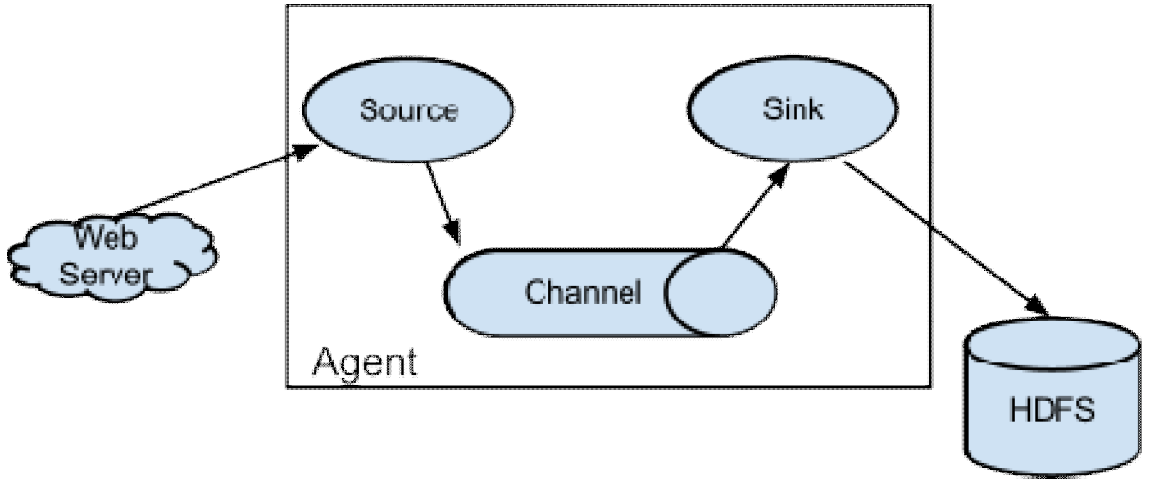
ECE 4813A

Project

Topic-1: Batch Processing (Python)

- Tools/Frameworks: Flume, MapReduce and Pig

  - MapReduce & Pig discussed earlier
  - Apache Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming data into the Hadoop Distributed File System (HDFS).



Topic-2: Micro-Batch Processing (Java)

- Tools/Frameworks: Storm Trident & Redis

  - Storm Trident is a micro-batch processing framework.
  - Trident is a high-level abstraction for doing realtime computing on top of Storm.
  - Trident has joins, aggregations, grouping, functions, and filters.
  - Trident adds primitives for doing stateful, incremental processing on top of any database or persistence store.

  - Redis is an open source, advanced key-value cache and store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets, sorted sets, bitmaps and hyperloglogs.

Topic-3: Stream Processing (Python)

- Tools/Frameworks: Storm & Kafka & Cassandra

  - Storm & Kafka discussed earlier

  - Apache Cassandra is a massively scalable open source NoSQL database.
  - Cassandra has a “masterless” architecture, meaning all nodes are the same.
  - Cassandra provides automatic data distribution across all nodes that participate in a “ring” or database cluster.
  - Cassandra delivers continuous availability, linear scalability, and operational simplicity across many commodity servers with no single point of failure.
  - Cassandra offers a powerful data model designed for maximum flexibility and fast response times.

Topic-4: In-Memory Processing (Python)

- Tools/Frameworks: Spark Streaming, Kafka & DynamoDB
  - Apache Spark is a fast and general-purpose cluster computing system.
  - Spark's primary abstraction is a distributed collection of items called a Resilient Distributed Dataset (RDD).
  - Spark Streaming is an extension of the core Spark API that enables scalable, high-throughput, fault-tolerant stream processing of live data streams.

Topic-5: NoSQL (Python)

- Tools/Frameworks: HBase and Hive

  - HBase discussed earlier

  - Apache Hive allows interactive SQL queries over petabytes of data in Hadoop.
  - The tables in Hive are similar to tables in a relational database, and data units are organized in a taxonomy from larger to more granular units.
  - Databases are comprised of tables, which are made up of partitions.
  - Data can be accessed via a simple query language and Hive supports overwriting or appending data.

