Journal of Computer Science

Research Article

Enhancing Database Availability: A Combined Approach Using SQL Always on Failover Cluster Instance and Availability Groups

Faisal Rahman and Benfano Soewito

Department of Computer Science, Bina Nusantara University, Jakarta, Indonesia

Article history
Received: 26-01-2025
Revised: 12-05-2025
Accepted: 26-05-2025

Corresponding Author:
Faisal Rahman
Department of Computer Science,
Bina Nusantara University, Jakarta,
Indonesia
Email:
faisal.rahman003@binus.ac.id

Abstract: Database availability is a critical factor in supporting business processes to ensure uninterrupted operations. When the production server experiences failures, data may become inaccessible, disrupting business operations. To mitigate this risk, separating the production server from the reporting server is essential. However, this approach introduces challenges related to data synchronization and automatic failover during system failures. Existing High Availability (HA) solutions have significant trade-offs: FCI enables fast instance-level failover but lacks database replication, making it vulnerable to shared storage failures. Meanwhile, AG ensures continuous database synchronization but has higher failover latency due to log shipping overhead. While FCI provides automatic failover at the instance level, AG offers database-level synchronization and protection, including a read-only access option. To overcome these limitations, this study integrates SQL Always On Failover Cluster Instance (FCI) and Availability Groups (AG), combining FCI's rapid failover capabilities with AG's robust database replication. This hybrid approach enhances availability beyond standalone FCI or AG. Testing over 30 days demonstrated a 99.97% availability rate with an average downtime of 12.3 minutes, offering a practical solution for improving operational efficiency and minimizing system failures.

Keywords: Database Availability, Failover Cluster Instance, Availability Groups, Downtime

Introduction

In the current business environment, data is a vital asset for every organization. The availability of ready-to-use data at any time is a crucial factor in generating reports for decision-making processes. The data used for reporting purposes must be synchronized with the production server. However, data corruption in synchronized systems can disrupt reporting processes, potentially leading to financial losses. According to a survey conducted by Lawrence & Simon (2023), as shown in Figure (1), 14% of 385 companies experienced database synchronization issues or data corruption in the past three years, leading to data unavailability. This situation highlights the need for improved database availability and synchronization between production and reporting servers.

To address this issue, two options are available: upgrading to a server with greater resources or distributing the workload between the production and reporting servers. The primary challenge lies in ensuring that data remains available and consistently synchronized with the allocated resources, even with the separation of production and reporting servers. This approach aims to maintain server performance and eliminate any synchronization issues. Addressing synchronization issues between production and reporting servers is not a straightforward task and must be executed accurately. Failure to do so may result in problems during backup processes, report generation, or data analysis, ultimately affecting decision-making processes.

Therefore, several studies have explored the separation of production and reporting servers. In previous research, Zhang *et al.* (2021) utilized a High Availability (HA) solution based on redundancy within a single physical storage system, with data synchronization between a local disk and a network disk. In this configuration, the local disk functions as the primary storage device, while the network disk serves as a backup. In the event of a failure, the system

DOI: 10.3844/jcssp.2025.1332.1342

automatically switches to the network disk without any service disruption.

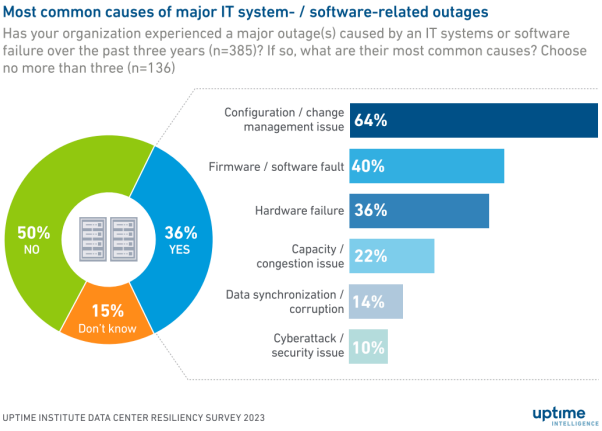

Fig. 1: Uptime Institute (Lawrence & Simon, 2023)

Literature Review

Hannum *et al.*(2019) in their study, employed High Availability SQL (HASQL) within the open-source database system Comdb2. This system is capable of recovering and resuming interrupted transactions caused by connection loss by redirecting users to another cluster node, thereby preventing data loss.

From these two studies, the authors observe that there remains a potential for data corruption or loss despite the allocation of resources for storage synchronization and data recovery. If the storage used experiences physical damage, the nodes or disks that have been distributed will no longer be accessible to users and additional time will be required for the database recovery process.

This study applies a combination of high-availability methods with system recovery to separate the functions of production and reporting databases while enhancing database availability. To achieve this objective, the study combines two features of SQL: Always Failover Cluster Instance (FCI) and Availability Groups (AG). These technologies provide automatic database replication and failover, ensuring continuous access and minimizing the risk of data loss (Carter, 2020).

In contrast to standalone FCI or AG implementations, the combination of FCI + AG ensures both instance-level and database-level redundancy, providing a more resilient failover mechanism. While FCI alone offers rapid failover, it lacks database replication, making it vulnerable to storage failures. Conversely, AG ensures data consistency across multiple replicas but suffers from failover delays due to synchronization overhead. By integrating FCI and AG, this study aims to create a hybrid HA solution that minimizes downtime while maintaining high data consistency, addressing the limitations of both approaches.

