Cloud-Based Data Pipelines and Interfaces for Batch Data

DAREDATA ENGINEERING

Sam Hopkins

Co-Founder and Technical Lead
sam@daredata.engineering
https://daredata.engineering

Main Points

- How you end up deciding to use batch processing technologies.
- An example architecture that uses Snowflake and AWS batch processing technologies to satisfy some real life business requirements.

General Advice

At DareData, we always start with use cases and work back from there. We actually use User Stories!

General Advice

At DareData, we always start with use cases and work back from there. We actually use User Stories!

You shouldn’t select your technologies based upon the type of data that you’re working with.

General Advice

At DareData, we always start with use cases and work back from there. We actually use User Stories!

You shouldn’t select your technologies based upon the type of data that you’re working with.

Rather, you should use some combination of data and business requirements to select a final data interface and work backward from there.

When Selecting an Interface

Make sure that the end user can use it.

When Selecting an Interface

Make sure that the end user can use it.

Choose protocol over technology ALWAYS.

> Choosing a data interface that is a technology rather than an open protocol leads to lockin and painful migrations down the line.

When Selecting an Interface

Make sure that the end user can use it.

Choose protocol over technology ALWAYS.

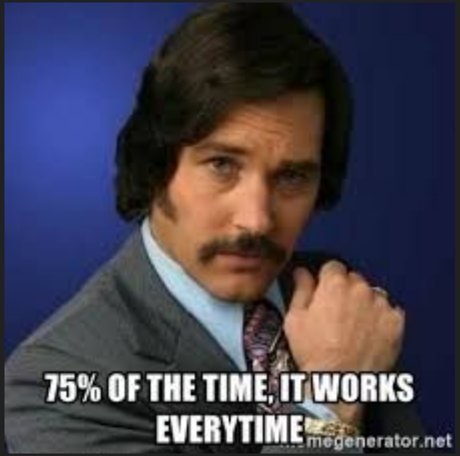
> Choosing a data interface that is a technology rather than an open protocol leads to lockin and painful migrations down the line.

Try to future-proof if possible. If you have two choices that serve your current use cases equally well and one is more open than the other, go with the more open one.

If you follow these guidelines...

75% of the time, it works every time

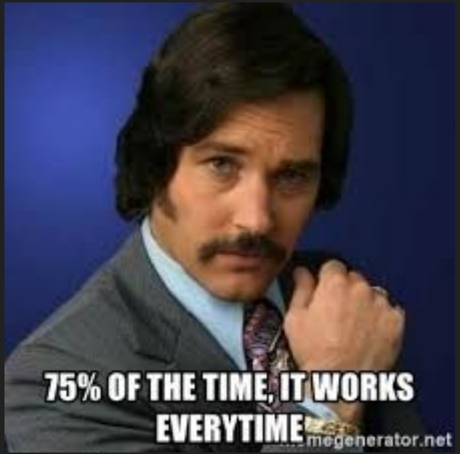
So, if you follow these guidelines, you’ll usually end up choosing SQL as your main data interface.



75% of the time, it works every time

So, if you follow these guidelines, you'll usually end up choosing SQL as your main data interface.

It's not a protocol per-se but the implementations typically share enough in common that SQL knowledge gained using one implementation is largely transferable to another.



Batch or Streaming Data

How do you know which one you’ve got?

Batch or Streaming Data?

Let’s first take a flawed (but still useful) look at the types of data that you would use streaming or batch technologies to process.

Streaming Data

Streaming technologies are used on data that

Typically comes from sensors. These can be anywhere from temperature / humidity sensors in a brewery to a mouse movement tracker on a web page.

Streaming technologies are used on data that

Typically comes from sensors. These can be anywhere from temperature / humidity sensors in a brewery to a mouse movement tracker on a web page.

When represented in tabular form, usually has a column with a timestamp and the rest as measurements in different dimensions.

Streaming technologies are used on data that

Typically comes from sensors. These can be anywhere from temperature / humidity sensors in a brewery to a mouse movement tracker on a web page.

When represented in tabular form, usually has a column with a timestamp and the rest as measurements in different dimensions.

Is typically time series data or another type of data where the ORDER IS VERY IMPORTANT.

Streaming technologies are used on data that

Typically comes from sensors. These can be anywhere from temperature / humidity sensors in a brewery to a mouse movement tracker on a web page.

When represented in tabular form, usually has a column with a timestamp and the rest as measurements in different dimensions.

