called a journal receiver. When the journal receiver accumulates a preset number of journal entries, the journal receiver is 'changed' and a new, empty journal receiver is then associated with the journal. The changed journal receiver is then available to be saved to tape.

When journaling is used with tape saves in a backup and recovery strategy, the saved journal receivers can be retrieved and the journal entries can be 'applied' to the data in the restored production file in the event of a system failure between tape saves. This effectively restores the data back to the state it was in at the point in time that last journal receiver was saved to tape.

When remote journaling is added to the journaling process, the system essentially sends a copy of all relevant journal entries to a copy of the journal receiver on a second iSeries system, thereby creating a real-time 'back up' of journaled information.

All high availability solutions use journaling to track data changes to the objects that are selected to be mirrored by the HA software on a backup system (target machine). As changes are made to data on the production system (source

machine) the HA software ‘harvests’ the data-change information from the journal entries and ‘applies’ this information to copies of the objects on the backup system.

‘Harvesting’ Journalled Data

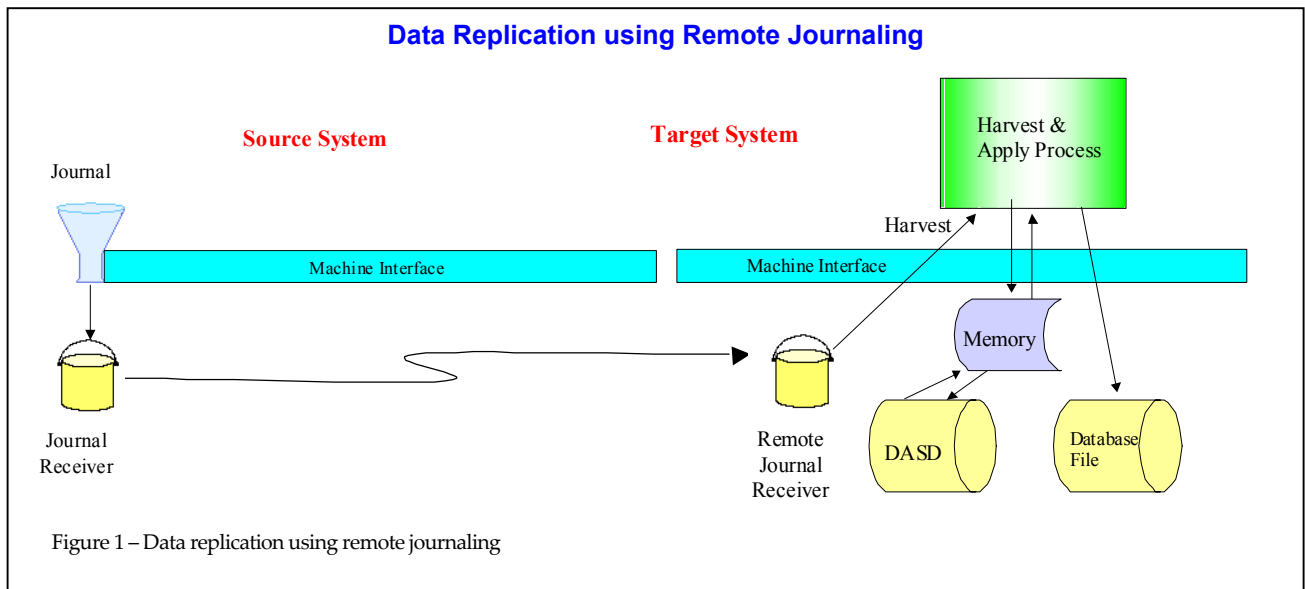
One of the primary differences between HA solutions are which machine that the solution uses to “harvest” the data from journal entries. HA solutions that harvest journal entries from the source system use their own proprietary process to extract, filter and send this information to the target system. HA solutions that harvest journal entries from the target system incorporate remote journaling since the journal entries must first be moved to the target before they are harvested. The difference between these two methods is illustrated in the next two graphics.

