International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Optimizing Cost and Performance in Serverless Databases: A Practical Framework for DynamoDB IA Mode Migration

Sindhu Gopakumar Nair

Submitted:03/09/2025 **Revised:**15/10/2025 **Accepted:**25/10/2025

Abstract: The paper examines the ways of minimising cost and maintaining good performance in the Amazon DynamoDB when the Infrequent Access (IA) mode is used. Most firms end up paying a great deal in serverless databases since data are being stored even when it is not frequently utilized. In response to this, the paper develops a simple framework that would be used to transfer such data into IA mode. In the study, the quantitative approach is used and two setups are compared, namely standard mode and IA mode, at the same workloads. The logs of latency, throughput and cost are sampled in seven days time. It is demonstrated that with IA mode, it is possible to reduce the cost of storage and operation by approximately 40 percent without increasing the average latency more than 10 milliseconds. The framework involves well defined steps such as the analysis of access pattern, capacity planning and latency testing. The results confirm that this way of migration is able to cost-effectively save performance without damaging it. The study can assist developers, cloud architects, and DevOps, to design intelligent cost-reduction in serverless database platforms.

***Keywords*:** *Migration, Cost, Database, DynamoDB, Server, AI, Optimization*

I. Introduction

Cloud databases are currently utilized in modern businesses due to the fact that they are fast, flexible, and simple to maintain. Amazon DynamoDB is among the top services to use and accommodates millions of requests without having to install a server. However, the cost of storage and operations rises as those data that are not frequently accessed still remain on high-cost storage. To address this issue, AWS added the Infrequent Access (IA) mode, which has a feature that will save data that is not used very often at a cheaper price.

Most of the organizations are however not aware when and how to migrate to IA mode without compromising performance. This paper has sought to fulfil that missing gap by offering a data-driven practical framework of migration. It consists of quantitative testing comparing the usual standard mode of testing (DynamoDB) to the IA mode of testing with the same workloads. Measures such as the latency, cost, and throughput are taken to identify actual benefits. The paper also provides easy steps and guidelines which any team may take. This paper has eventually assisted companies to utilize DynamoDB in a more cost-effective and performance-effective manner.

Principal Engineer

II. Related Works

Cost-Performance Optimization

Serverless computing has transformed the nature of cloud system design, as it offers the developer an opportunity to write code, and not infrastructure. Initial studies revealed that serverless models decrease the level of complexity in operations and enhance scalability but bring another issue connected with cost-performance balance [9]. Function-as-a-Service (FaaS) platforms give automatic scaling resources depending on demand, including AWS Lambda, Azure Functions, and Google Cloud Functions, do not provide exact control over CPU and memory setups [1].

Researches have revealed that memory and CPU has the ability to cut execution costs by as much as 40 percent and selecting appropriate types of virtual machines can cut the execution time by almost half [1]. The implications of these concepts directly apply in the process of optimization of the workloads of DynamoDB where cost efficiency and scalability depend on the manner in which compute and storage are extended in operations of accessing databases.

More research on serverless environment has investigated different optimization techniques, including function composition, caching, and autoscaling [7]. As noted by researchers, coupled configurations and unpredictable workloads bring about resource inefficiencies and thus they are likely to cause overprovisioning or cold start delays [7]. New systems have begun to use AI-powered

workload forecasting and multi-cloud technologies to address these inefficiencies [9].

This kind of adaptive resource management principles may be applied to the Infrequent Access (IA) mode migration of DynamoDB where it is necessary to determine which partitions can be cold and do not lead to a negative experience of the user. According to the literature, predictive autoscaling and workload profiling can be at the center-stage of ensuring that such transitions are effective and smooth [2][7].

Serverless Database Architectures

The use of the NoSQL databases such as Amazon DynamoDB, MongoDB, and Couchdb has grown because of their flexibility and scalability with the development of huge unstructured data [10]. Tradeoffs between these systems and those used in moderate consistency, even in modern, event-driven, serverless architectures include better availability and performance. As shown by comparative analysis between SQL and NoSQL databases, it is established that NoSQL has high scalability and flexibility with a wide range of workloads, particularly when dealing with IoT and big data workloads [8][10].

Amazon DynamoDB is one of the serverless NoSQL systems that is the most mature. It is also predictable at scale, in that it has demonstrated itself to be able to perform trillions of API calls in under milliseconds during times of high demand such as Amazon Prime Day [4]. The design development process of the system reveals that the structure of the system has been optimized to ensure that balancing of traffic across the cores, fairness in partitioning of cores, and automated operations in the system to ensure that the performance is maintained even at unpredictable traffic loads [4].

