Big Data Platforms - Recitation Workshops

Workshop II
Data & Cloud Storage

21 December 2022

Yarden Fogel & Roei Arpaly

Data & Cloud Storage Agenda

Data & Engines

- Data Lakes, Data Warehouse
- SQL, NoSQL and Databases
- Files (text, csv, image, json, yaml, etc)
- Parquets (partitions, predicate pushdown, columnar storage)
- Big data engines, query optimisation, scaleup
- PySpark

Storage

- Cloud Storage, File System and Object Storage
- Storing Models (Pickle, Joblib)
- Model Store / Feature Store

Data Lake and Data Warehouse

- A **data lake** is a repository of data stored in its raw format.
- A data lake can include:
    - **structured** data from relational databases (rows and columns)
    - **semi-structured** data (logs, CSV, XML, JSON)
    - **unstructured** data (images, audio, video, binary data)
- Most of the data lakes nowadays are resides in a cloud storage (using the service of vendors such as Amazon, Microsoft or Google).

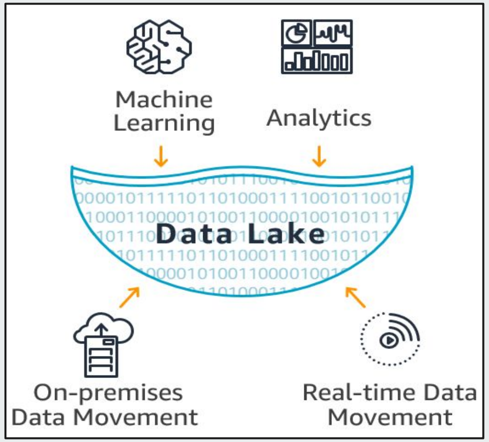
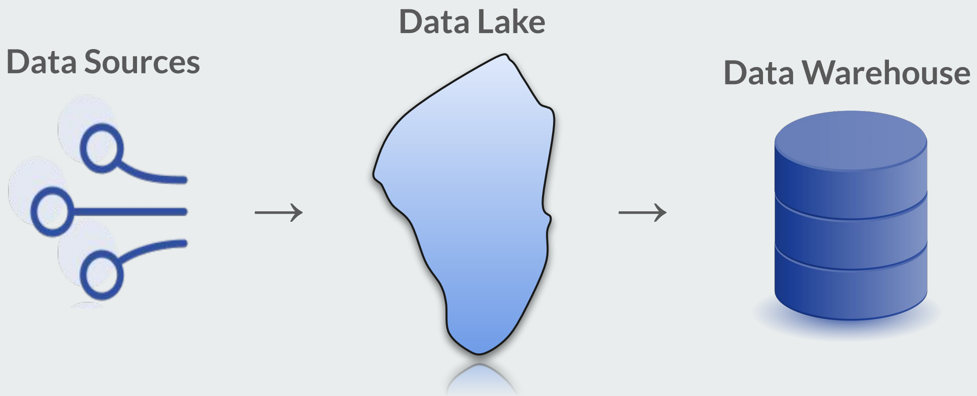


Image from AWS

Data Lake and Data Warehouse (Cont'd)

- A **data warehouse** (DW or DWH) is a system used for reporting and data analysis and is considered a core component of business intelligence (BI).
- They store processed (transformed) historical data in one single place that is used for tasks such as reporting, visualization, advanced analytics and machine learning.
- ETL and ELT are the two main approaches used to build a DW system (will be discussed later).



Relational and Non-Relational databases

- **SQL -** Structured Query Language, working with relational databases which store data in the form of schema, tables, columns, and rows.
  - Hard to scale, hard to manage data with different data models
  - Examples: SQLite, MySQL, PostgreSQL, MS SQL, etc

- **NoSQL** - a NoSQL database is one that is less structured, allows for more flexibility and adaptability. No predefined relationships or constraints in the database.
  - Popular types: key-value, document, graph and column.
  - MongoDB, DynamoDB, etc

Big Data Engines

- A Big data engine is a processing engine which is responsible for processing data, usually retrieved from cloud storage.
- Big data engines utilize a distributed parallel programming framework that enables processing very large amounts of data distributed across multiple nodes.
- **Master/slave** (or coordinator/worker) relationship
- **Optimisation** (analyzer, execution-plan, optimisation (predicate-pushdown etc))

Big Data Engines (Cont'd)

- Why is sorting an expensive task even with multiple workers (parallelisation)?
- Why is limiting the amount of rows after sorting more efficient in parallel?

