upsolver

Technical Whitepaper:

Partitioning Data on S3 to Improve Performance in Athena/Presto

Executive Summary

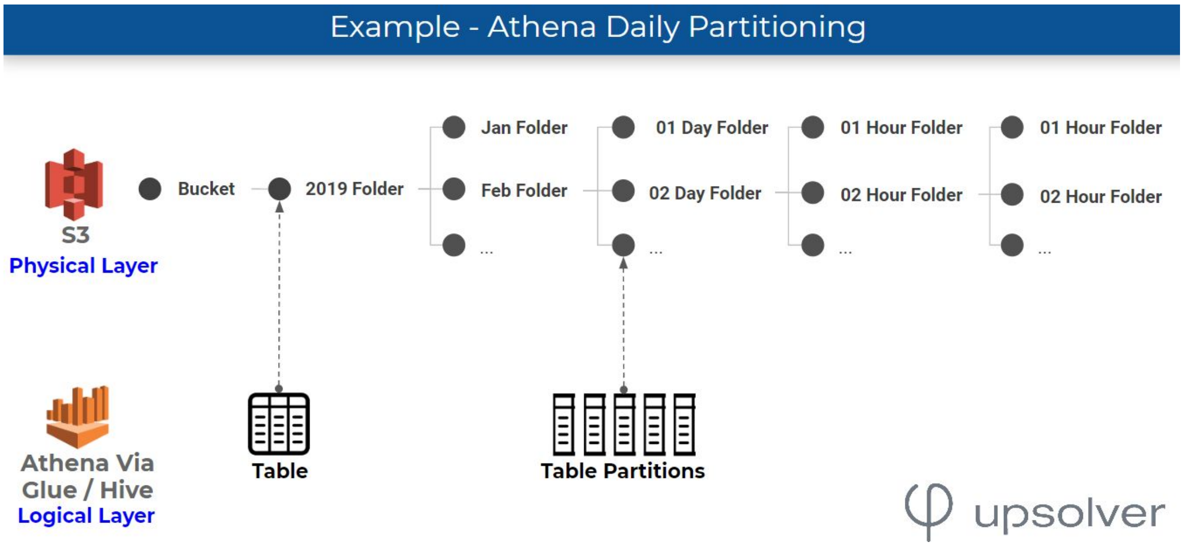
In an AWS S3 data lake architecture, partitioning plays a crucial role when querying data in Amazon Athena or Redshift Spectrum since it limits the volume of data scanned, dramatically accelerating queries and reducing costs ($5 / TB scanned). This article will cover the S3 data partitioning best practices you need to know in order to optimize your analytics infrastructure for performance.

Data partitioning is difficult, but Upsolver makes it easy. To learn how you can get your engineers to focus on features rather than pipelines, you can try Upsolver now for free or check out our guide to comparing streaming ETL solutions.

How to Partition for Performance

How partitioning works: folders where data is stored on S3, which are physical entities, are mapped to partitions, which are logical entities, in a metadata store such as Glue Data Catalog or Hive Metastore. As covered in AWS documentation, Athena leverages these partitions in order to retrieve the list of folders that contain relevant data for a query.

Data is commonly partitioned by time, so that folders on S3 and Hive partitions are based on hourly / daily / weekly / etc. values found in a timestamp field in an event stream. Here's an example of how Athena partitioning would look for data that is partitioned by day:



Matching Partitions to Common Queries

Athena matches the predicates in a SQL WHERE clause with the table partition key. The best partitioning strategy enables Athena to answer the queries you are likely to ask while scanning as little data as possible, which means you’re aiming to filter out as many partitions as you can.

For example - if we’re typically querying data from the last 24 hours, it makes sense to use daily or hourly partitions.

Monthly partitions will cause Athena to scan a month's worth of data to answer that single day query, which means we are scanning ~30x the amount of data we actually need, with all the performance and cost implication.

On the other hand, each partition adds metadata to our Hive / Glue metastore, and processing this metadata can add latency. A rough rule of thumb is that each 100 partitions scanned adds about 1 second of latency to your query in Amazon Athena. This is why minutely or hourly partitions are rarely used - typically you would choosing between daily, weekly, and monthly partitions, depending on the nature of your queries.

Partitioning by Timestamp: Best Practices

Athena runs on S3 so users have the freedom to choose whatever partitioning strategy they want to optimize costs and performance based on their specific use case. This would not be the case in a database architecture such as Google BigQuery, which only supports partitioning by time.

However, more freedom comes with more risks, and choosing the wrong partitioning strategy can result in poor performance, high costs, or unreasonable amount of engineering time being spent on ETL coding in Spark/Hadoop - although we will note that this would not be an issue if you're using Upsolver for data lake ETL.