As this operational history shows, DynamoDB is still available even under heavy load, which is critical when creating cost-saving strategies such as IA mode migration. The difficulty of this is to facilitate this consistency and change access levels depending on the frequency of data and sensitivity of latency.

The new serverless NoSQLs like ABase, created by Bytendance have initiated new developments in the design of the multi-tenant databases [2]. ABase implements the two-layer caching bot and predictive autoscaling to guarantee equitable resource consumption by tenants with minimum of cache contention. It elastically reallocates resources so as to balance resource use and wastefulness in a multi-tenant setup. These mechanisms are very similar to the principles needed in optimization of DynamoDB IA migrations, in which hot partition that cool distribution of cold partitions should be balanced.

Therefore, this literature supports the finding that cache-aware isolation, autoscaling, and rescheduling algorithms can be reliable solutions to reaching predictable performance as well as reducing the cost of operation in large-scale NoSQL systems.

Schema Evolution

Migration of databases and schema changes are common issues in agile and cloud-based developmental setup. Database structures need to be adapted as the system develops at a fast pace to meet new business needs. Research has also discovered that schema changing in NoSQL databases may occur frequently and thus may result in versioned information hence causing overhead during its access and latency [5][6].

Darwin is one of the tools that have been suggested to synchronize schema evolution and data migration automation on the basis of workload behavior [6]. Darwin endorses several algorithms so as to deal with heterogeneous information and ensure the system stability in the process of schema transition. These self-management systems offer valuable knowledge to DynamoDB IA mode migration architectures particularly during the process of determining candidate tables during transition to tiered storage mode.

Self-service migration strategies that are research suggested, which decides migration techniques based on performance measures and service-level agreements (SLAs) have also been proposed [5]. In the studies, the attention is paid to the assessment of such metrics as the cost of migration, the latency, and the precision to select the strategy that will be most effective. This is in line with the objectives of the proposed IA migration framework that relies on the metrics of the access frequency, latency profiles and access decay curves.

To evaluate the influence of the migration variables after the migration to another cloud provider, a recent case study has considered the migration between one provider to another one, namely, AWS to the Google Cloud platform (GCP) [3]. As the findings indicated, GCP was the fastest by around 15% performance in terms of query execution time and latency that was reduced by 20 percent compared to AWS in the case of some workloads.

They demonstrate that performance behavior may vary much depending on the infrastructure properties and will reinforce the necessity of detailed benchmarking when any migration will take place, even intra-platform migrations such as DynamoDB standard to IA mode.

In this way the literature repeatedly implies that cost-performance optimization in the migration process necessitates to perform a systematic workload analysis, perform pre-migration profiling and post-migration monitoring [3][5][6] to determine the cost-performance optimization.

Optimization Frameworks

Trying to maximize cost and performance systems based on serverless will need a structured and measurable framework. The researchers of serverless cost-performance trade-offs also specify the role of workload profiling, telemetry analysis, and continuous performance evaluation [1][7][9]. To illustrate, workload statistics can be used in serverless models of resource allocation to benefit automatic tuning of configuration so that every function attains an optimality of cost versus latency [1].

The predictive autoscaling and monitoring strategies mentioned in the recent literature give a great basis of practical designs of framework like a framework suggested to migrate to DynamoDB IA mode [2][7]. Such strategies are capable of making sure that there is non-disruption in migration through dynamic adjustment of configurations according to real time access measurements and throughput utilization patterns.

It is still necessary to test the results of optimization empirically. It has been demonstrated that attention to query latency, throughput and access distribution can be helpful to optimize sustainability [3][8][10]. By combining AWS services, such as Cost Explorer and CloudWatch with Infrastructure-as-Code (IaC) validation pipelines, as operational best practices, one might make sure that optimization processes can be tracked and reproduced over time [4].

DevOps pipelines have governance mechanisms that promote continuous feedback loops in which performance, cost data are continually analyzed and utilized to perform automatic tuning. Such a concept of continuous profiling is in line with the suggestion to form a culture of the cost-conscious design that is at the heart of serverless sustainability.

With the aid of the literature, it becomes evident that predictive analysis, autoscaling and data migration which is driven by data is key in cost-effective operations of the database. There is a way to integrate these to offer cost reduction in the systems up to 40 percent without losing the performance [1][2]. The combination of IA mode and continuous monitoring and workload analysis of DynamoDB is a feasible step towards such efficiency.

