A Broadcom Company
technologies

White Paper

CA Smart/RESTART

Beyond Restart and Concurrency: z/OS System Extensions for Restartable Batch Applications

Introduction

This document examines the challenges facing organizations with mission critical z/OS batch workloads. It presents restart capability for z/OS batch applications as a strategic investment in infrastructure and describes the benefits to be derived from restart enablement. It considers the requirements and characteristics of a generic restart solution along several dimensions to facilitate a rigorous evaluation and selection process.

As the enterprise extends its hours of operation steadily towards a *24x7* environment, there is less and less time to tolerate failures in z/OS batch production. A robust and reliable restart facility minimizes the impact of z/OS batch application failures and prevents them from escalating into business problems. Investing z/OS batch applications with restart capability is a proven means to leverage existing IT assets and optimize the applications already in place. A system-wide restart facility provides a resilient *safety net* for z/OS batch applications and an essential foundation for z/OS batch application processing.

This document discusses the following topics:

- A description of a system-wide restart facility
- How to implement restartable z/OS batch applications
- The benefits of a restartable operation
- How to evaluate a restart solution

System-Wide Restart Facility Description

A system-wide restart facility that is suitable for the organization's entire portfolio of z/OS batch applications is implemented as an extension to the z/OS operating system. It can track and manage the state of the z/OS batch application throughout its processing and provides an architected solution that is resilient, scalable, and manageable. Such a facility creates a fault tolerant execution environment where additional capabilities can be layered. These additional capabilities are described in Benefits of Restartable Operation.

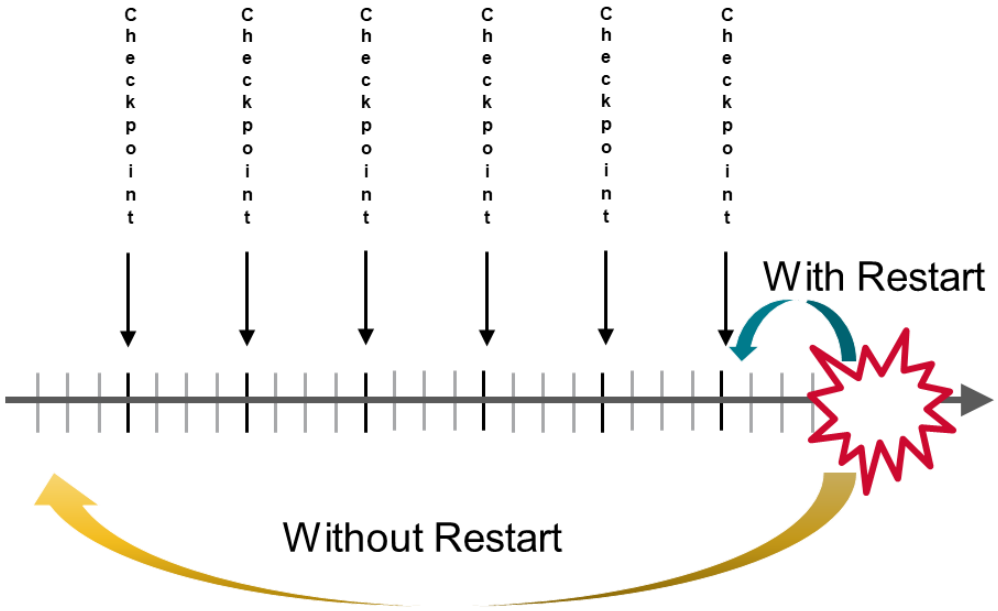
Checkpoint and restart is an enabling technology that allows an application to restart from a checkpoint near the point of failure with all its resources in a consistent state. The restart solution must complete the following actions for a checkpoint restart:

- Reposition sequentially accessed input and output files
- Reposition Db2 cursors
- Log VSAM I/O so uncommitted changes can be backed out if necessary
- Save selected areas of application storage at checkpoint time and restore them at restart time
- Inform the application as to whether an initial run or restart run is in progress

The commit and rollback services provided by products like Db2 and IMS are only partial solutions. Successful restart requires a more comprehensive unit of work whose scope encompasses all the resources the application may access. The scope includes not only changes to Db2, IMS, VSAM, WebSphere MQ and other RRS compliant resources, but also changes to application storage, VSAM updates and sequential file and Db2 cursor position.

The system-wide restart facility functions as the resource manager for these orphan resources and keeps them in sync with resources managed by formal MVS subsystems like Db2 and IMS. When a restartable-unit of work is identified as a checkpoint, it establishes a point of consistency from which processing can successfully resume upon restart. Figure 1 shows an example of this checkpoint process with restart. Upon restart, this application can resume its execution from a checkpoint near the point of failure with all its resources in a consistent state. This restart eliminates redundant processing and ensures processing always proceeds in a forward direction. There is much less wasted processing because only work performed since the last successful checkpoint is backed out and discarded, not the entire run.

