perform Interactive analysis (Notebook)

Job

Declarative Pipeline (ETL)

SQL

compute is a very important thing to understand.

- All purpose compute (Classic compute)

Dedicated server to run your things.

infra (azure)

softwares (databricks)

DBU (databricks Unit)

0.75 DBU/h

$1/DBU cost

generally the compute to memory ratio is 1:4

16 gb ram / 4 cpu cores

32 gb ram / 4 cpu cores

16 gb ram / 8 cpu cores

https://www.databricks.com/product/pricing/datascience-ml

$0.4 (databricks per hour for this cluster)

https://azure.microsoft.com/en-in/pricing/details/virtual-machines/windows/#pricin
g

$0.2 (infra)

.2 (azure) + .4 (databricks) = $0.6
==========

All purpose compute: Provisioned compute used to analyse data in notebooks.

policies
==========

personal compute - allows users to create an individually assigned single node compute resource with minimal configurations.

shared compute: allows users to create larger multi node resources. Intended for multiple users to share.

power user compute - allows users to create larger multi node cluster.

job compute: allow users to create a general purpose default compute for jobs.

custom intern policy
====================

```
{
 "node_type_id": {
   "type": "fixed",
   "value": "Standard_D4s_v3"
 },
 "spark_version": {
    "type": "fixed",
    "value": "16.4.x-scala2.13"
  },
  "autotermination_minutes": {
    "type": "fixed",
    "value": 20
  }
```

}

permissions on compute:

- can attach to: allows you to attach your notebook to the compute
- can restart: allows you to start, restart and terminate the compute. also includes can attach to permissions.
- can manage: allows you to edit compute details, permissions and size. also includes can attach to and can restart permissions.

Pools - used to reduce the start and autoscaling times.

DBR (Databricks Runtime version)

this adds many features including

Delta Lake,
Installed Java, scala, python and R libraries
Ubuntu and its system libraries
GPU libraries

16.4

major version - 16

feature version - 4

LTS (long term support version)

Serverless compute
===================

classic compute

wait time for classic compute is a lot 5-7 minutes
choose so many infra settings for performance / cost

in serverless you can just pick a goal
- standard mode
- performance

server (infra + software)

classic

infra - Azure
software - Databricks (DBU's)

in case of serverless

infra - Databricks
software - Databricks

Environment in the right pane is applicable only for serverless

Memory - it refers to the memory used for this REPL

this notebook acts like a REPL -> Spark connect -> Spark Cluster

budget policy -> go to settings -> compute

importance of this buget policy - this can help us to know how many dbu's a user has consumed.

to check this, we have to query the system tables.

https://learn.microsoft.com/en-us/azure/databricks/release-notes/serverless/environment-version/one

SQL

Jobs

Notebooks

Declarative pipelines (DLT)

Simple / performant / maintainance free

serverless compute is a versionless product.

serverless overspend protection - to control long running queries.
serverless notebooks have a default execution timeout of 2.5 hours.

Python and SQL are the only supported languages when you go with serverless.

Delta Lake Session 1

======================

Its a file format..

1. what is a data lake?

A data lake is a storage repository that holds a vast amount of raw data in its

native format.

Data lake stores all the data in the form of files.

can hold any kind of data.

examples are: Amazon s3, ADLS gen2, GCS etc..

2. advantages of a data lake

cost effective

scalable

any kind of data (structured, semi structured, unstructured)

3. challenges of a data lake

ACID guarantees

when we perform transactions we required ACID guarantees.

Any database will provide these ACID guarantees.

HDFC Bank account

25000 Rs in your account

your friend initially has 50k

min balance of 5000

ACID

Atomicity- all or none

10k will be deducted from your account

your friends account will be added with 10k

Consistency

10k will be deducted from your account

your friends account will be added with 10k

lets say you are transferring 25k to your friend

0 Rs

75k Rs

Isolation

before 10 pm

10k will be deducted from your account- 10 pm

your friends account will be added with 10k- 10.02 pm

Reader- 10.01 (your balance and your friend)

10 < 10.01 < 10.02

your balance is 15k

your friends balance is 50k

Durability- changes should be permanent and system failure should not hamper the results..

ACID properties in a database

Delta Lake Session 2

=====================

Data lake do not provide ACID guarantees

when you run a spark job then multiple tasks underneath and each generate a part file..

1. A job is failing while appending the data

output folder- 5 part files.

you are running a job which will also create 5 part files.

2 files are generated...

after that the job fails...

7 files

the reader will read 7 files

Atomicity, Consistency

2. A job is failing while overwriting the data

overwrite is a 2 step process..

1) deleting what is already there

2) write the new files

5 part files are already present in the output folder

and after that we ran the job

2 part files are generated and then the job fails

Atomicty, Consistency, Durability

3. Simultaneous reads/writes

5 part available

and then you invoked the spark job

2 part are written

reader is trying to read while part file 3 is getting processed.

totally 7 files

here isolation has gone for a toss

4. Append the data with a different Schema

you have 5 part files... and each of these files have 5 columns..