The diagram in Figure 1 illustrates the data replication process in which remote journaling is used to transport the data changes between

identical journal receiver on the target system. Remote journaling transports this data very quickly—typically, the journal entry is placed on the communication wire on the source machine in less than 5 milliseconds. The reason for this remarkable speed is the entire process is performed within the operating system (beneath the machine interface at the SLIC level).

Once the journal entry lands in the journal receiver on the target system, a ‘harvest and apply’ process within the HA software then extracts information from the remote journal receiver, filters and validates the data, and then applies the change to the database file on the target system.

Figure 2 illustrates a ‘harvest-and-send’ transport process in an HA solution. As with the previous diagram, journaling is used to detect changes on the source system. In this process, a job within the HA software harvests information from the journal entry on the



machines. As changes are made to application data, journaling detects these changes on the source, creates journal entries, and automatically transmits a duplicate of each journal entry via remote journaling to an

source system. Once harvested, other jobs within the HA software on the source system validate and filter the information, while another HA job transmits the information to the target system. This process results in moving

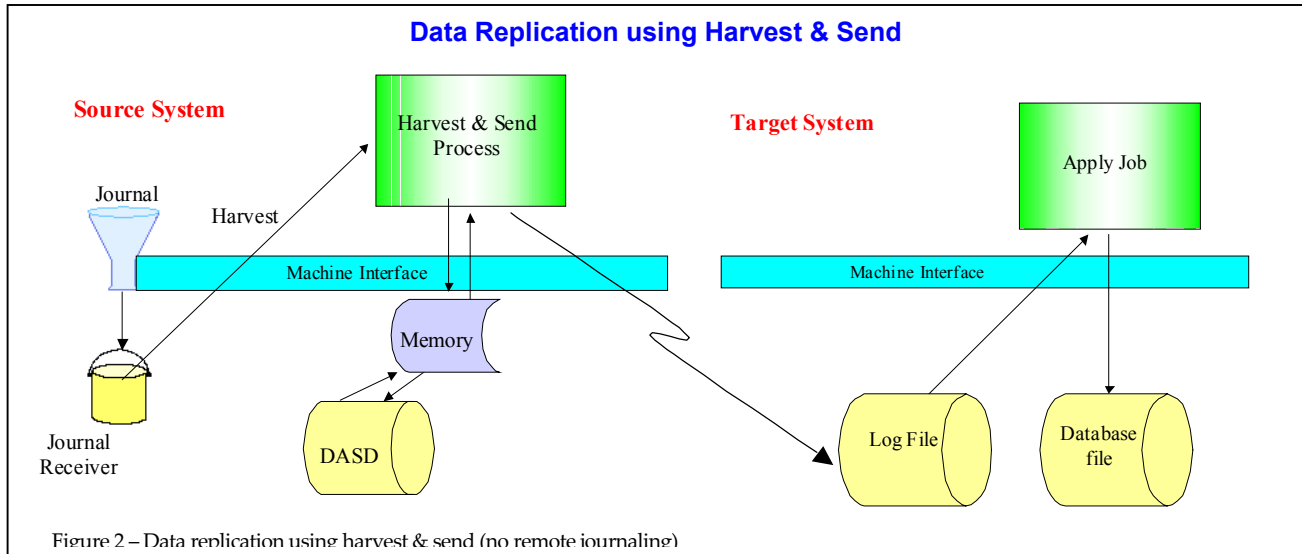


Figure 2 – Data replication using harvest & send (no remote journaling)

and mapping data many times through the machine interface, which creates a significant amount of overhead on the source system (more on this later).

When the harvested information reaches the target system, it is put in a temporary storage area—typically a log file. An apply job within the HA software then extracts this information and applies it to the necessary database files to bring it current with the source machine.