Figure 1: Restart Capability Enables a z/OS Batch Application to Restart from the Last Checkpoint



A system-wide restart facility can reposition sequential files and sequentially access VSAM datasets. The restart facility uses optimized, parallel I/O techniques during restart that outperform the QSAM access method used by COBOL by roughly 10%. As a result, z/OS batch applications endowed with restart capability exhibit run times close to native processing times. The z/OS batch applications that fail and require restart will finish processing before their non-restartable counterparts. Performance is improved because the z/OS batch applications resume execution from near the point of failure, rather than from the beginning.

Implementing Restartable z/OS Batch Applications

Enabling an application for restartable operation can considerably complicate development. By some estimates, restart logic can double application size and significantly impair your ability to comprehend and maintain it. Also, in the absence of a generic restart solution, each application must make its own provisions for restart.

Although restart poses numerous technical problems, a generic restart solution enables developers to delegate these responsibilities to reliable system software. Since the system-wide restart facility handles the mechanics of checkpoint and restart, developers can concentrate on defining the logical unit of work that successful restart requires.

To drive more comprehensive checkpointing services, a system-wide restart facility makes it easier to endow both new and existing z/OS batch applications with restart capability by using the commit logic within the application (for example, standard checkpointing requests like SQL COMMIT and IMS CHKP). These services extend the scope of a unit of work to encompass all resources the application accesses and ensure they remain consistent and *in sync*.

It is the developers responsibility to identify the unit of work for an application and request a checkpoint when it is complete. This is not a task that can be delegated to the restart facility since it knows neither the application requirements, or when application data is in a consistent state. What the restart solution can do is establish a point of consistency from which processing can resume if restart is required. The developer must also code a flow of control that resumes execution in the event of restart at a unit of work boundary.

The unit of work can be comprised of one or more transactions since each transaction constitutes a point of consistency. Batch applications typically bunch multiple transactions within a single unit of work and process them within a loop. For the sake of clarity, the unit of work should be defined within the high level logic in the main source module of an application. Main modules should always be *restart enabled.* Subroutines do not need to be enabled, unless they contain storage that must be saved and restored (a practice that is not encouraged).

A system-wide restart facility can process I/O to sequential files and VSAM datasets using standard compiler I/O statements. This practice allows COBOL programmers, for example, to code standard SELECT and FD statements and use familiar OPEN, CLOSE, READ, WRITE and START verbs to access both sequential files and VSAM datasets. In the event restart is required, the restart facility will automatically reposition sequentially accessed files when driven by a native compiler statement like OPEN.

Developers can optionally identify specific storage areas to be saved at checkpoint time and restored at restart time. Application storage can often be saved in its entirety—which requires no specification. The exception is if the storage is extremely large and non-volatile.

NOTE: Some restart facilities permit the unit of work to be defined, and a checkpoint request to be triggered external to the application source code (such as instream with the JCL). This practice can be useful if source code is unavailable or cannot be modified.

Benefits of Restartable Operation

A restart facility implemented as an extension to the operating system can improve the performance, availability, and operational efficiency of the entire z/OS batch workload. The restart facility can also provide the following benefits:

- Enhance data integrity, availability, and quality
- Increase business flexibility and improve service quality
- Complement existing business resiliency and DR processes
- Promote compliance with requirements for auditing and disaster recovery as mandated by Sarbanes Oxley legislation

The following sections examine these benefits in further detail.

Recovery and Restart Benefits

The failure of a critical z/OS batch application can halt activity, delay decisions and disrupt downstream users and processes that depend on the application's timely completion. Although failures and abends will inevitably occur, restart capability mitigates their impact with facilities that provide the following benefits:

- Provide standardized and simplified procedures for restart
- Eliminate the time consuming back out of changes (which can take twice as long as the original processing)
- Greatly reduce the need to back out and redo successfully processed work
- Completely eliminate the need to back out and reprocess committed work
- Provide automated error handling and recovery
- Reduce the need to back up shared files, Db2 tables, IMS databases, and WebSphere message queues (to the extent that precautionary copies are made to allow job restart)
- Permit the same JCL to be used in both initial and restart runs (which eliminates the errors and delays associated with modifying JCL for restart)

