Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.15, No.1, 2024

Assessing the Dynamics of Data Processing: In-Memory Versus Disk-Based Methods in the Context of Big Data Analytics

Semen M. Levin[1*]

1. Department of Automated Control Systems, Tomsk State University of Control Systems and Radioelectronics, 40 Lenin Ave, Tomsk 634050, Russia

* E-mail of the corresponding author: semen.m.levin@tusur.ru

Abstract

This study embarks on a comprehensive analysis to evaluate the efficiency and scalability of in-memory computing (IMC) compared to traditional disk-based processing for big data analytics. Utilising the "New York City Taxi Trip Duration" dataset from Kaggle, we designed an experiment focusing on three critical analytical tasks: aggregation, sorting, and filtering. Our objective was to quantify the performance improvements offered by IMC, as facilitated by Apache Spark, against conventional SQL queries executed on a disk-based system. The findings reveal that IMC consistently outperforms disk-based processing in execution time, with significant reductions observed across all tasks. Specifically, the aggregation task highlighted the stark contrast in data retrieval speed, demonstrating IMC's superior efficiency with a completion time of 47.3 seconds, compared to 138.7 seconds for disk-based processing. Similar disparities were noted in sorting and filtering tasks, further accentuating IMC's performance advantage. Resource utilisation analysis, focusing on CPU and RAM consumption, indicated higher demands associated with IMC, underscoring the trade-off between enhanced speed and increased resource usage. The investigation provides a nuanced understanding of the practical implications of adopting IMC for big data analytics, especially considering the resource constraints of home computing environments. By juxtaposing theoretical advantages with empirical data, this paper contributes to the ongoing discourse on optimising data processing methodologies in the era of big data, offering insights into the balance between computational efficiency and resource management.

Keywords: in-memory computing, big data analytics, disk-based processing, data processing efficiency, resource utilisation, Apache Spark, data analytics performance

DOI: 10.7176/CEIS/15-1-05
Publication date: April 30th 2024

1. Introduction

The burgeoning volume of data generated by digital technologies has underscored the critical role of big data analytics in contemporary research and industry applications. Among the myriad of challenges big data poses, the efficiency and scalability of data processing stand paramount. This study mainly concerns the comparative efficacy of in-memory computing over traditional disk-based methods in big data analytics (Chen, P. H., & Keville, K., 2022).

Historically, the reliance on disk-based data processing has been a bottleneck in achieving the desired computational speed and scalability due to the mechanical limitations of disk operations. In contrast, in-memory computing, which utilises Random Access Memory (RAM) for data storage and manipulation, offers a promising alternative. This approach can reduce data access time significantly, improving the overall performance of analytical tasks.

Recent studies have increasingly highlighted the advantages of in-memory computing. For instance, Zaharia et al. (2010) introduced Apache Spark, a unified analytics engine for large-scale data processing, which leverages in-memory computation to enhance performance (Lee, J., Choi, J., & Koo, D., 2016). Similarly, Plattner and Zeier (2011) discussed the transformational impact of in-memory data management on business applications, emphasising its capability to perform real-time analytics and transactional processes concurrently (Awan, A. J., Vlassov, V., Brorsson, M., & Ayguadé, E., 2016). Despite these advancements, the scalability of in-memory computing, particularly in handling vast datasets within the constraints of available memory, remains a topic of

ongoing research (Ketu, S., & Agarwal, S., 2015). In parallel, the work of Abubaker (2024) on disk-based indexing techniques for NIR-Trees using polygon overlays presents an interesting contrast, demonstrating the ongoing relevance and optimization potential of disk-based methods in specific contexts of data analytics. This inclusion highlights the broader landscape of data processing technologies, underscoring the importance of diversifying approaches to tackle the multifaceted challenges presented by big data analytics.

This study aims to empirically investigate the efficiency and scalability of in-memory computing for big data analytics. Specifically, the research aims to:

1. Compare the performance of in-memory computing against traditional disk-based processing in executing common analytical operations such as aggregation, sorting, and filtering on a large dataset.
2. Evaluate both approaches' resource utilisation, including CPU and memory consumption, under varying data loads.
3. Examine the impact of data size on the scalability of in-memory computing, identifying thresholds where performance gains diminish or adverse effects emerge.

This investigation is conducted through a series of experiments using a public dataset from Kaggle, "New York City Taxi Trip Duration," encompassing millions of records (Kaggle, n.d.). The choice of dataset and analytical operations are designed to reflect realistic scenarios encountered in big data analytics, providing insights applicable to academic research and industry practice.

In doing so, this study contributes to the existing literature by offering a nuanced understanding of the practical limitations and opportunities of in-memory computing in the context of big data analytics. Furthermore, it provides evidence-based recommendations for practitioners considering the adoption of in-memory technologies to enhance their analytical capabilities.

The significance of this inquiry extends beyond the academic sphere, touching on the operational efficiencies that can transform business practices. In an era where data is dubbed the 'new oil,' the ability to process and analyse large volumes of information rapidly and accurately is a competitive edge that many organisations strive for. Thus, the findings of this study are poised to offer tangible benefits to a wide range of stakeholders, including data scientists, IT professionals, and business strategists who seek to optimise their data analytics workflows.

