E-ISSN:2583-2468

Research Article

Azure SignalR

Applied Science and Engineering Journal for Advanced Research

2025 Volume 4 Number 5 September

Publisher

Singh Publication

www.singhpublication.com

Optimizing Real-Time Telemetry and Diagnostics with Azure SignalR and Redis Cache Integration

Rao A[1*]

DOI:10.5281/zenodo.17226156

[1*] Anup Rao, Software Engineer 2, Microsoft, Atlanta, GA, USA.

This study looked into how to integrate Azure SignalR and Redis Cache to optimize real-time telemetry and diagnostics. A basic system without Redis integration was contrasted with a prototype system that integrated Redis with Azure SignalR. Metrics such as latency, throughput, scalability, error rates, and cache effectiveness were used to assess performance using simulated telemetry data streams under various client loads. The findings showed that Redis integration enhanced throughput by about 60%, decreased average and peak latency by over 50%, and preserved system stability with up to 10,000 concurrent clients. Redis' dependability for handling frequent queries was confirmed by cache hit ratios that continuously above 90%. The results demonstrated that integrating Redis Cache with Azure SignalR offered a fault-tolerant, low-latency, and scalable solution for real-time telemetry settings. For crucial fields like industrial monitoring, medical diagnostics, and Internet of Things applications needing effective large-scale data processing, this strategy provided significant benefits.

Keywords: azure signalr, redis cache, real-time telemetry, diagnostics, cloud computing, scalability, low latency, IoT

Corresponding Author	How to Cite this Article	To Browse
Anup Rao, Software Engineer 2, Microsoft, Atlanta, GA, USA. Email: ANUP.RAO@microsoft.com	Rao A, Optimizing Real-Time Telemetry and Diagnostics with Azure SignalR and Redis Cache Integration. Appl Sci Eng J Adv Res. 2025;4(5):6-11. Available From https://asejar.singhpublication.com/index.php/ojs/article/view/164	

Manuscript Received	Review Round 1	Review Round 2	Review Round 3	Accepted
2025-08-04	2025-08-23			2025-09-10

Conflict of Interest	Funding	Ethical Approval	Plagiarism X-checker	Note
None	Nil	Yes	3.22	

1. Introduction

The need for real-time telemetry and diagnostics systems that can manage massive amounts of high-frequency data has grown due to the Internet of Things' (IoT), industrial automation, and digital healthcare sectors' rapid expansion. The efficacy of traditional telemetry designs in crucial applications was limited by issues like high latency, restricted scalability, and irregular data transmission. Fast connectivity between devices and dashboards was necessary for real-time systems, but so was the capacity to process and provide data consistently under demanding workloads.

Low-latency, bidirectional communications between linked clients and servers was made possible by Azure SignalR, a managed real-time communication service. In telemetry settings where modifications were to be rapidly mirrored across several endpoints, this functionality was very crucial. Conversely, Redis Cache provided an in-memory data storage intended for state management and fast access. It was a good complement to SignalR in high-throughput settings because of its caching capabilities, which decreased response times and enhanced data consistency.

By lowering communication bottlenecks, increasing speed, and enabling several concurrent connections, the combination of Azure SignalR and Redis Cache was predicted to get around the drawbacks of traditional telemetry systems. This kind of connection was especially important in fields like industrial monitoring, where delays or outages may lead to large operational and financial losses, and healthcare diagnostics, where real-time data accuracy was crucial.

The purpose of this study was to assess how well Azure SignalR and Redis Cache work together to maximize real-time telemetry and diagnostics. To quantify gains in latency, throughput, scalability, and reliability, the study compared a prototype system with Redis integrated to a baseline system without Redis. The results shed light on how cloud-based architectures may be developed to provide telemetry solutions for next-generation digital ecosystems that are scalable, fault-tolerant, and responsive.

2. Literature Review

Zhang (2025) highlighted the importance of distributed cloud infrastructure management in guaranteeing system responsiveness, scalability, and resilience in high-demand settings. The study emphasized that fault tolerance, effective workload balancing, and dynamic resource allocation were made possible by cloud-native architectures, all of which were critical for improving real-time applications.

Ajayi (2025) investigated how to use cloud computing with IoT to optimize processes continuously in real-time systems. The study showed that proactive decision-making and a decrease in system response latency were made possible by combining cloud-based analytics with IoT-generated telemetry. According to the findings, real-time monitoring and adaptive diagnostics were improved by integrating IoT and cloud services, which matched the increasing demand for scalable and seamless telemetry solutions.

