IBM Data Gate: Making On-Premises Mainframe Databases Available to Cloud Applications

Knut Stolze[1] Felix Beier[1] Vassil Dimov[1] Eirini Kalogeiton[1] Mateo Tošić[1]

Abstract:

Many companies use databases on the mainframe for their mission critical applications. It is important to exploit this existing data for analysis and business decisions via modern applications that are often built exclusively for cloud environments. IBM Db2 for z/OS Data Gate (Data Gate) is bridging the gap between mainframe databases and such cloud-native applications. It offers high-performance data synchronization for connecting both worlds, while providing data coherence at the level of individual transactions. Data Gate is a hybrid cloud solution that protects existing systems, applications, and investments into those, while enabling new use cases to work with mainframe data. In this paper, we give an overview of Data Gate and discuss how it evolved from IBM Db2 Analytics Accelerator (IDAA) technology by adjusting the system architecture and some of the functionality in order to make IBM Db2 for z/OS (Db2/z) data a first-class citizen the cloud.

1 Introduction

Database systems are the well-established approach to manage application data since the relational model was invented. In 1983, IBM released Db2/z, a relational DBMS for IBM's *mainframe*. As of today, Db2 and Db2/z are the core data management products for many organizations, as they ensure excellent availability, performance, scalability, and storage saving options. Nowadays, the need of organizations has grown to take advantage of their huge amount of data that is generated or collected every day. To cope with this need, cloud solutions are introduced, providing a simple, efficient, cost-effective, scalable, and flexible environment [Ch15]. IBM offers various solutions for making on-premises data sources, such as Db2/z, available as first-class citizens in cloud-native environments while addressing a series of requirements, like ensuring that business-critical systems are not impacted negatively, existing investments are protected, and risks to such systems are minimized.

This paper focusses on an approach for *integrating* on-premises databases into the new cloud-native landscape rather than *migrating* such systems. While this integration methodology protects decades of customers' investments in their data management infrastructure, it rises a set of requirements that have to be addressed in order to seamlessly integrate business-critical databases into a managed cloud ecosystem, without having to relocate existing applications and/or workloads. First, integration solutions have to optimize data placement with caching strategies in order to collocate cloud-based data accesses. Those cloud-based caches need

[1] IBM Germany Research & Development GmbH, {stolze,febe}@de.ibm.com, {Vassil.Dimov1,Eirini.Kalogeiton, Mateo.Tosic}@ibm.com

doi:10.18420/BTW2023-06

to be synchronized with the on-premises data sources, which comes with high requirements w.r.t. change replication performance. Second, any data replication mechanism inevitably introduces latencies that need to be hidden by some cache coherency protocols to guarantee consistent views when data is accessed by consuming applications. Third, compatibility aspects need to be considered. Although the SQL standard pretends a uniform specification of relational data management and processing, the actual systems implementing them differ largely in available features and functions. These differences need to be hidden from customer applications for a truly transparent data access experience. Finally, various non-functional requirements w.r.t. security, administration, monitoring, stability, etc. have to be satisfied for enterprise readiness. In the following, we will discuss how Data Gate [IB22e] solved these challenges for making Db2/z data available and synchronized in the cloud. Based on well-proven database accelerator technologies of IDAA [IB22d], Data Gate replicates Db2/z data into a cloud database and uses patented cache coherence protocols to guarantee data consistency, irrespective from where the data is accessed. Data Gate can be considered the cloud evolution of IDAA. We will present important aspects of Data Gate's architecture and will also shed some light on architectural decisions that led to dead ends.

2 Related Work

Data Gate is primarly an integration component that makes data available from mainframe databases (ie. Db2/z) in established cloud databases. Generally, different approaches exist to integrate on-premise data sources in the cloud The most common ones are described below.