Topic-6: Interactive SQL (Python)

- Tools/Frameworks: Google BigQuery

  - Google BigQuery enables fast, SQL-like queries against append-only tables, using the processing power of Google's infrastructure.
  - To query data, it is first loaded into BigQuery using the BigQuery console or BigQuery command line tool or BigQuery API.
  - Data can be either in CSV or JSON format.
  - The uploaded data can be queried using BigQuery’s SQL dialect.

Topic-7: Interactive SQL (Python)

- Tools/Frameworks: Amazon RedShift

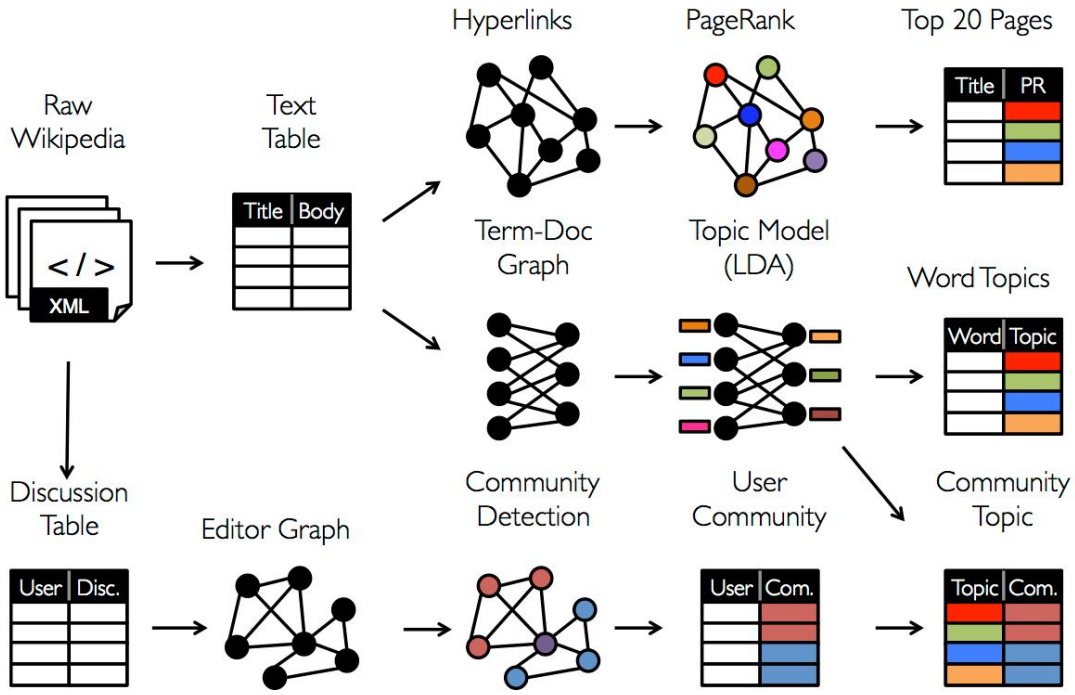
  - Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud.
  - An Amazon Redshift data warehouse is a collection of computing resources called *nodes*, which are organized into a group called a *cluster*.
  - Each cluster runs an Amazon Redshift engine and contains one or more databases.

Topic-8: Directed Acyclic Graphs (Python)

- Tools/Frameworks: Apache Spark
  - Spark features an advanced Directed Acyclic Graph (DAG) engine supporting cyclic data flow.
  - Each Spark job creates a DAG of task stages to be performed on the cluster.
  - Compared to MapReduce, which creates a DAG with two predefined stages - Map and Reduce, DAGs created by Spark can contain any number of stages.
  - This allows some jobs to complete faster than they would in MapReduce, with simple jobs completing after just one stage, and more complex tasks completing in a single run of many stages, rather than having to be split into multiple jobs.

Topic-9: Distributed Graph Processing (Java)

- Tools/Frameworks: Apache Giraph
  - Apache Giraph is an iterative graph processing system built for high scalability.
  - Currently used at Facebook to analyze the social graph formed by users and their connections.