IBM’s Bench Tests

The advantages and disadvantages of the remote journaling and harvest-and-send transport processes will be examined more closely in the remainder of this white paper in relation to findings from studies conducted by IBM and documented in their Redbook, Striving for Optimal Journal Performance on DB2 Universal Database for iSeries (SFOJP)¹.

Chapter 6 of the SFOJP Redbook compares the data transport abilities of remote journaling to proprietary harvest-and-save processes through a variety of tests. Most of the findings clearly support remote journaling as the replication engine of choice. Consider the following from the Redbook regarding the performance aspects of these two processes that came from tests

conducted in a specialized, controlled computing environment.²

The remote journal function is a part of the base OS/400 system and is not a separate product or feature. It is implemented at the Licensed Internal Code layer. The benefits of the remote journal function include:

- *It lowers the CPU consumption by as much as 10% on the source machine by shifting the processing required to harvest the journal entries from the source system to the target system.*
- *It eliminates the need to buffer copies of harvested journal entries to a temporary area before transmitting them from the source system. This translates into less disk writes and greater DASD efficiency on the source system.*
- *Since it is implemented in microcode, it significantly improves the replication performance and transmission efficiency of sending of journal entries to the target system.*

IBM: Striving for Optimal Journal Performance, Chapter 6.1

As shown previously in Figure 2, when HA software uses a proprietary harvest-and-send process, the data must be harvested from journal entries on the source machine, stored in a temporary area on that machine, and then validated and filtered before being sent. This translates into several passes through the machine interface on the source machine,

including disk writes and reads, creating a significant amount of additional overhead on that machine. In addition, because of this increased activity on the source machine, data is passed in and out of memory buffers creating more vulnerability should the source machine abruptly end.

Any [harvest and send] approach which must wait until alerted that a new journal entry has arrived, then seek to be scheduled to use the CPU, then harvest the new journal entries, and finally filter out the interesting ones and place them on the wire, obviously will have more path length and a natural delay of some magnitude. The non-remote-journal-enabled versions of most [HA] products fall in this category and though they try heroically to keep current, it's difficult for them to achieve this objective...especially when the volume of journal traffic is as high as we employed during [our] tests.

IBM :Striving for Optimal Journal Performance, Chapter 7.1

In one test conducted by IBM using a harvest-and-send transport process, 7% additional CPU consumption was required on the source machine for this process. In contrast, the same test was conducted using a remote journaling transport process, which only added about 0.5% additional CPU.³ The reason for the difference is that with remote journaling, most of the

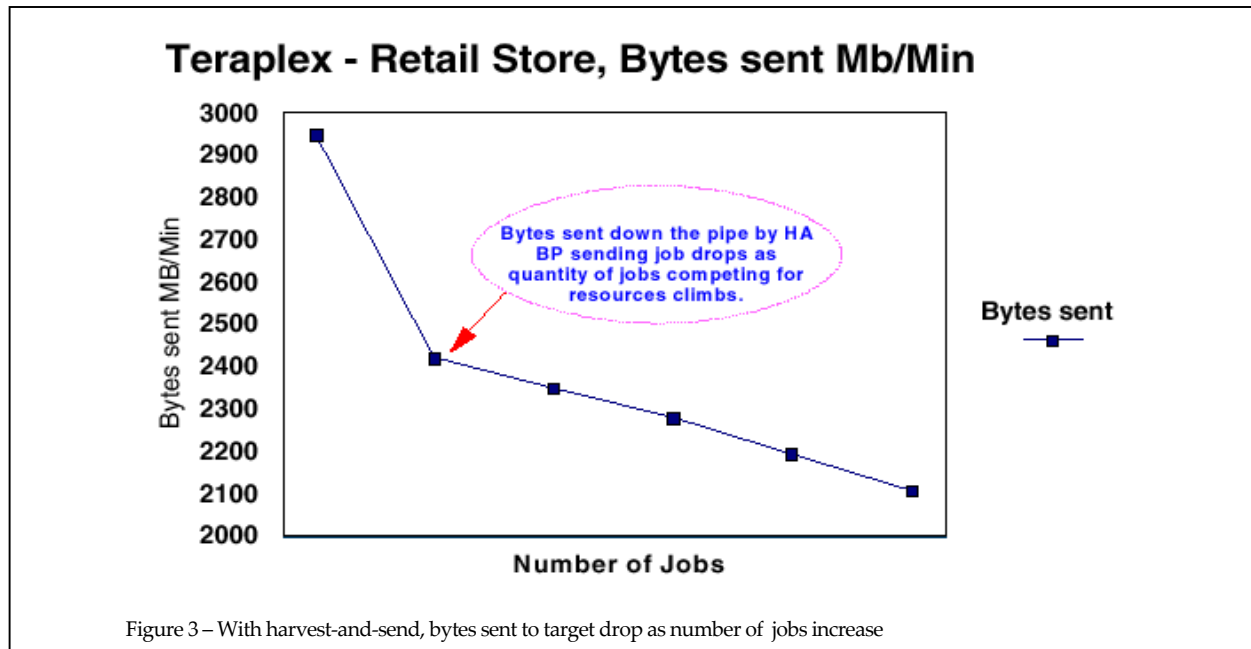
processing is shifted to the target machine.