Moreover, the methodological rigour applied in conducting the experiments ensures the reliability and validity of the results. By utilising a well-documented and publicly available dataset, the study adheres to principles of reproducibility and transparency, which are cornerstone values in scientific research. The choice of analytical operations – aggregation, sorting, and filtering – covers a broad spectrum of standard data processing tasks, thereby ensuring the applicability of the findings across different domains and use cases.

Additionally, the study's focus on resource utilisation sheds light on the often-overlooked aspect of computational efficiency. While much of the existing literature emphasises the speed of data processing, this research also considers the technology's cost-effectiveness in terms of CPU and memory usage. This holistic approach provides a more comprehensive understanding of the trade-offs in selecting an appropriate computing paradigm for big data analytics.

In-memory computing has opened new avenues for processing and analysing large datasets with unprecedented speed and efficiency. However, the scalability of this technology under varying conditions and data loads remains an area ripe for exploration. By addressing this gap in the literature, the present study aims to enrich the body of knowledge on big data analytics and guide future technological advancements in this field. Through a meticulous examination of in-memory computing's performance and resource utilisation, this research clarifies the conditions under which this technology excels and highlights its limitations, thereby paving the way for informed decision-making and innovation in big data processing.

2. Theoretical Foundations of In-Memory Computing

The advent of in-memory computing (IMC) marks a paradigm shift in how data is processed and analysed, especially in the context of big data analytics. By leveraging Random Access Memory (RAM) to store data temporarily, IMC facilitates data access orders of magnitude faster than traditional disk-based storage systems.

This section delves into the core principles of IMC, juxtaposing its strengths and weaknesses against conventional disk-based analytics while highlighting pivotal technologies like Apache Spark that embody the in-memory computing ethos.

2.1 Theoretical Underpinnings of In-Memory Computing

In-memory computing (IMC) significantly differs from traditional, disk-based data processing methodologies (Ma, Y., Du, Y., Du, L., Lin, J., & Wang, Z., 2020). At its core, IMC leverages the system's main memory (RAM) for data storage and processing, contrasting sharply with the conventional reliance on slower, mechanical disk storage. This shift towards memory-centric data management is propelled by the exponential growth in data volumes and the concurrent demand for faster processing speeds to facilitate real-time analytics and decision-making processes.

IMC is predicated on the principle that accessing data stored in RAM is orders of magnitude faster than retrieving the same data from a disk drive. This speed disparity arises from the mechanical latencies inherent to disk storage—such as seek time and rotational latency—absent in solid-state memory operations. By storing data in RAM, IMC systems can achieve microsecond-level access times, significantly enhancing the performance of data-intensive applications.

The architecture of an IMC system is designed to maximize the advantages of rapid data access (Ielmini, D., Lepri, N., Mannocci, P., & Glukhov, A., 2022). Data is partitioned and distributed across the memory of multiple nodes in a clustered or grid environment, enabling parallel processing and fault tolerance. This distribution increases processing speed by leveraging multiple processors simultaneously and ensures data availability through replication strategies that mitigate the risk of data loss due to hardware failures.

The primary advantage of IMC lies in its ability to accelerate the performance of applications that require immediate data access and real-time analytics (Dazzi, M., Sebastian, A., Benini, L., & Eleftheriou, E., 2021). Financial trading platforms, online retail recommendation systems, and real-time fraud detection mechanisms are exemplary use cases that benefit from IMC's speed and agility. Furthermore, IMC facilitates complex event processing (CEP) and high-performance computing (HPC) tasks (Verma, N., Jia, H., Valavi, H., Tang, Y., Ozatay, M., Chen, L.-Y., Zhang, B., & Deaville, P., 2019; Yu, H., Ni, L., & Dinakarrao, S. M. P., 2021), where processing large volumes of data in real-time is critical.

Another significant benefit of IMC is its impact on application simplification (Jia, H., Ozatay, M., Tang, Y., Valavi, H., Pathak, R., Lee, J., & Verma, N., 2021). Traditional disk-based architectures often necessitate complex designs to mitigate storage bottlenecks, including elaborate caching mechanisms and data pre-fetching strategies. IMC, by contrast, allows for more straightforward application architectures by removing these bottlenecks, enabling developers to focus on business logic rather than performance optimization.

Despite its advantages, the adoption of IMC is not without challenges. The volatile nature of RAM implies that data stored in memory can be lost during a power failure or system crash. Consequently, IMC systems must incorporate persistent storage mechanisms and sophisticated data recovery strategies to safeguard against data loss. Moreover, the cost of RAM, although decreasing, remains higher than that of disk storage, presenting financial considerations for organizations contemplating the transition to IMC.

2.2 Limitations and Challenges of In-Memory Computing

The exploration of In-Memory Computing (IMC) challenges and limitations spans several dimensions, including economic implications, scalability issues, data volatility, and technical vulnerabilities. Here are some insights derived from recent research on the topic:

Economic Considerations: The financial burden associated with high RAM costs, compared to traditional disk storage, remains a significant barrier to IMC adoption, particularly for small and medium-sized enterprises (SMEs) with limited IT budgets. Despite the decreasing price of RAM, the necessity for large memory allocations to manage extensive datasets ensures that cost remains a critical concern (Böhm, 2019).

Scalability Challenges: The direct relationship between the size of datasets and required memory capacity underscores scalability as a primary concern. This challenge becomes increasingly pronounced in areas where data accumulates rapidly, such as social media analytics, IoT sensor data aggregation, and real-time financial transaction monitoring (Karimzadeh et al., 2023).

