Real Time Data Analytics on Active Data Guard

Krishna C Gonugunta[1], Meng Chen[2]
[1]Sr. Database Admin/SCS Operational Manager, Nevada System of Higher Education (NSHE), 2601 Enterprise Rd, Reno NV 89512
[2]Centre for Innovative and Lifelong Learning, ICDT

ABSTRACT

Oracle Database (ODB) offers significant acceleration for analytic queries with its highly compressed, transactionally consistent, memory-optimized Column Store. Customers may utilize Oracle ODB to make real-time judgments by evaluating extensive data sets at remarkable rates. Active Data Guard (ADG) is Oracle's all-encompassing solution for high availability and disaster recovery of the Oracle Database. Oracle ADG mitigates the substantial expense of idle redundancy by enabling reporting applications, ad-hoc queries, and data extracts to be delegated to the synchronized, physical Standby database duplicated via Oracle ADG. In Oracle 12.2, we expanded the ODB benefit to the Oracle ADG architecture. ODB-on-ADG markedly enhances the performance of analytic, read-only workloads executed on the physical Standby database, while the Primary database persists in managing high-velocity OLTP workloads. Customers may distribute their data across the In-Memory Column Stores on both the Primary and Standby databases according to access patterns, so achieving fault tolerance and workload segregation without sacrificing essential performance SLAs. This study examines the principal issues associated with constructing the ODB-on-ADG architecture, particularly the synchronized maintenance of the In-Memory Column Store on the Standby database, amid ongoing high-speed OLTP activity that continually alters data on the Primary database.

Keywords: Data Guard Configuration, Active Data Guard, Data Replication, Real-time Data Synchroniztion, Data Guard Broker, Real-Time Data Protection, Standby Database Analytics, Real-Time Querying, Data Guard Snapshot Standby, Zero Data Loss, Read-Only Standby database, Real-Time Data.

Introduction

Oracle Active Data Guard (ADG) offers active-standby replication for the Oracle Database. The Standby database is a synchronized, physical replica of the Primary database, usually located at a distant site. The Primary interacts with the Standby database using a network protocol such as TCP/IP. The Standby database is crucial for disaster recovery, usually trailing the Primary database by sub-second latency, thereby offering nearly real-time, read-only access to database items. Sub-second delays are acceptable for many reporting systems that handle extensive datasets, such as Big Data analytics. Transferring read-only workloads to the Standby database alleviates CPU use on the Primary (Production) database for OLTP and segregates batch reporting demands from live operations [3]. This study elucidates the enhancement of ODB functionality [12] inside the Oracle ADG architecture, enabling analytic tasks offloaded to the Standby database to execute with much improved speed. The ODB-on-ADG architecture facilitates the independent scalability of both Primary and Standby databases using Oracle Real Application Clusters (RAC), granting enterprise-scale clients complete autonomy in augmenting their OLTP or read-only workloads [1-5].

Capacity Expansion Capability: When the In-Memory Column Store (IMCS) is setup for both Primary and Standby databases, the data contained inside the IMCS of the two databases may consist of entirely distinct sets of objects. This method efficiently enlarges the IMCS. In a standard setup, clients can establish three services: Standby-only, Primary-only, and Primary-and-Standby utilizing Oracle's Services Infrastructure [7]. Figure 2 illustrates that the most recent month of the SALES fact table data is stored in the Primary instance's IMCS, while the complete year's SALES data is retained in the Standby instance for analytical purposes. The dimension tables may be loaded in both cases to enhance join processing efficiency. For each partition of SALES data, the customer designates either the standby or primary service, and for each dimension table, the client selects a service including both primary and standby database instances. Customers may accomplish workload separation with the IMCS capacity extension feature of the ODB-on-ADG architecture, therefore benefiting from accelerated analytics for workloads executed on both instances [6-16].

The Oracle ADG's distinctive replication architecture required a redesign of essential elements within the Oracle ODB infrastructure, particularly those responsible for populating data in the IMCS and ensuring its transactional integrity. A significant issue in building the ODB-on-ADG architecture was to preserve the primary advantage of ADG — its catastrophe recoverability. The recoverability from disaster is contingent upon the rapidity with which the Standby database can synchronize with the redo logs generated by the Primary database. Consequently, the ODB-on-ADG infrastructure is engineered to impose minimal overhead on the ADG architecture. The approach executes minimum processing on crucial pathways and leverages the extensive parallelism utilized by the Oracle ADG architecture.

This article is structured as follows: Section II presents an introduction of Oracle ADG and Database In-Memory; Section III elucidates the ODB-on-ADG architecture and the design of its components; Section IV offers performance findings; and Section V finishes with a discussion of future work.

An Overview of Oracle ADG and ODB Architectures

Summary of Oracle ADG Architecture

Oracle ADG offers active-standby replication for the Oracle Database. The replica, referred to as the Standby database, is a synchronized, physical duplicate of the Primary database, preserved by Redo Apply (also known as Media Recovery) [6].