The **Lift-&-Shift** approach transfers not only the data, but the complete on-premises environments, consisting of data stores and the applications working with them, to a virtualized cloud-based infrastructure. This is beneficial for data stores and applications running on platforms that are available in the cloud already. *Lift-&-Shift* may require recompiling code, changes to the packaging/installation procedures or security and user management due to the different underlying environment. Additionally, data and file conversions may change [Me18]. If suitable, this approach eliminates the need for on-premises environments, as it benefits from reusability and is faster then rewriting existing applications [In20]. On the other hand, future enhancements are slower to adopt because the software stack is not modernized [PP20].

Custom data ingestion pipelines use a combination of tools for bulk loading and continuous data replication to copy the data from on-premises databases to a cache in the cloud for minimizing data access costs of cloud-native applications. The required tools can be implemented from scratch or by combining open-source technologies, like Kafka [Kr11], Debezium [Co22] and others [De21]. Alternatively, a combination of existing products with different performance characteristics and consistency semantics can be used, e. g., AWS Database Migration Service [Ge22]. Such data ingestion pipelines minimize risks for existing applications because the source database systems are not impacted and allow to embed data transformations that may be required by consuming cloud applications. On the other hand, this approach has a longer time-to-market because multiple components are involved that have to be separately implemented and integration-tested as a whole system.

Many companies offer **integrated data replication tools** that provide all benefits of custom data ingestion pipelines while abstracting the implementation complexity, e. g. Oracle Golden Gate [Gu16] and Cyniti Data Replication [Sy19]. Those tools replicate data from the on-premises data store to a target database in the cloud, offering features for initial bulk data ingestion, continuous data replication, and utilities to orchestrate the whole process. These tools take care of encryption and data transformations and, usually, offer better data movement performance and superior consistency semantics than home-grown solutions. An integration with other cloud services is provided to ease data consumption in cloud environments. Examples for such integrations are cataloging data in enterprise data catalogs, automated data governance, and data profiling. Data Gate falls into this group.

While the previous approaches involve movement of data from on-premises data sources to the cloud, **data virtualization** allows consuming the data from cloud-based applications without creating a cached data copy. Inconsistencies between different data versions are avoided because all data consumers operate on the same data. To achieve that, a data virtualization layer is defined that combines separate data sources from different platforms. The ease of integrating various data platforms at one place enables rapid prototyping of new cloud applications [MAT19]. On the other hand, the on-premises data store has to facilitate the whole workload, analytical and transactional, stemming from both, on-premises and cloud applications. Thus, a virtualization approach can be more costly in terms of operations of the source data store and pose a risk for existing business-critical applications.

3 Evolution from IDAA to Data Gate

Data Gate can be categorized as integrated data replication tool that facilitates the movement of data from an on-premises mainframe database to a cloud database. Data Gate is integrated into IBM Cloud Pak for Data (CP4D) [IB22c], IBM's cloud platform for data analytics with a unified service architecture for user management, encryption, and common user experience for working with a wide variety of data sources. It interacts with the platform's metadata catalog for making mainframe data discoverable and consumable by any cloud application. Applications can access the data in the cloud database at any time with the guarantee that they always operate on the latest, consistent view of the data. Furthermore, Data Gate supports seamless routing of analytic workload stemming from Db2/z to the cloud. It evolved from IDAA, an on-premises accelerator appliance for processing analytical Db2/z queries. The architectural evolution is illustrated in Figure 1.

The architecture of IDAA that acts as the base for Data Gate is illustrated in Figure 1A. IDAA is a pre-configured cluster that runs a tuned IBM Db2 Warehouse (Db2 WH) installation and an accelerator server middleware. IDAA is deeply integrated into Db2/z as internal resource for processing analytic queries (green flow in Figure 1A). IDAA exists only as an extension to Db2/z and the data residing on the accelerator can only be accessed through Db2/z. IDAA's main purpose is to make mainframe workload more predictable and to execute the analytical part of it more efficiently. Queries that are submitted by client applications are analyzed by the Db2/z optimizer which decides to route the query to IDAA if it is