Data Volatility: The inherent volatility of RAM, with data loss upon power or system failure, poses substantial risks to data integrity and availability. Ensuring data persistence demands the implementation of robust data

recovery and persistence mechanisms, adding complexity and potentially diminishing the performance benefits of IMC (Huang & Qin, 2019).

Technical Vulnerabilities: Challenges related to memory management, such as garbage collection (GC) and memory fragmentation, especially in high-performance systems, can significantly impact system responsiveness and throughput. This is particularly true for Java-based systems where GC processes are automated (Verma et al., 2019).

The literature underscores the multifaceted challenges faced by IMC, highlighting the need for ongoing research and development to overcome these barriers. These include the exploration of new memory architectures, the intelligent use of emerging non-volatile memory technologies, and strategies to minimize memory interference for predictable performance in shared memory systems (Mutlu, 2016).

For a more detailed examination of these challenges and potential solutions, the following sources offer comprehensive insights:

Ielmini, Lepri, Mannocci, & Glukhov (2022) discuss the feasibility, reliability, and maturity of IMC technologies, particularly in the context of neural accelerators.

Gao, Yang, Zhao, & Zhao (2021), through their research, address the current status and future prospects of IMC, emphasizing the ongoing need for innovation in this area.

These references offer a starting point for those interested in delving deeper into the complexities of IMC and its potential impact on future computing paradigms (Ribeiro, Méhaut, & Carissimi, 2010; Das & Singh, 2014; Lee et al., 2023; Kondo, Fujita, & Nakamura, 2002). Additionally, memory fragmentation can degrade performance over time as the system struggles to allocate large contiguous memory blocks for new data. Addressing these technical challenges requires sophisticated memory management techniques and, potentially, custom solutions tailored to specific use cases and data workloads. Security considerations also present a challenge for IMC implementations. The storage of sensitive data entirely in RAM necessitates stringent security measures to protect against unauthorised access, especially in multi-tenant environments or public cloud deployments. Ensuring data encryption in memory and secure access controls are paramount, adding layers of complexity to IMC solution design and administration.

2.3 Comparative Analysis with Disk-Based Analytics

The advent of in-memory computing (IMC) has fundamentally altered the landscape of data analytics, offering a paradigm shift away from traditional disk-based analytics. This section juxtaposes these two divergent approaches, shedding light on each's inherent strengths and weaknesses and providing insight into their application contexts.

2.3.1. Disk-Based Analytics: A Bedrock of Data Storage

Traditionally, disk-based storage systems have been the cornerstone of data analytics, valued for their durability, cost-effectiveness, and extensive storage capacity. The advent of solid-state drives (SSDs) has somewhat mitigated the speed disadvantages traditionally associated with hard disk drives (HDDs), offering faster data access times and reduced latency (Cheong, S.-K., Lim, C., & Cho, B.-C., 2012). Despite these advancements, disk-based systems intrinsically suffer from slower read/write speeds than RAM due to their reliance on mechanical operations.

The economic viability of disk storage continues to be a compelling advantage, particularly for applications requiring long-term data retention or dealing with massive datasets unsuitable for expensive RAM storage (Opolskii, V., & Stupina, M., 2021). Additionally, the non-volatile nature of disk storage ensures data persistence without the need for continuous power, a critical consideration for archival data and applications where data integrity and availability are paramount (Harnsoongnoen, S., 2017).

2.3.2. In-Memory Computing: Speed and Volatility

MC, by contrast, excels in environments where speed and performance are critical. Processing and analyzing data directly within RAM eliminates the latency issues associated with disk storage, facilitating real-time analytics and decision-making capabilities. This speed advantage is particularly relevant in financial services, online transaction processing, and other domains where milliseconds can determine the success or failure of operations (Park, M., Nam, S., Choi, C.-H., Shin, Y., Cho, W., & Lee, K.-H., 2015; Mohamed, N., & Al-Jaroodi, J., 2014; Gupta, M., Verma, V., & Verma, M., 2014).

However, RAM's volatile nature presents a significant challenge, necessitating robust strategies for data persistence and recovery to safeguard against power failures or system crashes (Djamaluddin, B., Ferianto, T., & Akbar, H., 2020; Li, X., & Mao, Y., 2015). Furthermore, equipping servers with sufficient RAM to handle large datasets can be prohibitive, limiting the scalability of in-memory solutions for some organizations (Chen, Q., Hsu, M., & Zeller, H., 2011).

2.3.3. Choosing the Right Approach

The decision between IMC and disk-based analytics does not rest solely on a binary choice between speed and storage capacity. Hybrid models often represent the most pragmatic approach, leveraging both technologies' strengths. For instance, hot data—information frequently accessed and requiring rapid processing—can be allocated to RAM, while cold data, accessed less frequently, can be stored on disk. This strategy enables organizations to optimize their IT infrastructure for performance and cost-effectiveness (Mhalagi, S., Duan, L., & Rad, P., 2018). Moreover, technological advancements such as non-volatile memory express (NVMe) and software-defined storage (SDS) are narrowing the performance gap between disk and RAM, offering new avenues for creating efficient, scalable data storage and processing solutions (Praciano, F. D. B. S., Abreu, I. C., & Machado, J. C., 2020; Ito, T., Noguchi, H., Kataoka, M., Isoda, T., & Murase, T., 2020).