Sorting example:

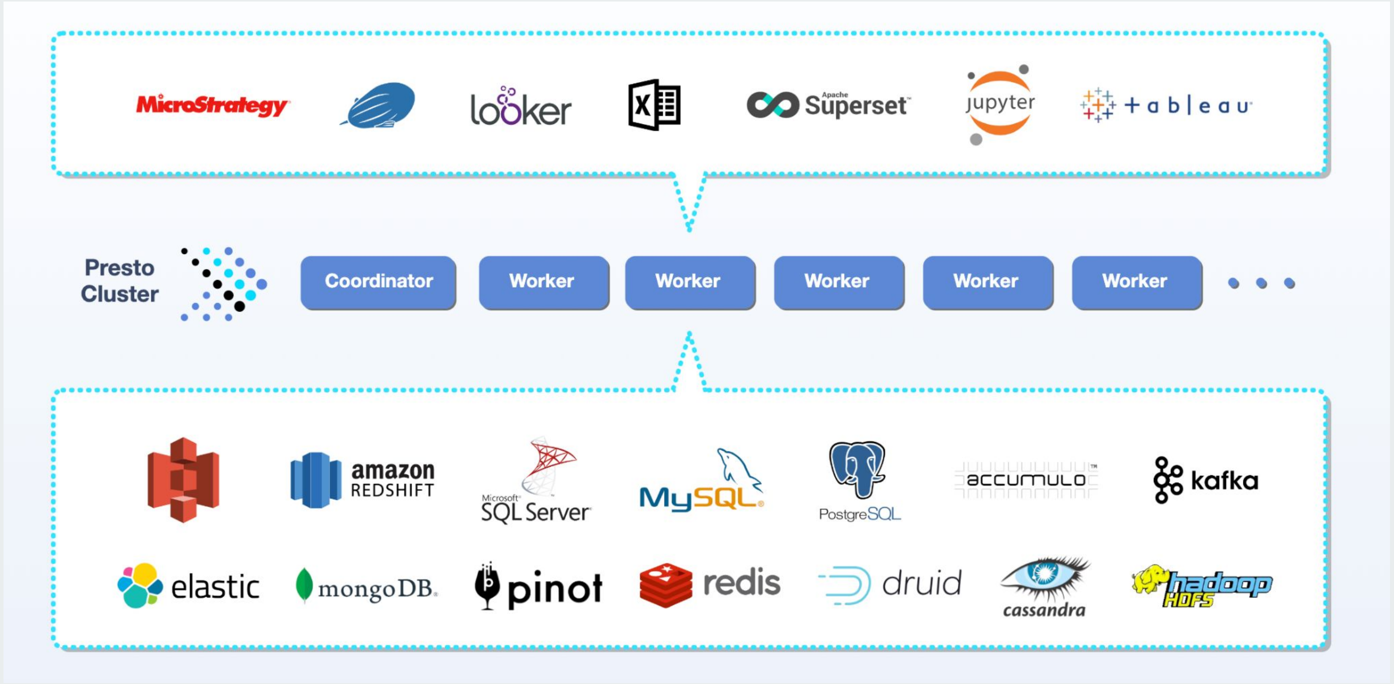
- Master with 10 Workers, 100M rows.

