Ensemble Behavior after a System Failure and Restart

By David Loveluck

This paper describes how Ensemble behaves after a system crash or similar uncontrolled shutdown or failover. Ensemble is normally configured to start processing messages automatically when the operating system restarts or when the system has failed over to a cluster member or mirror member. No manual intervention is required. But, in order for your system to robustly handle system failure and restart, you must understand how Ensemble handles these conditions and develop your productions using the rules and guidelines in this document.

The behavior in the event of a restart is very dependent on the solution and the participating applications. Ultimately, it is the responsibility of the solution designer to address each possible failure mode.

Guarantee of Delivery at Least Once

With wire protocols such as SOAP over HTTP or HL7 over TCP/IP, it is impossible to guarantee that messages are sent once and only once. For example, consider a client that makes a web service request and then the client system crashes. If the client system crashes before it receives the reply to its request, it cannot distinguish between the case where the server did not receive the message and the case where the server received the message but the client crashed before receiving the reply. Consequently, the client must repeat the request to ensure it has been delivered.

Ensemble guarantees that if it has received and acknowledged a request, it will deliver it. If Ensemble has delivered a request but has not received the response before a system crash, it will resend the message on the restart to guarantee that the message is delivered at least once. In some cases, Ensemble will be sending a duplicate message to the target. For example, if a business operation was invoking an external web service when the system crashed, Ensemble has no way to determine after a restart whether the remote service completed the request, so the business operation will resend it. Any application that is receiving a message from an Ensemble operation must be able to gracefully handle duplicate identical messages.

Ensemble is able to guarantee sending a message at least once by using information in the journal. In order to ensure that journal entries are durable, you must select the 'sync-commit' option. This setting requires some additional resources but is the recommended setting for mission-critical systems. You should control the sync-commit option by setting a global in the namespace at the critical code sections that need it.

If a business service has received a message, but has not yet acknowledged it when the system crashes, it may send the request but cannot guarantee that the message was persisted and processed. Consequently, if an application has sent a message to a business service but does not receive an acknowledgement, it should resend the request to the business service.

Careful Use of Transactions

Transactions are an important tool to maintain consistency over a system crash and restart, but you must follow the recommendations in this section.

When a request or response is sent, developers of Ensemble applications do not need to do anything to ensure queues, message headers and message bodies are saved in a consistent manner following a restart. This is handled by the use of transactions in the Ensemble framework where appropriate. Consequently, you do not need to code any transactions to protect message integrity.

If your application code in a business service, process, or operation updates the Caché database, your code must ensure that the database is left in a satisfactory state if the system crashes. You can use transactions to ensure that database will not be left in an inconsistent state. If the system crashes while a transaction is open, any updates to the database made while the transaction is open will be rolled back upon a system restart. Once your code has completed updating the updates and the database is in a consistent state, it can close the transaction.

Although transactions are important in maintaining database consistency, you must use care in coding them. Poorly coded transactions can lead to deadlocks, slow recovery time after failures, and, in some cases, loss of data.

When coding transactions, you should follow these guidelines:

- Code transactions to have a short duration. By ensuring that transactions are short, you minimize the possibility of deadlocks. If your application requires a long duration to complete a full transaction, consider breaking it into multiple intermediate transactions. When your code starts, it should check to see if the database is in one of these intermediate steps. If it is, your code should undo the changes and restart from a clean state.

- Ensure that if the system does not crash, the transaction is always closed. To do this, you must safely handle any error that occurs during the transaction and there should be no code paths that do not close it.

- Do not send a request to another configuration item or wait on an asynchronous request when a transaction is open.

How Ensemble Uses the Active Message after a Restart

Ensemble is responsible for ensuring that messages are successfully transmitted to and from business services, processes, and operations. When a business process or operation takes a request off a queue and begins to process it, Ensemble saves the message header and body as the 'active message'. When the process or operation successfully completes, Ensemble clears the active message. If the system crashes before the business service, process, or operation completes, the message is preserved as the active message.

When Ensemble restarts, it detects the active message and resends it to the business service, process, or operation. Your application code does not have to have any special code to handle the message requeueing, but your application is responsible for handling any explicit database updates that it made.

For a business process, a response to an asynchronous call is treated as a message. This means that after a restart, the business process can resume from the last point it was saved to disk and it will get the response as Ensemble replays it from the active message.

Recovery after Restart

When Ensemble restarts after a system crash, or similar event, Ensemble first restores the database to a consistent state by applying database updates from the journals and then rolling back any incomplete transactions.

Database updates made in the application code that implement the business process may or may not appear in the database after restart, depending on the timing of the failure and the use of transactions in the application code. Since all transactions must be closed before completion of the business process

and control is passed back to the Ensemble framework, there is always a chance of a system failure occurring after the application transaction is completed and before the response from the business process or operation is secured to disk.

Once the database has been restored to the proper state, Ensemble checks to see if messages were in progress at the time of the crash by looking at the active message global. If Ensemble detects that an asynchronous request has been taken off the queue for processing but processing has not been completed, the request is returned to the head of the queue and the message will be processed again as soon as the production starts.

Synchronous requests are not replayed. The job making a synchronous call, whether it is a business service or process, no longer exists after a restart and consequently there is no active element to receive the request. If all requests in a session since the primary request to a business service have been synchronous, the requests are lost; but in this case the business service will not have sent any acknowledgement back to the caller so nothing has been guaranteed.

Note that BPL 'synchronous' responses are really asynchronous and are treated as such in the context of this paper. The request is made asynchronously, but the code waits for the response before continuing.

All Code Must Be Repeatable

Because messages can be replayed, when developing an Ensemble production, you must ensure that code can be run repeatedly and provide a consistent result. Certain actions, such as inserting records into tables with unique constraints can cause failures if repeated and the business logic must account for these possibilities.

It is possible that a message is sent to an external system which replies, but the acknowledgement was not processed by Ensemble before a crash. After restart this message will be sent again, and therefore all external applications must also be able to tolerate repeated identical requests.

Care must be taken with any code that isn't exactly repeatable. For example if data is saved using a timestamp as part of the key, there may be no way to detect that a single message processed at two different times should only update one set of central data. If it is just a log file recording activity that isn't a problem, but if it has meaning to the business logic there may be a problem.

Recovery with File Adapters

Adapters that write or read files have special considerations because they are not protected by the transactional nature of the database. There are many different situations that need to be considered.

If a business service using an inbound file adapter uses a workpath directory, the file is first moved from the Filepath directory. If the system restarts during processing of the file, it will be left in the workpath directory and after restart the inbound adapter will not find it in the inbound directory. To ensure that the file is correctly processed, either an external process or a manual intervention must move the file back to the filepath directory.

For a business operation, a system restart may leave partial data in the outbound file. The business operation will reprocess its active message and the data will be written out to the file again. If the business operation is configured to append to the outbound file, the partial data will remain and then be repeated. The application receiving this file must be able to cope with the incomplete data.

If the business operation is configured with the 'overwrite' option the partial data will be replaced by a

complete set of data, but there are still cases that may cause problems. If the downstream application picks up the partial file before it can be replaced, it will have to handle the partial data followed by the complete data in a second file. Conceivably, the downstream application may lock the file and cause the business operation to fail. Additionally, the name of the file created when the active message is processed again, may not be the same as the first and the partial data will then be left in the output directory.

Note that Caché mirroring only mirrors the Caché database. If your application depends on external files, you must provide some other mechanism to ensure that the files are transferred from the failed system to the mirror.