Is typically time series data or another type of data where the ORDER IS VERY IMPORTANT.

Quite often, measures a low number of dimensions (temperature, humidity) or (x, y coordinates of a mouse click) or (page element that was clicked).

Streaming technologies are used on data that

When processing a single datapoint, you don’t need to look at much of the rest of the dataset, just the ones that are “near” it.

Streaming technologies are used on data that

When processing a single datapoint, you don’t need to look at much of the rest of the dataset, just the ones that are “near” it.

Usually works well with time series visualizations.

Batch Data

Batch technologies are used on data that

Typically comes from some kind of reporting.

Batch technologies are used on data that

Typically comes from some kind of reporting.

Either does not have a timestamp or if it does, is not required to be ordered when visualizing.

Batch technologies are used on data that

Typically comes from some kind of reporting.

Either does not have a timestamp or if it does, is not required to be ordered when visualizing.

Tends to have many more fields, lends itself to tabular representation, is a mix of types, and mixes have structured as well as unstructured data.

Batch technologies are used on data that

Typically comes from some kind of reporting.

Either does not have a timestamp or if it does, is not required to be ordered when visualizing.

Tends to have many more fields, lends itself to tabular representation, is a mix of types, and mixes have structured as well as unstructured data.

When processing it, you need access to large portions of the dataset to compute aggregate statistics

Batch or Streaming Tech

Which one should you use?

Batch or Streaming Tech?

Now let’s look at the most important question to determine whether or not you need batch or streaming technologies.

Batch or Streaming Tech?

Now let’s look at the most important question to determine whether or not you need batch or streaming technologies:

> Do you need to process and react to the data in real time?

Batch or Streaming Tech?

Now let's look at the most important question to determine whether or not you need batch or streaming technologies:

Do you need to process and react to the data in real time?

If the answer is "yes", you need streaming data technologies.

Batch or Streaming Tech?

Now let’s look at the most important question to determine whether or not you need batch or streaming technologies:

Do you need to process and react to the data in real time?

If the answer is “yes”, you need streaming data technologies

If the answer is “no”, you can use batch processing technologies.

Some Useful Exercises

Question

I need to raise an alarm within 30 seconds if the temperature in my brewery goes above 25 degrees

Question

I need to raise an alarm within 30 seconds if the temperature in my brewery goes above 25 degrees

Streaming

Question

I need to know how much we sold yesterday before the manager’s meeting at 9am every morning.

Question

I need to know how much we sold yesterday before the manager's meeting at 9am every morning.

Batch

Question

What was the average price of a product over the last 2 years.

Question

What was the average price of a product over the last 2 years.

Batch

Question

I need to preprocess my clickstream data by deduplicating clicks on the same element if they are within a few milliseconds of each other.

Question

I need to preprocess my clickstream data by deduplicating clicks on the same element if they are within a few milliseconds of each other.

This one could be more tricky… might depend on how quickly you need that data deduplicated and for what use case!

Fact

Fact

Most use cases can be satisfied with SQL as the interface and with pipelines (ETLs) implemented using batch processing technologies.

Implementing Batch Data Pipelines

Use Case

Let’s pick a real live use-case from a previous client called Daltix that provides price analytics via web scraping.

Daltix Context

Daltix is b2b.

Daltix Context

Daltix is b2b.

Plus, data is their thing.

Daltix Context

Daltix is b2b.

Plus, data is their thing.

They sell data, sometimes via a service or an app but the end product is always tabular data.

First, Select an Interface

First, we select the fundamental data interface.

First, Select an Interface

First, we select the fundamental data interface.

We’re going to build lots of solutions on top of it (web products and reports in particular) as well as make the data accessible to analysts.

First, Select an Interface

First, we select the fundamental data interface.

We’re going to build lots of solutions on top of it (web products and reports in particular) as well as make the data accessible to analysts.

Furthermore, we may want to give direct access to the data to Daltix’s clients and we don’t want to make them use a particular programming language or even something like a REST API as we aren’t sure of their programming abilities.

First, Select an Interface

First, we select the fundamental data interface.

We’re going to build lots of solutions on top of it (web products and reports in particular) as well as make the data accessible to analysts.

Furthermore, we may want to give direct access to the data to Daltix’s clients and we don’t want to make them use a particular programming language or even something like a REST API as we aren’t sure of their programming abilities.