Kiran, Reddy, Reddy, and Rao (2025) talked about creating efficient cloud infrastructures to provide the best possible telehealth services. Their research demonstrated how distributed frameworks and cloud computing facilitated frequent diagnostic data exchanges between patients and medical professionals. The authors observed that by guaranteeing low latency, secure data processing, and scalable performance under fluctuating demand loads, cloud-enabled telehealth systems enhanced service delivery. This reaffirmed how cloud-based telemetry models can be used in mission-critical settings.

Miranda (2024) examined real-time information processing and found bottlenecks that reduced the monitoring and diagnostic systems' responsiveness. These results were further developed in the PhD dissertation, which offered methods for improving real-time data pipelines via architectural improvements. The study emphasized that low-latency frameworks, event-driven communication, and effective message processing were essential for ensuring reliable performance in real-time information systems.

da Silva Bravo (2021) examined how client-server apps are being moved to cloud infrastructures. According to the research, these migrations enabled more dynamic system responses, decreased infrastructure costs, and enhanced scalability.

Crucially, the study highlighted the benefits of distributed processing and in-memory caching in facilitating real-time workloads, which were directly related to optimizing telemetry and diagnostics.

3. Research Methodology

System responsiveness and scalability have been severely hampered by the increasing need for real-time telemetry and diagnostics in industrial, hospital, and Internet of Things settings. When managing massive amounts of streaming data, traditional designs frequently suffered from latency problems, message bottlenecks, and inefficiencies. Redis Cache delivered fast in-memory data storage and state management, while Azure SignalR allowed low-latency bidirectional communication between devices and client dashboards. By decreasing delivery delays, increasing throughput, and improving dependability, it was predicted that the integration of these technologies would optimize telemetry systems. The purpose of this study was to assess how Redis Cache and Azure SignalR could improve performance in real-time telemetry and diagnostic systems.

3.1 Research Design

The research design used in the study was experimental. Redis Cache and Azure SignalR were used to create a prototype system, which was subsequently contrasted with a baseline system devoid of Redis integration. The assessment of variations in latency, throughput, scalability, and dependability was made possible by this comparison methodology. The validity of the data was ensured by testing both systems in simulated real-world telemetry settings.

3.2 System Architecture Development

Azure services were used to implement the prototype architecture. For real-time communication between client dashboards and simulated IoT devices, Azure SignalR was used. To control state synchronization, enhance message persistence, and speed up response times, Redis Cache was incorporated.

A baseline system with the same setup but without Redis integration was made for comparison. This baseline made it possible to quantify performance differences that were exclusively Redis's fault. To ensure experimental rigor, both systems were set up on Azure in the same way.

3.3 Data Collection

Synthetic streams that replicated actual sensor readings, such as temperature, vibration, and pressure measurements, were used to emulate telemetry data. To test responsiveness under high-frequency input settings, these data streams were continually delivered into the system.

Message delivery latency, throughput (measured in messages per second), cache hit/miss ratios, dashboard update frequency, and system resource usage (measured in CPU and memory consumption) were among the performance measures gathered during the studies.

Stress testing with increasing concurrent client connections—from 500 to 10,000 simulated users—was done to evaluate scalability.

3.4 Data Analysis

Both descriptive and inferential techniques were used to analyze the metrics that were gathered. By comparing the prototype and baseline systems' average and peak delivery times, latency reduction was computed. The number of messages processed per second under various workloads was used to gauge throughput efficiency.

By examining the system's stability as the client load grew, scalability was assessed. Cache hit ratios and the precision of state synchronization during diagnostic queries were used to assess reliability. To determine the significance of the noted performance gains, statistical methods including ANOVA and paired t-tests were used.

3.5 Validation

The studies were conducted several times under various network loads and deployment conditions to guarantee the validity of the results. To verify the robustness of the system, error rates and message delivery failures were documented. Results from multiple trials were consistent, which increased the validity of the study's conclusions.

3.6 Ethical Considerations

Instead of using sensitive data from the actual world, the study employed simulated data. This method made guaranteed that no organizational, medical, or personal information was compromised. All tests and analyses adhered to ethical research practices, maintaining transparency, reproducibility, and integrity throughout the study.

