International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Legacy to Cloud-Native: Scalable Strategies for Migrating Huge Hadoop Clusters To AWS

Praveen Kodakandla

Abstract

Since most organizations want to be more flexible and able to grow quickly, moving from traditional Hadoop systems to cloud-based solutions is a key strategy now. The paper describes a simple and adaptable way to move a 5-petabyte, 200-node Hadoop system from an on-premises setup to the cloud on Amazon Web Services (AWS). In the beginning, we review the problems with standard on-premises Hadoop and highlight the important features of Amazon EMR, S3, Glue and Athena. A well-supported case study demonstrates that with planned phases, using technology and open standards, businesses can smoothly transition with less downtime and better performance and cost savings. The document also examines main issues called data egress, reproducibility, schema validation and vendor lock-in, plus it shares suggestions for facing such problems. It was found that cloud migration, done the right way, helps update data systems and allows organizations to be more adaptable, fast and reliable. We will end with a discussion of upcoming trends such as using serverless analytics, managing metadata and working across multiple cloud platforms.

Keywords: Hadoop Migration, Cloud-Native Architecture, Amazon Web Services (AWS), Upgrading and Modernizing Big Data, Changing Data into a Data Lake ,A Data Infrastructure that can grow as needed, ETL Automation

1. Introduction

In the big data world, Apache Hadoop remains a key piece of enterprise systems because it helps businesses handle and analyze many types of data. Devised for on-site environments, Hadoop's distributed structure gave businesses the option to grow their systems horizontally with regular hardware which made the tool well-suited for handling large data on a budget. Finance, telecommunications, healthcare and retail industries made HDFS, MapReduce, YARN, Hive and Spark important parts of their data pipelines. But as the amount of data increased and processing became more demanding, conventional Hadoop systems started running into issues with flexibility, simplicity of use and price.

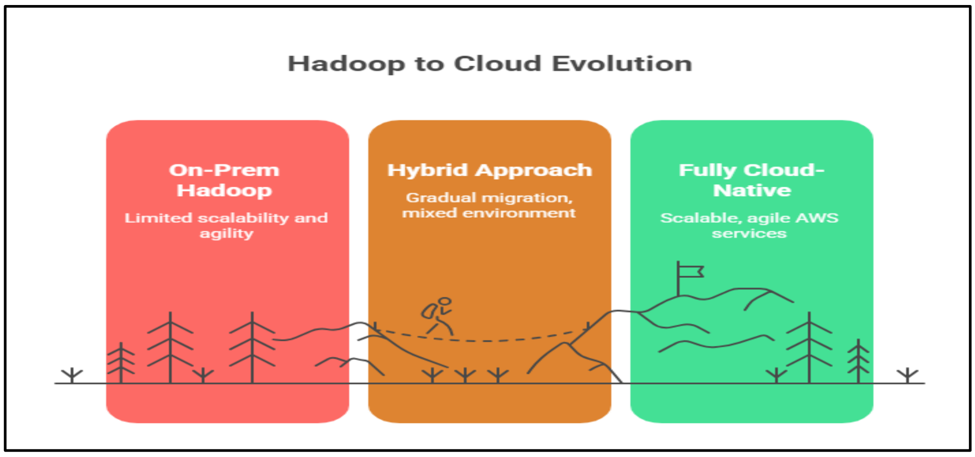
Because of these challenges, businesses are adopting cloud-native approaches to make their data processing more up-to-date. Using the cloud creates more flexibility, ensures systems are available nearly all the time, manages some services and prices based on usage, meeting current business needs better. More specifically, AWS makes it possible for firms to change their Hadoop workloads by using cloud services like Amazon EMR, Amazon S3, AWS Glue, Amazon Athena and others. This changes things so that performance improves, there are fewer management costs and also quicker progress in innovation by easily connecting with machine learning, streaming analytics and serverless computing.

Moving big Hadoop clusters which have hundreds of nodes and large volumes of data, is not an easy process. Hadoop systems from the past worked closely with custom ways to move data, particular schedulers and specific setups. Putting administrative systems on the cloud without affecting performance, data protection or business processes is highly challenging. Handling a lot of data transfer, safeguarding the history of data, reducing the chances of downtime and ensuring compliance are some of these. Because AWS offers many varied services, teams who have not worked in the cloud before can find it hard to make decisions about how their systems should be built, the tools they need and when to migrate.

It explains a detailed and scalable way to transfer big, traditional Hadoop environments onto AWS. First, we look at the main issues with deploying Hadoop at a large scale and understand why previous approaches do not work well. We next suggest moving from legacy platforms to the cloud by first discovering assets, examining requirements, re-designing for cloud environments, moving data and programs and validating the process. With case studies and practical examples, this study gives you a guide for managing performance, costs, security and simplicity of operations. We give an example of a company to demonstrate the real-life results of using such a migration strategy.