Example problems

- Social media trends - trending topics
- Sentiment analysis from aggregated social media feeds
- Wikipedia data/Common Crawl data analysis – e.g. PageRank
- Clickstream analytics
- Customer recommendations
- Real-time machine sensor data analysis
- Credit card fraud detection - using real-time transactions
- Shipment monitoring - for fragile/expensive shipments
- Remote vehicle diagnostics - for a fleet of trucks
- Forest fire detection
- Weather monitoring

Datasets

Below is a list of pages where you can find public datasets for various problem domains:

- https://aws.amazon.com/datasets/
- http://hadoopilluminated.com/hadoop_illuminated/Public_Bigdata_Sets.html
- https://www.yelp.com/academic_dataset
- http://webscope.sandbox.yahoo.com/catalog.php

Design Patterns

- Cache-aside Pattern
- Circuit Breaker Pattern
- Compensating Transaction Pattern
- Competing Consumers Pattern
- Compute Resource Consolidation Pattern
- Command and Query Responsibility Segregation (CQRS) Pattern
- Event Sourcing Pattern
- External Configuration Store Pattern
- Federated Identity Pattern
- Gatekeeper Pattern
- Health Endpoint Monitoring Pattern
- Index Table Pattern
- Leader Election Pattern
- Materialized View Pattern
- Pipes and Filters Pattern
- Priority Queue Pattern
- Queue-based Load Leveling Pattern
- Retry Pattern
- Runtime Reconfiguration Pattern
- Scheduler Agent Supervisor Pattern
- Sharding Pattern
- Static Content Hosting Pattern
- Throttling Pattern
- Valet Key Pattern

Tasks

- Define the problem
- Identify format of data and some example queries
- Obtain a public dataset or implement a synthetic data generator
- Implement the system using the specified tool/framework
- Demonstrate use cases, example queries
- Indentify the design patterns applicable

Weekly Presentations

- Every Thursday teams will present their progress
- April-2: All teams to present for 6-8 minutes each
- April-9: Teams 1-4 to present for 15 minutes each
- April 16: Teams 5-9 to present for 15 minutes each
- April 21/23: CDR presentations and live demos

Preliminary Design Review (PDR)

- For the PDR, each team will be required to submit a presentation (5-10 slides) describing:
  - Problem selected for the project
  - Selection of the public dataset (or plan to implement synthetic data generator)
  - Proposed architecture design
  - Use cases and example queries

Critical Design Review (CDR)

- For the CDR, each team will be required to submit a report (15-20 pages), complete project source code and a presentation (10-15 slides) describing:
  - Problem
  - Tools/Frameworks used
  - Dataset used
  - Architecture design
  - Implementation/deployment details
  - Use cases, example queries
  - Results

Evaluation

- Problem identification, use of public dataset or synthetic data generator – 10 points
- PDR presentation – 10 points
- Architecture design and use of specified tool/framework – 20 points
- Source code – 20 points
- Use case demonstration – 10 points
- Use of patterns – 10 points
- CDR Report and presentation – 20 points

Topic Assignments

- Topic-1: Batch Processing (Python) - Flume, MapReduce and Pig
- Topic-2: Micro-Batch Processing (Java) - Storm Trident & Redis
- Topic-3: Stream Processing (Python) - Storm & Kafka & Cassandra
- Topic-4: In-Memory Processing (Python) - Spark Streaming, Kafka & DynamoDB
- Topic-5: NoSQL (Python) - HBase and Hive
- Topic-6: Interactive SQL (Python) - Google BigQuery
- Topic-7: Interactive SQL (Python) - Amazon RedShift
- Topic-8: Directed Acyclic Graphs (Python) - Apache Spark
- Topic-9: Distributed Graph Processing (Java) - Apache Giraph

References

- Hadoop: https://hadoop.apache.org/
- Flume: http://flume.apache.org/
- Pig: https://pig.apache.org/
- Storm: https://storm.apache.org/
- Storm Trident: https://storm.apache.org/documentation/Trident-tutorial.html
- Redis: http://redis.io/
- Spark Streaming: https://spark.apache.org/streaming/
- Kafka: http://kafka.apache.org/
- DynamoDB: http://aws.amazon.com/dynamodb/
- Cassandra: http://cassandra.apache.org/
- HBase: http://hbase.apache.org/
- Hive: https://hive.apache.org/
- Google BigQuery: https://cloud.google.com/bigquery/
- Amazon RedShift: http://aws.amazon.com/redshift/
- Spark GraphX: https://spark.apache.org/graphx/
- Apache Giraph: http://giraph.apache.org/
- Apache Tez: http://tez.apache.org/
- Design Patterns: https://msdn.microsoft.com/en-us/library/dn568099.aspx