It would be nice if they could be given the same interface that is used internally so we could leverage internal knowledge in providing support.

Surprise! SQL!

As what usually happens, we find that SQL satisfies the hard requirements as well as the general guidelines we outlined earlier.

Surprise! SQL!

As what usually happens, we find that SQL satisfies the hard requirements as well as the general guidelines we outlined earlier.

Daltix’s internal analysts use SQL.

Surprise! SQL!

As what usually happens, we find that SQL satisfies the hard requirements as well as the general guidelines we outlined earlier.

Daltix’s internal analysts use SQL.

The application developers know SQL.

Surprise! SQL!

As what usually happens, we find that SQL satisfies the hard requirements as well as the general guidelines we outlined earlier.

Daltix’s internal analysts use SQL.

The application developers know SQL.

The clients know SQL.

Surprise! SQL!

As what usually happens, we find that SQL satisfies the hard requirements as well as the general guidelines we outlined earlier.

Daltix's internal analysts use SQL.

The application developers know SQL.

The clients know SQL.

SQL brings much joy to the world.

Business Requirements

Business Requirements

Next we gather some other requirements of the data.

Business Requirements

Next we gather some other requirements of the data.

Namely, the question of real-timeness.

Business Requirements

Next we gather some other requirements of the data.

Namely, the question of real-timeness.

It turns out that the clients don’t need real-time data.

Business Requirements

Next we gather some other requirements of the data.

Namely, the question of real-timeness.

It turns out that the clients don't need real-time data.

The most stringent requirements for delivery are that once per day, the data needs to be in delivered around mid-morning.

Sooooo Batch

Given that the raw data has typically been scraped before 7am, we are clearly in batch territory as we have several hours to process something on the order of millions of json structures.

Raw Input / Output

The raw output of the scrapers is json which is super nice and the business cases are mostly satisfied by taking the json data structures and flattening them into tabular ones.

Extra Requirements

However, we have one extra fun little requirement which is that some very non-trivial code needs to do some NLP on a few of the unstructured fields to turn un-structured promo strings into structured data.

Architecture v1

Batch Pipeline

scrapers -> AWS s3 (scraper output data lake)

Batch Pipeline

scrapers -> AWS s3 (scraper output data lake)

-> Spark (Partitioning the data) -> AWS s3 (partitioned data lake)

Batch Pipeline

scrapers -> AWS s3 (scraper output data lake)

-> Spark (Partitioning the data) -> AWS s3 (partitioned data lake)

-> AWS Lambdas (promo extraction) -> AWS s3 (promo output data lake)

Batch Pipeline

scrapers -> AWS s3 (scraper output data lake)

-> Spark (Partitioning the data) -> AWS s3 (partitioned data lake)

-> AWS Lambdas (promo extraction) -> AWS s3 (promo output data lake)

-> Snowflake (Data Warehouses) -> Snowflake (Data Marts)

Tangent Time!

Airflow (Quick Note)

All of these operations are scheduled and coordinated using Airflow.

More Tangent Time!

About SQL Data Warehouses and Data Marts

A Data Warehouse’s main job is to be complete, accurate, and up-to-date and is not optimized for any specific use-case.

About SQL Data Warehouses and Data Marts

A Data Warehouse’s main job is to be complete, accurate, and up-to-date and is not optimized for any specific use-case.

A Data Mart’s main job is to ensure that specific business requirements can be met.

About SQL Data Warehouses and Data Marts

A Data Warehouse’s main job is to be complete, accurate, and up-to-date and is not optimized for any specific use-case.

A Data Mart’s main job is to ensure that specific business requirements can be met.

You want it to be AS EASY AS POSSIBLE to create Data Marts (because this is where the business value is directly generated) which means that you really do want your Data Warehouse and your Data Marts to be implemented in the same technology / network in order to avoid expensive export / import operations typically involved in ETLs.

Back On Track

Going through the components

scrapers -> AWS s3 (scraper output data lake)

-> Spark (Partitioning the data) -> AWS s3 (partitioned data lake)

-> AWS Lambdas (promo extraction) -> AWS s3 (promo output data lake)

-> Snowflake (Data Warehouses) -> Snowflake (Data Marts)

scrapers -> AWS s3 (scraper output data lake)

The scrapers write out to an S3 data lake in a way that is just segmented by the date that it was crawled. This is the very beginning of the pipeline.

