In-Memory Database Architecture for High-Speed Computing

Binoy Kurikaparambil Revi

Independent Researcher, USA
Email: binoyrevi@live.com

Abstract

The high-speed computing has transformed dramatically, largely driven by remarkable advancements in graphics processing units (GPUs) and algorithms. With these technological strides, there has been a parallel surge in the volume of data that must be efficiently managed during high-speed computing operations. This increase in data complexity poses challenges, yet it is met with innovative solutions. One of the most noteworthy developments is the rise of affordable, high-capacity memory chips, which have fundamentally altered how data is managed in memory. As a result, software architects are now empowered to design sophisticated high-speed computing architectures that leverage various in-memory techniques. Among these techniques are in-memory databases, which allow rapid data retrieval and manipulation along with publish-and-subscribe functionality, which ensures that data is immediately accessible to applications as soon as it is generated. This technology enables high-speed applications to operate seamlessly and efficiently, making real-time data available and enhancing overall computational capability. Many industries and domains can benefit from using in-memory database techniques to process large volumes of data. For example, high-speed control systems in the power and automotive industries can take advantage of these techniques. Fields such as artificial intelligence and data mining can also leverage in-memory databases for improved performance.

Keywords: High-Speed Computing, Data Processing, In-Memory Database, Redis

Introduction:

In-memory databases were initially used mainly in web applications, primarily for managing user sessions and caching small data sets for faster retrieval. These databases operated in the server's volatile memory, enabling quick access times compared to traditional disk-based storage solutions. Over the years, significant advancements in hardware technology have reduced the cost of Random Access Memory (RAM) while improving its speed and capacity. As a result of these improvements, in-memory databases have expanded beyond their original applications. They are now integrated into high-speed computing architectures to support a broader range of scenarios, such as high-speed time-series data management, publish-and-subscribe services, complex event processing, and large-scale data processing tasks. The ability to handle larger datasets while maintaining high performance makes in-memory databases crucial in modern data-intensive applications.

Related Work:

Abdullah Talha Kabakus and Resul Kara conducted performance evaluations of in-memory databases. They compared the performance of various in-memory databases, such as Redis and H2, and traditional databases like MongoDB and Apache Cassandra. The results of their experiments were quite interesting, as they

indicated that not only does pure in-memory play a crucial role but the technologies and algorithms used to build the database are also significant. Based on the evaluation by Abdullah Talha Kabakus and Resul Kara, Redis emerged as the top performer, utilizing memory very efficiently.The figure 1 is sourced from the evaluation by Abdullah Talha Kabakus and Resul Kara.

Database	Number of records			
	1,000	10,000	100,000	1,000,000
Redis	34	214	1666	14.638
MongoDB	904	3482	26.030	253.898
Memcached	23	100	276	2813
Cassandra	1202	4487	15.482	140.842
H2	147	475	1648	7394

Figure 1: The calculated time to write key-value pairs (ms)[1] from Abdullah Talha Kabakus and Resul Kara conducted a performance evaluation of in-memory databases

In their insightful paper titled "In-memory Databases: Challenges and Opportunities From Software and Hardware Perspectives," Kian-Lee Tan, Qingchao Cai, Beng Chin Ooi, Weng-Fai Wong, Chang Yao, and Hao Zhang thoroughly investigate the multifaceted challenges and promising opportunities that In-Memory databases present. The authors emphasize the importance of approaching the topic from both software and hardware angles, exploring innovative strategies that could mitigate existing limitations while enhancing performance. Their comprehensive analysis leads to a significant conclusion: to fully harness the capabilities of In-Memory databases, an integrated approach that combines advancements in both hardware and software is extremely important. This assertion is further reinforced by the rigorous experiments conducted by Abdullah Talha Kabakus and Resul Kara, which meticulously assess the performance metrics of In-Memory databases, providing a deeper understanding of their operational efficiencies and potential improvements.