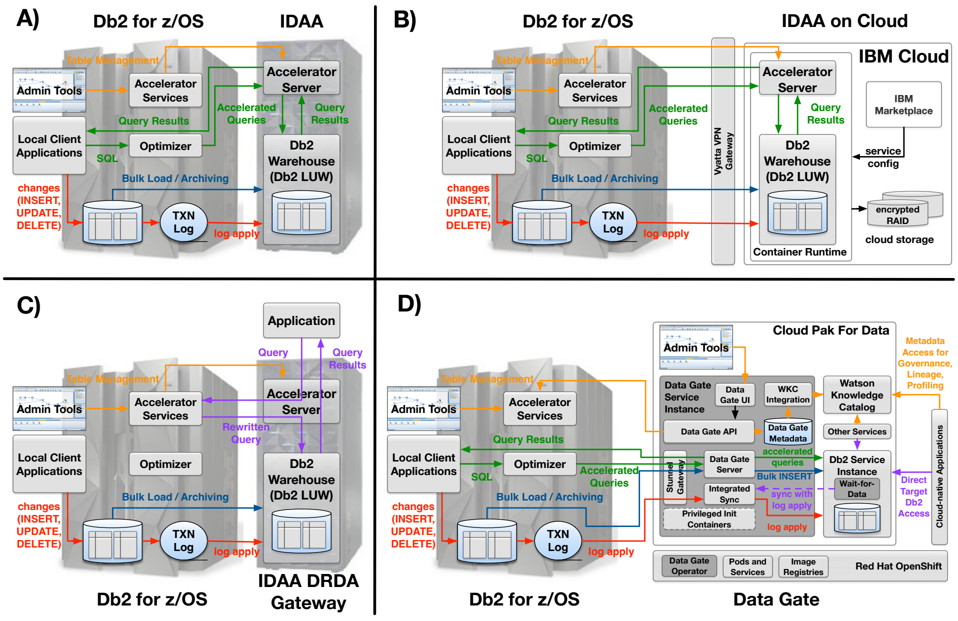

Fig. 1: IDAA and Data Gate Architecture Overview

an analytical one. Otherwise, it is executed locally. In case it is routed to IDAA, it will be transformed to the Db2 WH SQL dialect, executed on the appliance, and its result set is streamed back to the application via Db2/z. The offloading is, therefore, completely transparent to the application and no modifications are needed. That is a critical requirement for many Db2/z users. The data residing on IDAA is always kept in the same encoding as on the source database so that queries will deliver the same results as if they are run on Db2/z.

Accelerated queries are executed on a copy of the Db2/z data. An administrator selects tables that should be accelerated and adds them to the appliance (orange flow in Figure 1A), which registers the table's schema. The actual copy of the table data is triggered via a bulk load mechanism that unloads table partitions from Db2/z in an efficient way using a dedicated utility (see [IB22b]). The partition data is transferred to the accelerator and inserted in parallel to Db2 WH (blue flow Figure 1A). The load process creates a snapshot of the tables that can later be updated with another bulk load in case many changes accumulated before the accelerator data copy needs to be refreshed.

Consistency of the copied data in the accelerator is realized by a multi-version concurrency control (MVCC) mechanism that employs views to define visible vs. invisible rows [SBM19]. Therefore, each bulk load assigns a unique ID to all processed rows which is stored in an internal column of the target table. This ID is used by a view for filtering rows when queries are executed. The view is updated after each load operation and synchronized with potential concurrent incremental update operations. Cross-table consistency is guaranteed because DDL statements are fully transactional in Db2 WH and adhere to the ACID properties.