SQL Always On Failover Cluster Instance (FCI) isan instance of SQL Server installed across multiple nodes within a Windows Server Failover Cluster (WSFC). This type of instance relies on resources such as storage and a virtual network name (Microsoft, 2024). If the primary node experiences hardware, operating system, application, or service failures, the SQL Server instance will be transferred to another node within the WSFC.

SQL Always On Availability Groups (AG) is a high availability and disaster recovery solution that provides an advanced alternative to database mirroring. AG maximizes database availability for users within an organization (SQL Server, 2024). AG is capable of protecting a set of databases on the primary replica by continuously transferring transaction log blocks from all AG databases on the primary replica to the corresponding databases on all secondary replicas.

According to SQL Server (2024), Windows Server Failover Cluster (WSFC) allows multiple servers to work together to maintain service availability. In this setup, each server, referred to as a node, can be either a physical or virtual machine. These nodes communicate through various hardware and software components and WSFC continuously monitors their health. If one node fails, its services are automatically reassigned to another node to ensure uninterrupted operation. In practical implementations—particularly within banking institutions and enterprise-scale data centers—Windows Server Failover Cluster (WSFC) is commonly deployed using a multi-node and multi-subnet configuration. This design enables nodes to be geographically distributed, supporting robust disaster recovery strategies. Each node is typically connected to shared storage via a Storage Area Network (SAN), facilitating the use of SQL Server Failover Cluster Instances (FCI) to achieve rapid instance-level failover. WSFC employs quorum models such as Node Majority or Node and File Share Majority to ensure safe failover decisions and prevent split-brain scenarios in cases of inter-node communication failure. When integrated with SQL Server Always On Availability Groups (AG), WSFC forms a hybrid high availability architecture where FCI manages instance failover and AG provides synchronous database replication and read-only secondary replicas. This layered architecture allows WSFC to function as both a failover decision engine and a connection director via intelligent listener routing. The topology supports real-time monitoring of nodes, storage, and network health using tools like Failover Cluster Manager or third-party solutions. The integration of WSFC in a hybrid FCI + AG model delivers key benefits including minimized downtime, seamless client access continuity, and enhanced data resilience. By leveraging both instance-level and database-level failover capabilities, organizations can maintain the availability of mission-

DOI: 10.3844/jcssp.2025.1332.1342

critical systems even in the event of hardware failures or network disruptions.

High Availability (HA) refers to an IT system, component, or application's ability to operate continuously at a high level of performance without interruption over a specified period of time (Cisco, 2024). According to Zamanian *et al.* (2019), server failures must not result in service unavailability or data loss. To ensure service continuity, a failover process is required. Failover is a procedure that transfers the active node to a passive node, ensuring uninterrupted service for users (Chen & Tsai, 2019). In the failover process, it is essential to calculate the availability time of a system to ensure it can be accessed again. The percentage of availability can be determined using the following formula(Kit & Aibin, 2023):

$$Availability\,(\%) = \frac{MTBF}{MTBF+MTTR} \times 100$$

Mean Time Between Failures (MTBF) refers to the average time elapsed between one failure and the next in a system, representing the reliability of the system. Meanwhile, the Mean Time To Recover (MTTR) is the average time required to repair and restore the system to operational status after a failure occurs. Both metrics are crucial for evaluating system performance and planning maintenance strategies.

Several technical and environmental factors can significantly reduce MTBF in SQL Server environments. These include hardware degradation such as disk failures or memory faults, improper configuration of cluster nodes, overheating due to insufficient cooling, and network instability that leads to replication timeouts. Additionally, frequent software updates without thorough validation, high transactional workloads that result in resource contention, and misconfigured quorum models in Windows Server Failover Clustering (WSFC) can also trigger unplanned failovers or service interruptions. Identifying and mitigating these risks through real-time system monitoring, infrastructure optimization, and preventive maintenance practices is essential to sustain a high MTBF and ensure continuous service availability in mission-critical operations.

Related Works

Several approaches have been employed to ensure data consistency and secure business processes from system failures. However, these approaches have limitations, such as high implementation complexity and scalability challenges when the system operates within a large organizational environment.

Previous studies have explored SQL Server High Availability (HA) mechanisms separately. For instance, Zhang *et al.* (2021) investigated storage-based redundancy for HA, while Hannum *et al.* (2019) proposed HASQL for open-source database clustering. However, these studies did not address the combined impact of FCI and AG on performance and failover efficiency.

Other research, such as Chen & Tsai (2019), analyzed HA using PostgreSQL and MySQL but lacked insights into SQL Server's built-in failover mechanisms. Meanwhile, Khan & Sabri (2021) focused on SQL Server Always On AG but did not consider its integration with FCI. This study builds upon these works by integrating FCI and AG, leveraging FCI's rapid failover and AG's robust data replication to achieve an optimal balance between speed and reliability. Unlike prior studies, this research systematically evaluates failover efficiency, resource consumption, and downtime reduction across multiple configurations.

