



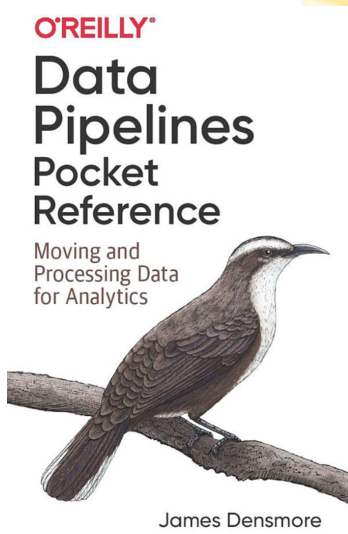
UMD DATA605 - Big Data Systems

Lesson 2.2: Data Pipelines

Instructor: Dr. GP Saggese, gsaggese@umd.edu

Data Pipelines: Resources

- Concepts in the slides
- Class project
- Mastery
 - [Data Pipelines Pocket Reference](#)

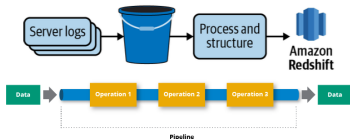


Data as a Product

- **Many services today "sell" data**
 - Services are typically powered by data and machine learning, e.g.,
 - Personalized search engine (Google)
 - Sentiment analysis on user-generated data (Facebook)
 - E-commerce + recommendation engine (Amazon)
 - Streaming data (Netflix, Spotify)
- **Several steps are required to generate data products**
 - Data ingestion
 - Data pre-processing
 - Cleaning, tokenization, feature computation
 - Model training
 - Model deployment
 - MLOps
 - Model monitoring
 - Is model working?
 - Is model getting slower?
 - Are model performance getting worse?
 - Collect feedback from deployment
 - E.g., recommendations vs what users bought
 - Ingest data from production for future versions of the model

Data Pipelines

- “Data is the new oil” ... but oil needs to be refined
- **Data pipelines**
 - Processes that move and transform data
 - **Goal:** derive new value from data through analytics, reporting, machine learning
- **Data needs to be:**
 - Collected
 - Pre-processed / cleaned
 - Validated
 - Processed
 - Combined
- **Data ingestion**
 - Simplest data pipeline
 - Extract data (e.g., from REST API)
 - Load data into DB (e.g., SQL table)



Roles in Building Data Pipelines

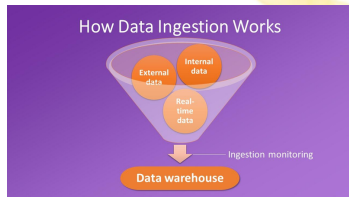
- **Data engineers**
 - Build and maintain data pipelines
 - Tools:
 - Python / Java / Go / No-code
 - SQL / NoSQL stores
 - Hadoop / MapReduce / Spark
 - Cloud computing
- **Data scientists**
 - Build predictive models
 - Tools:
 - Python / R / Julia
 - Hadoop / MapReduce / Spark
 - Cloud computing
- **Data analysts**
 - E.g., marketing, MBAs, sales, ...
 - Build metrics and dashboards
 - Tools:
 - Excel spreadsheets
 - GUI tools (e.g., Tableaux)
 - Desktop

Practical problems in Data Pipeline Organization

- **Who is responsible for the data?**
- **Issues with scaling**
 - Performance
 - Memory
 - Disk
- **Build-vs-buy**
 - Which tools?
 - Open-source vs proprietary?
- **Architecture**
 - Who is in charge of it?
 - Conventions
 - Documentation
- **Service level agreement (SLA)**
- **Talk to stakeholders on a regular basis**

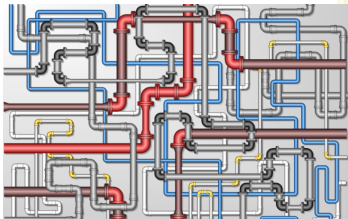
Data Ingestion

- **Data ingestion**
 - = extract data from one source and load it into another store
- **Data sources / sinks**
 - DBs
 - E.g., Postgres, MongoDB
 - REST API
 - Abstraction layer on top of DBs
 - Network file system / cloud
 - E.g., CSV files, Parquet files
 - Data warehouses
 - Data lakes
- **Source ownership**
 - An organization can use 10-1000s of data sources
 - Internal
 - E.g., DB storing shopping carts for a e-commerce site
 - 3rd-parties
 - E.g., Google analytics tracking website usage



Data Pipeline Paradigms

- There are **several styles of building data pipelines**
- **Multiple phases**
 - Extract
 - Load
 - Transform
- **Phases arranged in different ways** depending on philosophy about data / roles
 - ETL
 - ELT
 - EtLT



ETL Paradigm: Phases

- **Extract**

- Gather data from various data sources, e.g.,
 - Internal / external data warehouse
 - REST API
 - Data downloading from API
 - Web scraping

- **Transform**

- Raw data is combined and formatted to become useful for analysis step

- **Load**

- Move data into the final destination, e.g.,
 - Data warehouse
 - Data lake

- **Data ingestion pipeline = E + L**

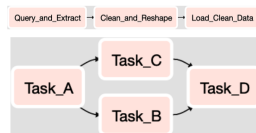
- Move data from one point to another
- Format the data
- Make a copy
- Have different tools to operate on the data

ETL Paradigm: Example

- **Extract**
 - Buy-vs-build data ingestion tools
 - Vendor lock-in
- **Transform**
 - Data conversion (e.g., parsing timestamp)
 - Create new columns from multiple source columns
 - E.g., year, month, day → yyyy/mm/dd
 - Aggregate / filter through business logic
 - Try not to filter, better to add tags / mark data
 - Anonymize data
- **Load**
 - Organize data in a format optimized for data analysis
 - E.g., load data in relational DB
 - Finally data modeling