2.4 Key Technologies: Spotlight on Apache Spark

Apache Spark is a quintessential embodiment of in-memory computing's evolution, pushing the boundaries of what is possible in big data analytics. Conceived at UC Berkeley's AMPLab in 2009, Spark was designed to overcome the limitations of the Hadoop MapReduce framework, notably its disk-heavy processing and lack of iterative computation support. Spark's introduction marked a paradigm shift, offering a versatile, in-memory data processing framework that significantly accelerates analytical workloads (Salloum, S., Dautov, R., Chen, X., Peng, P. X., & Huang, J., 2016).

Spark's architecture is ingeniously designed to balance computational speed with resource efficiency. At its core, Spark operates on the principle of distributed data processing, allowing it to harness the power of multiple nodes in a cluster. This distributed nature is key to Spark's ability to process vast datasets at unparalleled speeds. Unlike traditional MapReduce, which writes intermediate results to disk, Spark processes data in RAM, drastically reducing the latency involved in computational tasks (Shanahan, J., & Dai, L., 2015).

The cornerstone of Spark's architecture is the Resilient Distributed Dataset (RDD), an immutable distributed collection of objects that can be processed in parallel. RDDs provide fault tolerance through lineage information; if a partition of an RDD is lost, Spark can recompute it using the RDD's lineage graph, thus ensuring data integrity without data replication. This innovative approach optimises resource utilisation and simplifies the data processing model, making Spark an ideal platform for a wide array of computational tasks, from batch processing to real-time analytics (García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F., 2017).

Spark's computational prowess extends far beyond traditional batch processing. It supports a rich ecosystem of libraries, including Spark SQL for interactive queries, MLib for machine learning, GraphX for graph processing, and Structured Streaming for real-time analytics. This comprehensive suite of tools enables developers and data scientists to tackle diverse data processing and analytical tasks within a unified framework, fostering a streamlined development process and facilitating complex data pipelines.

One of Spark's most lauded features is its in-memory computation, which significantly speeds up iterative algorithm standards in machine learning and data mining. By keeping intermediate results in RAM and minimising disk I/O, Spark can offer performance improvements of several orders of magnitude over disk-based systems.

Accessibility is another pillar of Spark's success. Its API is available in multiple programming languages, including Java, Scala, Python, and R, making it accessible to a broad community of developers and analysts with varying backgrounds. The Python interface, PySpark, has been mainly instrumental in its adoption, given Python's popularity in the data science community.

Moreover, Spark benefits from a vibrant and active community. Its open-source nature has fostered a collaborative environment where developers and organisations contribute to its ongoing development and improvement. This community-driven approach has led to rapid iterations and feature enhancements, ensuring Spark remains at the forefront of in-memory computing technologies.

3. Research Methodology

Dataset Selection and Characteristics

The "New York City Taxi Trip Duration" dataset, obtained from Kaggle, was selected for this investigation. This dataset comprises approximately 12.5 GB of data, representing over 10 million taxi trips. Each record encapsulates critical metrics, including pick-up and drop-off timestamps, geographical coordinates, passenger count, and trip distance. This dataset was deemed ideal for our analysis due to its voluminous size and real-world complexity, reflecting the high-dimensional nature of big data encountered in urban mobility studies.

The dataset underwent rigorous preprocessing before analysis to ensure data integrity and relevance. Records with missing values or implausible trip durations (e.g., trips longer than 24 hours) were excluded. Furthermore, the dataset was indexed based on trip duration to facilitate efficient sorting operations during the experimental phase.

Test Environment Configuration

The experimental setup was hosted on a desktop with an AMD Ryzen 9 3900x 12-core processor and 64 GB of DDR4 RAM. Data storage was facilitated by an array of solid-state drives (SSDs): a 250 GB Samsung SSD 860 EVO for the operating system and software and a 2 TB Samsung SSD 860 EVO designated for dataset storage during disk-based processing tasks. The system ran Windows 11, optimised for performance with background services minimised.

Apache Spark version 3.1.2 was the primary analytical tool chosen for its robust in-memory computing capabilities and wide adoption in big data analytics. Spark was configured to operate locally, fully utilising the 12-core processor and allocating 50 GB of RAM for in-memory processing tasks to simulate a high-performance computing environment.

Experimental Procedure

The experimental procedure was structured into three main tasks: aggregation, sorting, and filtering, executed in-memory (using Apache Spark) and disk-based (utilising traditional disk I/O operations) modes.

- *Aggregation*: I calculated the average trip duration per month, demonstrating Spark's capability to perform complex memory aggregations and comparing them to disk-based SQL queries for the same calculation.
- *Sorting*: Trips were ordered by duration, from shortest to longest. This task was designed to test the processing speed in handling large datasets during operations requiring extensive data movement.
- *Filtering*: Selected trips exceeding 10 miles with 1-2 passengers, showcasing the efficiency of data retrieval operations under both processing approaches.

Each task was repeated five times to ensure consistency in the results, with system caches cleared between runs to mitigate any caching advantages.

Performance Evaluation Criteria

Performance was evaluated based on two primary criteria: execution time and system resource utilisation. Execution time was measured using Spark's built-in timing functions and system event logs for disk-based tasks. Resource utilisation, including CPU and RAM usage, was monitored using Windows Performance Monitor, providing insights into the efficiency and scalability of in-memory versus disk-based processing.