An alternative to bulk loading is the incremental update strategy, called IBM Integrated Synchronization (InSync) (red flow in Figure 1A). It can be used if changes happen frequently and accelerated queries should run on the latest data version. InSync captures changes to observed tables from the Db2/z transaction log and applies them to the target database [Bu20]. The overall data flow is highly optimized and avoids data transformations, e. g., the raw log format written by Db2/z is directly consumed by Db2 WH. The specialized incremental update implementation is further exploited during query processing. An application can request to process the data as it was in Db2/z at the time when a query was submitted. Upon such a request, the accelerator captures the current head-of-transaction-log from Db2/z or the newest log position for the tables specified in the query. Query execution is delayed until this log position has been applied by InSync to the target database.

IDAA provides a High Performance Storage Saver (HPSS) feature to move data from Db2/z to Db2 WH. Historical partitions that do not change anymore on Db2/z can be loaded to IDAA and are removed from the source database. Only partitions that are still being updated remain on Db2/z. Although the "archived" data is not present on Db2/z anymore, it can still be queried via the accelerator [IB22a]. HPSS reduces the data volume on Db2/z which leads to faster index maintenance, more efficient reorganizations and statistics collection, and, thus, processing overall.

Another feature allows tables to exist only on the accelerator without being present on Db2/z at all [BSM16]. For those Accelerator-only Tables (AOTs), not only analytical queries but also all Data Manipulation Language (DML) statements are routed to the accelerator. AOTs can be used for efficient in-database transformations, data preparation tasks, and are a perfect fit for storing temporary data for subsequent queries or reports.

IDAA's architecture has been adjusted several times in the past in order to enable additional use cases and data access patterns. Figure 1B) illustrates the IDAA on Cloud modification that provided query acceleration capabilities as standalone cloud service [BS17]. The data flow and use cases are comparable to the appliance form factor of IDAA. From a high-level perspective, just the deployment of the accelerator software stack changed so that it can be hosted in IBM's public and private cloud environments. Therefore, the Db2 WH as well as the accelerator middleware have been containerized. To meet the security requirements of a cloud-based database service offering, encryption mechanisms were introduced, which were not needed before in the isolated on-premises environment of IDAA. Both, data at rest, i. e. the table copies stored inside Db2 WH, as well as data in motion, i. e. data that flows between Db2/z and the accelerator, are protected via encrypted storage and VPN gateways.

A second direction of the IDAA architecture aimed at opening the encapsulated target database to minimize data transfer overheads for large result sets to external applications. The corresponding IDAA DRDA Gateway architecture is depicted in Figure 1C). An application can directly connect to the accelerator for executing a query via the DRDA protocol [DRD03][2] (purple flow in Figure 1C). From there, the query is passed on to Db2/z

[2] For IBM's database systems, the DRDA protocol is the underlying technology used by ODBC/CLI or JDBC.

where the statement is validated (privileges, object existence, etc.) and rewritten according to the SQL dialect of Db2 WH. Db2/z routes the query to the accelerator, which hands it over to the cloud database for execution. Contrary to normal query execution, IDAA returns an empty result set to Db2/z while forwarding the actual result set from Db2 WH to the application.

Although the DRDA gateway showed a 10x performance improvement for transferring result sets, data could not be directly accessed. All queries had to be processed by Db2/z during preparation and IDAA was handling the execution in Db2 WH. IDAA itself was an additional hop for transferring result sets and applications had to use the SQL constructs of Db2/z and could not take advantage of native constructs available in the cloud database. Hence, the product never got beyond the prototyping stage. The standalone IDAA on Cloud service was also discontinued because it lacked a deep integration into a cloud-based data analytics environment.

A third direction was the integration of additional data processing engines with the target database. For example, IBM Netezza Analytics Stored Procedures [IB16, BSM16] have been made available as separate package that could be added to IDAA. The package enabled additional stored procedures for analytics that directly run in the target database. The stored procedures were callable in Db2/z, but processing was forwarded to the accelerator. The intention was to generalize the query acceleration idea to custom analytical packages. Spark was also provided as additional processing engine that was collocated with the target database, Netezza at that point in time. Stored procedures could be used to submit Spark jobs. However, a shift in the underlying database technology from Netezza to Db2 WH lead to a setback.