Problem Description:

In the embedded systems and control systems industry, there are numerous use cases that require high-speed data processing to achieve accurate control or the timely execution of tasks. Often, these data processing tasks involve large amounts of data to perform precise computations. However, when memory is limited or

data is not well managed, applications tend to use only the minimum amount of data necessary for computations. This represents the first problem that must be addressed. To address this critical issue, there is a pressing need for advancements in data management techniques that optimize memory usage. By improving how data is organized, stored, and accessed, it becomes possible to incorporate larger datasets into the processing framework. This, in turn, enhances the granularity and accuracy of computations, enabling systems to make better-informed decisions and operate more effectively. As a result, tackling the challenges associated with data management is essential for advancing the capabilities of embedded and control systems in handling complex tasks efficiently.

The second issue at hand is the capability to process vast volumes of data at rapid speeds. One common approach to managing extensive datasets is through the implementation of databases. However, conventional on-disk or hybrid databases often fall short when faced with the rigorous demands of high-speed control systems or the substantial computational loads required in real-time embedded systems. While these traditional database solutions might initially seem adequate to meet the performance needs, as the volume of data necessary for executing essential functions continues to grow, the performance can diminish significantly, leading to a system that may become increasingly unreliable. This challenge highlights the urgent need for in-memory databases, which can provide the required speed and efficiency, along with complementary technologies for persistent storage when necessary.

The third challenge examined in this analysis revolves around the methods employed for capturing data and making it accessible to the core data processing engine. In numerous scenarios involving embedded systems and control systems, there is a critical necessity for data to be available for data processing immediately upon receipt by the system. This urgency is even more pronounced within control systems, where the output generated by the control processing unit must be transmitted to the appropriate destination unit without any delay or instantaneously. Consequently, it becomes imperative for embedded and control systems to guarantee instantaneous access to data and timely delivery of outputs. Meeting these stringent requirements is essential for the effective operation of these systems, highlighting the importance of reliable data capture and swift communication pathways.

Handling data series, particularly time series data, poses significant challenges for real-time computations. This is largely due to the continuous accumulation of data over time, each data point accompanied by its corresponding timestamp. The intervals at which this data is collected can vary greatly, spanning from minutes to milliseconds. When it comes to managing such vast amounts of data in a traditional on-disk database, the task can quickly become overwhelming for data processing engines. The data processing unit is most efficient when accessing information stored in memory as it is generated, thereby avoiding the delays and complexities of on-disk or hybrid data operations.

In-Memory Database System Architecture:

The data management and data transfer techniques used in embedded systems or control systems are crucial for their performance and robustness. Utilizing in-memory databases can be essential for achieving this. However, it's important to consider the system configuration. The amount of memory required for the system does not only depend on the tasks of the operating system but also on the volume of data that needs to be stored and processed using the in-memory database.

Based on the research and performance evaluation conducted on various databases by Abdullah Talha Kabakus and Resul Kara[1], it is clear that the Redis database performs exceptionally well. In my further study of the Redis database, I found it fascinating to learn about the features and techniques it provides for efficient data management and transaction handling, which significantly support high-speed computing.

Figure 2 depicts a reference architecture for a typical embedded or control system that collects data from various sources, processes it, and sends the data to both internal and external systems. This architecture

includes modules for different standard communication protocols through the network, along with external systems that function as publishers. Furthermore, the system distributes data to both external and internal systems.