Redo logs transmitted from the Primary database encompass redo records, which may be produced by many Oracle database instances (utilizing Oracle RAC [9]). A redo record may include numerous Redo Change Vectors (CVs), each corresponding to a specific database block indicated by the Database Block Address (DBA). All CVs in a redo record are deemed to have been produced at the identical SCN (System Change Number), which signifies the database timestamp when alterations were executed on the database blocks. Every CV is assigned a transaction identification. It is essential to recognize that the SCN linked to a redo record may not correspond to the database time at which the transaction commits

(commitSCN). The commitSCN is, indeed, the SCN linked to a 'commit' CV, which is utilized for a certain block. In the Standby instance, a Log Merger procedure organizes the redo data according to their SCN. The SCN-ordered logs may be utilized to apply to the respective database blocks, therefore generating a physical replica of the underlying datafiles on the Standby database. Upon the application of all CVs up to a certain SCN value, the Standby database is deemed to have synchronized with the Primary database at that SCN. This represents the most basic iteration of redo application [17-27].

Logs on the Primary database may be produced by several simultaneous transactions. The serialization of log application to the Standby database can significantly impede redo application, hence exacerbating the latency between the Primary and Standby databases. This undermines the primary objective of the Standby Database – catastrophe recovery. Consequently, the redo apply process is extensively parallelized for Oracle ADG by allocating the SCN-ordered collection of CVs among recovery worker processes via a hashing mechanism. Figure 3 illustrates the overarching architecture of Parallelized Redo Apply. Each DBA is mapped to a specific recovery worker identification, enabling a recovery worker process to autonomously process its allocated CVs and apply them to database blocks in SCN order.

While parallelization accelerates redo application, it presents a possible issue of transactional inconsistency. Due to the varying rates at which recovery workers may apply the change vectors, the transactional sequence of modifications on the Standby database may be disrupted. In Figure 3, recovery worker 1 continues to implement the CV from SCN 100, but recovery worker N has advanced and is utilizing the CV from SCN 110, which incorporates a modification to the database at a subsequent time. If the modifications to DBA 7 and DBA 150 constitute a single transaction, the alteration to DBA 150 must remain imperceptible to queries until the modification to DBA 7 is likewise perceptible, in accordance with the atomicity feature of transactions. A centralized coordinator must ensure awareness of the implemented changes on the Standby database, leading us to the notion of QuerySCN in Active Data Guard (ADG).

A recovery coordinator process monitors the advancement of all recovery worker processes and sets a consistency point at which all workers have finalized the redo application. The consistency point is identified as the 'QuerySCN' on ADG. The QuerySCN functions as the Consistent Read (CR) snapshot for queries performed on the Standby database, until a more recent consistency point (i.e., a higher QuerySCN) is created by the

Recovery Coordinator. Due to the varying speeds at which recovery worker processes apply redo, the QuerySCN on ADG often exhibits leapfrogging rather than a continuous sequence of successive SCNs.

Specific elements of the ODB-on-ADG architecture are intentionally placed to create a uniform consistency point for the IMCS. This allows queries executed at the QuerySCN to utilize the IMCS on the Standby Database. Section III examines these elements and their positioning comprehensively [28-41].

Summary of Oracle Database

Row-stores are optimal for OLTP workloads, where transactions engage a limited number of rows but several columns inside each row, whereas column stores are appropriate for analytic workloads that often access a vast number of rows, but only a few columns per row. Oracle ODB [12] implemented a dual-format architecture that preserves two versions of identical data: the conventional row-format on-disk and a columnar format in the IMCS. The ODB Transaction Manager maintains consistency between the column store and the current transactional activities in the row store. Oracle ODB is therefore designed to accelerate mixed-OLTP workloads that execute transaction processing and analytic queries.

The IMCS data consists of read-only In-Memory Columnar Units (IMCUs). IMCUs utilize methods like as data compression and encoding to optimize the packing of the IMCS. The In-Memory Scan Engine utilizes techniques like as SIMD vector processing, in-memory storage indexes, and efficient predicate evaluation and aggregation to enhance the performance of analytic queries. Data loading in the IMCS, referred to as Population, is often executed as a background process and does not impact concurrent transactions and queries. The population creates a snapshot SCN for each IMCU, and the IMCU is populated with data that aligns with the snapshot SCN according to Oracle's Consistent Read (CR) paradigm.

Upon loading, data in the IMCUs is synced with active transaction processing using specific methodologies. A Snapshot Metadata Unit (SMU) is associated with each IMCU and monitors the integrity of the data contained inside its related IMCU at many levels of granularity: block level, row level, and column level. The In-Memory Scan Engine synchronizes the IMCU data with the SMU to guarantee that incorrect or outdated data is not retrieved from the IMCS, but rather from the database buffer cache (i.e., the row-store). As transactions continue to alter the underlying row-store, an increasing proportion of the data in an IMCU becomes invalid over time. The repopulation approach is utilized to update the data in an IMCU with a more recent snapshot SCN, hence optimizing query speed when scanning the IMCS. Repopulation, like to population, is entirely online, transparent to queries and transactions interfacing with the IMCU, and is executed as a background process. A collection of heuristics is employed to initiate repopulation and adjust the repopulation frequency of each IMCU.

Besides ensuring consistency guarantees for data within IMCUs, SMUs facilitate concurrency management and synchronize processes such as repopulation, scanning, and removal of IMCUs. Figure 4 depicts a comprehensive overview of the ODB architecture, highlighting access patterns for various components.