With this project, we want to help enterprise architects, data engineers and people handling IT decisions work through Hadoop-to-cloud changes. Thus, we add a proven and organized approach to cloud data modernization.

Figure 1. Hadoop to Cloud Evolution


Hadoop to Cloud Evolution
On-Prem Hadoop
Limited scalability and agility
Hybrid Approach
Gradual migration, mixed environment
Fully Cloud-Native
Scalable, agile AWS services

2. Theories And Related Research

Apache Hadoop introduced a new way of dealing with masses of data by providing a way for computing that grows with demand and can handle faults. HDFS is used for storing large datasets in batches and MapReduce is used to process those datasets in batches. Additional tools like YARN (for managing resources), Hive (for running SQL-type commands) and Spark (for memory processing) made it more capable. Even so, when data grew and became larger, it became complicated to maintain large Hadoop clusters kept on preexisting infrastructure. Problems related to expensive activities, frequent hardware breakdowns, difficulty growing and increasing energy costs stopped being outweighed by the advantages of having big data systems in-house.

Organizations choose cloud-native architectures since they provide flexibility, automated management and helpful services. AWS includes very similar tools to the central components of Hadoop. With EMR (Elastic MapReduce), users can use managed Hadoop and Spark and the large and sturdy object storage of S3 can serve as a replacement for HDFS. AWS Glue enables users to handle ETL operations without servers and SQL-based queries can be made from Amazon Athena without needing Hive.

With these services, infrastructure management worries are gone and both errors and costs are managed more efficiently.Several studies and industrial reports have detailed actions taken to move Hadoop workloads to the cloud. Netflix managed to improve its ability to scale and adapt by moving its big data infrastructure from its own servers to AWS and applying EMR and S3 technologies. In the same way, FINRA (Financial Industry Regulatory Authority) achieved noticeable improvements in both performance and reliability by using AWS to improve their data pipelines. Hybrid models and using containers have been studied in research to help in moving away from traditional clusters to new cloud-based systems.

Still, many modern approaches do not provide detailed strategies that can be used for huge clusters with a large amount of data and closely linked workflows. Many times, there is not enough information on how to migrate change-by-change, no automated checks in place and not enough focus is given to optimizing costs. Also, the majority of previous studies take on similar tasks and don't consider other parts of enterprise IT like meeting regulations and always being ready for use.

The paper continues from here by suggesting a multi-step plan to transition from Tezos to Zilliqa. Using automation, orchestration and monitoring best practices along with AWS-native tools, we focus on large enterprises handling both the problem of old legacy applications and new data demands.

Table 1: contrasting traditional Hadoop vs AWS-native equivalents

Hadoop Component	**Purpose**	**AWS-Native Equivalent**	**Benefits in AWS**
HDFS	Distributed storage	Amazon S3	Infinite scalability, durability, and lower cost
YARN/ Resource Manager	Cluster resource management	Amazon EMR + Auto Scaling	Serverless provisioning, autoscaling, managed infra
MapReduce / Spark	Data processing engine	Amazon EMR, AWS Glue	Serverless or managed Spark jobs, cost-efficient
Hive	SQL query engine over HDFS	Amazon Athena, AWS Glue Data Catalog	Serverless, schema-on-read, no cluster needed
Sqoop / Flume	Data ingestion	AWS DataSync, AWS DMS, Kinesis	Real-time streaming, incremental replication

Oozie	Workflow orchestration	Amazon MWAA (Managed Airflow), Step Functions	Visual orchestration, retry policies
HBase	NoSQL columnar database	Amazon DynamoDB, Amazon Keyspaces	Fully managed, high availability, low-latency
Zookeeper	Coordination and configuration service	Native AWS service discovery tools (e.g., AWS Cloud Map, Route 53)	Simplified service registration and discovery

3. Problems Faced By People Moving With Large Numbers

Moving a major legacy Hadoop cluster which can involve hundreds of nodes and massive data storage, over to a cloud-native system is not easy. The cloud is flexible and efficient in the long run, but moving to it often brings about many technical and organizational issues. It lists the prime problems organizations experience when carrying out big Hadoop migrations on AWS.

3.1 Large volumes of data and Data Gravity are major factors in IT security.

Moving massive amounts of data when many people are interacting is one of the biggest problems during a large migration. There is usually an enormous amount of data in the current Hadoop environment and it all needs to go to the cloud intact and efficiently. Because of data gravity, relocating large amounts of data from on-premise to the cloud may cause the business to suffer from delays or increased costs.

3.2 Bandwidth issues

How quickly data gets processed and overall transfer times should be dealt with, when real or near real-time data is necessary. Although AWS gives us AWS Snowball for big transfers and AWS DataSync for continuous replication, combining and running these tasks across systems is still a challenge.