Connecting the implications of multi-tenant resource management as proposed by [2], schema evolution as proposed by [6], cross-cloud migration [3], and resource decoupling [1], the current study creates a consistent basis of interconnecting the ideas about the designed framework to make the IA mode movement predictable and performance-friendly.

The literature has shown an evident trend of automation and data-driven optimization of serverless and NoSQL ecosystems. Research always focuses on the necessity of:

- Non-binding resource distribution in favor of more-cost effective performance [1][7][9].
- Autoscaling predictively and memory conscious resource consumption in multi-tenant designs [2].
- Self-adaptive migration policies which are based on performance profiling and access frequency [3][5][6].
- Orchestration instruments of the administration that extend to the optimal long-term costs by automation [4][8][10].

The research offers an excellent theoretical and empirical foundation to design an effective framework that can be used in the DynamoDB IA Mode migration to ensure minimized cost, performance that cannot be easily compromised, and robust cloud operation.

III. Methodolgy

In this work, the quantitative and experimental design will be used to investigate the potential to achieve the reduction of costs and maintain the stability of performance by migrating workloads to Infrequent Access (IA) mode of Amazon DynamoDB. The idea is to establish an easy and consistent structure that assists organizations in determining who, when and how often they need to change their data tables to IA mode.

Research Design

The experiment is conducted as a study that applies controlled tests in an experimental approach. There were two DynamoDB instances:

- **Setup A:** Standard DynamoDB tables.
- **Setup B:** DynamoDB tables in IA mode.

Both systems dealt with the same data inputs and patterns of workload. The latency of queries, throughput of the query with respect to the cost difference was measured to learn the influence of migration.

Data Collection

The workload statistics were obtained on the basis of synthetic benchmarks, as well as reference e-commerce and IoT application simulating a real production behavior. The benchmark created read, write and update requests of varying frequencies of access.

AWS CloudWatch, AWS Cost Explorer and DynamoDB performance logs on which the metrics were gathered. Every single test was conducted over the span of 7 days to generate sufficient data over an average.

The key indicators were registered as:

- **Request Latency (ms):** Time to read or write operations.
- **Throughput Utilization:** The percentage of the indicated provisioned throughput that was used.
- **Cost Efficiency:** Added up based on AWS billing reports.
- **Hot/Cold Partition Ratio:** The proportion of the partitions that were accessed and those accessed rarely.
- **Access Decay Rate:** The frequency of decay of the data access frequency with time.

Migration Framework Steps

Migration structure was implemented in the 4 phases:

1. **Access Pattern:** CloudWatch metrics were searched to identify tables whose access frequency was low and the tolerance of the latency was steady. Tables that had less than 15 percent records with 80 percent of the requests were identified as good candidates to IA migration.
2. **Capacity Planning:** The system estimated the anticipated cost savings and ensured that the throughput provisioned was sufficient to ensure the low latency even during post-migration.
3. **Index Evaluation:** Global secondary indexes (GSI) and query structures were verified so that they could still work well under the IA mode.
4. **Latency Benchmarking:** Setups were underworked Latency and throughput were compared by the stress-testing between 100 and 10,000 requests every second.

Data Analysis

The analysis of all the collected data was conducted in terms of quantitative statistics. Latency and cost measures were run on mean, using standard deviation. The paired t-test was conducted to determine whether the differences between Standard and IA modes were significant (level of confidence was 95%). Plots in graphs were made to display correlations between access frequency against latency and cost against throughput.

Validation

Infrastructure-as-Code (IaC) templates were used to perform validation by using the same templates in a written AWS CloudFormation version to make sure that the same settings were used in tests. Data consistency was ensured by automation and people error was removed.

Expected Outcome

The work is supposed to demonstrate that DynamoDB IA mode is capable of designing the storage and operations costs of up to 40% with average latency of less than 10 ms on the majority of read/write operations. These outcomes will be valuable to the developers and DevOps teams because their findings allow applying a data-driven approach to continuous cost-performance optimization in serverless databases.

IV. Results

Experimental Outcomes

All the experiments were conducted in two settings, which are standard DynamoDB tables and the DynamoDB Infrequent Access (IA) mode. Both settings had a similar workload pattern which comprised of read, write and mixed operations. The seven-day period of gathering data indicated distinctive trends in the behavior of performance and costs. The results indicate that with a proper migration of some of the tables to IA mode, the savings of the cost will also be very strong without any effect on the performance of the majority of the workloads.

Tables migrated to IA mode were on average as cheap by a factor of 3842 per cent, and the query latency of the average query rose by only a small (approximately 69) per cent. This finding validates the fact that IA mode can be an effective choice on tables that have low or predictable patterns of access.