Comparison with Alternative HA Solutions

In addition to SQL Always On solutions, several high-availability architectures are widely used in database systems:

1. Oracle Data Guard: Provides database-level failover but lacks instance-level protection, making it inefficient for SQL Server instance failures
2. MariaDB Galera Cluster: Uses synchronous multi-master replication but requires all nodes to synchronize before committing transactions, increasing latency
3. PostgreSQL Streaming Replication: Ensures redundancy via log shipping but lacks automatic failover, requiring external tools like Patroni for cluster management

Prior studies, such as Shrestha & Tandel (2023); and Zamanian *et al.* (2019), have explored the benefits and limitations of Oracle Data Guard, MariaDB Galera, and PostgreSQL Streaming Replication. Their findings indicate that while Oracle Data Guard provides robust replication, it does not offer instance-level failover like FCI. Similarly, the MariaDB Galera Cluster ensures synchronous replication but introduces high transaction latency, making it less efficient for real-time applications. This study builds upon these findings by demonstrating how a hybrid approach (FCI + AG) can mitigate these limitations by combining fast instance failover with reliable database replication.

Materials

This study utilizes various configurations of processor cores and memory to identify the minimum specifications required to meet the needs of FCI, AG, and FCI + AG in designing the separation of functions between production and reporting servers to increase database availability. Table (1) provides the environment

DOI: 10.3844/jcssp.2025.1332.1342

specifications used as the baseline for testing the performance and requirements of each configuration.

Table 1: Environment specification

Component	FCI	AG	FCI + AG
CPU (Cores)	4 up to 8	4 up to 8	4 up to 8
Memory (GB)	4 up to 8	4 up to 8	4 up to 8
Disk (GB)	300	300	300
OS	Win Server 2012	Win Server 2012	Win Server 2012
SQL Version	2012	2012	2012

Methods

This study involves several research stages and designs that focus on the separation of functions between the production server and the reporting server to increase database availability.

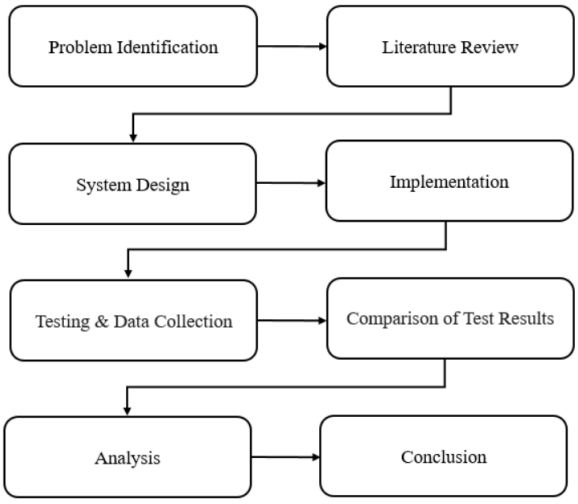

Fig. 2: Research stages

Research Stages

The research stages depicted in Figure (2) present the systematic process involved in the combination of SQL Always On FCI and AG, which is divided into five phases as follows:

1. Problem Identification: Identifying issues related to the need to increase database availability
2. Literature Review: Collecting relevant literature on solutions to database availability issues
3. System Design: Based on the problem and literature review, methods for database synchronization are designed
4. Implementation: The proposed synchronization methods are implemented and tested
5. Testing and Data Collection: Conduct tests under various scenarios and compile the results into comparative tables
6. Analysis and Conclusion: Analyze the collected data and draw conclusions based on the research findings

System Design

SQL Always On FCI

Figure (3) presents the system design of SQL Always On Failover Cluster Instances (FCI), where the production server and standby server utilize a shared storage architecture to ensure high availability. The Availability Group (AG) listener directs transactions to the production server and in the event of a failure, the failover cluster automatically transfers the workload to the standby server. The failover process is triggered when the system detects an anomaly in the production server through continuous heartbeat monitoring, SQL Server health checks, or node unavailability within Windows Server Failover Clustering (WSFC). If the production server stops responding due to network failure, hardware crash, or service interruption, the WSFC quorum mechanism determines the failure and initiates the failover process.

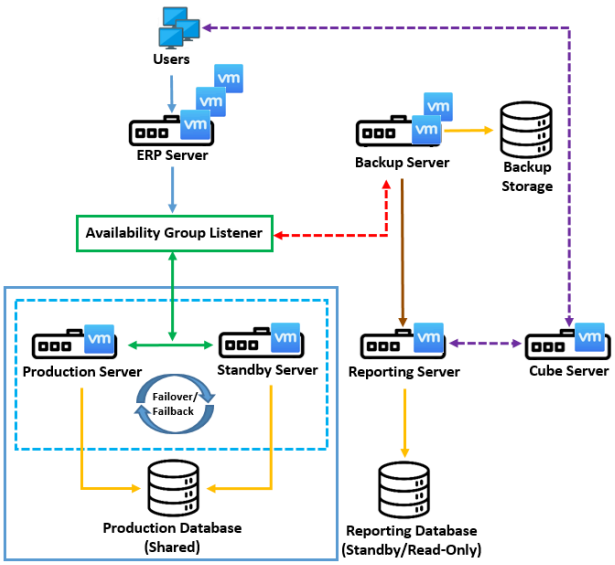

Fig. 3: System design of SQL always on FCI

During failover, the AG listener detects the transition and automatically reroutes client connections to the standby server, ensuring minimal downtime. Transactions that were in progress during the failure may be rolled back or retried, depending on the failover policy and the database transaction model used. The production database remains synchronized with the standby server through synchronous replication, ensuring that the standby server contains the latest committed transactions before taking over as the primary server.