ODB-on-ADG Infrastructure

The Standby database does continuous redo application and generates consistency points that ensure queries provide consistent results. The ODB-on-ADG architecture utilizes specialized components to populate the IMCS and preserve its transactional consistency at

certain consistency points. Figure 5 illustrates the principal components of the ODB-on-ADG system [42-53].

The ODB-on-ADG infrastructure engages with the QuerySCN progression on the Standby database to get a consistent snapshot SCN for the IMCS population. Upon population, the transactional consistency of the IMCS is kept via strategically located components:

The Mining Component leverages recovery workers to detect changes to items within the IMCS. The metadata extracted by the Mining Component is stored in the In-Memory ADG (IM-ADG) Journal. The Invalidation Flush Component transmits this metadata to SMUs during QuerySCN progression, therefore rendering updated data in the IMCUs invalid.

Nuanced improvements to the previously described components enable ODB-on-ADG architecture to expand effortlessly over ADG-RAC and accommodate schema modifications. Moreover, specific redo generation may be utilized on the Primary database to maintain the consistency of IMCS during ADG instance restarts. The subsequent subsections elucidate the function and design of these components comprehensively.

Population of the IMCS on ADG

The IMCS population in the Standby database, like to that in the Primary database, is engineered to remain entirely online and does not impede continuous queries on the IMCS. A segment loader divides an object into data block ranges, while background population worker processes create IMCUs for the DBA ranges. Queries and redo application on the Standby continue uninterrupted throughout the populating of the IMCS.

In contrast to the Primary database, the Standby database disseminates distinct consistency points that align with the QuerySCNs. Consequently, the snapshot SCN of an IMCU is invariably the QuerySCN determined at that moment. This is crucial because the population infrastructure may experience a transient state of the database if it selects a snapshot that is not a consistency point. Synchronization is essential between the recovery coordinator disseminating a fresh QuerySCN and the population infrastructure recording the snapshot SCN. This is accomplished via the 'Quiesce Period' on the Standby Database. Prior to publishing a new QuerySCN, the recovery coordinator acquires the 'Quiesce lock' to signify the commencement of the Quiesce Period on the instance. The population infrastructure is prohibited from capturing the snapshot SCN for IMCUs during the Quiesce Period. Upon the publication of the new QuerySCN, the Quiesce Period concludes, allowing the population infrastructure to acquire the snapshot SCN for IMCUs. Background procedures inside the population infrastructure verify the conclusion of the Quiesce Period and maintain the Quiesce lock while recording the snapshot SCN for an IMCU.

Upon populating the IMCUs in the Standby database instance, the query engine operating on the Standby database—identical to that of the Primary database—can leverage all optimizations and methodologies devised by the In-Memory Scan Engine to scan the IMCS, thereby delivering exceptionally rapid query responses. The subsequent primary objective is to maintain the IMCS on the Standby database in alignment with the consistency point or

QuerySCN being disseminated, ensuring that queries get the most current and consistent results.

Mining Element

Recovery personnel on the Standby database (ADG) implement Redo Change Vectors (CVs) to the foundational data blocks. The ODB-on-ADG Mining Component leverages the recovery workers to detect each CV. When the CV alters an object designated for loading in the IMCS on the Standby database, a tuple comprising the Object Identifier, Data Block Identifier (DBA), and the list of modified rows inside the data block is recorded in the IM-ADG Journal. Given that ODB-on-ADG operates with Oracle Database facilitating multi-tenant applications, the tuple further comprises tenant information. Each tuple extracted from analyzing a CV is referred to as a 'Invalidation Record' (refer to Figure 6). As modifications to the data blocks are this tuple is tagged with its transaction identifier, ensuring atomicity at transaction borders.

Besides extracting alterations to the data within the IMCS, ODB-on-ADG methods must also extract certain control information. Every transaction possesses a distinct commit point, or commitSCN, at which the alterations of a transaction are deemed atomic, persistent, and accessible to queries in accordance with Oracle's Consistent Read model. The IMCS on the Standby database must also comply with these promises. Consequently, the ODB-on-ADG Mining Component extracts control information regarding transactions – namely. Transaction state transitions, including Transaction Begin, Prepare, Commit, and Abort, along with the corresponding commitSCN for each transaction. Invalidation records are linked to this control information via the Transaction Identifier.

One may naturally inquire, "Why cannot an invalidation record be promptly flushed to the SMU following its construction?"The SMU must document this information to ensure transactional consistency of the IMCUs. The rationale for postponing the flush is twofold. Initially, because the populating of an IMCU occurs as a background process, independent of the redo application, it is plausible that the corresponding SMU has not yet been established. Secondly, even if the SMU is present, prematurely purging the invalidation records results in the exposure of a transaction's alterations prior to its commitSCN. Although this may appear to be a precautionary measure, certain processes are necessary to ensure the SMU's persistence and retention of the invalidation information until the QuerySCN on Standby aligns with the transaction's commitSCN. Given that population and repopulation occur in the background in an entirely online fashion, it is exceedingly challenging to offer such assurances.

To avert these occurrences, ODB-on-ADG protocols save invalidation entries in a 'IM-ADG Journal' and thereafter transmit them to the SMUs at an ideal juncture.

IM-ADG Journal to Cache the Invalidation Records

The IM-ADG Journal enables the documentation and storage of invalidation entries extracted by the ODB-on-ADG Mining Component. The IM-ADG Journal is intended to

operate in conjunction with the massively parallel redo apply, while preserving the principle that modifications must be atomic at transaction borders.