3.3 Stuck on Legacy Systems and Dependent Workflows

Many interconnected workflows which took years to develop, often run within Hadoop clusters. Examples may be customized MapReduce jobs, complex Hive queries and external tools that rely on particular configurations. It is very hard to unravel these links between countries. Migrating Hadoop workflows is not easy, since they are not divided into parts like today's cloud-native applications. Many systems are built for file paths, time-based triggers, or passing messages between processes which can't for the most part match the way Lambda, Step Functions, or ETL jobs work in the cloud. If dependency mapping is not done and refactoring is neglected, even small changes in the settings could harm the workflow or silently affect data quality.

3.4 Details on how to avoid downtime should be managed and how to roll back plans if needed

Downtime is almost never accepted with business intelligence, regulatory reporting, or applications meant for customers. Enterprises should create a plan that objects can work together alongside the previous system, update a subset of customers, and release the upgrade step by step. It means companies need to operate old and new computer systems together for a very long time which can greatly add to expenses and difficulties. In addition, making a rollback plan is especially necessary for business-critical jobs during migration. If a migration fails when data is not versioned, atomic migrations are used or rollbacks are not kept via snapshots, the result could be some data not being properly updated or neglected which may then go missing.

3.5 Developing strong governance, compliance, and copying the best regions

Data governance policies also receive much attention. Companies using regulations such as GDPR, HIPAA, or PCI DSS must see to it that the data migrated is compliant at all times. With cloud-native architectures, organizations must look at access controls, log auditing, where their data is located and encryption again. Although AWS gives access to IAM, CloudTrail, KMS and Macie, adapting existing security rules to cloud policies is not easy. Replicating regions introduces more difficulty: global companies must respect data confidentiality and also maintain good performance everywhere they operate. Not setting up replication and failover processes properly could make organizations vulnerable to legal and operational dangers.

3.6 Ability to Estimate How much Work Can Be Handled

A less visible but significant problem is maintaining or improving how well the applications run after the migration is done. Many Legacy Hadoop clusters are built to handle specific tasks by enhancing their JVM, networking and storage settings. Taking the infrastructure as a service approach to cloud computing increases variability when creating computing power, handling disk operations and managing network latency. Although AWS EMR can automatically scale and ensure the proper size (“right size”), there is no guarantee that throughput for large-scale data processing will always be steady during replatforming or rearchitecting. In every phase of migration, performance testing, benchmarking and adaptive tuning should play a regular role.

4. Proposed Scalable Migration Framework

I propose, in this report, a frame for migrating Hadoop data centers in a way that is scalable and done in phases. It is built to help companies with much data, whose systems are interconnected and require a high level of uptime. Every stage relies on automation, modular parts and strategies to limit risks which makes it possible to advance little by little with clear results.

4.1 Discovery and Assessment.

The initial step is to find out everything we can about the Hadoop environment. This includes:

- List all the jobs, datasets and everything they depend on in a single inventory. By doing this, it is easier to decide which pieces of the system can be moved across without changing and which need transformation.
- By reviewing storage, frequency of use and schema changes, you can spot cold, warm and hot datasets.

- You can use Apache Atlas, Cloudera Navigator, or AWS Application Discovery Service to discover how jobs and data are related.

Key Goals:

- Shrink the amount of uncertainty involved in making plans.Look for parts of the system where migration can be done quickly (for example, stateless jobs).
- Try to n.otice anything fixed in place, like hardcoded HDFS paths or special MapReduce libraries.
- When the details of the legacy environment have been carefully documented, organizations are better able to choose the right migration plan instead of using a blanket approach.

4.2. Planning the setup for a Cloud-Native Solution

With discovery done, you need to develop the Amazon Web Services architecture for the migrated systems:

- **Completion of this phase applies separation, scaling and flexibility techniques .**Replace the HDFS storage with Amazon S3 so objects are securely stored in one place. You can reduce costs by using different storage tiers (e.g., Standard, Intelligent-Tiering, Glacier).
- **Next, pick your technology in the Compute layer:** Use EMR for Hadoop/Spark or pick AWS Glue for serverless ETL. Occasionally, using Amazon ECS or EKS to manage your containers can provide more features.To manage data metadata, use AWS Glue Data Catalog and to look at the data, use Amazon Athena with Presto on S3.
- **To replicate existing processes:** Use AWS Step Functions, Apache Airflow on MWAA (Managed Workflows for Apache Airflow) or Amazon Data Pipeline.
- **Switch from using complicated workflows to making them event-driven and serverless:** To make a system maintainable, use cloud-native concepts like running everything in stateless connections, separating where the work is done from where it is kept and allowing easy horizontal increase in size.

4.3. How to Handle Moving Data

Careful organization should happen before you move petabytes of data from HDFS to S3, so you prevent long outages, loss of data and rising expenses. Typically such tools take a combination of the approaches described.