The typical failover time can vary depending on network latency and cluster settings but generally completes within seconds to minimize service disruption. Once the failed production server is restored, a failback mechanism can be initiated to reinstate its role as the

DOI: 10.3844/jcssp.2025.1332.1342

primary server. Before failback occurs, the system ensures that all transaction logs and data are resynchronized between the standby and production servers to prevent data loss. Depending on the system configuration, failback can be performed manually by an administrator or automatically through a preconfigured policy within the failover cluster manager. To enhance system reliability and reduce the load on the primary database, data from the production database is routinely backed up to dedicated backup storage and replicated to a reporting database in a read-only mode. This reporting architecture enables analytical processing through a cube server, allowing complex queries and reports to be generated without impacting the performance of the production server.

Furthermore, the reporting server retrieves data asynchronously from the backup server, ensuring optimized resource allocation and operational efficiency. The interaction between the reporting and backup servers is further illustrated in Figure (4).

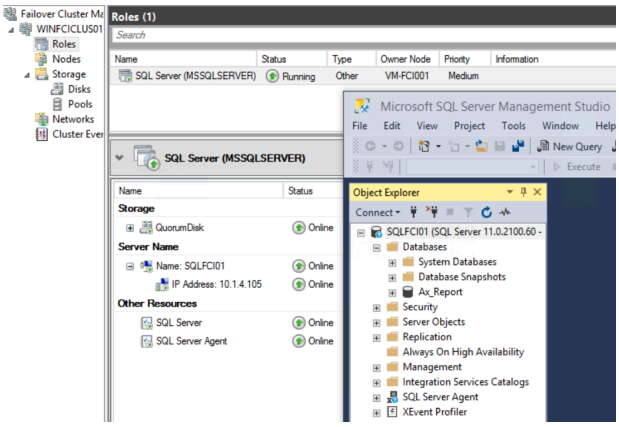

Fig. 4: SQL Always On the FCI dashboard

SQL Always On AG

Figure (5) presents the system design of SQL Always On AG, which ensures high availability through real-time synchronization and automatic failover. The AG listener directs connections to the primary replica (production database), with data synchronously replicated to the standby replica and the read-only replica (reporting database) for reporting purposes.

SQL Always On AG supports two replication modes: synchronous and asynchronous. In synchronous mode, transactions are committed simultaneously on both the primary and standby replicas, ensuring zero data loss but potentially increasing transaction latency. This mode is typically used within the same data center or low-latency network environments to guarantee data consistency.

In contrast, the asynchronous mode allows transactions to be committed on the primary replica without waiting for acknowledgment from the secondary replicas, improving performance but introducing the risk of data loss in case of failure. As shown in Figure (5), the primary and standby replicas operate in synchronous mode to ensure high availability, while the reporting replica utilizes asynchronous replication to optimize read performance without impacting transaction speed on the production server. The automatic failover process allows the standby replica to take over the role of the primary replica in the event of a failure, without disrupting access to the reporting database, which operates with a dedicated IP address. Reporting is carried out via a cube server, processing data from the reporting server without placing a load on the production server.

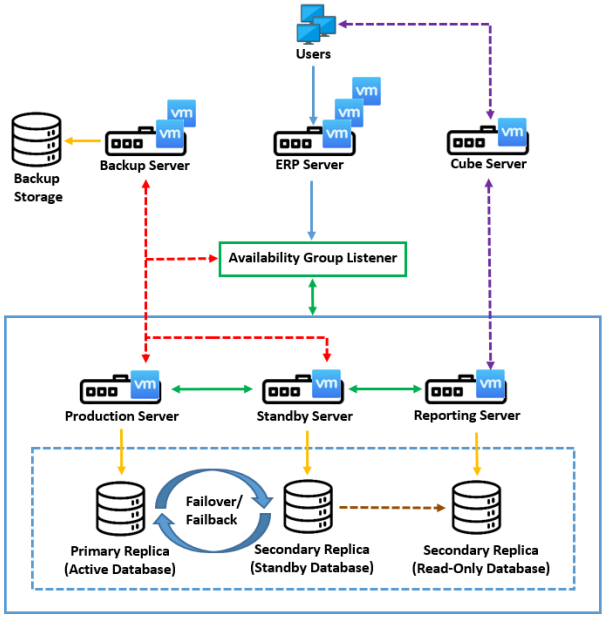

Fig. 5: System design of SQL Always On AG

This design ensures high availability and efficiency in database management. Each instance is equipped with dedicated storage, where synchronized data is automatically and in real-time replicated from the production server to both the standby and reporting servers, eliminating any data loss, as shown in Figure (6).

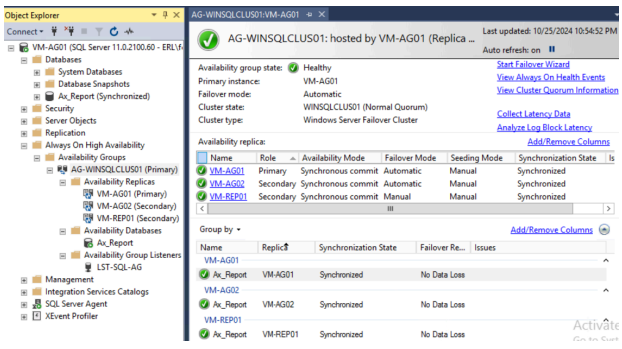

Fig. 6: SQL is always On the AG dashboard

DOI: 10.3844/jcssp.2025.1332.1342

SQL Always On FCI+AG

Figure (7) presents the system design that integrates SQL Always On Failover Cluster Instances (FCI) and Availability Groups (AG), providing comprehensive protection. FCI offers automatic failover at the SQL Server instance level, while AG ensures synchronization and protection at the database level, with a read-only access option on secondary replicas for reporting or backup purposes. Figure (8) presents two groups combined into a single dashboard.