The path of Db2/z data towards the cloud has been paved by Data Gate which can generally be regarded as consolidation of the previous architectures. Data Gate's architecture is illustrated in Figure 1D), which will be explained in more details in Section 4.

4 Deep Dive into Data Gate

4.1 Data Gate Architecture

Data Gate mainly inherited its functionality from IDAA on Cloud. IDAA and Data Gate can coexist and be connected to the same Db2/z system. While IDAA is attached locally and used for accelerating analytic queries, Data Gate is an extension of Db2/z to a cloud-based environment where it maintains a cached twin copy of the tables. In contrast to IDAA, Data Gate allows direct access from applications to the target database. In fact, it is the user's responsibility to provide, configure, and maintain a Db2 on Cloud instance and connect it to Db2/z via Data Gate. This offers the flexibility to use the cached Db2/z data for various use cases in the cloud-based environment.

Unlike IDAA on Cloud, Data Gate is integrated in a common cloud architecture (cf. section 4.2), which is more flexible since it allows organizations to tailor their system architecture towards specific needs of the target applications by allowing:

- Independent configuration of the computation and storage resources that should be allocated for Data Gate and the cloud database instance
- Selection between multiple storage types with different capabilities with regard to failover and performance characteristics
- Selection of different cloud database form factors: row store for OLTP workloads, Db2 WH for analytics, and Db2 WH with query acceleration for OLAP workloads from cloud-native applications and accelerated query routing via Db2/z, like IDAA

Internally, Data Gate uses a microservices architecture. Therefore, IDAA's monolithic middleware was split into more fine-granular component containers which are loosely coupled and controlled over an API layer that maintains stateful information and metadata. These APIs communicate with Db2/z over secure connections and invoke administrative stored procedures that are used to control IDAA or Data Gate. Data Gate uses encryption in all layers of communication and data propagation (cf. section 4.2). This differs from IDAA where a dedicated, private network is used to avoid any encryption-related overhead.

4.2 Cloud Platform Mandates

Data Gate is available through CP4D that is deployed on top of an Red Hat OpenShift (OpenShift) cluster. One advantage of OpenShift is the use of operators for integrating all components. A Data Gate operator observes container registries and automatically reconciles the cluster on updates, e. g., for security patches, which can be applied in short time intervals. Moreover, CP4D provides a set of common services that can be used by any service offering, like user and credentials management, logging and diagnostics, and a common user interface. Unlike IDAA, Data Gate enables direct access to the cloud database, which allows to consume Db2/z data by cloud-native applications. Since the target database is not fully controlled by Data Gate, high security requirements are implemented by:

- Providing fewer privileges for run-time users
- Deploying a dedicated init container for separation of duties and target database tuning
- Encrypting data everywhere: at rest and in motion between all components, e. g., TLS encryption between Db2/z and Data Gate is provided via a dedicated *stunnel* container

4.3 Integration with Cloud Services

Most organizations have huge amounts of data stored in many forms in various locations. Finding relevant data quickly and connecting disparate data sources can be challenging and time-consuming. IBM Watson Knowledge Catalog (WKC) [IB22f] unites all information assets into a single metadata catalog. A single click in Data Gate's User Interface (UI) is sufficient to publish metadata about the source database connection and its data assets (tables), along with the target database connection and all replicated data assets to WKC (orange data/metadata flow in Figure 1D). In addition to making the metadata discoverable, the main purpose of the integration is to enable WKC's built-in tools for data governance, lineage, and profiling, as well as numerous other cloud-native applications, e. g., Watson Studio for data analysis. This is particularly useful to ensure that each user can only see data according to their roles, where WKC masks and randomizes sensitive data.