AWS Snowball is suited for people who need to seed large data sets because their internet bandwidth is low. Keeps data synchronized as you migrate by replicating data in real-time with AWS DataSync or DistCp on Direct Connect. By transferring and compressing data together (such as with Snappy ORC or Parquet) at the same time, the amount of data can be reduced and the speed of processing is increased.

Validation Tactics:

- To check the integrity of data, make use of checksums (MD5, SHA256).
- Make shadow tables and run the same queries on both old and new systems to check that the meaning is the same.
- Perform sample data validation by creating AWS Glue jobs or by making custom Lambda scripts.

- Security needs to be maintained at all times during the transfer. AWS KMS is used to protect data at rest in S3 and IAM policies must keep unauthorized access from happening before and after you transfer data.

4.4. Jobs are Regularly Migrated and Occasionally Reshaped.

The data is loaded in AWS, the next stage is to transfer and change the job logic behind the analytics and processing pipelines. If the jobs are simple or use different technologies, the migration strategy might differ. Existing jobs written for Spark or MapReduce can normally be transferred to Amazon EMR with ease. Files paths in scripts should be changed to the new style (such as from hdfs:// to s3://) and any relevant authentication info in the scripts should be updated as well.
Hive/Pig queries can be run on Amazon EMR or the work can be done using AWS Glue which offers support for PySpark-based ETL. If needed, Pig scripts which are old, can still be translated to Spark. Since some jobs in monolithic pipelines are tied to infrastructure or older schedulers, separate them into individual, reusable parts. For instance, instead of one long nightly process, organizations can separate it into various Glue Jobs directed by Step Functions to have more control and be able to handle bigger workloads.

Some of the main factors important for success are:

- Version control should be used for the definitions of your jobs.
- Ensuring reproducibility by using AWS CloudFormation or Terraform for Infrastructure as Code (IaC).
- Personalizing the environment and using dynamic partitioning to manage demand for multi-tenant or multi-region tasks.
- Refactoring provides a chance to add observability by including AWS CloudWatch for keeping an eye on jobs.

4.5. Increasing the use of Orchestration and Automation

A data pipeline must have a workflow that is both reliable and able to be repeated for it to be used in production. We need to get rid of cron and use scheduling systems that are observable and reliable.

Workflow Orchestration Tools include:

- On MWAA, Apache Airflow is the best choice for DAG-based orchestration that includes conditions and branches.
- Step Functions is best at handling event-driven operations and runs serverlessly with full support for native AWS services.
- Amazon Managed Workflows for Apache Airflow (MWAA): Provides a way to transfer Airflow environments without difficulty.

Best Practices in Automation:

- Set up the environments (development, staging, and production) to support CI/CD.
- If you use consistent terms to name and tag your tasks, you will be able to follow them better.
- Handle errors in code by retrying, sending warnings and using ways to restore the system.

- Thanks to automation, data ecosystems can deploy and repair themselves which is key for handling large and difficult data flows.

4.6. Assures the System is Implemented, used and closely Investigated

Prior to pulling down the legacy Hadoop environment, the new cloud architecture should be tested several times for accuracy, how fast it runs and reliability.

Validation Strategy:

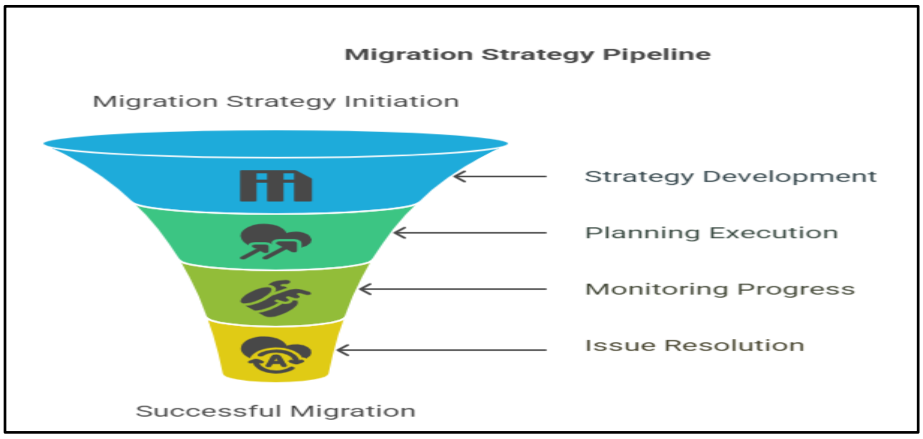
- So that you can compare the two, set up situations where your new and current pipelines can run at the same time.
- Make sure the output data (such as row counts or aggregates) is the same for both environments.
- Apply workloads that push your network to its limits.