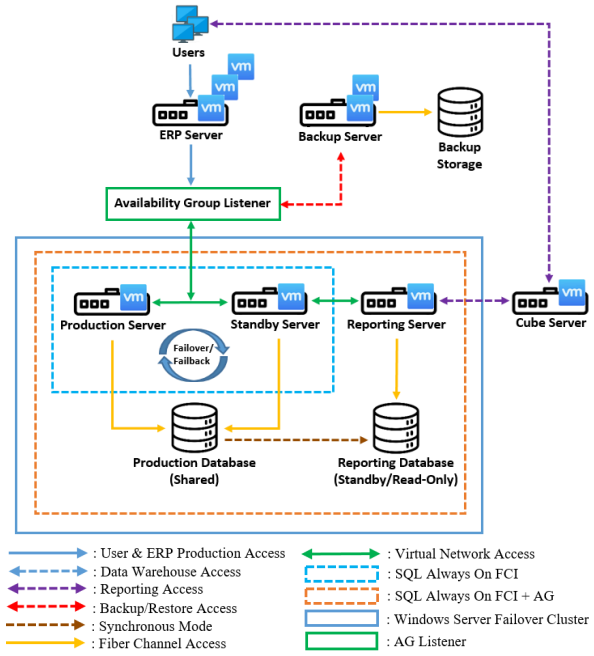

Fig. 7: System design of SQL Always On FCI + AG

SQLDBCLUS01 consists of the production and standby servers using FCI as the production database, synchronizing data in real time to VM-AG03, which serves as the reporting database. VM-AG03 uses AG with a status of "synchronized" and "no data loss." The integration of FCI and AG introduces distinct failover mechanisms that interact within the system. FCI provides automatic failover at the instance level using Windows Server Failover Clustering (WSFC), ensuring that the entire SQL Server instance moves to the standby server in case of failure.

In contrast, AG handles failover at the database level, allowing specific databases to transition independently between replicas. The diagram in Figure (7) indicates that the production and standby servers are clustered under FCI, sharing the production database. In the event of a failure, FCI manages the transition of the SQL instance between the production and standby servers, maintaining high availability. Meanwhile, AG ensures that the reporting database remains accessible, as it operates asynchronously on a separate instance.

The failover handling of these two systems can be independent or coordinated. If the FCI cluster fails over, the production instance shifts between the nodes without affecting the AG replicas. However, if an AG-level failure occurs on the reporting replica, AG will handle the transition independently without impacting the production database. This design ensures that system availability is maximized, preventing single points of failure while maintaining data consistency and accessibility across instances.

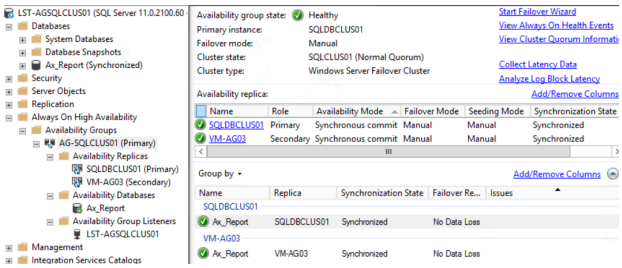

Fig. 8: SQL Always On FCI + AG dashboard

Testing Conditions

Testing was conducted in a virtualized environment using Windows Server 2012 R2 and SQL Server 2012, deployed on virtual machines with varying configurations (4–8 CPU cores, 4–8GB RAM and 300GB SSD storage). The evaluation involved three setups: (1) FCI-only, (2) AG-only, and (3) FCI + AG hybrid.

To simulate failover, intentional disruptions were introduced at different levels:

1. Node Failure Simulation: The primary node was forcefully shut down to test automatic failover
2. Network Partitioning: The connection between the primary and secondary replicas was severed to assess failover handling in AG mode
3. Manual Failover: Failover was manually triggered through SQL Server Management Studio to observe recovery time

SQL Server was configured with synchronous replication for the primary and standby replicas in AG mode, while FCI relied on shared storage failover. Performance metrics, including failover time, CPU/memory utilization, and transaction recovery time, were recorded over 30 days.

Results and Discussion

Server Utilization

Table (2) presents the monitoring results of server utilization over 60 min, using a sample configuration of an 8-core processor and 8GB RAM. The results underline the strengths and trade-offs of each

configuration. While AG provides robust data replication and high availability, it comes at the cost of higher CPU, memory, and disk utilization. FCI offers a lightweight solution with minimal resource requirements but lacks the advanced replication features necessary for environments requiring high data consistency. FCI + AG emerges as the most balanced configuration, leveraging the strengths of both FCI and AG. It ensures reliable failover and data protection with optimized resource utilization.