The following graphic (Figure 3) from Chapter 6.5.6 of the SFOJP Redbook illustrates from another angle how CPU is affected on the source machine when a harvest-and-send transport process is used (referred to as an 'HA BP sending job'). The statistics in this graphic come from a simulated retail store environment that was created at IBM's Teraplex testing center in Rochester, MN. You can see on the graphic that as the number of jobs increase, the number of bytes sent from the source machine to the target continue to drop.

As the number of application jobs ramp up, HA BP [harvest-and-send] jobs can get starved for CPU and can't send as many bytes as are being generated—hence they can fall behind on the source side (which is why use of remote journal is probably a wiser choice to get the journal entries sent promptly).

IBM :Striving for Optimal Journal Performance, Chapter 6.5.6

The consequence of bogging down the harvest-and-send process is the high potential for significant data loss caused by an accumulation of unsent transactions on the source machine during a sudden system failure. However,



when the CPU burden of harvesting and validating transactions is shifted to the target machine through a remote journaling transport process, this vulnerability is substantially reduced since journal entries are moved so quickly to the target.

This point is dramatically illustrated in another graphic (Figure 4) from the SFOJP Redbook, which compares the number of unsent transactions that occur from both a harvest-and-send transport process and a remote journal transport process after 22 million transactions are generated within a ten minute period. The data in this graphic is based on a test in IBM's iSeries Teraplex testing center, using the following criteria:

- 18,167 journal entries per second
- 91 journal entries per bundle (bundle is the unit sent across the communication line)
- 200 journal bundles put on the communication line per second
- 1.65 GB of journal receiver populated during this run

In Figure 4, the harvest-and-send transport process (HA BP) is designated as the 'BP Sending Job'. The remote journaling transport process is shown in two variations: 'Sync RJ' (synchronous remote journaling—wait for reply from target before returning control to program) and 'Async RJ' (asynchronous remote