Concurrency Benefits

In the past, recovery from z/OS batch application failure meant restoring data to its original state from backup copies, correcting the error, and then rerunning the application from the beginning. This is no longer feasible in a shared data environment because restoring data to a point in time prior to running the failed batch application will overlay changes made by online transactions and other batch applications that were active while the failed application was running. Since these concurrently active processes access the same shared data as the failing z/OS batch application, their committed changes are wiped out by a point-in-time restore. In contrast, restart enabled applications are well behaved in a shared data environment. They promote concurrent access and allow batch and online processing to coexist. They also provide the following benefits:

- Release locks that impact concurrency and degrade response times
- Reduce batch window requirements or eliminate them entirely
- Enable z/OS batch applications to schedule and execute at any time
- Reduce deadlocks, timeouts and resource unavailable conditions as a cause of batch application failure

NOTE: Committed changes become available immediately to other processes. In contrast, the uncommitted changes for a failing application never become visible.

Temporary failures due to resource contention are conducive to automatic restart and unit of work retry. A system-wide restart facility permits SQLCODE values such as -911, as well as selected IMS status codes and user abends to be predefined as eligible for *retry*. When the restart facility detects a retry condition, it can back out data and processing to the previous point of consistency and wait a site-defined interval of time. It can then restart the application automatically—without manual intervention or re-submission of JCL. The application can retry the failing unit of work in its entirety, and can further reduce timeouts, deadlocks and resource unavailable conditions as a cause of failure.

Checkpoint pacing facilities allow the commit frequency to be adjusted dynamically during execution—without changing the application. Both the checkpoint frequency and interval may be modified based on factors such as elapsed time, number of records processed, time of day, heuristic code, and operator request. Different checkpoint frequencies can also be pre-defined for separate shifts and take effect automatically.

The number of logical checkpoint requests received from the application before triggering a physical checkpoint can be varied as well. The physical checkpoint frequency represents a trade-off between speed and concurrency. The process is accelerated when the frequency of physical checkpoints is reduced, and concurrency is achieved when the frequency of physical checkpoints is increased. This balance allows applications to request a commit after every unit of work—without concern for performance.

The ability to adjust the physical checkpoint frequency can be useful in a variety of situations. For example, does a batch application running beyond its batch window hold so many locks that it causes contention with online transactions? Rather than cancel it, the application can be directed to take more frequent checkpoints.

Workload Scheduling and Switching Benefits

A system-wide restart facility provides additional benefits within the context of workload scheduling and process migration. Such a facility enhances z/OS job scheduling products that provide step restart capability by enabling a restartable z/OS batch application to resume execution from a checkpoint within a job step—something a job scheduler alone cannot do.

A system-wide restart facility can gracefully halt an application at a point-of-consistency in its processing. For example, an application running beyond its batch window can be quiesced and resumed at a more convenient time. Since execution halts at a unit of work boundary, there is no wasted processing or time consuming removal of uncommitted changes.

An entire workload can be quiesced should a subsystem or LPAR be subject to a planned outage (such as a shutdown to conduct routine maintenance, upgrade hardware, and so on). In such a scenario, the restart facility can gracefully halt each restartable application within the workload at a unit of work boundary. Workload execution can then resume on another MVS image within the Sysplex, and access another member of the Db2 data sharing group.

Development Benefits

A system-wide restart facility shields developers from the complexities of restart and provides an easy to use methodology that simplifies application development. The restart facility provides the following benefits:

- Eliminates the need to implement restart logic separately in each application so z/OS batch programs are smaller, easier to understand, faster to develop, and less prone to bugs.
- Handles the mechanics of checkpoint and restart so developers can concentrate on defining the logical unit of work that successful restart requires.
- Provides a comprehensive set of services through a consistent, problem state API.
- Requires only minimal changes to application source code (and often no changes at all).
- Supports applications written in COBOL, PL/I, C/C++, and Assembler languages.
- Supports all RRS compliant resource managers such as Db2, IMS, WebSphere MQ, and Transactional VSAM.
- Supports restartable applications comprised of multiple load modules and/or program objects. These executables may in turn be comprised of multiple, independently written source modules.

Operational Benefits

A system-wide restart facility enhances operations by speeding up recovery times and minimizing the risk of delays and service disruptions. The restart facility provides the enterprise with the following benefits:

- Provides a single system image and single point of control throughout the Sysplex
- Reduces or eliminate restart errors
- Orchestrates automatic recovery actions when failures occur
- Helps meet SLAs with the ability to set and dynamically change batch execution priorities

- Enables the same JCL to be used for both initial and restart runs, which greatly reduces the possibility of error in the manual process of updating JCL for restart
- Allows restartable applications to be tuned in accordance with established practices
- Provides monitoring and control facilities for the entire workload of restartable z/OS batch applications
- Permits database utilities that require exclusive control of a resource to run free of contention
- Predicts completion times so behind schedule warnings can be issued and remedial action can take place
- Prevents batch applications that should be restarted from being cold-started by mistake
- Provides a system programming interface for use by job schedulers, database utilities and other authorized programs