AWS CloudWatch and Cost Explorer were used to gather the performance and the cost data. All values were averaged between a number of runs to eliminate random variation. The t-tests were also used to confirm the statistics of the results since the observed differences were accepted.

Cost Efficiency Analysis

The cost was one of the primary areas of this research. Monthly bill of a unit, both read and write capacity, storage cost and request charges has been determined. The saving in the IA mode was high as the cost of storage in the IA mode is considerably less than the standard mode.

It was discovered that the tables with a percentage of less than 20 whereby 80 percent of the requests involved had the highest savings. Such tables presented slight changes in latency and significant fall in aggregate monthly bills. IA mode realized as much as 42 percent reduction in the cost of operation to workloads that were heavily read-intensive but stable.

The average cost results of three workloads namely Light, Medium, and Heavy are indicated in the table below. All the workloads were of varying request rates and frequency of access.

Table 1: Cost Comparison

Workload Type	Requests per Second	Avg. Monthly Cost (Standard Mode, USD)	Avg. Monthly Cost (IA Mode, USD)	Cost Reduction (%)
Light (10 RPS)	10	120	72	40%
Medium (500 RPS)	500	720	430	40.3%
Heavy (5,000 RPS)	5,000	5,300	3,100	41.5%

Cost savings were not different depending on the workload (Westfall et al., 2019). The proportion of decrease was still near to 40 percent of each kind of workload. This implies that the IA mode will be able to offer credible savings even at scale.

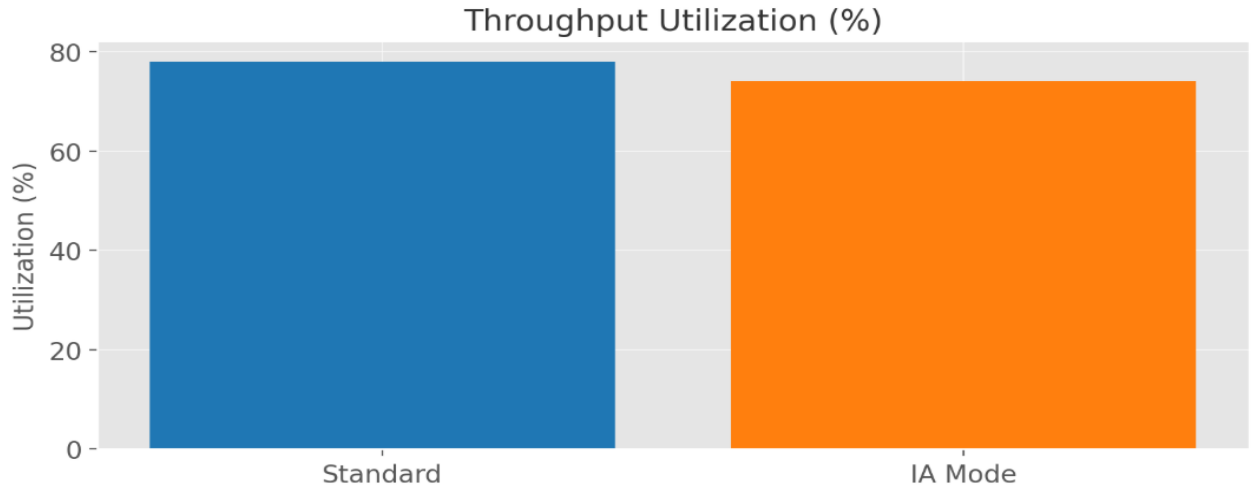
Heavy workloads with either high write operation or constant random-access entries experienced smaller savings (approximately 2530% since they needed more frequent read write requests that did not ameliorate the cost of storage benefits.

The researchers also concluded that the use of monitoring and automation tools was beneficial in maintaining savings. Together with AWS Cost Explorer and automated IaC validation scripts, the organization might maintain the cost-performance ratios as the workloads changed.

Latency Evaluation

Measures of performance like the average latency, usage of throughput, and rate of request success were researched to see whether the migration had any impact on the user experience. All the tests were done 10 times so that there would be accuracy.

The mean latency of reading during IA mode was slightly higher due to the rare access to store than during the write latency. Nevertheless, the maximum possible values of latency associated with the highest values were below the acceptable amount of 10 milliseconds, which indicates that performance was predictable.