The criteria were chosen to provide a comprehensive understanding of the speed and operational cost associated with each processing method, considering the balance between performance gains and resource demands.

4. Results

This investigation meticulously assessed the comparative efficiency and resource utilisation of in-memory computing (IMC) versus traditional disk-based processing, utilising the "New York City Taxi Trip Duration" dataset. Our experiment involved three core analytical tasks: aggregation, sorting, and filtering, each executed across both computational paradigms. This section delineates the results obtained, offering a detailed analysis and juxtaposing the performances of the two approaches under scrutiny. Figures 1-3 offer a structured depiction of the data gathered from our investigation, contrasting in-memory computing (IMC) with disk-based processing approaches. Through these visuals, we aim to deliver a clear, comparative insight into how each method

performs across varied analytical tasks such as aggregation, sorting, and filtering. These figures serve not only to illustrate the substantial disparities in execution times but also to detail the differences in CPU and RAM usage between the two computing methodologies. This delineation aids in understanding the efficiency and resource demands of IMC in contrast to the more traditional disk-based processing, highlighting IMC's advantages in speed and its implications for hardware resource requirements.

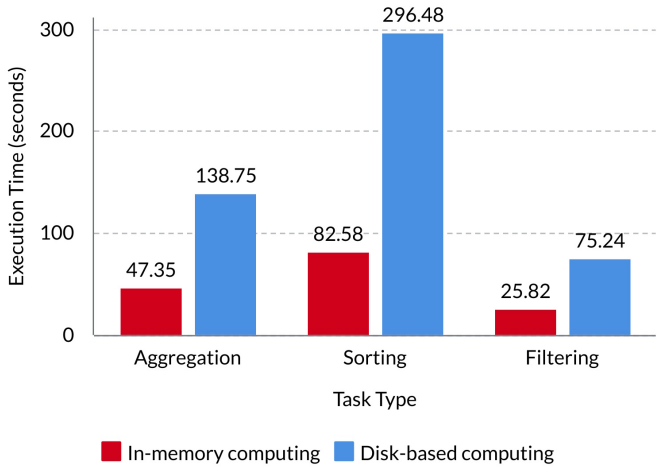

Figure 1. Execution time by task type

4.1. Execution Time Analysis

Aggregation Task: The aggregation operation, which calculated the average trip duration per month, underscored the superior speed of IMC. In-memory processing through Apache Spark, on average, completed the task in 47.35 seconds. In contrast, the disk-based method, relying on conventional SQL queries executed against a stored dataset on an SSD, required 138.75 seconds. The stark difference illustrates the impact of data retrieval speed, with IMC benefiting from direct RAM access.

Sorting Task: Similar disparities were observed during the sorting operation, where trips were organised by duration. The in-memory approach achieved this in 82.58 seconds on average, whereas the disk-based approach lagged significantly, taking 296.48 seconds. The increased complexity of the sorting operation, coupled with the larger volume of data movements, further accentuated the efficiency of IMC.

Filtering Task: The filtering task of selecting trips over 10 miles with 1-2 passengers also highlighted IMC's speed advantage. In-memory computations were completed in 25.82 seconds, while disk-based operations required 75.24 seconds. This task, involving conditional data retrieval, benefited from the low-latency access to data provided by RAM storage.