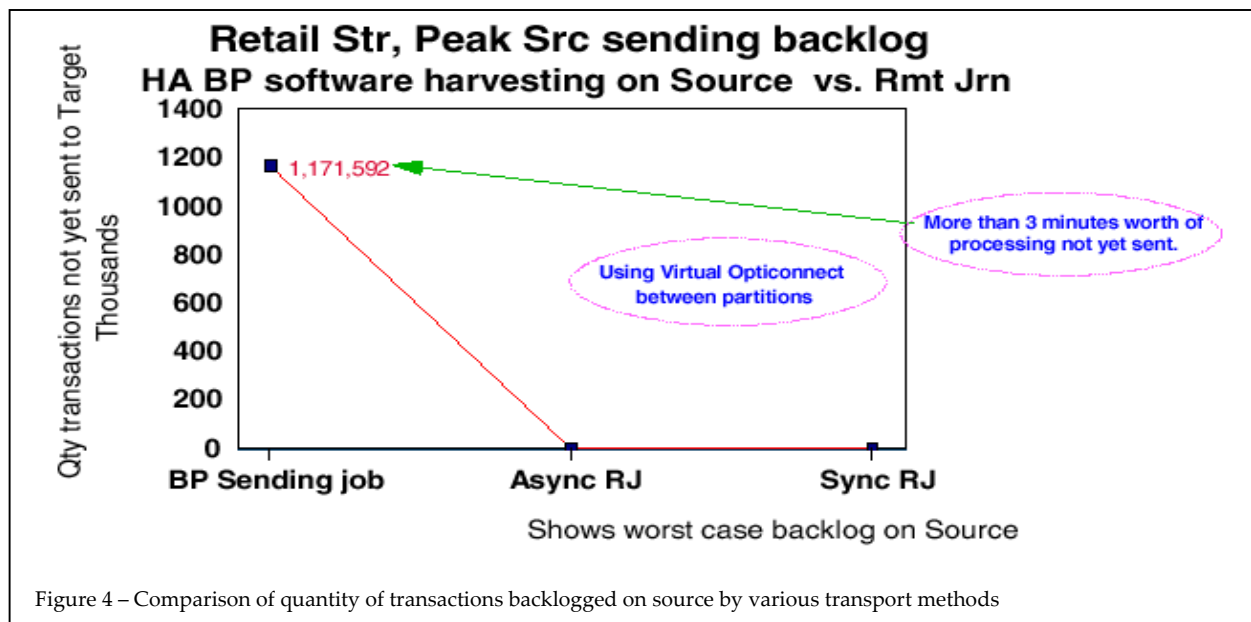


Figure 4 – Comparison of quantity of transactions backlogged on source by various transport methods

- 140 jobs running in a retail store environment without exhausting the CPU
- Using an 11 processor partition on an iSeries 840 (96GB of main memory, 1.9TB of DASD)
- Asynchronous remote journaling over 100 Megabit Ethernet line
- 22,280,812 transactions generated in 10 minutes:

journaling—doesn't wait for reply)⁴.

The results of this test are significant. When using a proprietary harvest-and-send process to transport transactions in this environment, nearly 1.2 million transactions are still unsent on the source machine after ten minutes, creating over three minutes of latency. On the other hand, remote journaling—whether of the synchronous or asynchronous kind—left zero transactions on the source system.

The async remote journal sending task on the source machine with a fast virtual opti-connect communications path between the partitions keeps up and doesn't fall behind even when the transaction rate on the source peaks at 382,998 entries/minute; by contrast the HA BP [harvest-and-send] software falls behind as much as 3 minutes worth after a 10 minutes run...Not a single one of the [remote] journal bundles fell "behind"...These remarkable results illustrate how effective and efficient the remote journal approach can be.

IBM: Striving for Optimal Journal Performance,
Chapter 6.5.7

Further:

Both the async and sync varieties of remote journal support have an inherent advantage over any HA BP-like traditional approach [harvest-and-send]. The remote journal transport technology doesn't have to incur the overhead to convert the journal entries from internal SLIC representation to external mapped representation, this obviously saves CPU cycles and time. In addition, the internal representation takes up fewer bytes per journal entry and hence not as many bytes need to flow down the wire.

IBM: Striving for Optimal Journal Performance,
Chapter 6.5.7

One of the major arguments of HA vendors who include a proprietary harvest-and-send process in their solution is that their process filters the harvested data on the source before it is transmitted, removing any unnecessary data and journal entries, thus helping to make data transmission faster—especially in situations where bandwidth is limited. Because of this, there are scenarios where source side filtering makes good sense.