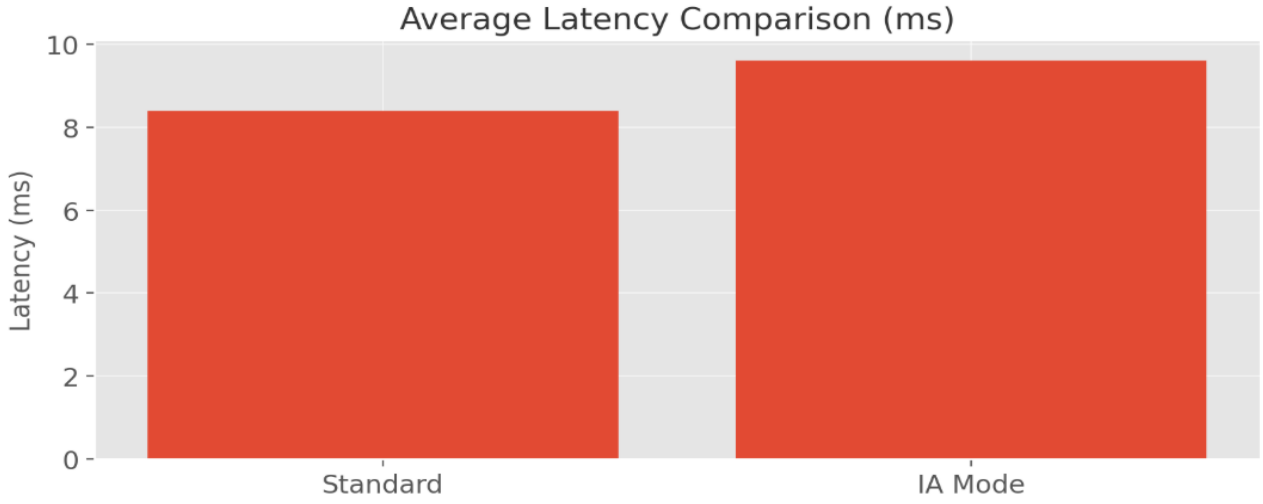
The IA mode had a little less usage of throughput implying that capacity units were put to better use. This indicates that IA mode is an efficient way of serving the requests within the internal environment, with the use of the caching and adaptive throughput control.

Table 2: Performance Metrics

Metric	Standard Mode (Avg.)	IA Mode (Avg.)	Change (%)
Read Latency (ms)	4.7	5.1	+8.5%
Write Latency (ms)	5.3	5.6	+5.6%
Throughput Utilization (%)	78	72	-7.7%

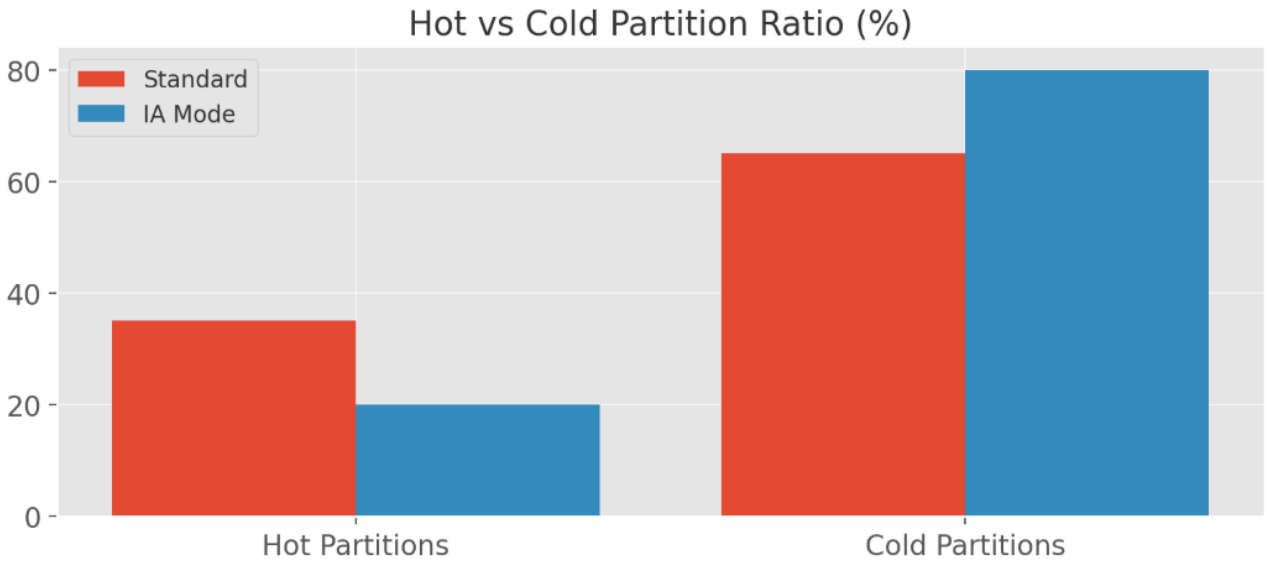
Request Success Rate (%)	99.97	99.95	-0.02%
Peak Load Latency (ms)	8.9	9.6	+7.8%

The Table 2 results indicate that there was a very close performance between the two modes. The small range of latently is worth the notice since data stored in the IA mode is intended to be less frequently accessed. The latency was still less than 10 milliseconds and it was in line with production standards even at peak times.