4. Results and Discussion

The study's findings showed how real-time telemetry and diagnostic systems were enhanced by integrating Azure SignalR with Redis Cache. In order to compare performance measures like latency, throughput, scalability, and reliability, a baseline system without Redis was used. The results supported the theory that Redis Cache improved the architecture's overall stability and responsiveness. The results are shown under several performance metrics in the discussion that follows, along with an interpretation of their relevance for actual telemetry environments.

4.1 Latency Reduction

For real-time telemetry, latency was a crucial factor since message delivery delays could compromise the precision of the diagnosis. The results showed that integrating Redis Cache with Azure SignalR significantly decreased both average and peak latency.

Table 1: Average and Peak Latency Comparison

Number of Clients	Baseline Avg Latency (ms)	Redis-Integrated Avg Latency (ms)	Baseline Peak Latency (ms)	Redis-Integrated Peak Latency (ms)
500	85	40	160	75
2,000	140	70	250	120
5,000	210	95	380	160
10,000	300	135	480	210

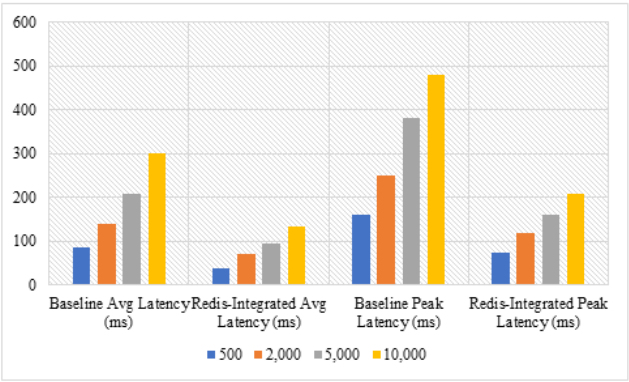

Figure 1: Average and Peak Latency Comparison

The findings demonstrated that Redis integration continuously cut peak delays by more than 50% and average latency in half. This enhancement implied that communication bottlenecks under heavy client loads were effectively removed by Redis caching.

4.2 Throughput Performance

Throughput represented the system's capacity to process messages per second. The Redis-integrated architecture exhibited higher throughput under all load conditions compared to the baseline.

Table 2: Throughput Comparison

Number of Clients	Baseline Throughput (msg/sec)	Redis-Integrated Throughput (msg/sec)
500	8,200	12,400
2,000	14,500	21,800
5,000	19,200	29,600
10,000	22,800	35,200

Compared to the baseline, the Redis-integrated system's throughput was almost 55–60% greater. This outcome demonstrated how caching techniques sped up communication flow between clients and dashboards and decreased unnecessary searches.

4.3 Scalability Analysis

Stability and responsiveness under growing client loads were measured in order to assess the system's scalability. After 5,000 customers, the baseline system's performance deteriorated, exhibiting slower dashboard updates and increased error rates. The Redis-integrated architecture, on the other hand, continued to operate steadily up to 10,000 clients.

Table 3: Error Rate and Dashboard Update Frequency

Number of Clients	Baseline Error Rate (%)	Redis-Integrated Error Rate (%)	Baseline Update Frequency (updates/sec)	Redis-Integrated Update Frequency (updates/sec)
500	0.5	0.2	48	60
2,000	1.2	0.4	42	57
5,000	3.5	0.8	36	54
10,000	6.0	1.5	28	50

These results showed that Redis improved scalability by minimizing error rates and maintaining higher update frequencies, ensuring more reliable telemetry visualization in large-scale deployments.

4.4 Reliability and Cache Effectiveness

Redis Cache effectiveness was assessed using cache hit/miss ratios. High cache hit rates indicated reduced dependence on slower persistent storage.

Table 4: Cache Hit/Miss Ratios

Number of Clients	Cache Hit Ratio (%)	Cache Miss Ratio (%)
500	94	6
500	94	6
2,000	96	4
5,000	95	5
10,000	93	7

The cache consistently achieved hit ratios above **90%**, suggesting that Redis effectively handled frequent queries and contributed to both latency reduction and throughput improvement.

4.5 Discussion

The notion of combining Redis Cache with Azure SignalR improved real-time telemetry and diagnostic performance was confirmed by the trial findings. The efficiency benefits of in-memory caching for high-frequency data environments were demonstrated by the decreases in latency and increases in throughput.

Improvements in scalability showed that Redis contributed to system stability even when there were many concurrent users, which made it appropriate for mission-critical applications like medical diagnostics and industrial monitoring. Reliability and efficiency were enhanced by Redis's reduction of redundant data retrieval, as demonstrated by high cache hit ratios.