Workflow Orchestration

- Companies have **many data pipelines** (10-1000s)
- **Orchestration tools**, e.g.,
 - Apache Airflow (from AirBnB)
 - Luigi (from Spotify)
 - AWS Glue
 - Kubeflow
- **Schedule and manage** flow of tasks according to their dependencies
 - Pipeline and jobs are represented through DAGs
- Monitor, retry, and send alarms



ELT Paradigm

- **ETL** has been the standard approach for long time
 - Extract → Transform → Load
 - **Cons**
 - Need to understand the data at ingestion time
 - Need to know how the data will be used
- Today **ELT** is becoming the pattern of choice
 - Extract → Load → Transform
 - **Pro:**
 - No need to know how the data will be used
 - Separate data engineers and data scientists / analysts
 - Data engineers focus on data ingestion (E + L)
 - Data scientists focus on transform (T)
 - ETL → ELT **enabled by new technologies**
 - Large storage to save all the raw data (cloud computing)
 - Distributed data storage and querying (e.g., HDFS)
 - Columnar DBs
 - Data compression



Row-based vs Columnar DBs

- **Row-based DBs**

- E.g., MySQL, Postgres
- Optimized for reading / writing rows
- Read / write small amounts of data frequently

OrderId	CustomerId	ShippingCountry	OrderTotal
1	1258	US	55.25
2	5698	AUS	125.36
3	2265	US	776.95
4	8954	CA	32.16

- **Columnar DBs**

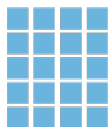
- E.g., Amazon Redshift, Snowflake
- Read / write large amounts of data infrequently
- Analytics requires a few columns
- Better data compression

Block 1	1, 1258, US, 55.25
Block 2	2, 5698, AUS, 125.36
Block 3	3, 2265, US, 776.95
Block 4	4, 8954, CA, 32.16

- **ETL**
 - Extract → Transform → Load
- **ELT**
 - Extract → Load → Transform
 - Transformation / data modeling ("T") according to business logic
- **EtLT**
 - Sometimes transformations with limited scope ("t") are needed
 - De-duplicate records
 - Parse URLs into individual components
 - Obfuscate sensitive data (for legal or security reasons)
 - Then implement rest of "LT" pipeline

Structure in Data (or Lack Thereof)

- **Structured data:** there is a schema
 - Relational DB
 - CSV
 - DataFrame
 - Parquet
- **Semi-structured:** subsets of data have different schema
 - Logs
 - HTML pages
 - XML
 - Nested JSON
 - NoSQL data
- **Unstructured:** no schema
 - Text
 - Pictures
 - Movies
 - Blobs of data



Structured Data

VS



Unstructured Data

Data Cleaning

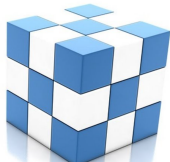
- **Data cleanliness**
 - Quality of source data varies greatly
 - Data is typically messy
 - Duplicated records
 - Incomplete or missing records (nans)
 - Inconsistent formats (e.g., phone with / without dashes)
 - Misabeled or unlabeled data
- **When to clean it?**
 - As soon as possible!
 - As late as possible!
 - In different stages
 - → Pipeline style: ETL vs ELT vs EtLT
- **Heuristics when dealing with data**
 - Hope for the best, assume the worst
 - Validate data early and often
 - Don't trust anything
 - Be defensive



OLAP vs OLTP Workloads

- There are two classes of data workloads
- **OLTP**
 - On-Line Transactional Processing
 - Execute large numbers of transactions by a large number of processes in real-time
 - **Lots of concurrent small read / write transactions**
 - E.g., online banking, e-commerce, travel reservations
- **OLAP**
 - On-Line Analytical Processing
 - Perform multi-dimensional analysis on large volumes of data
 - **Few large read or write transactions**
 - E.g., data mining, business intelligence

OLAP



OLTP



Challenges with Data Pipelines

- High-volume vs low-volume
 - Lots of small reads / writes
 - A few large reads / writes
- Batch vs streaming
 - Real-time constraints
- API rate limits / throttling
- Connection time-outs
- Slow downloads
- Incremental mode vs catch-up

Data Warehouse vs Data Lake

- **Data warehouse**

- = DB storing data from different systems in a structured way
- Corresponds to ETL data pipeline style
- E.g., a large Postgres instance with many DBs and tables
- E.g.,
 - AWS Athena, RDS
 - Google BigQuery



- **Data lake**

- Data stored semi-structured or unstructured
- Corresponds to ELT data pipeline style
- E.g., AWS S3 bucket storing blog posts, flat files, JSON objects, images



Data Lake: Pros and Cons

- **Data lake**
 - Stores semi-structured or unstructured data
- **Pros**
 - Cheaper cloud storage
 - Easier changes to types or properties
 - E.g., JSON documents
 - Data scientists
 - Initially unsure how to access and use data
 - Want to explore raw data
- **Cons**
 - Not optimized for querying like structured data warehouse
 - Tools query data lake similar to SQL
 - E.g., AWS Athena, Redshift Spectrum

Advantages of Cloud Computing

- **Ease of building and deploying:**
 - Data pipelines
 - Data warehouses
 - Data lakes
- **Managed services**
 - No need for admin and deploy
 - Highly scalable DBs
 - E.g., Amazon Redshift, Google BigQuery, Snowflake
- **Rent-vs-buy**
 - Easy to scale up and out
 - Easy to upgrade
 - Better cash-flow
- **Cost of storage and compute is continuously dropping**
 - Economies of scale
- **Cons**
 - The flexibility has a cost (2x-3x more expensive than owning)
 - Vendor lock-in