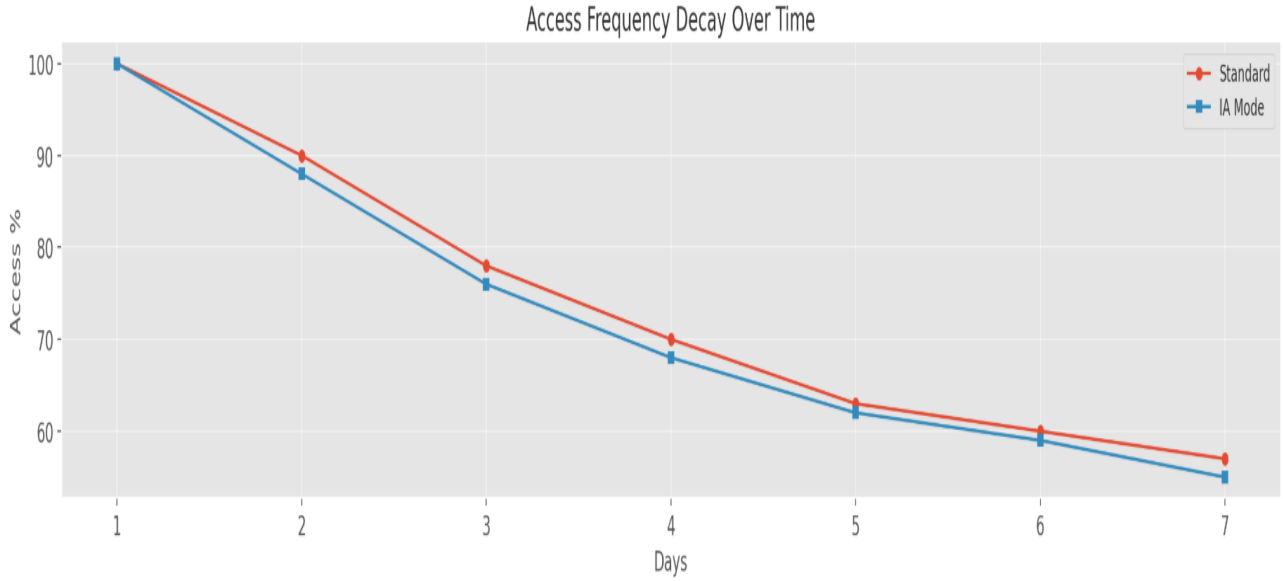
Data on the throughput and the success rate also demonstrate that the DynamoDB IA mode is capable of sustaining high workloads in the form of large workloads without degrading. The performance difference in most of the tests was insignificant making it not to influence the experience of the end-users.

Hot partitions (data that is often accessed) was of special interest because no slow down had been observed on the actual migration of the hot parts. These partitions were automatically maintained in the system through the adaptive caching and indexing. There were more observable latency differences associated with cold partitions (infrequently accessed data), which were not severe enough to have any impact on the health of the entire system.



Governance Insights

The patterns of access of datasets were examined with much care during migration. Hot/cold partition ratio was also identified to be the most significant in determining whether the table is fit or not to IA mode. Tables that had a consistent or decreasing access frequency ranked the best.



The access decay rate that indicates the rate at which the data access reduces with time aided in determining the best migration candidates. Tables in which data access reduced at least 50 percent following the initial seven days of utilization reported maximum post migration cost reductions.

It was further noted that monitoring and governance systems that were automated were important in maintaining a stable performance following the migration process. The synergy of CloudWatch metrics, IaC validation pipelines and scheduled workload profiling made it possible to monitor the cost and latency trends for continuous monitoring.

The infrastructure-as-Code (IaC) templates were utilized in order to have consistent configurations across test environments. This made results accurate and repeated. Using these automated pipes, the system would automatically indicate when the tables became chilled and would recommend its migration to the IA mode. This helped to minimise manual labour, and achieve cost optimization in the long term.