-> Spark (Partitioning the data) -> AWS s3 (partitioned data lake)

Every hour, there is a spark job that reads in all of the data from the past hour and writes it out to another s3 bucket in a more organized way (i.e. more partitions).

-> Spark (Partitioning the data) -> AWS s3 (partitioned data lake)

Every hour, there is a spark job that reads in all of the data from the past hour and writes it out to another s3 bucket in a more organized way (i.e. more partitions).

It also does a few relatively simple transformations and deduplications. The output is JSON.

-> AWS Lambdas (promo extraction) -> AWS s3 (promo output data lake)

AWS Lambdas are triggered when the Spark job writes out the json files to the newly partitioned structure.

-> AWS Lambdas (promo extraction) -> AWS s3 (promo output data lake)

AWS Lambdas are triggered when the Spark job writes out the json files to the newly partitioned structure.

This trigger executes the code that turns the unstructured promo strings into structured data about the depth and types of the discount.

-> AWS Lambdas (promo extraction) -> AWS s3 (promo output data lake)

AWS Lambdas are triggered when the Spark job writes out the json files to the newly partitioned structure.

This trigger executes the code that turns the unstructured promo strings into structured data about the depth and types of the discount.

This type of processing requirement is significant because it requires code and is not something that can be expressed in simple set-theory based transformations that you might do in SQL.

-> Snowflake (Data Warehouses) -> Snowflake (Data Marts)

Now we are ready to ingest the data into a data warehousing technology that has an SQL interface.

-> Snowflake (Data Warehouses) -> Snowflake (Data Marts)

Now we are ready to ingest the data into a data warehousing technology that has an SQL interface.

At this stage, since we are working with data warehousing, we will pretty much bring the data as-is by flattening our JSON into a tabular structure.

-> Snowflake (Data Warehouses) -> Snowflake (Data Marts)

Now we are ready to ingest the data into a data warehousing technology that has an SQL interface.

At this stage, since we are working with data warehousing, we will pretty much bring the data as-is by flattening our JSON into a tabular structure.

Remember: Data Warehouses main job is to be complete, not necessarily optimized or easy to use.

Data Transformation in Snowflake

Now that we have a data warehouse in an SQL format that supports a DDL, we can subset and transform the data into use-case-specific tables that are derived from our data warehouses.

Data Transformation in Snowflake

IMPORTANT: at this point, data mart creation is pretty much self-service.

Data Transformation in Snowflake

IMPORTANT: at this point, data mart creation is pretty much self-service.

Anyone who knows SQL and can read a data dictionary can now create their own use-case-specific data marts whether it be for web products, reports, clients, etc.

Data Transformation in Snowflake

IMPORTANT: at this point, data mart creation is pretty much self-service.

Anyone who knows SQL and can read a data dictionary can now create their own use-case-specific data marts whether it be for web products, reports, clients, etc.

This is absolutely critical and does as much for future-proofing your tech stack as any other single thing.

Data Transformation in Snowflake

IMPORTANT: at this point, data mart creation is pretty much self-service.

Anyone who knows SQL and can read a data dictionary can now create their own use-case-specific data marts whether it be for web products, reports, clients, etc.

This is absolutely critical and does as much for future-proofing your tech stack as any other single thing.

Compare this to using a proprietary interface like SAS or SPSS and you are light years ahead in terms of flexibility, speed, and cost.

Architecture V3

(yes, we skipped V2)

Learnings and Changes Since I Was At Daltix

There have been some iterations and soon, we they hope to be replacing Spark and Lambda with Dask using an AWS batch backend.

Learnings and Changes Since I Was At Daltix

There have been some iterations and soon, we they hope to be replacing Spark and Lambda with Dask using an AWS batch backend.

Spark code and clusters are just too much hassle to maintain.

Learnings and Changes Since I Was At Daltix

There have been some iterations and soon, we they hope to be replacing Spark and Lambda with Dask using an AWS batch backend.

Spark code and clusters are just too much hassle to maintain.

There aren’t enough use cases for Lambdas to justify introducing a new technology into the stack.

Conclusions

Conclusions

Start with your business requirements.

Conclusions

Start with your business requirements.

Then select your technologies.

Conclusions

Start with your business requirements.

Then select your technologies.

Always prefer protocol to technology.

Conclusions

Start with your business requirements.

Then select your technologies.

Always prefer protocol to technology.

Batch processing covers the vast majority of use cases.

Fim

Questions?