IBM Smart SOA Application Foundation: The Right Choice for Proven, Reliable Business Transactions

Transactional Integrity - An Important Consideration When Evaluating JBoss

Introduction

It is a different world today – one that is increasingly interconnected, instrumented and intelligent.[1] Transactions in every facet of life and business are growing exponentially, yet organizations in all industries need to be able to execute them reliably.

Reliable transactions are critical to business

There are 15 million bank transactions mined each day. Poor quality data will cost the banking industry 27 billion USD in operating costs.[2]

In the insurance industry, inaccurate data can make it difficult to properly underwrite and manage risk.[4]

In retail, an error between billing and shipping can have a ripple effect throughout the supply chain.

It's clear that businesses and consumers cannot afford compromised data for two main reasons: trust is paramount and any transactional error is a direct strike against it, and accurate, reliable data is vital to decision-making. Don't you want an application foundation that enables you to deliver responsive and reliable application transactions that your customers can trust?

The impact of a failed transaction on a consumer is clear. The consumer does not receive the funds, product, service or information they have requested. To the business, a failed transaction means loss of revenue and information.

Reliable transactions are critical to business. There are 15 million bank transactions mined each day. Poor quality data will cost the banking industry 27 billion USD in operating costs. [2] Half of customers will give their banks only two times to make a mistake before considering a change to another bank.[3]

Worldwide data shows insurance customers value honesty and trustworthiness even more highly than price, yet transparency has been lacking for years. And without accurate data, insurers cannot properly underwrite and manage risk.[4]

Approximately 91% of consumers are spending less in some discretionary areas, and when they do buy, they want a retailer they can trust.[5] An error between billing and shipping can have a ripple effect from consumers through the supply chain to stores, partners, suppliers and manufacturers.

It is imperative that companies and organizations execute reliable transactions. IBM WebSphere® Application Server makes an ideal platform for most business applications, including key company-wide and mission critical applications, because it is designed to ensure transactional integrity, helping to enable you to deliver business applications that your customers can trust.

By contrast, the open source-based JBoss Application Server seems to have demonstrated a lack of transactional integrity, which may significantly affect your ability to execute reliable business transactions and increase the risk of data loss. This could severely impact businesses in any industry. In this paper, we will cover examples in the banking, insurance and retail industries.

The ability to maintain data in a reliable and consistent state, regardless of any system or business failure that may occur, is ensuring transaction integrity. The application server is a critical part in handling and recovering from transaction failures.

What are Transactions?

Think of a transaction as having an all or nothing behavior. That is, a transaction is a series of operations that must successfully complete as a single unit, or else nothing should be completed. The operations of a transaction are used to move data between consistent states. If any part of the transaction cannot be completed, it must be rolled back to its original state, or be recoverable, so that if only part of the transaction is committed, the remaining operations will also commit so that the transaction is fully complete.

The transaction must retain a consistent state of the data it affects even in the event of application server crashes or failures.

A distributed transaction runs in multiple processes, potentially on many different machines where multiple databases are involved. A distributed transaction therefore has a higher potential for problems because of its complex environment.

Different types of transactions are used in various industries, sometimes within the same industry. Desired outcomes are still the same when a transaction is processed—either **all** data changes are completed successfully **or none** is.

Where Can Transaction Integrity Fail?

The ability to maintain data in a reliable and consistent state, regardless of any system or business failure that may occur, is ensuring transaction integrity. It is a key aspect and requirement for business readiness.

To handle transactions within a company's infrastructure, every component of an application needs to be able to recover from any IT or business failure. If not, you don't have transaction integrity. Failures can take place in the business process, the messaging layer or the application server as seen in Figure 1. The application server is a critical part in handling and recovering from transaction failures. This paper will focus on what can happen when it fails.