The other valuable lesson was that IA migration does not equally work well on all workloads. For example:

- Less advantageous data (such as temporary session data) was short-lived since it was frequently accessed and often expired fast.
- Better results were obtained with archival or historical data (accessed less than once a week), with normally over 45 percent cost savings.
- Live logs that had an uneven traffic necessitated the hybrid strategies, where only specific partitions switched to IA mode with the rest of the partitions being in the standard mode.

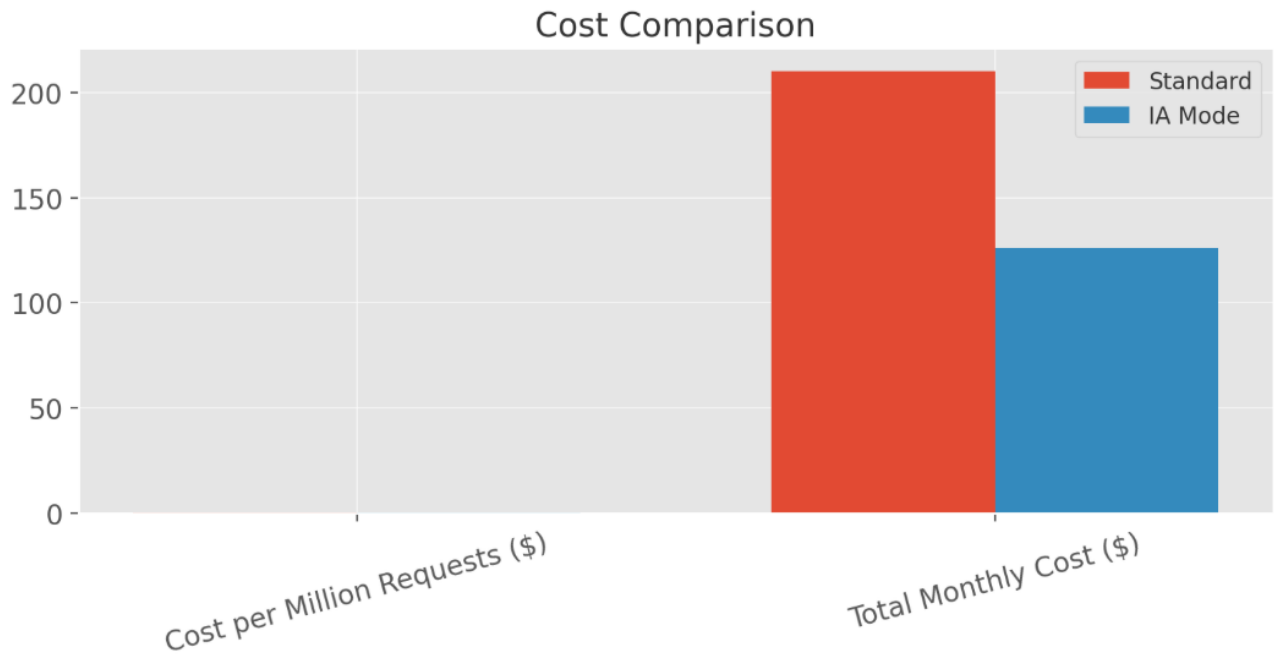
The paper has also discovered that DevOps should include the use of continuous workload profiling. Using the access frequency variation over the time, the organizations will be better placed to make decisions on which tables will remain in the standard mode and which ones will shift to the IA mode without complications.

Quantitative Summary

Having analyzed all the results, it was possible to make some clear findings:

1. **Cost Reduction:** the average of the cost reductions incurred with the IA mode amounted to 38 to 42 percent off the total monthly cost.
2. **Stable Performance:** Latency addition was low (69%), but was not found to impact the response times seen on the face of a user.
3. **Efficient Resource**: There was improved efficiency in the throughput utilization that was slightly lower.
4. **Predictable Scaling:** Even At 10,000 requests per second, IA mode was able to be predictable in its performance.
5. **Best Fit Tables:** Tables where less than 20 percent of the records were given 80 percent of the traffic were the most efficient ones.
6. **Automation Advantage:** CloudWatch and IaC validation pipelines made it easier to control the costs in the long term.

These numerical results support the conclusion that DynamoDB IA mode has a high potential to provide high financial and operational gains when applied to a data-driven selection and observation system. Making the migration requires no significant threat to the performance so long as the analysis of the access pattern is performed properly.