Although remote journaling transmits most journal entries related to the mirrored files to the target system, it is important to keep in mind that data is put on the communication wire so quickly with this process (again because it happens at the SLIC level) that in a majority of the cases, HA solutions that utilize remote journaling still have a significant performance advantage. Consider this from the [SFOJP Redbook](#):

When...transport is managed by the remote journal facility, it occurs in real time deep in the machine at the SLIC microcode level...as our measurements reveal, there's very little delay in getting such journal entries placed on the wire. Provided that your communication gear is fast enough, such remote-journal-managed...transports are generally on the wire before control is returned to your application.

IBM: Striving for Optimal Journal Performance,
Chapter 6.4.2

Another disadvantage of having a harvest-and-send process filter out journal entries is that it nullifies journaling's inherent ability to self-audit for data integrity. With journaling, each journal entry has a sequential number so the system can keep track of the order of journal entries in a journal receiver. Remote journaling uses these sequence numbers to track that a journal entry created on the source system arrives on the target system. When harvest-and-send filters out the journal entries it doesn't need, the ability to use journal entry sequence numbers to account for sent and received data transmissions is lost. Because of this, HA software that uses harvest-and-send must use CRC checks (checksums) to audit data integrity, which adds additional overhead to both the source and target systems.

Conclusion

IBM's [Striving for Optimal Journal Performance](#) Redbook makes clear the benefits of remote journaling for a majority of shops when used as the data transport mechanism in iSeries high availability solutions:

- *Reduced Overhead* – Remote journaling moves the data harvest and validation processes to the target system saving system overhead on the source system.
- *Shorter Time to Get Data on Wire* – Since the remote journaling data transport process is handled entirely at the level of the operating system, the data transport process is extremely efficient. Also, because the data

harvest and validation process occurs on the target system, this eliminates many operations before putting data on the wire.

- *Less Data Vulnerability* – Because of the reduced overhead on the source, and because of how quickly remote journaling puts data on the communication wire, there is less exposure to data loss in the event of a system failure on the source.

IBM calls remote journaling a ‘superior performance choice’ as a data transport engine between systems in iSeries high availability solutions:

We were pleasantly surprised to discover during our testing...[that] remote journal support over a sufficiently fast communication line never fell behind nor left recent journal entries trapped on the source system no matter how hard we pushed it. In fact in every instance we never saw more than 5 ms transpire between the time we produced the journal entry on the source system and the time it was sent down the communication wire...[remote journaling] appears to be a superior performance choice over any approach which harvests journal entries on the source system.

IBM: Striving for Optimal Journal Performance,
Chapter 6.2.1

Notes:

1 – The entire Redbook can be found at www.ibm.com/redbooks.

Search under book ID#: SG24-6286-00

2 – Keep in mind that all of the statistics shown in this white paper which are cited to IBM come from tests they conducted at under controlled conditions at their Teraplex testing center in Rochester, MN. Your own results may vary.

3 - IBM :Striving for Optimal Journal Performance, Chapter 6.5.6

4 – Remote journaling can be configured to work synchronously or asynchronously. If it is synchronous, then control is not returned to the job that created/changed the transaction until a message is received from the target that the journal entry related to the transaction has been received. Typically, synchronous remote journaling is only used when critical financial processes are being journaled. Synchronous remote journaling can dramatically affect system performance, depending on the system resources and bandwidth. Asynchronous remote journaling, on the other hand, does not wait for confirmation from the target that the journal entry arrived before returning control to the job.

iTera, Inc.
801-799-0300
info@iterainc.com
www.iterainc.com

© Copyright 2003, iTera, Inc. No portion of this document may be reproduced without permission from iTera, Inc. IBM, eServer, and iSeries are trademarks of International Business Machines Corporation. All other trademarks are property of their respective companies.