Now weare running a job which will write 5 more part files.. but all these new

files have 7

columns..

totally 10 files are there

5 files have 5 columns each

5 files have 7 columns each

Data Reliablity issues

no data validation

we are corrupting our data in failure cases

NODMLoperations

Data Quality issues

Difficult to maintain historical versions

=> Delta Lake comes for our rescue

Delta Lake Session 3

=====================

Some improvement on top of data lakes that solves the above challenges.

Delta lake is an open source storage layer

it is like a small utility installed on your spark cluster.

Parquet format- it is a column based file format- very well used with Apache spark- it embeds the metadata

Delta = Parquet + Transaction logs

df for orders data

df.write.format("parquet")

df.write.format("delta")

5 tasks are running

5 partfiles will be created

output folder- 5 partfiles

_delta_logs

00000.json

operation 1- write

=====================

part-000.parquet

part-001.parquet

part-002.parquet

part-003.parquet

part-004.parquet

_delta_log

00000.json

Operation 2- append

======================

part-005.parquet

part-006.parquet

_delta_log

00001.json

Operation 3- append

======================

part-007.parquet

_delta_log

00002.json

for each write operation- All part files is written first- A transaction log file is added to _delta_log folder (Json format) for each read operation- transaction log files are read first

- part files are read based on the above log files

1. A job failing while appending

part-000.parquet

part-001.parquet

part-002.parquet

part-003.parquet

part-004.parquet

_delta_log

00000.json

add part-000.parquet

add part-001.parquet

..

..

..

5 more part files

part-005.parquet

part-006.parquet--> job fails

If a reader reads they will see only 5 part files...

Atomicity, Consistency

2. A job failing while overwriting

overwrite is a 2 step process..

1) deleting what is already there

2) write the new files

5 part files are already present in the output folder

and after that we ran the job

2 part files are generated and then the job fails

Atomicty, Consistency, Durability

with Delta Lake the previous files are not deleted

it will start writing the new files

if all the 5 files are written properly...

_delta_log

00000.json

============

add part-000.parquet

add part-001.parquet

add part-002.parquet

add part-003.parquet

add part-004.parquet

00001.json

==========

add part-005.parquet

add part-006.parquet

add part-007.parquet

add part-008.parquet

add part-009.parquet

remove part-000.parquet

remove part-001.parquet

remove part-002.parquet

remove part-003.parquet

remove part-004.parquet

add part-005.parquet

add part-006.parquet

add part-007.parquet

add part-008.parquet

add part-009.parquet

3) Simultaneous reads/writes

Simultaneous reads/writes

5 part available

and then you invoked the spark job

2 part are written

reader is trying to read while part file 3 is getting processed.

totally 7 files

here isolation has gone for a toss

00000.json

============

add part-000.parquet

add part-001.parquet

add part-002.parquet

add part-003.parquet

add part-004.parquet

00001.json

============

when a reader is reading they will still see just 00000.json

Delta Lake
============

storage account - ttmystorageaccount001

databricks connector - deltalakeconnector01

blob data contributor

storage credential

external location -

abfss://unitycatalog@ttmystorageaccount001.dfs.core.windows.net/catalog

https://ttmystorageaccount001.blob.core.windows.net/unitycatalog/catalog/__unitystorage/catalogs/758280e5-e8ae-48b0-840c-9f10d52cc821/tables/2c9e6121-c089-4416-8602-38aeb7f7339c/_delta_log/00000000000000000001.json

==========

Metastore -> Catalog -> Schema / Database -> Tables, Volume, Function, View, Model

Tables - structured Data

Volume - unstructured data, semi structured data, structured

================

Managed tables

Volumes

External tables

container name - externaldata
directory - orders

abfss://externaldata@ttmystorageaccount001.dfs.core.windows.net/orders

json1 - add parquet1

json2 - add parquet2
                remove parquet1

add parquet2

1,2,3 - parquet1

delete id 3

2,3 - parquet2

json - add parquet2

remove parquet1

10 million records in a file

1 record

parquet1 - 10 million records

parquet2 - 10 million records - 1 records

remove parquet1
add parquet2

10 million records - parquet1

you are deleting 1 record

a deletion vector file is created - remove id 3

remove parquet1
add parquet1 along with deletion vector

file1 - 1,2,3
file2 - 4,5

deleted 1,5

deletionvector1 - id 1
deletionvector2 - id 5

remove file1
remove file2
add file1 with deletion vector1
add file2 with deletion vector2

2,3,4

file2 - 2,3,4

file3 - updated record 3
deletion vector for file2

add file 3
remove file 2
add file 2 along with deletion vector