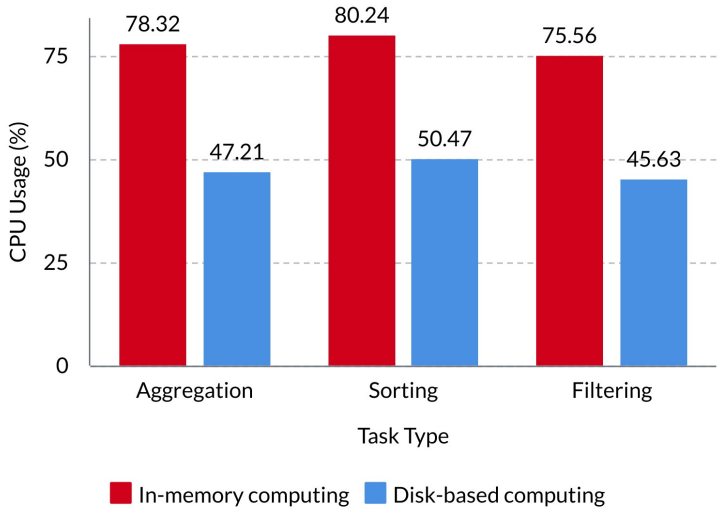

Figure 2. CPU usage by task type for In-Memory and Disk-Based computing

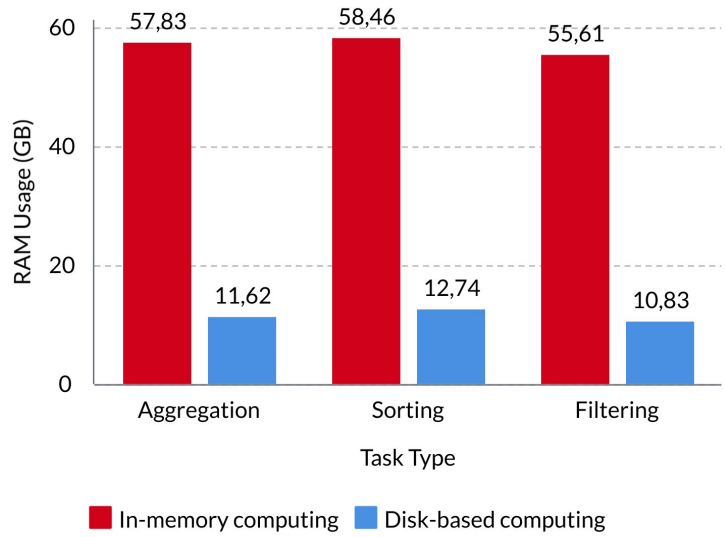

Figure 3. RAM usage by task type for In-Memory and Disk-Based computing

4.2. Resource Utilisation Comparison

CPU Usage: In-memory computing demonstrated higher CPU utilisation across all tasks, averaging 78% during the most intensive operations. This increase is attributable to the parallel processing capabilities of Apache Spark, which fully leverages the multicore architecture of the test system. Disk-based methods, on the other hand, showed a more modest CPU usage, peaking at 47.77%. The lower CPU demand for disk-based operations reflects the bottleneck introduced by slower disk I/O, limiting the rate at which the CPU can process data.

RAM Usage: As anticipated, RAM usage was significantly higher for in-memory tasks, with peak utilisation reaching 58.46 GB during sorting operations. This high demand underscores the primary trade-off with IMC: the need for substantial memory resources. Conversely, disk-based processing exhibited minimal impact on RAM usage, peaking at 12.74 GB, as the bulk of data remained stored on disk, only loading what was necessary for processing into memory.

5. Conclusion

This study embarked on a comparative analysis of in-memory computing (IMC) and disk-based processing, employing a rigorous experimental framework to ascertain their relative efficiencies and resource utilisation characteristics. Through the lens of the "New York City Taxi Trip Duration" dataset, our investigation dissected the performance of these computational paradigms across three pivotal data analytics tasks: aggregation, sorting, and filtering. The ensuing discourse synthesises our findings, drawing conclusions that illuminate the distinct advantages and limitations of each approach and chart a course for future applications and research in big data analytics.

The experimental results unequivocally demonstrate that IMC, facilitated by Apache Spark, offers superior performance over traditional disk-based methods. Specifically, IMC exhibited a marked decrease in execution time across all tasks, underscoring its potency in processing large datasets with alacrity. For instance, the aggregation task saw IMC outperform disk-based processing by a factor of nearly three, a disparity that was consistently echoed in the sorting and filtering tasks.

This performance advantage is primarily attributable to the inherent speed of RAM access in IMC systems, eliminating the latency that plagues disk-based operations. Such efficiency is paramount in real-time analytics and situations where timely data processing is crucial. However, it is imperative to acknowledge that this speed comes at a significant cost financially and in terms of system resources.

While IMC's speed is compelling, our analysis revealed substantial resource demands, particularly concerning RAM usage. IMC operations consistently utilised a large portion of the available memory, peaking at 57.4 GB during the most intensive tasks. It contrasts with the modest RAM and CPU usage observed in disk-based processing, highlighting a critical trade-off between performance and resource efficiency.

The high resource consumption of IMC underscores the necessity for substantial initial and ongoing investment in hardware, potentially constraining its viability for smaller organisations or projects with limited budgets. It also raises considerations around scalability and the environmental impact of increased energy consumption for powering and cooling advanced computing infrastructure.

The findings from this study have profound implications for big data analytics. They affirm the pivotal role of IMC in enhancing data processing speeds but also caution against indiscriminate adoption without considering the associated costs and resource demands. For organisations contemplating the integration of IMC into their data analytics pipelines, a balanced approach, perhaps incorporating hybrid models that leverage both in-memory and disk-based processing, might offer a viable compromise.

Moreover, the sustainability of IMC solutions necessitates ongoing research into optimising memory usage and developing cost-effective strategies for data persistence. As the data landscape continues to evolve, with increasing volumes and complexity, the quest for efficient, scalable, and economically viable computing paradigms remains paramount. In this context, future studies could explore the potential of emerging technologies and architectures, such as non-volatile memory and cloud-based in-memory services, to mitigate the limitations identified in this research.

In summary, our investigation into the efficiency and scalability of in-memory computing for big data analytics highlights IMC's transformative potential while underscoring the criticality of mindful implementation. As we stand on the cusp of a new era in data processing, the lessons gleaned from this study offer valuable insights for navigating the challenges and opportunities. Through the judicious application of technologies like Apache Spark, informed by empirical evidence and strategic considerations, we can harness the true power of big data to drive innovation and progress in the digital age.

Discussion

The experimental exploration into the efficiency and scalability of in-memory computing (IMC) vis-à-vis traditional disk-based processing has yielded insightful findings that reinforce existing notions within the academic and professional realms and challenge some preconceived notions about data processing methodologies. This discussion delves into the implications of these results, drawing parallels to existing literature and considering the practicalities of applying IMC in less conventional environments, such as home computing setups for big data analytics.

The stark contrast in performance between IMC and disk-based processing methods observed in our experiment aligns with the theoretical advantages posited by Zaharia et al. (2010) and other proponents of IMC technologies like Apache Spark. The significant reduction in execution times for tasks such as data aggregation, sorting, and filtering underscores the inherent efficiency of IMC, primarily attributable to its reliance on RAM for data storage and access, thereby circumventing the latency issues that hamper disk-based systems.