The fundamental architecture of the IM-ADG Journal comprises an in-memory hash table that associates a transaction identifier with its invalidation data. The hash table is dimensioned according to the degree of parallelism utilized by the ADG design to minimize conflict among the recovery worker processes.

Nevertheless, with an exceedingly high transaction throughput on the Primary database, some chaining in the hash buckets may be observed. The resultant hash chains are safeguarded by a 'bucket lock' to synchronize several recovery worker processes functioning on the same hash bucket. Figure 7 illustrates the overarching architecture of the IM-ADG Journal.

Each hashbucket has hashtable nodes that function as the anchor for invalidation records originating from a transaction. It is crucial to uphold transaction atomicity guarantees—either all modifications of a transaction must be accessible to a query, or none should be. Upon the creation of an anchor node for a transaction, each recovery worker is allocated a designated section within the anchor node to store the invalidation records it generates. This eliminates the necessity for synchronization among several recovery workers processing invalidation records for a transaction, which is a frequent occurrence. Figure 7 illustrates the storage of mined invalidation entries for transactions T1 and T2 within the IM-ADG Journal. T3 now lacks invalidation records; nonetheless, the anchor node would have been established upon the mining of the matching 'transaction begin' (control operation) by a recovery worker.

Advancement of QuerySCN and Invalidation Flush to SMU

When the Standby database is prepared to progress the QuerySCN to establish a more recent consistency point, the invalidation records collected in the IM-ADG Journal must be transmitted to the SMUs, but only if the transaction that generated those changes has a commitSCN that is less than or equal to the new QuerySCN. The IM-ADG Journal maintains distinct invalidation entries for each transaction, rendering this procedure straightforward. Nonetheless, a transaction may alter data across several IMCUs, necessitating the mapping of invalidation records to the relevant SMUs to facilitate a cost-effective flush operation. The Invalidation Flush Component does this by categorizing the invalidation data into 'Invalidation groups.' The recovery coordinator progressing the QuerySCN purges the invalidation groups to pertinent SMUs prior to disseminating the updated QuerySCN. Consequently, any queries executed at the new QuerySCN will render the associated data in the IMCU invalid.

Although this appears to be a simple procedure, it can result in considerable lag in publishing the new QuerySCN if the recovery coordinator does this task independently and sequentially. As stated in Section IIA, the Primary database produces logs in a multi-threaded fashion, executing hundreds of transactions per second, resulting in a rapid advancement of the SCN

on the Primary database. Consequently, the Standby database must rapidly elevate the consistency point to ever higher QuerySCNs. Any delay in creating the QuerySCN jeopardizes the Standby database's synchronization, hence compromising its failover capabilities. The Invalidation Flush is therefore on the critical path, making the optimization of this procedure essential.

The ODB-on-ADG architecture utilizes two primary methods to minimize the latency of Invalidation Flush during QuerySCN progression. A assistance structure known as the 'IM-ADG Commit Table' is established to facilitate rapid access to the IM-ADG Journal. Secondly, the recovery personnel are reassigned to execute a highly parallelized, collaborative flushing operation.

The ODB-on-ADG Mining Component preserves an in-memory, ordered linked list of transaction IDs and their corresponding commit SCN in the IM-ADG Commit Table. When specific control information on a transaction is extracted – namely. A 'Commit Table node' is generated during a transaction commit or transaction preparation. The Commit Table node includes the transaction identification and associated commitSCN, and is added into the linked list, which is organized by commitSCN. The Commit Table node directly references the anchor node in the IM-ADG Journal, which holds the transaction's invalidation entries.

When a new consistency point must be established, the ODB-on-ADG Invalidation Flush Component utilizes the recovery coordinator process to truncate the Commit Table and generate a Worklink (refer to Figure 8). All nodes in the worklink include transaction IDs for transactions that must be 'flushed' to the SMUs prior to the publication of the new consistency point. The Invalidation Flush Component does this by securing direct access to the IM-ADG Journal anchor node via the worklink. It collects all invalidation records for each transaction, organizes them into invalidation groups according to the DBA ranges for IMCUs, and transmits them to the corresponding SMUs.

To mitigate the bottleneck of insertion into a singular, sorted linked list by the Mining Component, the IM-ADG Commit Table may be partitioned to generate numerous sorted linked lists. A worklink is generated for each sorted list during the progress of QuerySCN.

Collaborative Flush:

Once the worklink is established, the invalidation records for various transactions inside the worklink may be parallelized with ease. The ODB-on-ADG Invalidation Flush Component employs recovery workers to facilitate this process, executing a 'Cooperative Flush.' Moreover, recovery personnel

During the execution of the redo apply task, periodically verify the creation of a worklink. If a worklink exists, the recovery workers assist the recovery coordinator in flushing a batch of nodes from the worklink prior to proceeding with redo application. The recovery

coordinator establishes the worklink, monitors its advancement, and disseminates the target QuerySCN as the new consistency point when the worklink has been depleted.