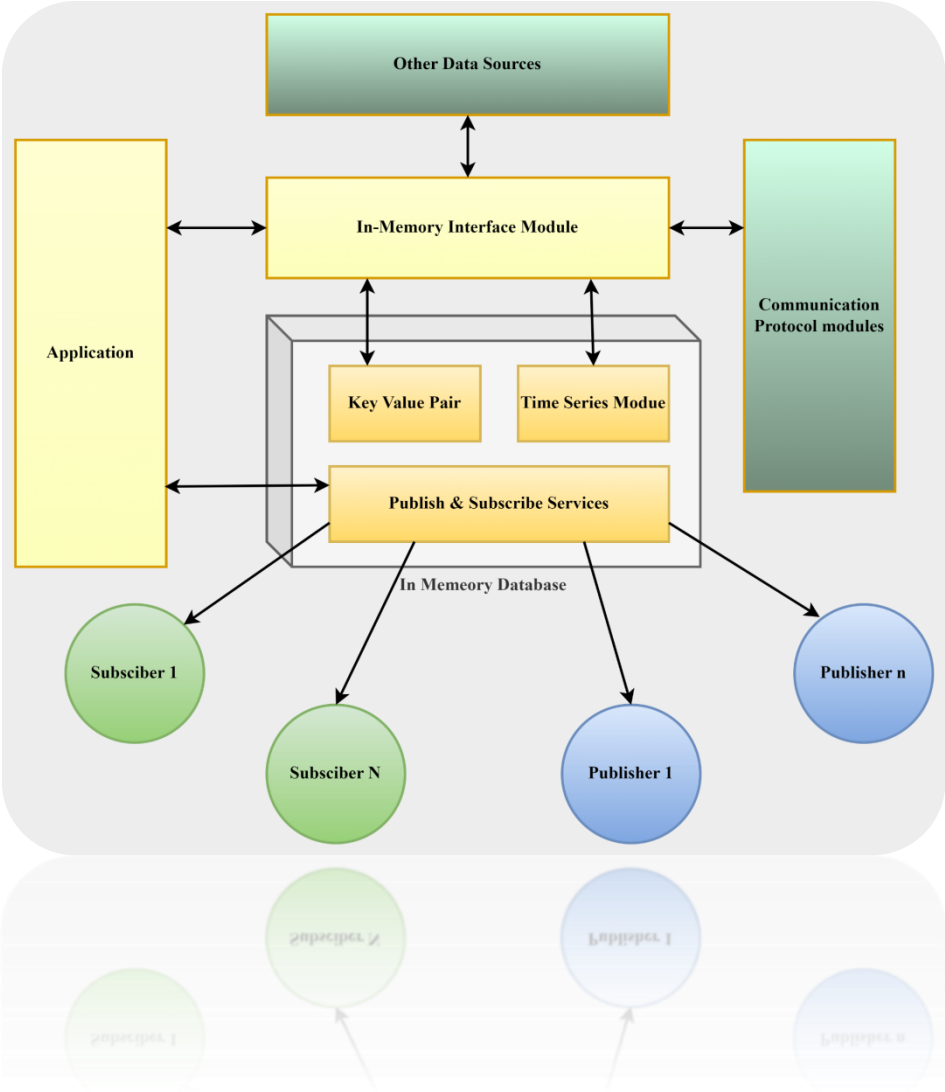

Figure 2: Architecture for a typical embedded or control system using Im-Memory Database

Let's discuss various major components that are included in this architecture.

1. Application: This program serves as the data processing engine and can also be considered a computing engine. It receives data from various sources, including different communication protocols and applications. The data is processed through the In-Memory Interface Module, which acts as an intelligent broker between the data sources and the applications.
2. In-Memory Interface Module: The In-Memory Interface Module is a crucial component of the architecture, significantly impacting the overall performance of the system. It serves as a broker between the application, the In-Memory database, data from network communication modules, and other application sources. The intelligent design of this component is essential, as it determines how incoming data is processed and how that data is made available to the application for further processing. Critical design decisions must be made regarding how the In-Memory Interface module will store data in the In-Memory database. This includes recognizing the data and, if necessary,

making it immediately available to applications. Additionally, handling data series, such as time series data, and defining priority levels are essential considerations.