Big Data Engines (Cont'd)

BI tool

Engine

Database / File system / Object Storage



https://prestodb.io/

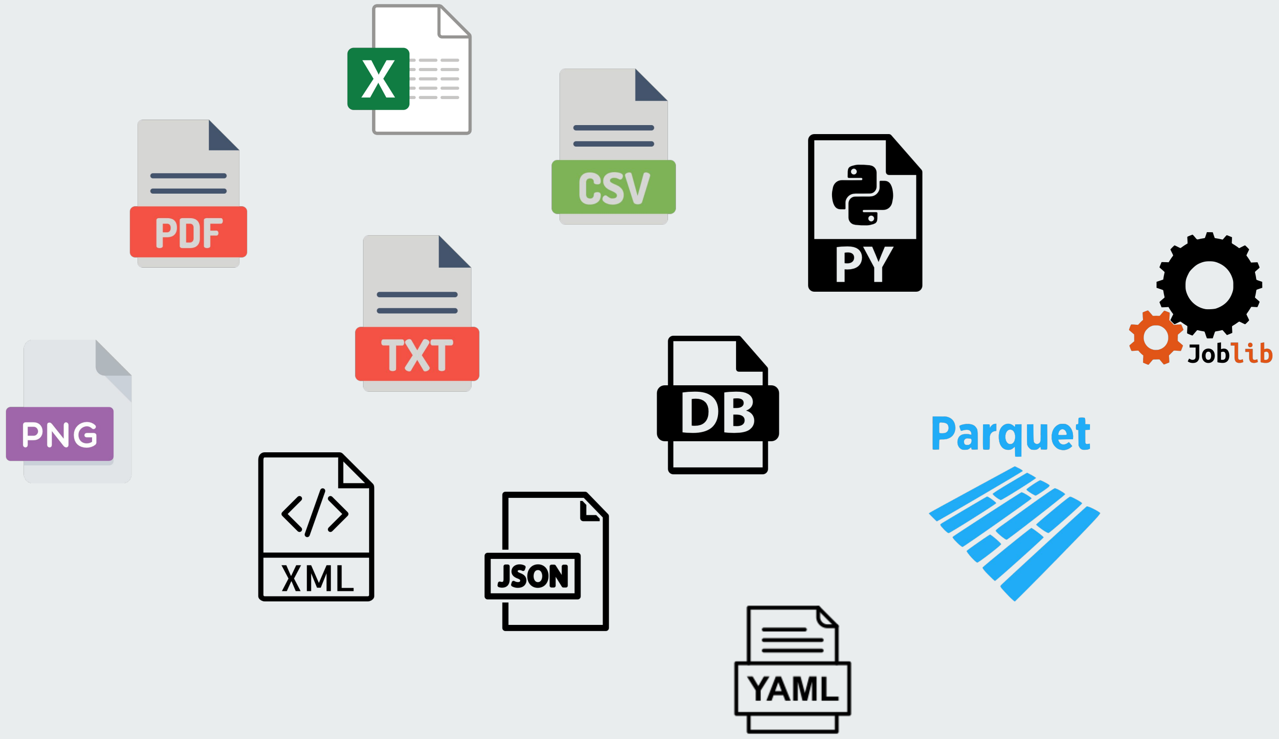
Reichman University

PySpark

- **PySpark SQL**: Spark SQL is a Spark module for structured data processing. It provides a programming abstraction called DataFrame (distributed datasets) and can also act as distributed SQL query engine.
    - Spark allows working with pandas API to scale your pandas workload out.

- **Pyspark MLlib:** Built on top of Spark, MLlib is a scalable machine learning library that provides a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.

Files



...and many more...

Reichman University

XML, JSON, YAML

XML

```
<note>
  <to>BDP students</to>
  <from>BDP staff</from>
  <heading>Recitation hours</heading>
  <body>Recitation will start today at 19:15</body>
</note>
```

JSON

```
{
 "to": "BDP students",
 "from": "BDP staff",
 "heading": "Recitation hours",
 "body": "Recitation will start today at 19:15"
}
```

YAML

```
to: BDP students
from: BDP staff
heading: Recitation hours
body: Recitation will start today at 19:15
```

Parquet

- **Apache Parquet** is an open source, column-oriented data file format for retrieval and storage of large amounts of data. It provides efficient data compression and encoding schemes.
- **Partitions:** A partition column is a column that the data files do not store values for. Instead, when writing the files, they will be divided (grouped) into folders / files based on partition column values.
- **Predicate Pushdown:** Predicate pushdown deals with what values will be scanned in the parquet files. If apply a filter on a specific column, to only return records with a specific value, the predicate push down will make parquet read only blocks that may contain this value.
- **Columnar Storage:** Parquet files allow the user to retrieve only the selected columns, reducing the amount of data transferred.