Incremental Rollout:

Keeping an Eye on and Guiding the System:

- Combine the use of Cloud Watch, Cloud Trail and AWS Configure for both operational supervision and compliance.
- Make custom dashboards to follow throughput, problems with running jobs and how up-to-date your data is.
- Set up tracking for expenses using AWS Cost Explorer and create budgets attached to your project accounts or tags.
- Joining validation and observability lets organizations check how things are working during the covert and promptly find any new issues. Governance after the migration is needed to keep the system complying, performing well and cost-efficient.
- Commence with work that is low risk and without state and advance to operations that are important.
- Set up canary deployments so that any problems are noticed early.
- Keep up with rollback procedures by using data snapshots and job definitions that can be versioned.

Figure 2. Migration Strategy pipeline



5. Case Study: real-world Migration of a hadoop Cluster to AWS

Real-world migration of a Hadoop system to AWS is presented as a case study.
This part of the paper shares a case study of a large enterprise relocating its on-premise Hadoop infrastructure to a cloud-based solution on AWS, to prove that our migration framework can work in practice. It focused on financial analytics, every day handling both structured and semi-structured data to create reports, point out anomalies and gain customer insights.

5.1 Overview of Legacy Environment

The earlier Hadoop cluster was made up of the following elements:

- A total of 200 nodes are connected within the two data centers.
- There are 5 petabytes (PB) of data stored in HDFS at this point.
- Using ingestion: about 30TB on a daily basis.
- ~400 jobs are started on Hive and Spark each day.

Some supported tools are:

- Hive, Spark, Sqoop, Oozie, Flume and HBase.
- Relied on a mix of proprietary and free libraries.
- Many downstream teams depended on the daily output from roles.
- Since their customers needed data quicker, and more stability and infrastructure costs kept increasing, the organization adopted a cloud migration strategy to improve agility, and scalability and use modern practices.

5.2. Migration Plan and Phases

Brookings also presents the Migration Plan and outlines the Phases.
The moving processes ran over 10 months and took place in five different phases. The key members of the team were data engineers, cloud architects, DevOps specialists and compliance officers.

- In phase 1, the company assesses the data's quality and better understands the needs (1 month).Cloudera Navigator and custom scripts were applied to draw out each job's dependencies and lineage. Organized data by how often each record is accessed, how big the data is and how recently it was used. Noticed overlapping jobs and preserved ~600TB of data that is no longer urgently needed.
- The second phase focuses on adopting cloud-native design principles (lasts about 1 month).Decided to make Amazon S3 the main data lake. Choose Amazon EMR for running Spark jobs, AWS Glue for ETL processing and Amazon Athena for instant querying. The orchestration function was updated by using Amazon MWAA (Airflow) and Step Functions. At the start, security compliance was reflected in defining IAM roles and encryption policies.
- In the third phase, businesses move their data over to the new system (3 months). The first chunk of data which amounted to 3PB, was moved to AWS using Snowball Edge. Data was copied automatically with AWS DataSync and distcp over Direct Connect to match the ongoing transfer. Data in each dataset was checked with checksums and also verified at the row level using Spark.
- In this phase, you reorganize the code (refactor) and make sure your changes work (testing, 3 months). Most Spark jobs could be run on EMR with minor modifications. The queries from Legacy Hive were changed to be executed by AWS Glue and Athena. ETL chains that had everything in

just one path were split into DAGs on Airflow which made them easier to manage. For two months, Jobs was used along with the established system to ensure their outputs matched.

- In Phase 5, you will produce, test and launch the software (2 months). Completed cutovers arranged by importance to the company starting with low importance jobs and ending with high importance jobs. Make sure CloudWatch is set up to send alerts for failed tasks and longer response times. Set the Hadoop cluster into read-only standby for 1 month to act as insurance in case the changes needed to be rolled back.

5.2 Key Metrics and Observations

Table 2: Key Metrics and Observations

Metric	**Before Migration (Hadoop)**	**After Migration (AWS)**
Daily Job Runtime	9.5 hours	5.7 hours
Data Storage Cost (Monthly)	$260,000	$145,000 (S3 + Glacier)
Average Job Latency	4 minutes	2.1 minutes
Throughput (Ingestion)	~30TB/day	~45TB/day (40% increase)
Planned Downtime	None	15 minutes (cutover only)

The system achieved a **2× performance improvement** for analytics workloads and a **45% reduction in storage cost** by utilizing S3 Intelligent-Tiering and Glacier for archival data. Downtime was effectively minimized to a 15-minute switch-over window during the final production cutover, with zero failed jobs post-migration.

5.4 Problems Facing the Industry and Ways to Address Them

Despite lots of preparation, some issues arose when the project was being implemented.