Table 2: Comparison of server utilization

Component	FCI	AG	FCI + AG
CPU	0.7331%	2.6738%	1.5391%
Memory	37.8363%	40.1525%	23.1686%
Disk	0.04218%	0.08416%	0.0825%

The results indicate that AG has the highest CPU usage at 2.6738%, significantly higher than FCI (0.7331%) and FCI + AG (1.5391%). This is primarily due to the continuous synchronization process in AG, which requires frequent transaction log captures and log shipping to secondary replicas. Additionally, AG's ability to handle multiple read-only replicas introduces additional overhead, as it must maintain consistency across instances while ensuring minimal data loss. Similarly, AG exhibits the highest memory utilization at 40.1525%, compared to FCI (37.8363%) and FCI + AG (23.1686%). This increased memory consumption stems from the need to manage multiple database replicas and process real-time transactional updates across nodes. The disk utilization pattern follows a similar trend, where AG (0.08416%) consumes nearly twice the disk resources compared to FCI (0.04218%), reflecting the impact of continuous log writes and replication overhead.

This makes the FCI + AG configuration ideal for environments demanding high data availability, consistency, and operational efficiency, particularly in systems with resource constraints. The trade-offs between resource usage and functionality suggest that FCI + AG is best suited for mission-critical applications where downtime and data loss must be minimized without overloading system resources.

Backup Performance

In this testing phase, two backup methods were implemented: full backup and differential backup, to evaluate the duration and throughput required during the backup process. The tests were conducted 30 times for each specification.

Full Backup

Table (3) presents a comparison of duration and throughput for the full backup process using FCI, AG, and FCI + AG configurations across various hardware specifications. AG demonstrated the best performance with an average duration of 289.69 sec and the highest throughput of 1,790.59 GB/h, particularly on the 6-core and 8 GB RAM specification, achieving a throughput of 2,186.66 GB/h.

FCI recorded the fastest average duration of 287.99 sec with a throughput of 1,691.64 GB/h, providing stability during the backup process without replication overhead, with its best performance observed on the 6-core and 4 GB RAM configurations.

FCI + AG had a longer average duration of 323.70 sec and a throughput of 1,587.14 GB/h but offered additional data protection through integrated failover and replication, showing optimal performance with 6 cores and 6 GB RAM. The AG configuration is ideal for efficiency and speed, while FCI + AG provides a balance between speed and data reliability.

Table 3: Comparison of full backup

No.	Spec (Proc & Memory)	FCI		AG		FCI + AG	
		Duration (s)	Throughput (GB/hr)	Duration (s)	Throughput (GB/hr)	Duration (s)	Throughput (GB/hr)
1	4 Cores, 4 GB	308.40	1,590.30	376.57	1,350.81	415.00	1,214.54
2	4 Cores, 6 GB	289.83	1,662.62	327.60	1,511.23	377.70	1,292.42
3	4 Cores, 8 GB	297.80	1,630.41	365.67	1,315.24	411.60	1,199.72
4	6 Cores, 4 GB	274.23	1,766.85	302.40	1,736.55	342.23	1,439.60
5	6 Cores, 6 GB	307.57	1,575.49	289.83	1,722.31	270.47	1,835.25
6	6 Cores, 8 GB	274.30	1,775.82	229.20	2,186.66	276.03	1,774.45
7	8 Cores, 4 GB	277.13	1,757.20	245.87	2,067.69	263.73	1,899.55
8	8 Cores, 6 GB	277.90	1,767.72	237.17	2,124.15	294.70	1,761.52
9	8 Cores, 8 GB	284.73	1,698.38	232.87	2,100.64	261.83	1,867.21
	Average	287.99	1,691.64	289.69	1,790.59	323.70	1,587.14

DOI: 10.3844/jcssp.2025.1332.1342

Differential Backup

Table (4) presents a comparison of duration and throughput for the differential backup process using FCI, AG, and FCI + AG configurations across various hardware specifications. FCI + AG achieved the best overall performance, with an average throughput of 7.33 GB/h and the highest throughput of 10.59 GB/h recorded on the 6-core and 8 GB RAM configurations. AG demonstrated good data transfer efficiency, with an average throughput of 5.00 GB/h and an average duration of 29.79 se, showing optimal performance on the 4-core and 4 GB RAM configuration, which achieved a throughput of 6.25 GB/h. FCI had the fastest average duration of 27.50 sec with an average throughput of 3.02 GB/h, offering high speed without additional replication overhead, with its best performance observed on the 6-core and 6 GB RAM configurations. With the highest throughput and additional data protection through integrated failover and replication, FCI + AG emerges as the best option for backup needs that prioritize efficiency and data reliability.

Table 4: Comparison of differential backup

No.	Spec (Proc & Memory)	FCI		AG		FCI + AG	
		Duration (s)	Throughput (GB/hr)	Duration (s)	Throughput (GB/hr)	Duration (s)	Throughput (GB/hr)
1	4 Cores, 4 GB	27.03	3.05	29.40	6.25	30.03	7.22
2	4 Cores, 6 GB	27.63	3.03	31.73	5.64	31.20	6.48
3	4 Cores, 8 GB	27.27	2.85	28.63	5.86	29.03	5.91
4	6 Cores, 4 GB	27.47	3.13	30.90	5.43	29.87	4.66
5	6 Cores, 6 GB	26.37	3.10	27.73	5.33	32.70	3.49
6	6 Cores, 8 GB	28.70	2.83	28.70	2.83	29.87	10.59
7	8 Cores, 4 GB	27.93	3.03	30.33	4.84	31.50	9.94
8	8 Cores, 6 GB	27.80	3.06	31.13	4.52	28.60	9.22
9	8 Cores, 8 GB	27.33	3.06	29.57	4.32	28.40	8.47
	Average	27.50	3.02	29.79	5.00	30.13	7.33