Parquet (Cont’d)

- **Parquet files management:** Parquet files are immutable, therefore only CREATE and OVERWRITE methods are supported.
- **Overwrite partitions:** If a parquet file is corrupted or is missing data, it can be overwritten. Hence, the file name should be easy to identify.
- **Time partitions**: Usually, one of the partition hierarchies should be time-based. Imagine you have a data pipeline which runs every day, in case there was a problem with a specific run, re-run will overwrite the files under the relevant time partition.
    - A dtstring (datetime in string format) column can be created for partitioning.

Query Optimization

- **Using partitions:** query over limited amount of folders of parquet files.
- **Predicate pushdown:** retrieve only limited information
- **Columnar format:** try not to use SELECT *
- **ORDER BY:** sorting only in the final query
- **JOIN:** small table to the bigger table
- **GROUP BY:** group by higher cardinality first, if possible join after the aggregation
- **DISTINCT / UNION:** use only if necessary
- **LIMIT:** retrieve less amount of data, important especially when sorting
- **APPROX aggregate functions:** more efficient functions (on behalf of accuracy)
- **REGEX:** more efficient than the LIKE operator

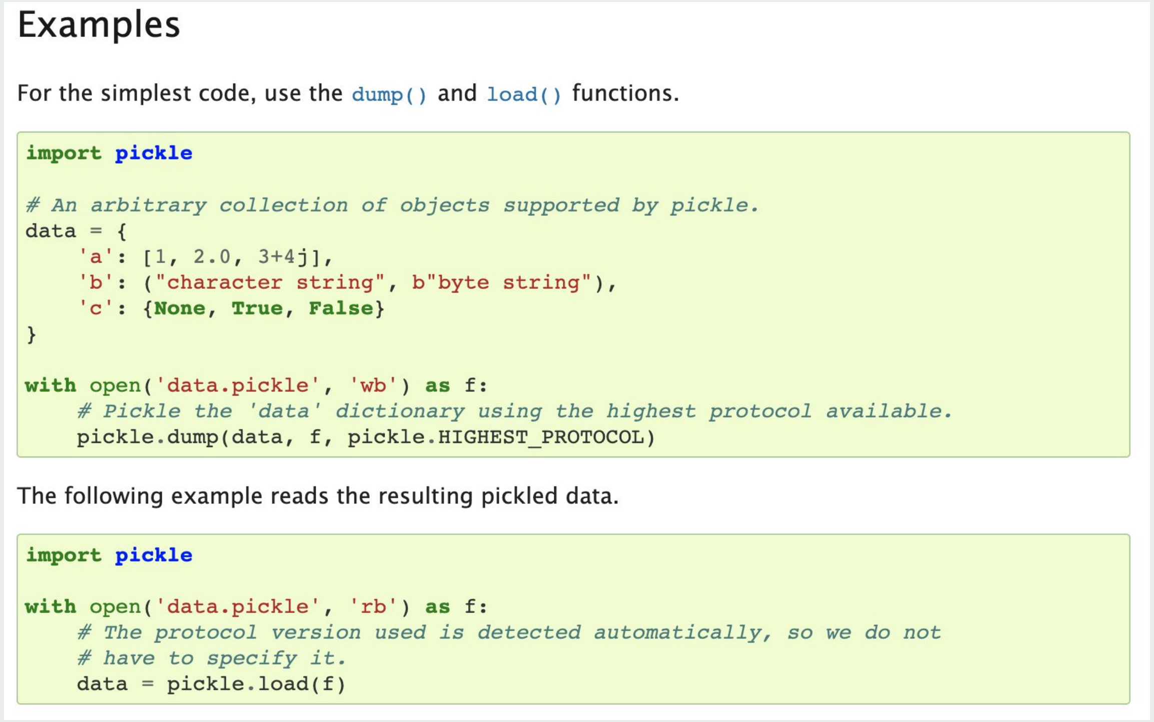
* Most engines try to automatically detect opportunities to optimize queries based on some of those examples and additional techniques