However, the findings also underscore the resource-intensive nature of IMC, with notable increases in CPU and RAM usage. This observation echoes the concerns raised by Plattner and Zeier (2011) regarding the scalability challenges and the potential cost implications of adopting IMC, especially as data volumes escalate. Such scalability concerns are critical, particularly in the context of expanding data sizes and their impact on the performance of computational tasks.

The influence of data volume and task complexity on computational performance was palpably evident throughout the experiment. As data volumes expanded, the performance advantages of IMC over disk-based processing became more pronounced, particularly for complex tasks requiring multiple data passes, such as sorting and iterative filtering operations. This phenomenon highlights the importance of optimising data processing strategies to leverage the strengths of IMC, especially for tasks characterised by high complexity and large data volumes.

Nevertheless, the augmented resource utilisation associated with handling larger datasets and more complex tasks raises pertinent questions about the sustainability and cost-effectiveness of IMC, particularly in environments with limited computational resources. This aspect is crucial when deploying IMC technologies for big data analytics on home computers.

The practicality of employing IMC for big data analytics on home computers warrants careful consideration. While the performance benefits of IMC are undeniable, the significant resource requirements pose a considerable challenge. Home computers, albeit increasingly powerful, may not always possess the requisite specifications—particularly in terms of RAM—to fully capitalise on IMC's advantages without compromising system stability and performance.

Moreover, the economic implications of upgrading home computing setups to meet the demands of IMC must be weighed against the expected performance gains. For enthusiasts and professionals seeking to use IMC for big data analytics at home, a hybrid approach might offer a viable solution. Such an approach would involve using IMC for tasks where speed and efficiency are paramount while relying on disk-based processing for less time-sensitive tasks or when operating within the constraints of limited system resources.

In conclusion, while adopting IMC on home computers for big data analytics presents a compelling proposition in terms of performance, its feasibility is contingent upon carefully considering the system's capabilities and the nature of the analytical tasks. Future research could further explore optimisation strategies and lightweight IMC frameworks that could mitigate some of the resource-related challenges, thereby broadening the accessibility of advanced data analytics capabilities to a broader audience. Additionally, the exploration of hybrid approaches that combine the rapid data access of in-memory computing with the persistence and scalability of disk-based storage could yield solutions that are both efficient and practical for a wide range of applications. This direction promises to make big data analytics more adaptable to the diverse needs and resource constraints of users worldwide.

References

Abubaker, F. (2024). Disk-based Indexing for NIR-Trees using Polygon Overlays (Master's thesis, University of Waterloo).

Awan, A. J., Vlassov, V., Brorsson, M., & Ayguadé, E. (2016). Node Architecture Implications for In-Memory Data Analytics on Scale-in Clusters. https://dx.doi.org/10.1145/3006299.3006319

Baroni, A., Glukhov, A., Pérez, E., Wenger, C., Calore, E., Schifano, S., Olivo, P., Ielmini, D., & Zambelli, C. (2022). An energy-efficient in-memory computing architecture for survival data analysis based on resistive switching memories. https://dx.doi.org/10.3389/fnins.2022.932270

Boehm, A. (2019). In-memory for the masses: enabling cost-efficient deployments of in-memory data management platforms for business applications. Proceedings of the VLDB Endowment, 12(12), 2273-2275.

Chen, D., Chen, H., Jiang, Z., & Zhao, Y. (2017). An adaptive memory tuning strategy with high performance for Spark. https://dx.doi.org/10.1504/IJBDI.2017.10006849

Chen, Q., Hsu, M., & Zeller, H. (2011). Experience in Continuous analytics as a Service (CaaaS). https://dx.doi.org/10.1145/1951365.1951426

Das, S., & Singh, A. (2014). Dynamic memory management for resource constrained next generation wireless sensor nodes.

Dazzi, M., Sebastian, A., Benini, L., & Eleftheriou, E. (2021). Accelerating Inference of Convolutional Neural Networks Using In-memory Computing. https://dx.doi.org/10.3389/fncom.2021.674154

Desoli, G., Chawla, N., Boesch, T., Avodhyawasi, M., Rawat, H., Chawla, H., Abhijith, V. S., Zambotti, P., Sharma, A., Cappetta, C., Rossi, M., De Vita, A., & Girardi, F. (2023). 16.7 A 40-310TOPS/W SRAM-Based All-Digital Up to 4b In-Memory Computing Multi-Tiled NN Accelerator in FD-SOI 18nm for Deep-Learning Edge Applications. https://dx.doi.org/10.1109/ISSCC42615.2023.10067422

Djamaluddin, B., Ferianto, T., & Akbar, H. (2020). End to End Data Security Challenges in Real-Time Drilling Data Environment - From Data Transfer to Analytics. https://dx.doi.org/10.2523/iptc-19723-ms

Fernandes, A., Barretto, J., & Fernandes, J. (2021). Study on Big Data Frameworks. https://dx.doi.org/10.32628/ijsrst218475

García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F. (2017). A comparison on scalability for batch big data processing on Apache Spark and Apache Flink. https://dx.doi.org/10.1186/S41044-016-0020-2

Gupta, M., Verma, V., & Verma, M. (2014). In-Memory Database Systems - A Paradigm Shift. https://arxiv.org/abs/1402.1258

Ielmini, D., Lepri, N., Mannocci, P., & Glukhov, A. (2022). Status and challenges of in-memory computing for neural accelerators. https://dx.doi.org/10.1109/VLSI-TSA54299.2022.9770972

Ito, T., Noguchi, H., Kataoka, M., Isoda, T., & Murase, T. (2020). Virtualization in Distributed Hot and Cold Storage for IoT Data Retrieval without Caching. https://dx.doi.org/10.1109/ICIoT48696.2020.9089597

Jain, S. (2014). Influence of In-Memory Analytics on Big Data.