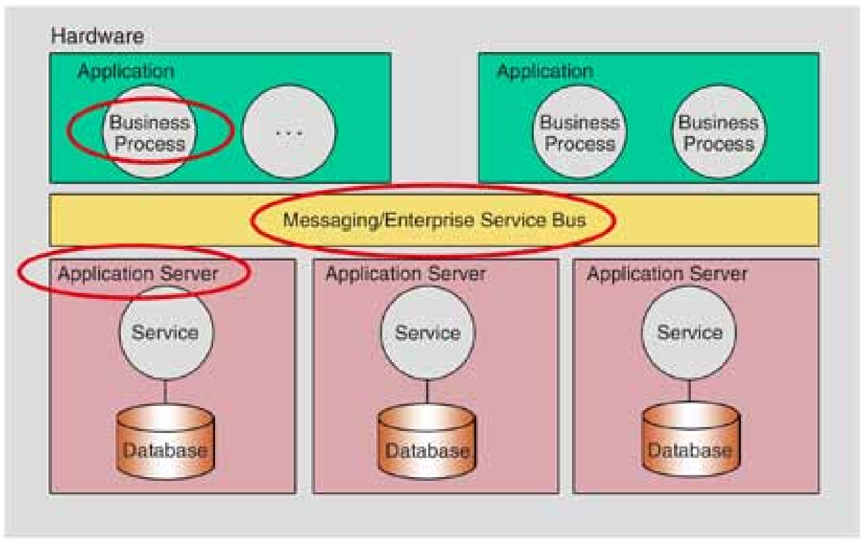

Figure 1. Areas where transaction failures can occur: business process, the messaging layer and the application server itself

What is an In-doubt Data Transaction?
Databases are typically used as transaction resources. There can be multiple databases in the typical distributed environment with transactions spanning across them.

To handle distributed transactions, it is important to use what is known as two-phase commit. With this algorithm, the application first (i.e. first phase) confirms that each database resource is able to complete its action, but does not ask it to do so. Next (i.e., second phase) each resource is committed, and locks are released. In this way, if any resource cannot complete the update, the other resources will not have committed their changes, leaving the databases as they were before the transaction began.

An in-doubt transaction can occur if a resource cannot commit the data. The data is in limbo until the application server or manual intervention handles it.

During the commit request phase, a message is sent to all resources involved in the transaction. Database records are locked, and the transaction manager waits for replies. If all are a “go,” then the commit phase follows. If even one says no, then a rollback occurs.

The two-phase commit algorithm is meant to update and enable commits in a short period. But even in short periods, things can fail, leaving the transaction in a state that is known as “in-doubt”. An in-doubt transaction can occur if a resource indicates in the first phase that it is available, but then fails in the second phase.

A failure, such as a network failure, caused after a resource is committed, but before another can commit, leads to an unresolved situation because the second database still has locked information. The data is in limbo and either waits for instructions from the application server’s transaction manager or manual intervention. That transaction is now in-doubt. Other transactions can follow that use different records in the database, and can also become in-doubt while the second database is off-line.

IBM WebSphere Application Server is used by 90% of the world’s 100 largest companies.[6]

Proven Transactional Integrity with WebSphere Application Server

IBM WebSphere Application Server is used by 90% of the 100 largest companies in the world[6], and is a leader in providing proven, reliable transaction integrity. WebSphere Application Server is rock solid in its recovery when issues with transactions occur.

WebSphere Application Server version 7 provides highly effective performance in terms of speed, scalability and reliability for transactions. Its transaction manager was built on top of a solid foundation from the time of its introduction over ten years ago. It is designed to automatically detect and recover from transaction problems that occur during two-phase commits. WebSphere Application Server V7 includes powerful tools, such as transaction log viewers and easy steps for specifying transaction recovery settings. These capabilities help ensure transaction integrity for WebSphere Application Server users.

This raises the question as to whether JBoss can automatically recover from all transaction integrity failures.

Can JBoss Ensure Transactional Integrity?

As the JBoss Application Server (JBoss AS) tries to mature as an application server, it seems to have gone through big changes from version to version. One area which appears to have changed dramatically is the JBoss transaction manager. JBoss AS has a relatively new integration with a third-party transaction manager. Public forums seem to indicate that this transaction manager is not proven, and the JBoss community has identified a defect in it. The documented bug is the JIRA JBoss document defect (JBAS-6244).[7] This defect describes a JBoss AS transactional integrity recovery issue, but does not specifically address how to fix the problem. This raises the question as to whether JBoss can automatically recover from all transaction integrity failures.

Because it is an open source community, the vast majority of JBoss AS documentation, discussions and defects can be found on the Web. For example, there are reference documents that talk about recovery, such as the JBoss Transaction Recovery Overview.[8] Another reference document, which can be found on the JBoss community's Wiki page,[9] describes XA datasource and other recovery methods using JBoss Application Server.

Moreover, there appears to be no documentation on how to configure or fix JBoss AS to automatically recover from the transactional integrity issues that we discussed earlier in this paper when in-doubt distributed transactions fail. Instead, there seem to be forum postings showing the specific transaction recovery problem with the JBoss Application Server,[10] or discussions about the transaction recovery problem that don't address how to actually fix it. In our opinion, the documents are few and often unclear, especially in how to handle Oracle database resources. It appears that most of the documentation is about messaging configurations, rather than transactions.