Specialized Redo Generation in the Primary Database

The Primary database remains largely indifferent to the existence of the IMCS on the Standby database, so no additional overheads are incurred when a transaction produces CVs to alter data on the Primary database. Nonetheless, an exception exists to this rule. Special redo generation may be executed on the Primary database by annotating the Commit Record of a transaction with a flag that signifies if the transaction altered any object designated for inclusion in the IMCS. This paragraph elucidates the utilization of this flag inside the ODB-on-ADG infrastructure.

The redo application on ADG is entirely independent of the Primary database. The database administrator may enable or disable redo apply on ADG and can arbitrarily shut down and restart the Standby database instances. ADG procedures retain sufficient state to facilitate recovery in such instances. Nevertheless, due to the IMCS lacking a permanent footprint aside from the foundational row-store objects, ODB-on-ADG components forfeit all their state upon instance restart. A transaction may be partially mined during a recovery session, after which the Standby database instance can be shut down and restarted, with the transaction commit information subsequently mined in a later session. If the commit SCN of the transaction exceeds the snapshot SCN of an IMCU, the invalidation records of the transaction must be sent to the corresponding SMU. The Commit Record of the transaction contains a flag to signify if any invalidation records are anticipated for this transaction. If they are, and the IM-ADG Journal contains none or just a partial set of invalidation data (indicated by the absence of a 'transaction begin' control information record), the Invalidation Flush Component employs a coarse invalidation technique to label all IMCUs for the designated renter as 'invalid.' Designating an IMCU as invalid prevents queries from accessing it until it is replenished.

Coarse invalidation causes considerable delay, but it occurs just when the Standby database instance is restarted. Therefore, if the populating of the IMCS is delayed briefly upon instance restart, we do not anticipate any coarse invalidation. It is important to acknowledge that special redo generation is not strictly necessary. ODB-on-ADG may conservatively presume that each transaction altered an object in the IMCS and initiate broad invalidation upon the detection of a missing 'transaction begin'. Nonetheless, to ensure optimal query speed, it is advisable to avoid initiating coarse invalidation.

Oracle Database on Oracle Data Guard with Real Application Clusters

Primary and Standby databases may be scaled autonomously with Oracle Real Application Clusters (RAC). Oracle Database In-Memory effortlessly scales over RAC, with IMCUs allocated across the IMCS on several Oracle RAC instances according to a hashing algorithm. The allocation of IMCUs to instances is recorded in a home-location map [5].

The Redo Apply process on the Standby database is often confined to a singular master instance, referred to as Single Instance Redo Apply (SIRA). A non-master instance does not execute Redo apply; instead, it operates a local recovery coordinator process that gets the QuerySCN from the master recovery coordinator and presents it to the queries handled by that instance. Consequently, the IM-ADG Journal and IM-ADG Commit Table are established solely on the master instance. During QuerySCN advancement, the ODB-on-ADG Invalidation Flush Component queries the home-location map and transfers the 'invalidation groups' to the specified instance. The local recovery coordinator on the receiving instance purges the invalidation groups to SMUs on that instance and confirms this to the master. To mitigate network latency's effect on QuerySCN progression, the ODB-on-ADG architecture utilizes batching and pipelined transmission of invalidation groups, as messaging across the network might create a bottleneck.

Interaction of IMCS with Schema Modifications

Oracle Database accommodates many Data Definition Languages (DDL) at the levels of table, partition, sub-partition, and column. DDL procedures often alter foundational schema objects. Specific DDL operations in Oracle are executed solely at the data dictionary level, resulting in no alterations to the underlying data blocks of the object. The In-Memory database on the primary database is closely connected with these DDL processes. For example, removing a column from a table in the IMCS eliminates the corresponding column from all IMCUs for that table, rendering it inaccessible to queries.

ODB-on-ADG is not afforded this privilege. DDL statements are executed through redo application on Active Data Guard. The ODB-on-ADG architecture consequently incorporates redo markers in the redo logs as a reaction to DDL operations. Redo markers resemble redo records but serve to denote alterations to non-persistent objects, namely the IMCUs inside the IMCS. Redo markers are extracted via the ODB-on-ADG Mining Component, and the information is stored in a distinct DDL Information Table, analogous to the IM-ADG Commit Table. When forwarding the QuerySCN,

Assessment of Performance

This section will outline the primary advantages of activating ODB on Oracle ADG. The performance enhancements of ODB for OLAP have been thoroughly evaluated in practical corporate scenarios. Our performance assessment tests aim to illustrate the benefits of utilizing Database In-Memory with Oracle ADG, while ensuring that the critical functionalities of redo apply and catch-up capabilities of Oracle ADG, essential for disaster recovery, remain uncompromised. We will examine two performance aspects: 1) Accelerating analytic workloads on Oracle ADG amidst OLTP on the Primary database, and 2) Evaluating Redo Apply performance on an ODB-enabled Standby database with high-throughput OLTP operating on the Primary database in a multi-tenant configuration on a 2-node Oracle RAC.

Analytics Workloads on Standby with and Without ODB-on-ADG Infrastructure

This section assesses the acceleration of analytic workloads on Oracle ADG in conjunction with OLTP on the Primary Database. In contemporary corporate entities, the capacity to integrate transactional processing with rapid on-demand analytics of real-time operational data is crucial for making informed business choices. Oracle ODB is a pioneering dual-format database that delivers rapid in-memory analytical performance while enhancing transactional processing. ODB-on-ADG enhances capabilities by offering isolation and workload segmentation, while accelerating reporting tasks.