4.4 Changes in the Backend Database

The target database is not owned by Data Gate but integrated as separate cloud service. To ensure high performance of the whole system the target database has to be tuned. For example, the archive log is disabled for better data synchronization performance in any deployment. Other parameters, such as automated statistics maintenance and table reorganizations, are enabled for the analytical use case only. For protecting the runtime environment, the tuning is done in a special init container that runs as privileged user in comparison to the normal Data Gate operations (see Figure 1D).

Data Gate caches data in the cloud for new applications which require data in UNICODE rather than EBCDIC encoding that is typically used in Db2/z. Thus, Data Gate is re-encoding the data when it is copied. Because such code page conversions may increase string lengths, the column widths of target tables is increased based on heuristics that implement a trade-off between the range of supported values, maximum column widths, and storage utilization.

In IDAA, queries are routed from Db2/z to the target database. With Data Gate, the cloud database is directly accessible by applications. The replicated database is impacted by the data synchronization latency and is only eventually consistent to the source database – similarly to most asynchronously replicated databases. Eventually consistent systems make no guarantees for the staleness of the data [Vo09]. Applications accessing the target database could retrieve data older than the one already persisted in the source database. Similarly to IDAA, Data Gate provides a way to run queries on the target database with the newest data from Db2/z. Query can retrieve transactionally consistent data as if the execution happens on the source database by using new SQL syntax in the cloud database, which uses Data Gate under the covers. A query can be annotated to *wait* for the newest data from the on-premises database to be replicated to the cloud database:

```
SET CURRENT QUERY WAITFORDATA = 10;
SELECT COUNT(*) FROM BANK.TRANSACTION;
```

The first line sets the `WAITFORDATA` special register of the cloud database to 10 seconds. The subsequent query will wait until the most recent changes from the source database have been replicated or the 10 seconds timeout has expired. Since Data Gate comes with very low latency (usually a few seconds only), query execution commences well before the timeout expires. If the timeout is reached, an appropriate error is returned. With this extension, *read-after-write inconsistencies* [Je] are avoided for applications that access data on the target database.

5 Performance Evaluation

After discussing the most important architectural changes that transform IDAA from a highly tuned on-premises analytical database accelerator to Data Gate as cloud database replication tool, this section sheds some light on the performance impact of these changes. From a use case perspective, we compared the performance of the initial data loading phase (bulk load), the incremental update performance for replicating changes from the source database to the target database, and also the query acceleration flow from Db2/z. Additional Data Gate use cases that employ different Db2 service versions as target database or different data access patterns, such as direct query processing from cloud applications on the target database, are out of scope for this paper.

Operation	IDAA	Data Gate
Initial Load Throughput (TB/h)	4.2	1.4
Average InSync Throughput (Tx/s)	511 k	364 k
Average InSync Latency (s)	6.4	10.4
Total Query Runtime (s)	969	1260

Tab. 1: Operations

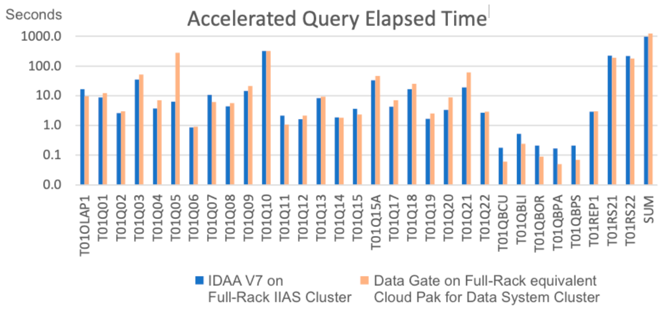

Fig. 2: Accelerated Query Performance

Since IDAA and Data Gate share large portions of the underlying code base, comparable performance results were expected with respect to the software stack. However, the cloud infrastructure abstraction layers (from OpenShift) and the new microservices architecture (introducing additional network communication between containers) may result in overhead reflected in the performance of Data Gate. The biggest performance impact is expected from the hardware resources and the Db2 service configuration that need to be specified when Data Gate is instantiated. While IDAA is preconfigured and tuned under lab conditions, Data Gate offers more options for its users. For the sake of space, just a single comparable configuration will be examined, without additional tuning on Data Gate side.

