CS 257 Database System Principles

BDaaS – Real time data stream processing and analytics using IOT connected devices

Team Members:

Charles MacKay
Jason Tellis
Piyush Bajaj

Description of the problem:

In the modern era of computing, there are millions if not billions of devices generating data. This large volume of streaming data can be sent to the cloud to process and generate useful insights for the user's business or product. We are designing a Big Data as a Service (BDaaS) framework. This will enable processing of multiple data streams from various IoT devices to provide real time statistics and critical messages for anomaly detection.

Motivation for the problem:

Use case examples of our software service can be a farmer with IoT connected sensors on his crops. They can monitor moisture levels and nutrient content in real time, as well as gather statistics over time regarding their crop's health. Machine learning can be applied to determine the correct amount of water and fertilizer to be applied to the crops for maximal yield and cost effectiveness.

We can provide the whole framework for feeding input data via REST APIs over a web interface and provides a scalable, high-throughput, fault-tolerant stream processing of live data streams on the fly.

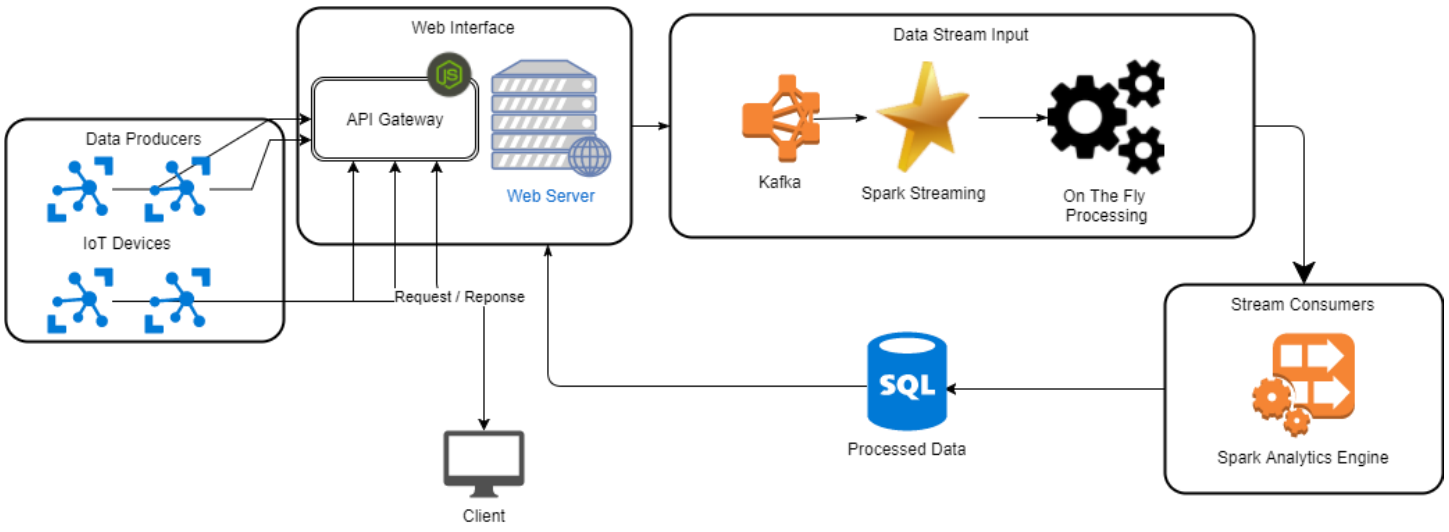
The challenging aspect of this project is to get the ecosystem of multiple software stacks working in harmony to provide a useful API interface to the end user, as well as developing algorithms for data crunching and analytics will have to be written from scratch.

Implementation

The basic architecture will consist of data generating sources (producers) that connect to a webserver via API gateway. This data will be fed into a stream processor Apache Kafka and Spark Streaming (consumer) that will then provide analytics. Processed data will be stored in a relational database (PostgreSQL, MySQL) that the API interface can allow users to query the database and retrieve analytic results. The API will also provide predictive analysis using machine learning modules.