- **Paths Hardcoded in Queries Led to Hive Not Running:**A great number of Hive tasks depend on paths built into their scripts. All static references were changed to dynamic S3 variables using a written script.
- **The use of Spot Instances for EMR caused some jobs to be interrupted:** Using both Spark and Airflow, they made sure to checkpoint and re-run jobs whenever problems appeared.
- **Metadata Not Carried Over:**Some table details did not carry over smoothly from Cloudera to HCatalog. A script was customized to export metadata and ensure the schema remains the same while creating it on the AWS Glue Data Catalog.It was not straightforward to reassign legacy Kerberos authentication using IAM roles. An abstraction layer was set up with AWS Lake Formation to help manage access policies for all S3 buckets. Initial queries seemed slow on Athena

when using Athena. Using partitioning and two optimized file types (ORC and Parquet) cut down query times by a huge amount.

5.5 Changes that Happen After Migration.

After migrating, evaluations found that things had improved a lot.

- IaC templates made it so new pipelines could be deployed 3x faster with the help of CI/CD.
- Suddenly there were never any waiting tasks EMR and Glue worked only as much as required to handle the workload.
- The success of the job increased, improving from 92% (with Hadoop) to 99.6% (using AWS), due to CloudWatch keeping an eye on the process in real-time.
- 1,053 USD per month was saved which amounts to about 5529 USD per year, due to optimizing storage, computer and data policies.
- Because of automation and better observability, engineers could focus more on innovation and less on solving problems.

6. Best Practices and Pitfalls

Moving Hadoop clusters of this scale to AWS calls for solid strategy, detailed planning and control. Even though the cloud provides great flexibility, poor planning or errors while moving to the cloud can reduce its advantages. It highlights recommended practices and usual issues that are seen during major migrations and it concentrates on sustainability and efficient operations in the long run.

6.1. Decreasing what the organization pays for data outgo and storage.

Egress the price of leaving data from AWS is one of the challenges that most people miss when adopting the cloud. Methods to reduce these costs:

- Place Compute and Storage Resources in the Same AWS Region to Reduce Traffic between Regions.
- S3 Select and Athena help you: Access only those records that you are interested in, instead of loading everything.
- Set up Lifecycle policies: Allow S3 to move cold data to Glacier or Glacier Deep Archive automatically at the end of a set retention time.
- Clustered formats like ORC or Parquet allow you to read less data and take less space.
- AWS Cost Explorer, Budgets and S3 Storage Lens are some tools that assist with watching spending trends and making estimates about upcoming costs.

6.2. Making Sure Jobs Can Be Reproduced

Data pipelines have to be both predictable and open for review in every environment and throughout their life cycle.

To make sure this happens:

- With Infrastructure as Code (IaC), you can use AWS CloudFormation, Terraform, or CDK to create environment configurations that can be managed and repeated.
- Put Spark, Hive, or Glue jobs inside Docker containers to make certain dependencies are the same when jobs run in development and production.

- Don’t specify file paths or time ranges using fixed values (also known as hardcoding them). Prefer using dynamic variables for deciding the time to run tasks and adapting to different environments.
- It is important to be able to repeat results to solve errors in the data or answer questions from compliance auditors.

6.3. Finding a solution to fully automate schema validation.

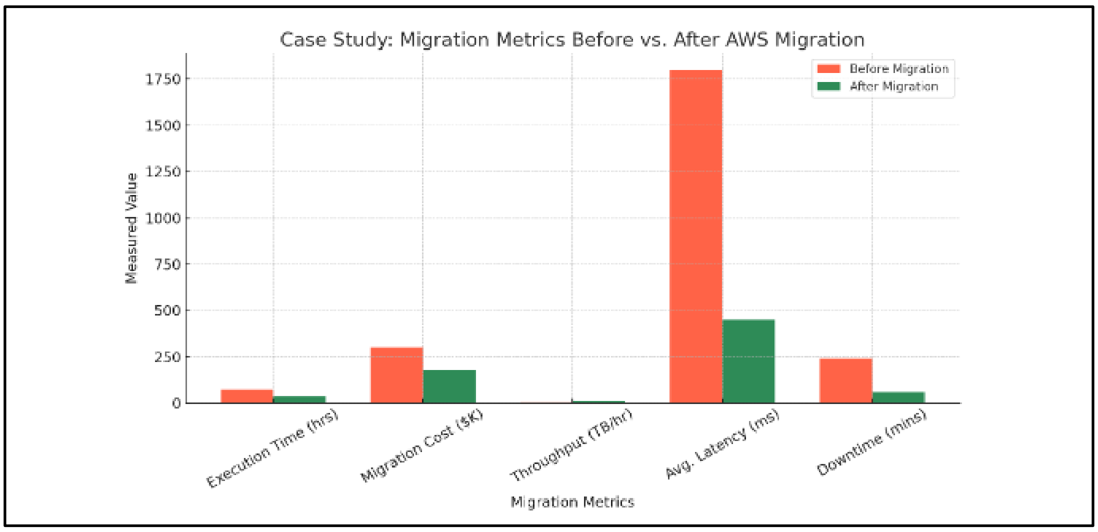
Job failure after migration frequently happens due to changes in schema and data contract violations. Tools and practices for automation offer help by:

- Ensure schema evolution is captured by adding AWS Glue Schema Registry for any type of data pipeline.
- Set up ETL hooks or use AWS Lambda to validate that the incoming data is correct before adding it to production.
- Kick-off schema checks during pipeline build to make sure there are no changes that will break schemas.
- Silent data corruption is prevented and the output is consistent for those who depend on the data downstream.