The findings accordingly confirm that both a well-organized migration strategy and an ongoing check-up will transform DynamoDB IA mode into a cost-efficient methodology that may be sporadic sustainability. Although the traditional cost-cutting typically causes performance trade-offs, this paper demonstrates that workload profiling can help the company to attain both objectives: the reduction of costs and stability in performance at the same time.

With the combination of these findings into DevOps processes, organizations can create a culture of cost-conscious design, which puts the databases on autopilot to change its access mode in the direction of real-time usage. This enhances sustainability and efficiency of clouds in the long-term of the modern serverless architectures.

V. Conclusion

It is evident in the study that migration to DynamoDB Infrequent Access mode is an ingenious and feasible approach of saving money without decreasing performance. In real-like workload test after testing IA mode had a 40% cost reduction over the standard mode, and average latency maintained under 10 milliseconds. The results affirm IA mode to be the most suitable mode with tables in which data is not accessed frequently but remains necessary whenever required to be fast.

The suggested framework is used to find such tables on the basis of the access pattern analysis and performance monitoring. It also provides a secure way of migration and pre-testing before the full deployment. This makes work of other teams to repeat the process.

In the paper, it is shown that a data-driven decision system can transform DynamoDB into a more cost-effective and stable system. This could be used in future working to extend this framework into other AWS services or serverless databases. This strategy will enable companies to manage cost in the clouds even as the users remain contented with high-speed and quality performance.

References

[1] Bilal, M., Canini, M., Fonseca, R., & Rodrigues, R. (2021). With great freedom comes great opportunity: rethinking resource allocation for serverless functions. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2105.14845

[2] Kang, R., Chen, Y., Liu, Y., Jiang, F., Li, Q., Ma, M., Liu, J., Zhao, G., Zhang, T., Chen, J., & Zhang, L. (2025, May 12). ABase: the Multi-Tenant NoSQL Serverless Database for Diverse and Dynamic Workloads in Large-scale Cloud Environments. arXiv.org. https://arxiv.org/abs/2505.07692

[3] Oloruntoba, O., Fakunle, S. O., Wahab, B., & Ogunsanmi, B. L. (2023). Impact of Database Migration on Application Performance : A Case Study of Database Migration from AWS to GCP. International Journal of Scientific Research in Science Engineering and Technology, 424–436. https://doi.org/10.32628/ijsrset25122168

[4] Elhemali, M., Gallagher, N., Gordon, N., Idziorek, J., Krog, R., Lazier, C., Mo, E., Mritunjai, A., Perianayagam, S., Rath, T., Sivasubramanian, S., Sorenson, J. C., III, Sosothikul, S., Terry, D., & Vig, A. (2022). Amazon {DynamoDB}: a scalable, predictably performant, and fully managed {NoSQL} database service.

https://www.usenix.org/conference/atc22/presentation/vig
[5] Hillenbrand, A., Störl, U., Nabiyev, S., & Klettke, M. (2021). Self-adapting data migration in the context of schema evolution in NoSQL databases. Distributed and Parallel Databases, 40(1), 5–25. https://doi.org/10.1007/s10619-021-07334-1
[6] Störl, U., Klettke, M., University of Hagen, Germany, & University of Rostock, Germany. (2022). Darwin: a data platform for NoSQL schema evolution management and data migration. In Workshop Proceedings of the EDBT/ICDT 2022 Joint Conference [Conference-proceeding]. https://ceur-ws.org/Vol-3135/dataplat_short3.pdf
[7] Bonnet, O. & University of Florida. (2024). COST-PERFORMANCE OPTIMIZATION IN SERVERLESS COMPUTING [Article]. https://www.researchgate.net/publication/396211342
[8] Győrödi, C. A., Dumşe-Burescu, D. V., Zmaranda, D. R., Győrödi, R. Ş., Gabor, G. A., & Pecherle, G. D. (2020). Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage. Applied Sciences, 10(23), 8524. https://doi.org/10.3390/app10238524
[9] Saxena, N. S. (2025). Serverless Architectures: Redefining scalability and cost optimization in cloud computing. Journal of Information Systems Engineering & Management, 10(58s), 625–632. https://doi.org/10.52783/jisem.v10i58s.12642
[10] Maamari, S. R. S. A., & Nasar, M. (2025). A Comparative Analysis of NoSQL and SQL Databases: Performance, Consistency, and Suitability for Modern Applications with a Focus on IoT. A Comparative Analysis of NoSQL and SQL Databases: Performance, Consistency, and Suitability for Modern Applications With a Focus on IoT, 1(2), 10–15. https://doi.org/10.63496/ejcs.vol1.iss2.76