We introduce a synthetic workload operating in various modes and the consequent improvement in scan response times for ad-hoc queries executing full table scans. All studies were conducted on the Oracle Exadata Database Machine, a cutting-edge symmetric multiprocessing server and storage cluster system.

The configuration comprises a synthetic OLTAP workload that emulates an insert/update workload interleaved with queries. The test comprises a large table containing 6 million rows and 101 columns, including 1 identity column, 50 numeric columns, and 50 varchar2 columns, with an index established on the identity column. The hardware configuration consisted of dual Intel Xeon E5-2690 processors operating at 2.90GHz, each with 8 cores, and 256GB of DRAM, of which just 60GB was allocated for the in-memory pool. The test was conducted for one hour with a target throughput of 4000 operations per second. The proportion of DMLs and analytical queries within the workload was adjustable. We exhibit enhancements in performance for ad-hoc searches using full-table scans executed on the Standby database, while the Primary concurrently manages a workload including various DML activities. We utilize measures like as query response time and CPU utilization to demonstrate the capabilities of the ODB-on-ADG system. A crucial aspect of the configuration is to ensure that the Oracle database buffer cache is adequately sufficient to prevent any physical I/O.

The achievement of the target throughput of 4000 ops/s is unattainable without ODB. There is considerable backpressure since the configuration utilizes the same thread pool for executing DMLs on the Primary database and queries on the Standby database. This results in a decrease in throughput. Dedicated threads may be utilized to sustain throughput for Data Manipulation Languages (DMLs).

1) Workload limited to updates

The update-only workload in the simulated OLTAP configuration generates 4000 operations per second, including 1% scan operations (40 scans per second) on the Standby database, while 70% updates (2800 updates per second) and 29% fetch operations via the index are performed on the Primary Database instance. We analyze the response times of queries Q1 and Q2 on the Standby Database, both with and without ODB-on-ADG. Figure 9 illustrates that the response time has enhanced by about 100-fold for the sample queries.

The expedited scans render the Standby a feasible option for task isolation while also decreasing CPU use on the Primary. In Update-only workloads, CPU utilization on the

Primary Database decreases from 11.7% when all activities are executed on the Primary to 4.7% when scans are delegated to the Standby Database. The CPU use of the Standby Database rises from 2% to 17%, with the disproportionate increase attributed to its architectural divergence from the Primary Database.

Update and Insert workload

The Update and Insert workload sustain table scans at 1% on the Standby Database, with a throughput of 4000 operations per second. It performs 25% inserts, 40% updates to the primary database, with the remainder consisting of index-based retrievals. Figure 10 juxtaposes the response times for Q1 and Q2 on the Standby database, both with and without ODB-on-ADG.

The use of ODB-on-ADG markedly enhances the performance of the Standby database, resulting in a nearly tenfold reduction in response time. However, the query response times pertain solely to the original table data. It is important to recognize that with inserts, the population infrastructure must employ much more CPU resources to integrate the newly entered data into the IMCS. The extensive concurrent invalidation and population activities on the edge IMCU related to new inserts result in a restricted performance advantage of the IMCS.

Comparison of Read-only Analytic Workloads on Primary and Standby Databases

This experiment demonstrates that the Primary and Standby databases exhibit equivalent performance under a scan-only demand.

A workload devoid of DMLs is executed independently on both the Primary and Standby databases. This indicates that searches for a specific data subset (e.g., a partition) devoid of DML activity may be effortlessly offloaded to the Standby, entirely transparent to the end-user.

The scan-only workload employs the identical simulated OLTAP configuration as subsection IVA, executing 4000 operations per second, including 25% ad-hoc queries doing full-table scans (1000 scans per second) and 75% fetch queries utilizing the index. Table 2 juxtaposes the response time for Q1 on the Primary and Standby databases with ODB activated on both.

Moreover, there is a direct transference of CPU use from the Primary to the Standby database instance—while the Primary's CPU usage diminishes from 8% to 0.5%, the Standby's CPU escalates from 0.3% to 7.9% during the execution of scans against the Standby database.

Execution of Redo Apply on the Standby Database

This experiment demonstrates that the ODB-on-ADG functionality does not substantially influence Redo Apply on the Standby database. The QuerySCN advancement rate is minimally impacted by the Invalidation Flush. The workload employed is a high-throughput transaction workload including a mix of short, medium, and long-running transactions

executed on the primary database utilizing Oracle multi-tenant architecture. The Primary and Standby databases are configured with 120 GB of DRAM.

The advancement of the redo log archiving on the Primary database operating with two Oracle RAC Instances (pri_log, pri_log2) for a duration of two hours. The archived redo is transmitted to the Standby database, and the status of the redo log application on the Standby database RAC instances 1 and 2, with the ODB-on-ADG feature on, is illustrated in the figure (std_log1, std_log2). The log catchup is very fast, and the Standby database exhibits negligible latency, despite the overheads caused by the ODB-on-ADG architecture.