IBM has done a study based on this public information to attempt to determine whether the transactional integrity problem referred to in JBoss defect JBAS-6244 can be recreated or if it has been fixed or resolved. When considering JBoss AS, keep in mind:

A failure in the system may cost a company millions of dollars in lost revenue and lost customer satisfaction.

- JBoss AS appears to be unable to automatically recover from all transaction integrity failures.
- There appears to be no documentation on how to configure or fix the documented JBoss AS transaction integrity defect.
- Based on IBM observations it is our belief that the problem still exists in JBoss AS version 5.1 GA.

Potential Business Ramifications of Failed Transactions

Businesses execute billions of transactions each day. A failure in the system may cost a company millions of dollars in lost revenue and lost customer satisfaction. As mentioned earlier, JBoss[10] forums and defect[7] indicates that JBoss automatic recovery options may fail.

Several examples prove how automatic recovery failures impact different industries.

Banking Industry

Assume that Account "A" is in one database, and that Account "B" is in a separate database. The consumer goes online to transfer $1000 from Account A to B. Each of the associated database records are locked, and they are ready to be committed. The first database records get committed (made permanent), completing the withdrawal. Before the second can commit, the second database suddenly goes off-line. It could be down because of a power failure, network connection interruption. The records are in-doubt.

Data is In-doubt. *With a transaction failure, money may be withdrawn, but not credited. The reverse can also happen. Money may be added to an account, and not yet withdrawn if the data becomes in-doubt. Regardless, either the consumer or the bank is at a loss. Would your customers tolerate inaccuracies in their accounts?*

This classic example of a transaction involves withdrawing money from one bank account and depositing it in another. If only the withdrawal completes, money is lost. The consumer is out $1000 in this case.

Debits and credits in double entry accounting must have both the debit and credit completed; otherwise, it should appear that neither operation was attempted. When power or the network is restored, the application server needs to know that there was a problem (via its logs) and recover once the connection to the database is available.

Until a recovery is made, records are locked in the second database. Applications cannot utilize the information from those records. If the consumer asks for an account balance, it will not include information based on those locked records. If they check Account "A", they'll see $1000 less in it. The accounts are corrupted and inaccurate at this point.

The application server should try to recover and should know of the problem via its logs. The logs need to be complete, clear and accurate so that manual intervention can also resolve the problem if the application server cannot.

No automatic recovery. *If the application server can't handle it, then manual intervention is required. If the logs are incomplete, recovery may not be possible. If tracing can be done, it involves intelligent coordination between the application server and data base administrators. Can your business afford the time and resources needed for manual intervention?*

If the application server cannot automatically fix the issue, it may loop and try to recover again. Each time the server loops, the transaction log grows. The application server administrator must look through the logs to trace and understand the problem, and to report the log information to others. More experts may need to be called in, assuming there might be a problem with the application software itself. Once it is determined that the database was interrupted, a database administrator who has authority to modify the databases would be called in. These administrators have to jointly fix the problem—one clears the logs, and the other corrects the databases. Depending on how the issue is handled, that could mean the transaction completed or that it was rolled back. Regardless, the fix is a manual, coordinated effort between highly skilled resources.

Insurance Industry

Policy management requires that every event impacting an insured customer be registered; the most common methods are through an agent or online. This information must then be accurately transmitted to the home office, where it is recorded, analyzed and stored. Policy adjustments are then made accordingly. These adjustments must be accurately communicated to all parties and reflected in online account information and agent systems. Finally, any adjusted rates must be billed without error. The failure of any one of these transactions results in erroneous data that may affect underwriting, revenue and customer confidence.

Audit and compliance issues. *Unresolved in-doubts can lead to a false sense of security. Eventually, it can lead to regulatory investigation of business processes and policy, along with wasted resources and time.*

With a transaction failure, the logs may not have been clear about the problem. By the time the administrator sees the log, it could have grown very large. The administrator may just delete the log so that the application server stops trying to recover in a loop. By then, traceability is lost and records are still in-doubt.

If accounts are not accurate, the insurance company may have accounting problems. They may no longer be in compliance with government regulations in terms of providing accurate customer account information to either the customer or to the government.