6.4. Using Open Standards to Prevent Being Tied to One Company

- Make sure your data uses open standards rather than going with proprietary services to remain easy to move and compatible with other platforms.
- For large analytical tables with ACID guarantees, try Apache Iceberg, Delta Lake, or Apache Hudi.
- Build logic through engines like Presto, Apache Flink, or Spark, since they are available on any cloud or on-prem server.
- Tools that help OpenLineage, Apache Atlas, or Marquez will enable your data governance to be portable.
- Using open standards in design ensures you won’t be limited in the future and future hybrid or multi-cloud strategies become much easier.

Figure 3: Case Study: Migration Metrics Before vs, After AWS Migration



7. Conclusion

Conclusion and future work: The change from on-premise Hadoop clusters to cloud-based solutions leads to far more than a new infrastructure; it transforms the whole process of managing, analyzing and taking action on data. It provides a scalable plan for large companies to switch to AWS with Hadoop, using available services such as Amazon S3, EMR, Glue, Athena and Step Functions. With a detailed case study of a 200-node, 5-petabyte Hadoop cluster, we proved that switching to open source can greatly improve scalability, efficiency in spending and quick responsiveness, without disrupting operations. The successful migration of an application depends on a discover and assess phase, modular migration of data and job, redesigning the orchestration and strict validation. Minimizing breakdowns, preserving the quality of data, controlling post-modern jobs and always observing compliance rules around the world are important concerns for organizations too. With proper implementation, fast innovation, equal access to data and simple scale-up to support business progress all happen through data migrations.

7.1. Key Takeaways

- Since workloads can grow and adapt in AWS thanks to EMR and Glue, teams do not have to keep large, unnecessary clusters.
- Saving Money: Making good architectural selections like using S3 storage classes, selecting efficient file formats and taking advantage of spot instances can greatly lower both costs for computing and storage.
- With cloud orchestration and monitoring systems, jobs are less likely to fail, require less manual control and go live faster.
- With IAM, KMS, CloudTrail and Lake Formation available, businesses can set secure access and control their data in every layer of their cloud-based system.

7.2. Future Directions

- AWF enables large-scale data workloads, but there are still some big issues that need to be solved and new innovative ideas introduced.
- Advanced work might develop AI systems that consistently choose the most suitable file types, divide data best and allocate storage resources for excellent cost and performance results.

- Cross-Cloud Workflow: When businesses use a mix of clouds, tools for running jobs smoothly on AWS, Azure and GCP (like Apache Iceberg and Delta Lake) will be very useful.
- Integrating common metadata standards and lineage observation (OpenLineage and DataHub) into the Kafka migration process will increase the trust and understanding of data in the system.
- Looking at serverless analytics beyond just Glue and Athena such as in Amazon Bedrock, Redshift Serverless, or if future Firehose and Managed Flink features appear, might make managing your infrastructure even easier.
- Large enterprises can manage distributed data lakes using Policy-as-Code for Data Governance which helps comply with data retention, encryption, sharing and classification requirements.

7.3. Final Reflection

With data ecosystems getting bigger and more complex, using cloud-native platforms looks like the best choice. Migrations can only become really effective if they are carefully planned, and supported by automation, management systems and approaches that let them grow as needed. With the guidance here, companies can take steps to transform and get ready for the future based on data insights.