Pickle, Joblib

- **Pickle:** The pickle module implements binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream (serialization), and "unpickling" is the inverse operation.
  https://docs.python.org/3/library/pickle.html

- **Joblib:** Joblib is a set of tools to provide lightweight pipelining in Python.
  In particular: transparent disk-caching of functions and lazy re-evaluation (memoize pattern), easy simple parallel computing. Joblib is optimized to be fast and robust on large data in particular and has specific optimizations for numpy arrays.
  https://joblib.readthedocs.io/en/latest/

Pickle - examples from the docs



Examples

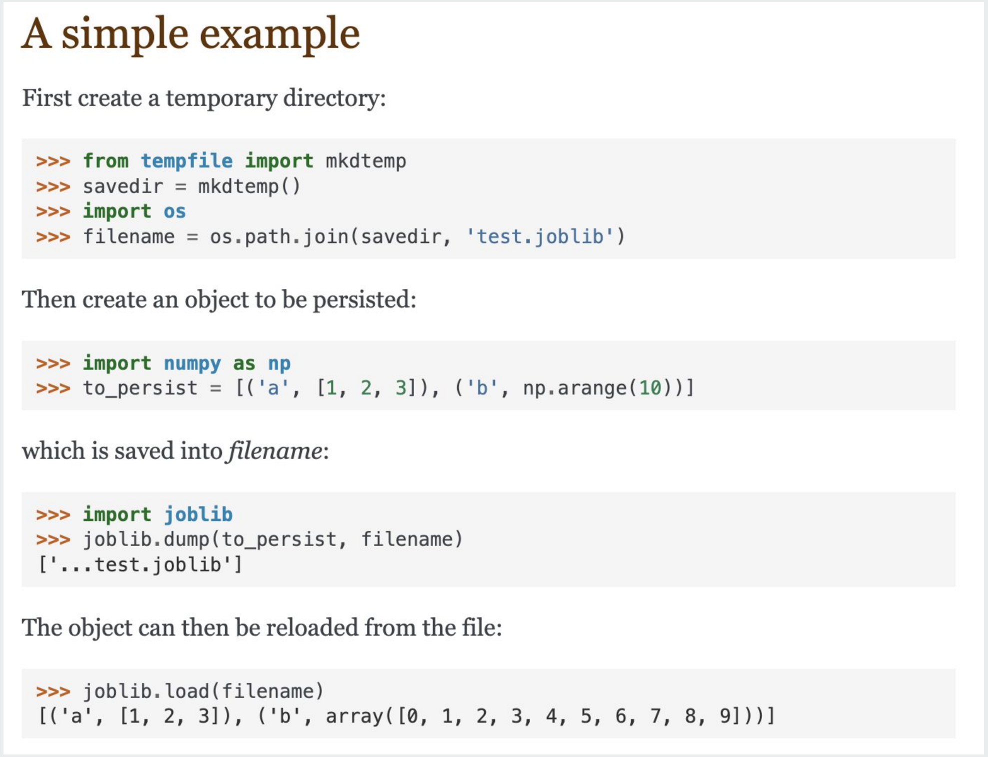
For the simplest code, use the `dump()` and `load()` functions.

```
import pickle

# An arbitrary collection of objects supported by pickle.
data = {
    'a': [1, 2.0, 3+4j],
    'b': ("character string", b"byte string"),
    'c': {None, True, False}
}

with open('data.pickle', 'wb') as f:
    # Pickle the 'data' dictionary using the highest protocol available.
    pickle.dump(data, f, pickle.HIGHEST_PROTOCOL)
```