Choosing a Restart Solution

When evaluating a system-wide restart facility, *IT managers and technicians* should consider the following features:

- Work without intrusive system hooks that can make z/OS service and upgrades problematic
- Provide same-day support for new z/OS releases and major subsystems like Db2, IMS, and WebSphere MQ
- Provide forward and backward compatibility with all releases of IBM software whose IBM support status is current
- Ensure that the restart environment will not break if service is applied or a z/OS upgrade occurs
- Scale from a single mainframe to the most complex, multi LPAR Sysplex configurations, and data sharing environments
- Support all RRS compliant resource managers—including Db2, IMS, and WebSphere MQ
- Support batch jobs initiated under JES or WLM control
- Support the latest DFSMS facilities like extended addressing volumes (EAV)
- Support widely used SMS features like compressed and extended format datasets
- Interoperate with DFSMS Advanced Copy Services that create dynamic and point-in-time copies of data
- Limit access to authorized restart functions through standard RACF, ACF2, and TopSecret definitions

In addition, *evaluators* should determine if the restart solution can perform the following tasks:

- Allow restart from near the point of failure after abends, recompiles or system IPLs—with all resources in a consistent state
- Implement restart functionality in system software rather than within the application program
- Provide common and simplified operating procedures for restart
- Enable the same JCL to be used for both initial and restart runs
- Provide checkpoint pacing so applications can commit after every unit of work without concern for performance
- Allow failing programs to be changed and recompiled if necessary, and then restarted
- Support automatic restart and unit of work retry after resource unavailable conditions like SQLCODE -911
- Reposition Db2 cursors automatically
- Enable applications without commit logic to take checkpoints at unit of work boundaries that are defined external to the application source code
- Exploit z/OS Parallel Sysplex capabilities to facilitate workload switching and process migration
- Provide facilities to monitor restartable applications, predict their completion times and govern their operation
- Allow restart support to be temporarily disabled or permanently removed should the need arise, on an application or site-wide basis
- Provide useful diagnostics such as compile date, compile options, record count and record contents at abend time
- Interoperate with z/OS job schedulers such as those from Broadcom, IBM and BMC, and provides additional function
- Fully support diagnostic tools and debugging aids such as XPEDITER, the IBM Debug Tool and Abend AID

If a vendor supplied solution is being considered, evaluators should determine if the vendor is committed to the successful implementation and deployment of the solution. Does the vendor enhance the software with regularly scheduled releases and provide ongoing service and support of the highest caliber? Can the organization leverage the experience, methodology and best practices gained from the vendor's successful implementations?

Conclusion

As 24x7 access to data and applications increasingly becomes the rule and an unconstrained batch window becomes the exception, the enterprise requires a robust and reliable restart capability with the following features:

- Ensures data integrity, availability, and recoverability
- Minimizes the impact of z/OS batch application failures and keep the enterprise on schedule
- Significantly reduces downtime and lower the risk of business disruption
- Shields the business as well as its partners and customers from the effects of batch application failure
- Improves batch job elapsed times
- Enhances the disaster recovery preparedness of the enterprise
- Promotes concurrency, ease batch window constraints, and improve business resilience

A system-wide restart solution provides a proven method for enhancing z/OS batch workloads in an evolutionary, non-disruptive fashion. With only minimal changes to source code, it avoids the risk, time and expense of a rip and replace approach. The nightly batch cycle becomes more predictable, reliable, secure, and takes less time. The enterprise as a whole benefits from shortened recovery times and improved service levels without impacting existing processes, procedures, or communities of users. The enterprise realizes significant operational benefits along with simplified development and maintenance.

With a growing recognition of the risks to which the enterprise is vulnerable, management must achieve the following objectives:

- Prepare for problems, preferably before they occur
- Cope with pervasive uncertainty, risk, and an exploding number of contingencies
- Envision disaster and failure scenarios that were unthinkable a short time ago
- Prepare for both planned and unplanned outages
- Devise procedures to detect, analyze, and solve problems

Restart capability is consistent with all these objectives. It plays a critical role in leveraging the legacy portfolio of z/OS batch applications that are already in place. Restart is increasingly recognized as a simple and cost-effective solution with proven return on investment characteristics. Since timing can make a big difference in the ROI calculation, management must decide when to make such an investment—whether to act now or wait and see. As restart capability becomes pervasive in the z/OS environment, its strategic rationale becomes increasingly evident and the business case more compelling by the day.