Conclusions

Oracle Active Data Guard is a distinctive architecture that facilitates query execution on a Standby database, whilst functioning as a disaster recovery solution. The ODB-on-ADG infrastructure allows queries performed on the Standby Database to use ODB advantages, significantly enhancing the response time of certain queries. ODB-on-ADG utilizes the highly parallelized architecture of ADG Recovery to synchronize the In-Memory Column Store on the Standby database with ongoing transactional activities on the Primary database, while guaranteeing the integrity of the Standby database stays dedicated to its objective of catastrophe recoverability. The extension of Database In-Memory capabilities to the Standby database allows customers to optimize their read-write and read-only workloads by segregating them across the Primary and Standby Databases, hence facilitating accelerated analytics for both types of workloads.

Activating ODB on the Standby database has granted access to several functionality offered by ODB. In-Memory Expressions [1] are now enabled on the Standby database, delivering enhanced performance for intricate analytical expressions utilized in reporting queries, including JSON processing. In-Memory Join Groups may also be established for the Standby database to expedite join processing. External data sources, such as Hadoop, can be utilized for population in the IMCS with the In-Memory External Tables functionality [7]. The innovative formats and techniques employed by ODB, like as in-memory storage indexes and aggregation push-down, are smoothly integrated into ADG, so enhancing the Standby database for real-time analytics processing.

References

[1] Joshi, D., Sayed, F., Jain, H., Beri, J., Bandi, Y., & Karamchandani, S. A Cloud Native Machine Learning based Approach for Detection and Impact of Cyclone and Hurricanes on Coastal Areas of Pacific and Atlantic Ocean.

[2] Agarwal, A. V., & Kumar, S. (2017, November). Unsupervised data responsive based monitoring of fields. In 2017 International Conference on Inventive Computing and Informatics (ICICI) (pp. 184-188). IEEE.

[3] Sai, K.M.V., M. Ramineni, M.V. Chowdary, and L. Deepthi. Data Hiding Scheme in Quad Channel Images using Square Block Algorithm. in 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2018. IEEE.

[4] Manduva, V.C. (2020) AI-Powered Edge Computing for Environmental Monitoring: A Cloud-Integrated Approach. The Computertech. 50-73.

[5] Tulli, S.K.C. (2023) An Analysis and Framework for Healthcare AI and Analytics Applications. International Journal of Acta Informatica. 1: 43-52.

[6] Pasham, S.D. (2023) Application of AI in Biotechnologies: A systematic review of main trends. International Journal of Acta Informatica. 2: 92-104.

[7] Manduva, V.C. (2020) How Artificial Intelligence Is Transformation Cloud Computing: Unlocking Possibilities for Businesses. International Journal of Modern Computing. 3(1): 1-22.

[8] Sakr, S., Liu, A., & Xie, M. (2020). Change data capture for scalable data migration. ACM Transactions on Database Systems, 45(3), 1-27.

[9] Tulli, S.K.C. (2023) Analysis of the Effects of Artificial Intelligence (AI) Technology on the Healthcare Sector: A Critical Examination of Both Perspectives. International Journal of Social Trends. 1(1): 112-127.

[10] Pasham, S.D. (2022) A Review of the Literature on the Subject of Ethical and Risk Considerations in the Context of Fast AI Development. International Journal of Modern Computing. 5(1): 24-43.

[11] Pasham, S.D. (2022) Enabling Students to Thrive in the AI Era. International Journal of Acta Informatica. 1(1): 31-40.

[12] Tulli, S.K.C. (2023) Utilisation of Artificial Intelligence in Healthcare Opportunities and Obstacles. The Metascience. 1(1): 81-92.

[13] Tulli, S.K.C. (2023) Warehouse Layout Optimization: Techniques for Improved Order Fulfillment Efficiency. International Journal of Acta Informatica. 2(1): 138-168.

[14] Manduva, V.C. (2020) The Convergence of Artificial Intelligence, Cloud Computing, and Edge Computing: Transforming the Tech Landscape. The Computertech. 1-24.

[15] Manduva, V.C. (2021) AI-Driven Predictive Analytics for Optimizing Resource Utilization in Edge-Cloud Data Centers. The Computertech. 21-37.

[16] Pasham, S.D. (2017) AI-Driven Cloud Cost Optimization for Small and Medium Enterprises (SMEs). The Computertech. 1-24.

[17] Pasham, S.D. (2018) Dynamic Resource Provisioning in Cloud Environments Using Predictive Analytics. The Computertech. 1-28.

[18] Manduva, V.C. (2021) Exploring the Role of Edge-AI in Autonomous Vehicle Decision-Making: A Case Study in Traffic Management. International Journal of Modern Computing. 4(1): 69-93.

[19] Memon, S., Bhatti, S., & Ali, A. (2019). Automated data migration strategies for enterprises. Future Generation Computer Systems, 91, 117-130.

[20] Manduva, V.C. (2021) Optimizing AI Workflows: The Synergy of Cloud Computing and Edge Devices. International Journal of Modern Computing. 4(1): 50-68.

[21] Manduva, V.C. (2021) Security Considerations in AI, Cloud Computing, and Edge Ecosystems. The Computertech. 37-60.

[22] Palanisamy, S., & Liu, L. (2019). Efficient privacy-preserving data masking for cloud-based machine learning applications. IEEE Transactions on Services Computing, 12(3),

444-457.

[23] Manduva, V.C. (2021) The Role of Cloud Computing In Driving Digitals Transformation. The Computertech. 18-36.