The following example reads the resulting pickled data.

```
import pickle

with open('data.pickle', 'rb') as f:
    # The protocol version used is detected automatically, so we do not
    # have to specify it.
    data = pickle.load(f)
```

Joblib - example from the docs



A simple example

First create a temporary directory:

```
>>> from tempfile import mkdtemp
>>> savedir = mkdtemp()
>>> import os
>>> filename = os.path.join(savedir, 'test.joblib')
```

Then create an object to be persisted:

```
>>> import numpy as np
>>> to_persist = [('a', [1, 2, 3]), ('b', np.arange(10))]
```

which is saved into *filename*:

```
>>> import joblib
>>> joblib.dump(to_persist, filename)
['...test.joblib']
```

The object can then be reloaded from the file:

```
>>> joblib.load(filename)
[('a', [1, 2, 3]), ('b', array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]))]
```

Cloud Storage, File System and Object Storage

- **Cloud Storage:** stored in logical pools, on remotely multiple physical servers, usually owned and managed by a hosting company.
- **File System:** data structure that a operating system uses to control how data is stored and retrieved. The data is separated into pieces (folders and files) with a name for easy identification.
- **Object Storage:** Object storage manages data as objects, as opposed to other storage architectures like file systems which manages data as a file hierarchy. Each object typically includes the data itself, a variable amount of metadata, and a globally unique identifier.

Model Store

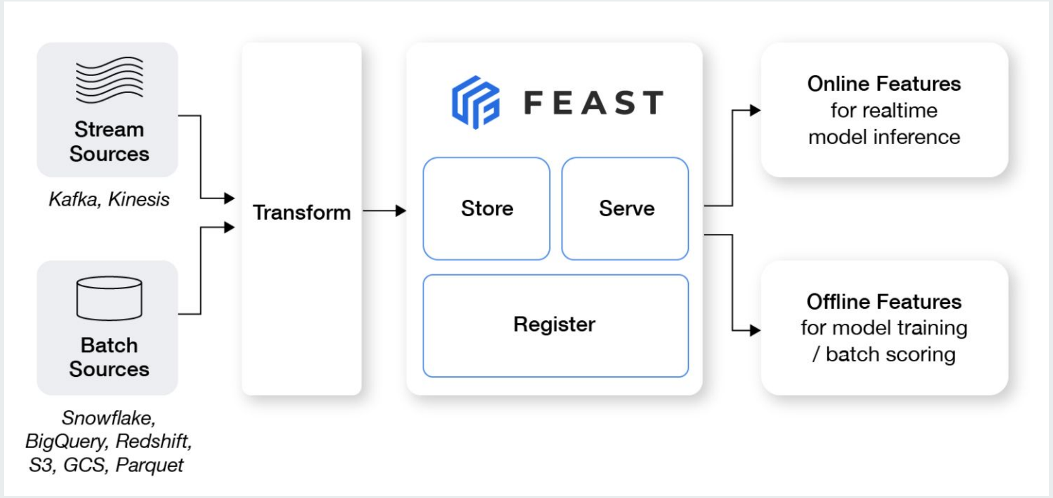
- **Model Store:** A model store is a central storage for ml engineers / data scientists to push, pull, version and manage their models and experiments, including the model files, artifacts, and metadata.

  A centralized storage for models allows to compare new models against existing deployed versions, track model performance over time, and more..

Source: https://imgur.com/gallery/w9DnEA7

Feature Store

- **Feature Store:** A feature store is an emerging infrastructure for organizing data for Machine Learning use cases, especially for real-time purposes. The features can be shared across different ML pipelines.


Stream Sources
Kafka, Kinesis
Batch Sources
Snowflake, BigQuery, Redshift, S3, GCS, Parquet
Transform
FEAST
Store
Serve
Register
Online Features for realtime model inference
Offline Features for model training / batch scoring

https://feast.dev/

Questions?

Open the Notebook