Table 5: Comparison of failover-failback performance

No.	Spec (Proc & Memory)	FCI			AG			FCI + AG		
		Auto	Manual	Failback	Auto	Manual	Failback	Auto	Manual	Failback
1	4 Cores, 4 GB	16.06	14.66	14.62	118.61	23.44	29.29	25.34	24.42	24.23
2	4 Cores, 6 GB	15.35	14.57	14.71	56.65	24.07	28.89	24.39	24.10	24.03
3	4 Cores, 8 GB	15.34	14.64	14.57	129.49	24.39	21.69	25.82	24.11	24.54
4	6 Cores, 4 GB	15.65	14.93	14.49	62.81	25.37	31.11	25.68	28.28	32.79
5	6 Cores, 6 GB	15.64	14.79	14.75	43.27	29.80	28.91	24.67	24.35	24.35
6	6 Cores, 8 GB	15.39	14.55	14.58	45.74	29.07	22.87	23.95	23.95	23.95
7	8 Cores, 4 GB	15.61	14.66	14.57	71.94	27.31	24.36	24.15	23.86	23.86
8	8 Cores, 6 GB	15.47	14.76	14.68	123.89	28.22	34.88	23.99	23.76	23.76
9	8 Cores, 8 GB	15.04	14.78	14.74	46.17	31.08	25.14	25.14	23.73	23.73
Average	15.51	14.70	14.63	77.62	26.97	27.46	24.76	25.03	25.03	

Failover-Failback Performance

Table (5) compares the failover-failback performance across FCI, AG, and FCI + AG configurations for automatic, manual, and failback modes. The FCI configuration recorded the fastest average times for all modes, with automatic failover at 15.51 sec, manual at 14.70 sec, and failback at 14.63 sec, attributed to its simple failover mechanism based on shared storage.

AG exhibited slower times for automatic failover (77.62 sec) and manual failover (26.97 sec) due to the synchronization process between replicas, with the longest failback time (27.46 sec) resulting from the need for full synchronization. FCI + AG achieved faster failover times compared to AG, with an average automatic failover time of 24.76 sec, manual failover at 25.03 sec, and failback at 25.03 sec, balancing FCI's failover speed with AG's synchronization reliability. The

combination of FCI + AG offers significant advantages by providing competitive failover times while ensuring high data protection, making it an ideal solution for environments that require a balance between operational speed and reliability.

The significant difference in automatic failover times across configurations can be attributed to multiple factors. In FCI, failover is nearly instantaneous because it relies on shared storage, eliminating the need for database synchronization. In contrast, AG experiences delays due to transaction log replication overhead and network latency. Since AG continuously synchronizes database changes between replicas, failover time is impacted by the volume of pending transactions at the moment of failure.

High transactional loads increase the time required to confirm consistency before promoting a secondary replica to primary status. Additionally, network overhead plays a crucial role, as AG failover involves log shipping and acknowledgment messages between nodes, leading to longer recovery times. FCI + AG reduces these delays by leveraging FCI's rapid instance-level failover while maintaining AG's database-level synchronization, striking a balance between speed and data consistency. However, the effectiveness of this approach still depends on storage performance and network bandwidth, which influence how quickly the system can resynchronize data post-failover.

Availability Percentage

Based on the 30-day monitoring results, the Mean Time Between Failures (MTBF) is calculated as 43,200 min (30 days × 24 h ×60 min). The *Mean Time to Recovery* (MTTR) is derived from the average automatic failover duration for FCI, AG, and FCI + AG, as shown in Table (5).

FCI

The average failover duration is 15.51 sec (converted to 0.25 min).

MTTR = 7.5 min (30-day average × 0.25 min). Availability percentage = 99.98%.

AG

The average failover duration is 77.62 sec (converted to 1.29 min).

MTTR = 38.7 min (30-day average × 1.29 min).

Availability percentage = 99.91%.

FCI + AG

The average failover duration is 24.92 sec (converted to 0.41 min).

MTTR = 12.3 min (30-day average × 0.41 min).

Availability percentage = 99.97%.

Table 6: Comparison of availability percentage

Feature	Avg Auto Failover (Minute)	MTBF (Minute)	MTTR (Minute)	Availability (%)
FCI	0.25	43,200	7.5	99.98
AG	1.29	43,200	38.7	99.91
FCI+AG	0.41	43,200	12.3	99.97

Table (6) shows that the FCI + AG configuration is the optimal choice for system availability. With an average automatic failover time of 0.41 min and an MTTR of 12.3 min, FCI + AG offers a balance between recovery speed and data synchronization reliability. Although its availability (99.97%) is slightly lower than that of FCI (99.98%), this configuration provides better data protection through AG replication, which FCI lacks. By combining the fast failover capabilities of FCI with the data protection of AG, FCI + AG emerges as an ideal solution for environments requiring minimal downtime and high reliability.