As testbed, a full-rack IDAA V7.5.8 on IBM Integrated Analytics System (IIAS) cluster with 168 cores, 3.5 TB RAM, SSD storage, and 20 Gbps network connection to Db2/z was used. As counterpart, a Data Gate 2.1 on a IBM Cloud Pak for Data System (CP4DS) cluster with comparable hardware specification was used. Since the CP4DS offers more resources than the IIAS cluster, the Db2 WH service was configured with less resources than the maximum to obtain a comparable target database. Data Gate uses Red Hat OpenShift Data Foundation (ODF) for storage which creates three replicas of each block that are distributed over multiple worker nodes. By using hostpath storage mapping, the performance could be increased because no data replicas are created and each worker node just mounts locally attached disks. However, since this configuration does not guarantee high availability, just the ODF results will be discussed. We used a 5 TB TPCH benchmark, extended by additional queries to match existing IDAA production workloads.

The performance results are outlined in Table 1. It can be seen that the initial load performance of Data Gate did not match expectations. The reason is a bottleneck in the network layer where additional tuning will be required. However, we do not consider this as a restriction because tables are typically bulk-loaded just once and then incremental update is used. Hence, InSync performance is more important. Table 1 shows that both IDAA and Data Gate perform well in terms of throughput and latency during the incremental update process. We highlight that both IDAA and Data Gate were validated in real environments where the InSync pipeline could keep up with the maximum change rate of the corresponding Db2/z subsystem. The query results of the benchmark were also almost on par. The drill-down in the query timings in Figure 2 shows that most queries perform equally good in both environments. Some regressed while others performed better, both stemming from the configuration differences.

Overall, the assumptions were met. The experiments have shown that the architecture changes of Data Gate work well. In most cases the requirements of current Data Gate users are already satisfied. But the load performance may be improved with additional tuning.

6 Conclusion & Outlook

In this paper, we gave an introduction on Data Gate and demonstrated how it makes Db2/z data accessible to cloud applications by replicating and subsequently synchronizing the data with a cached twin in a Db2 on Cloud service. We discussed how Data Gate was built on IDAA technology, which parts of it could be reused, and which parts had to be adjusted and why. Our performance measurements revealed that classic IDAA use cases can be executed by Data Gate without major performance impact and identified bottlenecks that can be addressed by additional architecture tuning.

The evolution is not done, however. Today, the interfaces between Data Gate and IDAA are kept very similar which is not practical in the long run. Applications accessing the data in the cloud database should not have to use Db2/z to obtain information, like replication latency or details about synchronization errors. We are working on providing such administrative and monitoring information directly in the cloud database.

Another feature we are working on is to use the cloud database as a replication source. Organizations have expressed interest to selectively propagate data provided by Data Gate on to other database for further processing. Such a daisy-chain replication requires adjustments in how Data Gate stores the data in its cloud database because internal abstraction layers, like views that are used today, cannot serve as sources for replication tools like CDC [Be12].

A third major functional enhancement is to enable data modifications in the cloud database, propagating such changes back to the mainframe, and making other tables that exist in the cloud database known on the mainframe. Of course, such features will break the concept of treating Db2/z as master of the data. In this respect, security, consistency, and durability are important concerns that needs to be taken into account.

Trademarks

Bibliography

[Be12] Beaton, A.; Noor, A.; Parkes, J.; Shubin, B.; Ballard, C.; Ketchie, M.; Ketelaars, F.; Rangarao, D.; Tichelen, W.V.: . Smarter Business: Dynamic Information with IBM InfoSphere Data Replication CDC. IBM Redbooks, 2012.

[BS17] Beier, Felix; Stolze, Knut: Architecture of a data analytics service in hybrid cloud environments. it-Information Technology, 59(3):151–158, 2017.