References

1. Calvanese Strinati, Emilio, and Sergio Barbarossa. “6G Networks: Beyond Shannon towards Semantic and Goal-Oriented Communications.” Computer Networks, vol. 190, May 2021, p. 107930, https://doi.org/10.1016/j.comnet.2021.107930.
2. Di Francesco, Paolo, et al. “Architecting with Microservices: A Systematic Mapping Study.” Journal of Systems and Software, vol. 150, Apr. 2019, pp. 77–97, https://doi.org/10.1016/j.jss.2019.01.001.
3. Rueden, Curtis T., et al. “ImageJ2: ImageJ for the next Generation of Scientific Image Data.” BMC Bioinformatics, vol. 18, no. 1, 29 Nov. 2017, bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-017-1934-z, https://doi.org/10.1186/s12859-017-1934-
4. Ueda, Hiroki R., et al. “Tissue Clearing and Its Applications in Neuroscience.” Nature Reviews Neuroscience, vol. 21, no. 2, 2 Jan. 2020, pp. 61–79, https://doi.org/10.1038/s41583-019-0250-1.
5. NIST Big Data Interoperability Framework: Oct. 2019, www.govinfo.gov/content/pkg/GOVPUB-C13 279cb6d2aabf3f3d9e0c05b026f9c19b/pdf/GOVPUB-C13- https://doi.org/10.6028/nist.sp.1500-7r2.
6. Ajah, Ifeyinwa, and Henry Nweke. “Big Data and Business Analytics: Trends, Platforms, Success Factors and Applications.” Big Data and Cognitive Computing, vol. 3, no. 2, 2019, p. 32. mdpi, www.mdpi.com/2504-2289/3/2/32, https://doi.org/10.3390/bdcc3020032
7. Grolinger, Katarina, et al. “Data Management in Cloud Environments: NoSQL and NewSQL Data Stores.” Journal of Cloud Computing: Advances, Systems and Applications, vol. 2, no. 1, 2013, p. 22, https://doi.org/10.1186/2192-113x-2-22.
8. Osman, Ahmed M. Shahat. “A Novel Big Data Analytics Framework for Smart Cities.” Future Generation Computer Systems, vol. 91, no. 1, Feb. 2019, pp. 620–633, https://doi.org/10.1016/j.future.2018.06.046

9. Sharp, Michael, et al. "A Survey of the Advancing Use and Development of Machine Learning in Smart Manufacturing." Journal of Manufacturing Systems, vol. 48, July 2018, pp. 170–179, https://doi.org/10.1016/j.jmsy.2018.02.004
10. Gill, Sukhpal Singh. "AI for next Generation Computing: Emerging Trends and Future Directions." Internet of Things, vol. 19, no. 1, Mar. 2022, p. 100514, www.sciencedirect.com/science/article/abs/pii/S254266052200018X, https://doi.org/10.1016/j.iot.2022.100514.
11. Tataria, Harsh, et al. "6G Wireless Systems: Vision, Requirements, Challenges, Insights, and Opportunities." Proceedings of the IEEE, vol. 109, no. 7, July 2021, pp. 1166–1199, https://doi.org/10.1109/jproc.2021.3061701.
12. Lenarduzzi, Valentina, et al. "Does Migrating a Monolithic System to Microservices Decrease the Technical Debt?" Journal of Systems and Software, vol. 169, Nov. 2020, p. 110710, https://doi.org/10.1016/j.jss.2020.110710.
13. Ribeiro de Almeida, Damião, et al. "A Survey on Big Data for Trajectory Analytics." ISPRS International Journal of Geo-Information, vol. 9, no. 2, 1 Feb. 2020, p. 88, https://doi.org/10.3390/ijgi9020088.
14. Zhang, Zhao, et al. "Kira: Processing Astronomy Imagery Using Big Data Technology." IEEE Transactions on Big Data, vol. 6, no. 2, 1 June 2020, pp. 369–381, ieeexplore.ieee.org/ielaam/6687317/9098178/7549106-aam.pdf, https://doi.org/10.1109/tbdata.2016.2599926.
15. Zhou, Xiang, et al. "Latent Error Prediction and Fault Localization for Microservice Applications by Learning from System Trace Logs." Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 12 Aug. 2019, https://doi.org/10.1145/3338906.3338961.
16. Abernathey, Ryan P., et al. "Cloud-Native Repositories for Big Scientific Data." Computing in Science & Engineering, vol. 23, no. 2, 1 Mar. 2021, pp. 26–35, par.nsf.gov/servlets/purl/10287683, https://doi.org/10.1109/mcse.2021.3059437.
17. Ali, Waqas, et al. "A Survey of RDF Stores & SPARQL Engines for Querying Knowledge Graphs." The VLDB Journal, vol. 31, no. 3, 13 Nov. 2021, pp. 1–26, https://doi.org/10.1007/s00778-021-00711-3.
18. Debauche, Olivier, et al. "Cloud and Distributed Architectures for Data Management in Agriculture 4.0 : Review and Future Trends." Journal of King Saud University - Computer and Information Sciences, vol. 34, no. 9, Oct. 2021, https://doi.org/10.1016/j.jksuci.2021.09.015.
19. Zhang, Xiaolong, et al. "Zeus: Improving Resource Efficiency via Workload Colocation for Massive Kubernetes Clusters." IEEE Access, vol. 9, 2021, pp. 105192–105204, https://doi.org/10.1109/access.2021.3100082.
20. Verbraeken, Joost, et al. "A Survey on Distributed Machine Learning." ACM Computing Surveys, vol. 53, no. 2, July 2020, pp. 1–33, https://doi.org/10.1145/337745