3. Communication Interface Modules: These modules implement communication protocols and operate as processes within the system. They receive data from the network, decode it, and provide it to the In-Memory Interface Module.
4. Other applications: This is optional and exists only if any other application, mainly third-party applications, runs on the system that can be considered a data source.
5. In_memory Database: In this architecture, we are using Redis as the in-memory database. One of the primary features of Redis is its ability to store data in key-value pairs. Additionally, the Redis Time Series module allows Redis to store time series data in memory, making it an excellent option for data processing applications. However, care must be taken to manage the data retention period to control and optimize memory usage. Redis also provides publish and subscribe services[3], which are valuable additions to its in-memory database capabilities. With these services, external applications can subscribe to the Redis database and instantly receive any updates published by the application or other publishers. It's important to note that the application itself acts as both a publisher and a subscriber in this service[6]. This functionality is extremely useful for implementing message passing and health monitoring requirements, serving as a critical data exchange mechanism.
6. Publishers: [3]The publishers are core applications or external services that can publish data to the Redis database using named channels.
7. Subscribers: [3]The subscribers are core applications, external services or applications, UI, and other data hubs that subscribe to the Redis Pub/Sub method to receive the published messages.

The internal data flow of the system is managed by the In-Memory Interface module, which alleviates the workload from the application, allowing it to focus on core critical tasks.In-Memory Interface module is also the best candidate to monitor the overall health of the system.

Conclusions:

The high-speed computing and data processing presents a variety of challenges, particularly when it comes to effectively handling and managing vast amounts of data in real-time. As the volume of data expands, traditional databases—whether on-disk or hybrid—can become significant bottlenecks, impairing overall system performance. In contrast, in-memory databases and their associated modules serve as vital software development tools, equipping developers with robust APIs to develop applications capable of high speed data processing. By storing data in the system's memory rather than on traditional disk drives, these databases facilitate rapid access to information, which is crucial for time-sensitive applications. Redis, an advanced in-memory database includes publish and subscribe services. This feature allows data to be disseminated to various applications and external services the moment it becomes available, enhancing responsiveness and interactivity. Such capabilities are essential for building event-driven applications that not only demand high performance but also adapt to real-time data changes.

References:

1. Abdullah Talha Kabakus, Resul Kara, A performance evaluation of in-memory databases, Journal of King Saud University - Computer and Information Sciences, Volume 29, Issue 4, 2017, Pages 520-525, ISSN 1319-1578, https://doi.org/10.1016/j.jksuci.2016.06.007.
2. Kian-Lee Tan, Qingchao Cai, Beng Chin Ooi, Weng-Fai Wong, Chang Yao, and Hao Zhang. 2015. In-memory Databases: Challenges and Opportunities From Software and Hardware Perspectives. SIGMOD Rec. 44, 2 (June 2015), 35–40. https://doi.org/10.1145/2814710.2814717
3. Redis Publish-Subscribe Service : https://redis.io/glossary/pub-sub/

4. Stephen Tu, Wenting Zheng, Eddie Kohler, Barbara Liskov, and Samuel Madden. 2013. Speedy transactions in multicore in-memory databases. In Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles (SOSP '13). Association for Computing Machinery, New York, NY, USA, 18–32. https://doi.org/10.1145/2517349.2522713
5. Y. Wang et al., "The Performance Survey of in Memory Database," 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), Melbourne, VIC, Australia, 2015, pp. 815-820, doi: 10.1109/ICPADS.2015.109.
6. Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. ACM Comput. Surv. 35, 2 (June 2003), 114–131. https://doi.org/10.1145/857076.857078
7. M. Hungyo and M. Pandey, "SDN based implementation of publish/subscribe paradigm using OpenFlow multicast," 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), Bangalore, India, 2016, pp. 1-6, doi: 10.1109/ANTS.2016.7947820.
keywords: {Switches;Delays;Protocols;Multicast communication;Servers;Routing;Clustering algorithms;Pub/Sub;SDN;multicasting;OpenFlow;QoS;P2P;API;flow table;delay;throughput},