[BSM16] Beier, Felix; Stolze, Knut; Martin, Daniel: Extending Database Accelerators for Data Transformations and Predictive Analytics. In: Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016. pp. 706–707, 2016.

[Bu20] Butterstein, Dennis; Martin, Daniel; Stolze, Knut; Beier, Felix; Zhong, Jia; Wang, Lingyun: Replication at the speed of change: a fast, scalable replication solution for near real-time HTAP processing. Proceedings of the VLDB Endowment, 13(12):3245–3257, 2020.

[Ch15] Chou, David C: Cloud computing: A value creation model. Computer Standards & Interfaces, 38:72–77, 2015.

[Co22] Community, Debezium: . Debezium Documentation, 2022. https://debezium.io/documentation/reference/stable/index.html.

[De21] Densmore, James: Data Pipelines Pocket Reference. O'Reilly Media, 2021.

[DRD03] The Open Group. DRDA V5 Vol. 1: Distributed Relational Database Architecture, 2003.

[Ge22] Ge, Zhiyu: Technologies and Strategies to Leverage Cloud Infrastructure for Data Integration. Future And Fintech, The: Abcdi And Beyond, p. 311, 2022.

[Gu16] Gupta, Ravinder: Introduction to Oracle GoldenGate (OGG). In: Mastering Oracle GoldenGate. Apress, Berkeley, CA, pp. 3–10, 2016.

[IB16] IBM: . Supported IBM Netezza Analytics stored procedures, 2016. https://www.ibm.com/docs/en/daafz/5.1?topic=procedures-support-netezza-analytics-remote-stored.

[IB22a] IBM: . Archiving partition or table data with the High-Performance Storage Saver, 2022.

[IB22b] IBM: . Db2 13 for z/OS documentation, 2022. https://www.ibm.com/docs/en/db2-for-zos/13?topic=utilities-unload.

[IB22c] IBM: . IBM Cloud Pak for Data 4.5 documentation, 2022.

[IB22d] IBM: . IBM DB2 Analytics Accelerator for z/OS 7.5, 2022.

[IB22e] IBM: . IBM DB2 for z/OS Data Gate Analytics Accelerator for z/OS 7.5, 2022.

[IB22f] IBM: . IBM Watson Knowledge Catalog 4.5.x documentation, 2022. https://www.ibm.com/docs/en/cloud-paks/cp-data/4.5.x?topic=services-watson-knowledge-catalog.

[In20] Inc, TmaxSoft: Lift, shift and modernize: proven mainframe modernization strategies that enable digital transformation. 2020.

[Je] Jeena, R; Saravanakumar, S; Bharathi, B Poornima; Priyancaa, RP: Providing Consistency in Cloud Using Read after Write Technique to Endusers.

[Kr11] Kreps, Jay; Narkhede, Neha; Rao, Jun et al.: Kafka: A distributed messaging system for log processing. In: Proceedings of the NetDB. volume 11, pp. 1–7, 2011.

[MAT19] Muniswamaiah, Manoj; Agerwala, Tilak; Tappert, Charles: Data Virtualization for Decision Making in Big Data. Int. J. Softw. Eng. Appl, 10(5):45–53, 2019.

[Me18] Mead, Larry: Microsoft:Migrating:Mainframe:Environments. 2018.

[PP20] Paul, Ajo; Paul, Dipyaman: Migrating Mainframe workloads to Azure. Mindtree, 2020. https://www.mindtree.com/sites/default/files/2020-11/Migrating-Mainframe-workloads-to-Azure-Whitepaper.pdf.

[SBM19] Stolze, Knut; Beier, Felix; Müller, Jens: Partial Reload of Incrementally Updated Tables in Analytic Database Accelerators. BTW 2019, 2019.

[Sy19] Syniti: . Syniti Data Replication - User Guide, 2019.

[Vo09] Vogels, Werner: Eventually consistent. Communications of the ACM, 52(1):40–44, 2009.