Conclusion

1. This study successfully combines the features of SQL Always On Failover Cluster Instance (FCI) and Availability Groups (AG) to increase database availability.
2. Based on the server performance testing over 60 minutes, the combination of FCI + AG demonstrates a balanced resource utilization, with CPU usage at 1.5391%, lower disk usage than AG at 0.0825%, and the lowest memory usage among the three configurations at 23.1686%. This combination can be an appropriate option for achieving both resource efficiency and improved performance.
3. Based on the full backup testing results, the use of AG with a 6-core processor and 8 GB RAM configuration achieved the fastest duration of 229.20 sec, with a throughput of 2,186.66 GB/h. Meanwhile, for differential backup, the combination of FCI + AG with the same 6-core processor and 8 GB RAM configuration recorded the fastest duration of 29.87 sec, with a throughput of 10.59 GB/h
4. Based on the availability percentage calculation over 30 days, the combination of FCI + AG achieved a database availability rate of 99.97%. This combination effectively leverages the fast failover capabilities of FCI and the scalability of AG.

Future Scope

In the current study, there are several limitations due to constraints in time, budget, and the authors'

perspectives. Many aspects can be further explored and developed in greater depth. Recommendations for future development include:

1. The combination of SQL Always On FCI and AG involves a high level of complexity. During the implementation process, it is crucial to clearly separate the roles of FCI and AG, perform routine system testing, and develop comprehensive planning and documentation. This approach is essential to ensure operational continuity and minimize the risk of system failures, thereby enabling the solution to operate optimally.
2. Conducting a comparative study between SQL Always-On and other high availability solutions, such as Oracle Data Guard or MariaDB Galera Cluster, to evaluate their strengths and weaknesses across various scenarios

Acknowledgment

The authors would like to express their gratitude to the Department of Computer Science, Binus Graduate Program, Bina Nusantara University, for providing the resources and support required to carry out this research. The authors also acknowledge the valuable feedback and encouragement received from peers and colleagues throughout the development of this manuscript.

Funding Information

This study was made possible through the generous funding provided by Bina Nusantara University. The authors express their heartfelt gratitude for the financial support from the University's Research Fund, which was instrumental in facilitating the research and enabling the dissemination of the findings presented in this journal paper.

Author's Contributions

Faisal Rahman: Carrying out all the research, performing the analysis, and writing the paper.

Benfano Soewito: Supervising the research and reviewed the paper.

Ethics

This manuscript represents original work by the authors and has not been published elsewhere. The authors have thoroughly reviewed and approved the content, confirming its accuracy and adherence to academic standards. The research and publication processes were conducted with a strong commitment to integrity and ethical principles. No ethical issues or conflicts of interest arose during this study. Furthermore, we adhered to the ethical guidelines established by Bina Nusantara University to ensure responsible conduct throughout the research.

References

Carter, P. (2020). *SQL Server 2019 Always On*.

Chen, Y.-J., & Tsai, H. (2019). Implementation and verification of high data availability on database. *Transactions on Networks and Communications*, *7*(5), 1-12. https://doi.org/10.14738/tnc.75.7292

Cisco. (2024). What Is High Availability? *Cisco*. https://www.cisco.com/c/en/us/solutions/hybrid-work/what-is-high-availability.html

Khan, M. A., & Sabri, M. S. (2021). An Analysisof Disaster Recovery with SQL Server Always On. *International Journal of Management, IT & Engineering*, *11*(4), 17-26.

Kit, N. K. K., & Aibin, M. (2023). Study on High Availability and Fault Tolerance. *2023 International Conference on Computing, Networking and Communications (ICNC)*, 72-82. https://doi.org/10.1109/icnc57223.2023.10074557

Lawrence, A., & Simon, L. (2023). Annual outages analysis 2023. *Uptime Institute*. https://datacenter.uptimeinstitute.com/rs/711-RIA-145/images/AnnualOutageAnalysis2023.03092023.pdf

Parui, U., & Sanil, V. (2016a). *Availability Groups in Microsoft Azure*. 283-307. https://doi.org/10.1007/978-1-4842-2071-9_18

Parui, U., & Sanil, V. (2016b). *Introduction to Windows Server Failover Clustering*. 45-52. https://doi.org/10.1007/978-1-4842-2071-9_5

Scotti, A., Hannum, M., Ponomarenko, M., Hogea, D., Sikarwar, A., Khullar, M., Zaimi, A., Leddy, J., Zhang, R., Angius, F., & Deng, L. (2016). Comdb2 Bloomberg's Highly Available Relational Database System. *Proceedings of the VLDB Endowment*, *9*(13), 1377-1388. https://doi.org/10.14778/3007263.3007275

Shrestha, R., & Tandel, T. (2023). An Evaluation Method and Comparison of Modern Cluster-Based Highly Available Database Solutions. *Proceedings of the 13th International Conference on Cloud Computing and Services Science*, 131-138. https://doi.org/10.5220/0011714400003488

Zamanian, E., Yu, X., Stonebraker, M., & Kraska, T. (2019). Rethinking database high availability with RDMA networks. *Proceedings of the VLDB Endowment*, *12*(11), 1637-1650. https://doi.org/10.14778/3342263.3342639

Zhang, Z., Tian, Y., Li, L., & Wang, Y. (2021). High Availability Services of Client in Large-scale Cluster System. *Journal of Physics: Conference Series*, *1792*(1), 012068. https://doi.org/10.1088/1742-6596/1792/1/012068