[24] Manduva, V.C. (2022) AI Inference Optimization: Bridging the Gap Between Cloud and Edge Processing. International Journal of Emerging Trends in Science and Technology. 1-15.

[25] Sen, A., & Sinha, S. (2020). Backup and rollback mechanisms for secure data migration in enterprises. Journal of Cyber Security and Mobility, 9(4), 369-392

[26] Manduva, V.C. (2022) Blockchain for Secure AI Development in Cloud and Edge Environments. The Computertech. 13-37.

[27] Manduva, V.C. (2022) Multi-Agent Reinforcement Learning for Efficient Task Scheduling in Edge-Cloud Systems. International Journal of Modern Computing. 5(1): 108-129.

[28] Manduva, V.C. (2022) Security and Privacy Challenges in AI-Enabled Edge Computing: A Zero-Trust Approach. International Journal of Acta Informatica. 1(1): 159-179.

[29] Pasham, S.D. (2021) Graph-Based Models for Multi-Tenant Security in Cloud Computing. International Journal of Modern Computing. 4(1): 1-28.

[30] Pasham, S.D. (2022) Graph-Based Algorithms for Optimizing Data Flow in Distributed Cloud Architectures. International Journal of Acta Informatica. 1(1): 67-95.

[31] Pasham, S.D. (2023) Privacy-preserving data sharing in big data analytics: A distributed computing approach. The Metascience. 1(1): 149-184.

[32] Manduva, V.C. (2022) The Role of Agile Methodologies in Enhancing Product Development Efficiency. International Journal of Acta Informatica. 1(1): 138-158.

[33] Manduva, V.C. (2023) Artificial Intelligence, Cloud Computing: The Role of AI in Enhancing Cyber security. International Journal of Acta Informatica. 2(1): 196-208.

[34] Manduva, V.C. (2023) Unlocking Growth Potential at the Intersection of AI, Robotics, and Synthetic Biology. International Journal of Modern Computing. 6(1): 53-63.

[35] Manduva, V.C. (2023) Artificial Intelligence and Electronic Health Records (HER) System. International Journal of Acta Informatica. 1: 116-128.

[36] Pasham, S.D. (2019) Energy-Efficient Task Scheduling in Distributed Edge Networks Using Reinforcement Learning. The Computertech. 1-23.

[37] Pasham, S.D. (2020) Fault-Tolerant Distributed Computing for Real-Time Applications in Critical Systems. The Computertech. 1-29.

[38] Pasham, S.D. (2023) Enhancing Cancer Management and Drug Discovery with the Use of AI and ML: A Comprehensive Review. International Journal of Modern Computing. 6(1): 27-40.

[39] Tulli, S.K.C. (2023) Enhancing Marketing, Sales, Innovation, and Financial Management Through Machine Learning. International Journal of Modern Computing. 6(1): 41-52.

[40] Manduva, V.C. (2023) Model Compression Techniques for Seamless Cloud-to-Edge AI Development. The Metascience. 1(1): 239-261.

[41] Manduva, V.C. (2023) Scalable AI Pipelines in Edge-Cloud Environments: Challenges and Solutions for Big Data Processing. International Journal of Acta Informatica. 2(1): 209-227.

[42] Manduva, V.C. (2023) The Rise of Platform Products: Strategies for Success in Multi-Sided Markets. The Computertech. 1-27.

[43] Tulli, S.K.C. (2023) Application of Artificial Intelligence in Pharmaceutical and Biotechnologies: A Systematic Literature Review. International Journal of Acta Informatica. 1: 105-115.

[44] Pasham, S.D. (2023) The function of artificial intelligence in healthcare: a systematic literature review. International Journal of Acta Informatica. 1: 32-42.

[45] Pasham, S.D. (2023) An Overview of Medical Artificial Intelligence Research in Artificial Intelligence-Assisted Medicine. International Journal of Social Trends. 1(1): 92-111.

[46] Pasham, S.D. (2023) Network Topology Optimization in Cloud Systems Using Advanced Graph Coloring Algorithms. The Metascience. 1(1): 122-148.

[47] Tulli, S.K.C. (2022) Technologies that Support Pavement Management Decisions Through the Use of Artificial Intelligence. International Journal of Modern Computing. 5(1): 44-60.

[48] Manduva, V.C.M. (2022) Leveraging AI, ML, and DL for Innovative Business Strategies: A Comprehensive Exploration. International Journal of Modern Computing. 5(1): 62-77.

[49] Manduva, V.C. (2023) AI-Driven Edge Computing in the Cloud Era: Challenges and Opportunities. International Journal of Modern Computing. 6(1): 64-95.

[50] Tulli, S.K.C. (2022) An Evaluation of AI in the Classroom. International Journal of Acta Informatica. 1(1): 41-66.

[51] Pasham, S.D. (2023) Opportunities and Difficulties of Artificial Intelligence in Medicine Existing Applications, Emerging Issues, and Solutions. The Metascience. 1(1): 67-80.

[52] Pasham, S.D. (2023) Optimizing Blockchain Scalability: A Distributed Computing Perspective. The Metascience. 1(1): 185-214.

[53] Tulli, S.K.C. (2023) The Role of Oracle NetSuite WMS in Streamlining Order Fulfillment Processes. International Journal of Acta Informatica. 2(1): 169-195.