A basic question you need to ask when partitioning by timestamp is which timestamp you are actually looking at. The main options are:

- **Server processing time -** the time an event was processed by the server that is writing to S3
- **Actual event time -** a timestamp found in the actual data indicating when the 'event' happened (e.g., the time a

user clicked on ad or a server log was generated)

- **Some combination of the two.**

Let’s take a closer look at the pros and cons of each of these options.

Partitioning by server processing time

When to use: if data is consistently ‘reaching’ Athena near the time it was generated, partitioning by processing time could make sense because the ETL is simpler and the difference between processing `and actual event time is negligible. Server data is a good example as logs are typically streamed to S3 immediately after being generated.

When not to use: if there are frequent delays between the real-world event and the time it is written to S3 and read by Athena, partitioning by server time could create an inaccurate picture of reality. Examples include user activity in mobile apps (can't rely on a consistent internet connection), and data replicated from databases which might have existed years before we moved it to S3.

ETL complexity: the main advantage of server-time processing is that ETL is relatively simple - since processing time always increases, data can be written in an append-only model and it's never necessary to go back and rewrite data from older partitions. On ingestion, it's possible to create files according to Athena's recommended file sizes for best performance.

Partitioning by actual event time

When to use: partitioning by event time will be useful when we’re working with events that are generated long before they are ingested to S3 - such as the above examples of mobile devices and database change-data-capture.

We want our partitiones to closely resemble the ‘reality’ of the data, as this would typically result in more accurate queries - e.g. most analytic queries will want to answer questions about what happened in a 24 hour block of time in our mobile applications, rather than the events we happened to catch in the same 24 hours, which could be decided by arbitrary considerations such as wi-fi availability.

When not to use: If it's possible to partition by processing time without hurting the integrity of our queries, we would often choose to do so in order to simplify the ETL coding required.

ETL Complexity: High - incoming data might be written to any partition so the ingestion process can't create files that are already optimized for queries. It's necessary to run additional processing (compaction) to re-write data according to Amazon Athena best practices.

Partitioning by a custom field and by time (multi-level partitioning)

When to use: we'll use multi-level partitioning when we want to create a distinction between types of events - such as when we are ingesting logs with different types of events and have queries that always run against a single event type. This might be the case in customer-facing analytics, where each customer needs to only see their own data.

It's usually recommended not to use daily sub-partitions with custom fields since the total number of partitions will be too high (see above). Monthly sub-partitions are often used when also partitioning by custom fields in order to improve performance.

When not to use: If you frequently need to perform full table scans that query the data without the custom fields, the extra partitions will take a major toll on your performance.

ETL complexity: High - managing sub-partitions requires more work and manual tuning. Also, some custom field values will be responsible for more data than others so we might end up with too much data in a single partition which nullifies the rest of our effort.

Combination of strategies

Since object storage such as Amazon S3 doesn't provide indexing, partitioning is the closest you can get to indexing in a cloud data lake.

If a company wants both internal analytics across multiple customers and external analytics that present data to each customer separately, it can make sense to duplicate the table data and use both strategies: time-based partitioning for internal analytics and custom-field partitioning for the customer facing analytics.

Choosing a Data Partitioning Strategy: 4 Questions to Ask

1. How will you manage data retention? If you’re pruning data, the easiest way to do so is to delete partitions, so deciding which data you want to retain can determine how data is partitioned.
2. Is data being ingested continuously, close to the time it is being generated? If so, you might lean towards partitioning by processing time.
3. Is the overall number of partitions too large? This could be detrimental to performance.
4. Is there a field besides the timestamp that is always being used in queries? If so, you might need multi-level partitioning by a custom field.

Automatic Partitioning for Athena

Partitioning data is typically done via manual ETL coding in Spark/Hadoop. As we've mentioned above, when you're trying to partition by event time, or employing any other partitioning technique that is not append-only, this process can get quite complex and time-consuming.

An alternative solution would be to use Upsolver, which automates the process of S3 partitioning and ensures your data is partitioned according to all relevant best practices and ready for consumption in Athena. Upsolver also merges small files and ensures data is stored in columnar Apache Parquet format, resulting in up to 100x improved performance.

Summary and Next Steps

As we've seen, S3 partitioning can get tricky, but getting it right will pay off big time when it comes to your overall costs and the performance of your analytic queries in Amazon Athena - and the same applies to other popular query engines that rely on a Hive metastore, such as Apache Presto.

Ready to take your data lake ETL to the next level? **Here's what you can do:**

- Learn how you can send data from Kafka to Athena in minutes.
- Get the Upsolver technical whitepaper to learn how we built our data lake ETL platform.
- Schedule a chat with one of our experts to learn how you can optimize your data lake for analytics.