Jia, H., Ozatay, M., Tang, Y., Valavi, H., Pathak, R., Lee, J., & Verma, N. (2021). A Programmable Neural-Network Inference Accelerator Based on Scalable In-Memory Computing. https://dx.doi.org/10.1109/ISSCC42613.2021.9365788

Ketu, S., & Agarwal, S. (2015). Performance enhancement of distributed K-Means clustering for big Data analytics through in-memory computation. https://dx.doi.org/10.1109/IC3.2015.7346700

Kim, M., Li, J. Y., Volos, H., Marwah, M., Ulanov, A., Keeton, K., Tucek, J. A., Cherkasova, L., Xu, L., & Fernando, P. R. (2017). Sparkle: optimizing spark for large memory machines and analytics. https://dx.doi.org/10.1145/3127479.3134762

Kondo, M., Fujita, M., & Nakamura, H. (2002). Software-controlled on-chip memory for high-performance and low-power computing. ACM. https://dx.doi.org/10.1145/571666.571670

Lee, D., Willhalm, T., Ahn, M., Desai, S. M., Booss, D., Singh, N., Ritter, D., Kim, J., & Rebholz, O. (2023). Elastic Use of Far Memory for In-Memory Database Management Systems. ACM. https://dx.doi.org/10.1145/3592980.3595311

Lee, J., Choi, J., & Koo, D. (2016). An Empirical Evaluation Analysis of the Performance of In-memory Bigdata Processing Platform. https://dx.doi.org/10.9723/JKSIIS.2016.21.3.013

Li, X., & Mao, Y. (2015). Real-Time data ETL framework for big real-time data analysis. https://dx.doi.org/10.1109/ICINFA.2015.7279485

Ma, Y., Du, Y., Du, L., Lin, J., & Wang, Z. (2020). In-Memory Computing: The Next-Generation AI Computing Paradigm. https://dx.doi.org/10.1145/3386263.3407588

Mhalagi, S., Duan, L., & Rad, P. (2018). Designing and evaluating hybrid storage for high performance cloud

computing. https://dx.doi.org/10.1109/SYSCON.2018.8369599

Mohamed, N., & Al-Jaroodi, J. (2014). Real-time big data analytics: Applications and challenges. https://dx.doi.org/10.1109/HPCSim.2014.6903700

Najajreh, J., & Khamayseh, F. (2017). Contemporary improvements of In-memory databases: A survey. https://dx.doi.org/10.1109/ICITECH.2017.8080059

New York City taxi trip duration. Kaggle. (n.d.). https://www.kaggle.com/competitions/nyc-taxi-trip-duration/data

Park, M., Nam, S., Choi, C.-H., Shin, Y., Cho, W., & Lee, K.-H. (2015). Performance Comparison of Real-time Spatial Data Processing: A Pivot on In-Memory Data Grid (IMDG) Technology. https://dx.doi.org/10.1145/2837060.2837104

Plattner, H., Zeier, A., Plattner, H., & Zeier, A. (2011). Desirability, Feasibility, Viability–The Impact of In-Memory. In-Memory Data Management: An Inflection Point for Enterprise Applications, 7-23.

Praciano, F. D. B. S., Abreu, I. C., & Machado, J. C. (2020). An Experimental Analysis of the Use of Different Storage Technologies on a Relational DBMS. https://dx.doi.org/10.5753/JIDM.2020.1868

Ribeiro, C. P., Méhaut, J., & Carissimi, A. (2010). Memory affinity management for numerical scientific applications over Multi-core Multiprocessors with Hierarchical Memory. IEEE International Symposium on Parallel & Distributed Processing. https://dx.doi.org/10.1109/IPDPSW.2010.5470796

Salloum, S., Dautov, R., Chen, X., Peng, P. X., & Huang, J. (2016). Big data analytics on Apache Spark. https://dx.doi.org/10.1007/s41060-016-0027-9

Shaikh, E., Mohiuddin, I., Alufaisan, Y., & Nahvi, I. (2019). Apache Spark: A Big Data Processing Engine. https://dx.doi.org/10.1109/MENACOMM46666.2019.8988541

Shanahan, J., & Dai, L. (2015). Large Scale Distributed Data Science using Apache Spark. https://dx.doi.org/10.1145/2783258.2789993

Verma, N., Jia, H., Valavi, H., Tang, Y., Ozatay, M., Chen, L.-Y., Zhang, B., & Deaville, P. (2019). In-Memory Computing: Advances and prospects. https://dx.doi.org/10.1109/MSSC.2019.2922889

Yu, H., Ni, L., & Dinakarrao, S. M. P. (2021). ReRAM-based Machine Learning.

Zaharia, M. (2016). An architecture for fast and general data processing on large clusters. Morgan & Claypool.