Multiple in-doubts are even more difficult to correct. *Tracing many in-doubts is much more difficult and may be impossible. Imagine an insurer realizing they owe greater coverage on claims than they had assessed, due to inaccurate data.*

If the problems remain unresolved, the company may have a false sense of security. It could allow the corruption of data to continue over time. The loss of data integrity may have profound consequences, including compromised risk management, loss of customers and ultimately, an impact on solvency, especially when the problem is discovered at a much later time.

Many in-doubts can occur. Other transactions may lock a different set of records. Unfortunately, many administrators let transactions occur during the day and administer at off-times, so the amount of in-doubts can be large, and tracing these could become much more difficult. If logs only state that there is a transaction problem with no account number or database reference, the administrator may not even be able to recover all of the transactions.

If these transactional errors go unresolved, the company may face many significant problems. With customers, the company may face the loss of trust, potential lawsuits, a diminished reputation and a compromised ability to attract new customers. On the operational front, it may face ineffective underwriting and risk management, capital loss and potential insolvency.

Retail Industry

Imagine a retailer who ships an online order to a customer but doesn't correctly record the transaction in its billing system. For example, the customer may never get billed, or customer may get billed, but the retailer does not ship the item. As a result, either the customer or the retailer is unhappy. Any discrepancy between the billing and shipping databases is inexcusable to today's customers, and can wreak havoc on a retailer's inventory management and revenue.

Locked records can mean lost orders. *Held records impact any application trying to access the database, potentially leading to lost revenue from an inability to accept new orders.*

If the in-doubt order doesn't get fixed, the records remain locked. As a result, the records in the database associated with that order cannot be modified until the locks are removed. That means that the company may lose money since it cannot take new orders!

Conclusion

Organizations and customers need reliable business transactions. The application server, as a main component in handling transaction failures and recovery, is critical in upholding transactional integrity.

IBM WebSphere Application Server is designed to ensure transactional integrity, helping to enable you to deliver business applications that your customers can trust. WebSphere Application Server is designed to ensure data consistency through automatic recovery when transaction failures occur, even under heavy loads, demonstrating high resiliency. IBM is “enterprise ready”.

By contrast, IBM believes the open source-based JBoss Application Server cannot automatically recover from distributed transaction failures. For customers looking at JBoss as an alternative to IBM, the potential loss of business from distributed transaction failures, in IBM’s opinion, makes JBoss a higher risk and cost solution. Would you base your business on questionable transaction processing?

For more information

For more information about IBM WebSphere solutions, please contact your IBM marketing representative, Business Partner, or visit:
ibm.com/software/webservers/appserv/wasproductline/

---

1 Smarter Planet Overview. http://www.**ibm.com/**ibm/ideasfromibm/us/smartplanet/20081106/index2.shtml?&re=sph
Three things have brought this about:

- By 2010, there will be a billion transistors per human, each one costing one ten-millionth of a cent.
- With a trillion networked things—cars, roadways, pipelines, appliances, pharmaceuticals and even livestock—the amount of information created by those interactions grows exponentially.
- Algorithms and powerful systems can analyze and turn those mountains of data into actual decisions and actions that make the world work better.

2 IBM “A Smarter Planet: Banking”
http://www.**ibm.com**/ibm/ideasfromibm/us/smartplanet/topics/finance/20090126/index.shtml?&re=spf

3 European Retail Finance article, “Delivering a better branch experience” http://www.erfonline.com/vendorsolutions/erf002_014_brickstream.htm AND http://www.**ibm.com**/ibm/ideasfromibm/us/smartplanet/topics/finance/20090126/index.shtml?&re=spf

4 2009 IBM Institute for Business Value Study: Growing Trust, Transparency and Technology:
http://www-935.**ibm.com**/services/us/index.wss/ibvstudy/gbs/a1030834?cntxt=a1000058

5 2008 IBM Study, “Shopper advocacy: Building consumer trust in the new economy“; AMR Research:
http://www.**ibm.com**/ibm/ideasfromibm/us/smartplanet/topics/retail/20090105/index.shtml?&re=spf

6 As measured through a cross reference of WebSphere Application Server customers and the Fortune Global 100.

7 JIRA JBoss document defect (JBAS-6244) at https://jira.jboss.org/jira/browse/JBAS-6244

8 JBoss Transaction Recovery Overview at http://www.jboss.org/community/wiki/TransactionRecovery

9 JBoss community’s Wiki page referencing XA and other recovery methods at
http://www.jboss.org/community/wiki/JBossTSRecoveryInAS

10 Problem cConfiguring Oracle for XA recovery at
http://www.jboss.org/index.html?module=bb&op=viewtopic&t=126834

WSW14068-USEN-01