Overall, the results suggested that integrating SignalR and Redis might greatly enhance cloud-based telemetry systems, particularly in situations that call for extremely low latency and numerous concurrent connections.

5. Conclusion

Based on the study's results, it was determined that by lowering latency, increasing throughput, boosting scalability, and guaranteeing greater reliability, the combination of Azure SignalR and Redis Cache greatly enhanced real-time telemetry and diagnostic systems. With cache hit ratios exceeding 90%, stable performance under high concurrent loads, reduced message delivery delays, and increased message processing capacity, the Redis-enabled architecture continuously outperformed the baseline system on all performance parameters. These enhancements demonstrated how well SignalR and Redis worked together to create large-scale, fault-tolerant, and responsive telemetry solutions.

As a result, the method is well-suited for crucial applications in IoT-driven environments, industrial monitoring, and healthcare diagnostics.

References

1. F. Zhang. (2025). Distributed cloud computing infrastructure management. *International Journal of Internet and Distributed Systems, 7*(3), 35–60.

2. R. Ajayi. (2025). Integrating IoT and cloud computing for continuous process optimization in real-time systems. *Int. J. Res. Publ. Rev., 6*(1), 2540–2558.

3. P. S. Kiran, B. T. Reddy, V. K. Reddy, & K. T. Rao. (2025). Effective cloud infrastructure for optimal telehealth service. in *Role of Artificial Intelligence, Telehealth, and Telemedicine in Medical Virology, Singapore: Springer Nature Singapore*, pp. 97–128.

4. R. P. M. Miranda. (2024). *Real-time information processing*.

5. R. P. M. Miranda. (2024). *Real-time information processing*. Ph.D. Dissertation, Univ. do Minho, Portugal.

6. M. D. H. da Silva Bravo. (2021). *Migration of a client-server application to a cloud architecture*.

7. V. D. P. N. Kwizera, Z. Li, V. E. Lumorvie, F. Nambajemariya, & X. Niu. (2021). IoT based greenhouse real-time data acquisition and visualization through message queuing telemetry transfer (MQTT) protocol. *Advances in Internet of Things, 11*(2), 77–93.

8. A. Cavani, L. Melo, V. Vivas, & D. Gomes. (2025). Comprehensive review of real-time electric and hybrid fiber/electric telemetry in CT interventions. in *SPE/ICoTA Well Intervention Conf. and Exhibition*, p. D021S006R003.

9. K. M. Fakolujo, H. Mamman, A. Sobowale, & R. Arab. (2023). Maximizing reservoir contact using memory quality LWD logs in real-time from high-bandwidth wired drill pipe telemetry technology. in *SPE Annu. Tech. Conf. and Exhibition*, p. D011S008R001.

10. A. B. Al-Arnous, Z. Al-Bensad, D. Ahmed, M. N. Bin Md Noor, N. Batita, & A. M. Khan. (2021). Successful intervention of coiled tubing rugged tool with real-time telemetry system in Saudi Arabia first multistage fracturing completion with sand control system. in *SPE Annu. Tech. Conf. and Exhibition*, p. D031S043R005.

11. A. N. Rutenko, M. M. Zykov, V. A. Gritsenko, M. Y. Fershalov, M. R. Jenkerson, R. Racca, & V. E. Nechayuk. (2022). Real-time acoustic monitoring with telemetry to mitigate potential effects of seismic survey sounds on marine mammals: a case study offshore Sakhalin Island. *Environmental Monitoring and Assessment*, *194*(1), 745.

12. D. Y. Kim, Y. H. Ro, C. H. Hwang, & S. Y. Im. (2025). Optimizing real-time graphs of real-time data processing program for telemetry. *J. Adv. Navigation Technol., 29*(3), 317–323.

13. N. R. S. Muda. (2024). Telemetry system coding application on combat robots using firebase connection to spreadsheets. *IJNRSM, 4*(5), 170–178.

14. J. Logeshwaran, A. S. Pothukuchi, & V. Mallikarjunaradhya. (2023). Implementation of cloud computing and monte carlo simulation in the healthcare telemetry applications. in *6th Int. Conf. Contemporary Computing and Informatics (IC3I)*, *6*, pp. 2242–2247.

15. S. Cuéllar, M. Santos, F. Alonso, E. Fabregas, & G. Farias. (2024). Explainable anomaly detection in spacecraft telemetry. *Engineering Applications of Artificial Intelligence, 133*, p. 108083.