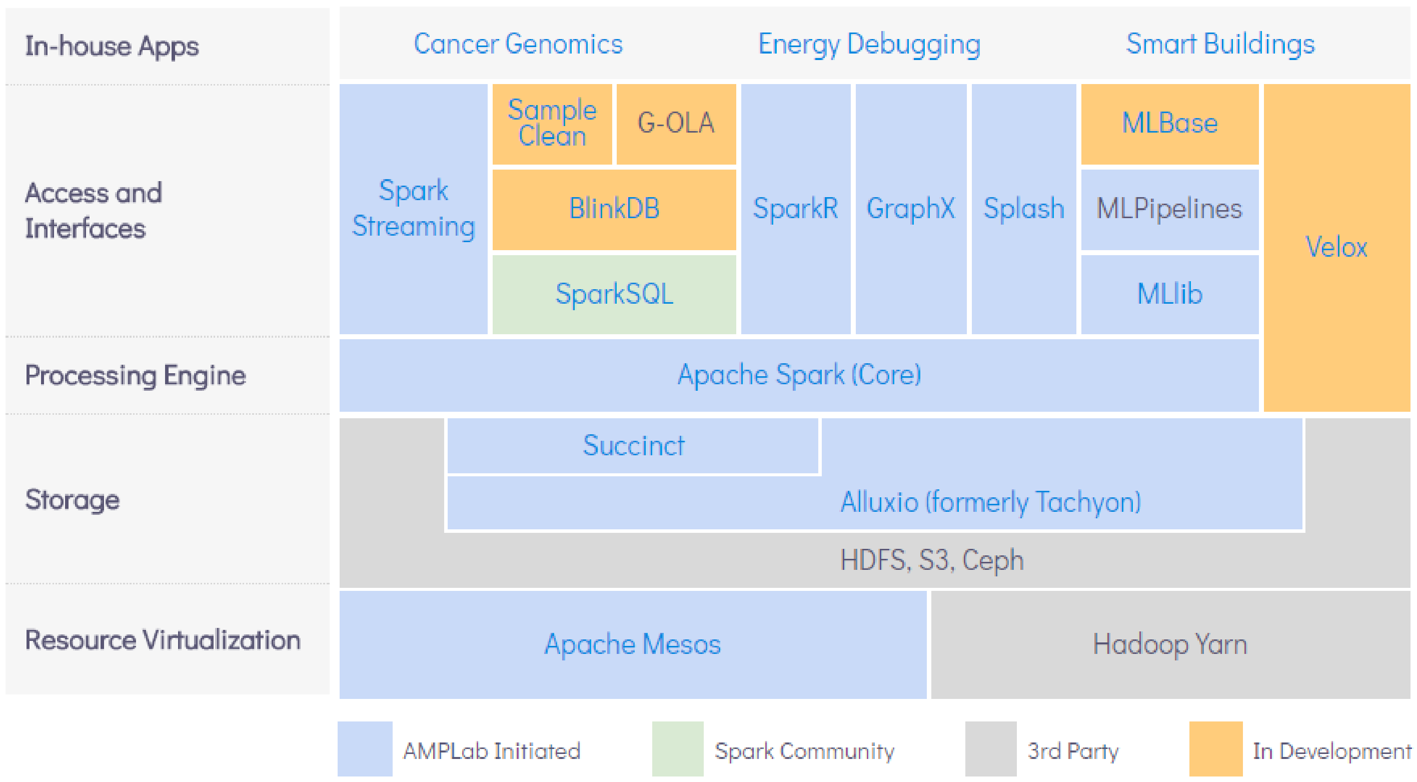
This framework can be really useful when a user wants to do data processing in real time using a minimal setup of the software stack and don't really want to invest time in the implementation of the data streaming behind the scenes. This will provide the user with statistics and graphable results they can use to analyze or predict future trends in their data.

System Diagram



Previous work related:

BDAS, the Berkeley Data Analytics Stack, is an open source software stack that integrates software components being built by the AMPLab to make sense of Big Data.



- https://amplab.cs.berkeley.edu/software/
- https://amplab.cs.berkeley.edu/publication/mllib-machine-learning-in-apache-spark/
- https://amplab.cs.berkeley.edu/publication/spark-sql-relational-data-processing-in-spark/
- https://amplab.cs.berkeley.edu/publication/faster-jobs-in-distributed-data-processing-using-multi-task-learning/

- https://aws.amazon.com/emr/
- https://mapr.com/products/mapr-converged-data-platform/

Methodology and plan for our project – Schedule and milestones:

0. **Week 0: (2017-10-9)**
    - Define the architecture, platform, and problem to solve
    - Find potential datasets
    - Submit proposal
    - Learn the Apache Big Data frameworks and ecosystems
1. **Week 1: (2017-10-16)**
    - Setup infrastructure
        - Setup hardware equipment
            - Make environment (Ubuntu, centos)
            - Open port on router to have access to outside world
        - Setup Big Data Ecosystem
            - Spark
            - Kafka
            - SparkSQL + RDBMS
        - Setup web API interface (Node, Express)
            - Node.js server REST API
                - Be able to accept POST requests from IOT devices
2. **Week 2 (2017-10-23)**
    - Setup pipeline
        - Connect web interface (Node.js) to big data engine
        - Feed incoming data into Big Data Engine
3. **Week 3 (2017-10-30)**
    - Develop algorithms for data crunching
    - Machine learning for anomaly detection and predictive analysis
    - Extend API to handle GET requests for analytics queries
4. **Week 4 (2017-11-6)**
    - Continue general work
5. **Week 5 (2017-11-13)**
    - Submit Milestone 1 status report
6. **Week 6 (2017-11-20)**
    - Handle heavy workloads from multiple data sources
    - Fix any bugs
7. **Week 7 (2017-11-27)**
    - Continuous improvements.
    - Have functioning system ready for demo
8. **Week 8 (2017-12-4)**
    - Work on final report and demos
9. **Week 9 (2017-12-11)**
    - Wrap up final project report and demo

Resources needed:

- Server (*nix)
  - Web Server
    - Node.js, Express
  - Big Data Engine
    - Spark, Spark Streaming, Spark MLib
    - Kafka
    - Spark SQL
      - Relational Database (MySQL, Postgres)

Workload distribution:

Charles MacKay:

- Machine learning for anomaly detection and predictive analysis.
- Setup Big Data Ecosystem

Jason Tellis:

- Extend API to handle GET requests for analytics queries.
- Setup hardware equipment.

Piyush Bajaj:

- Develop algorithms for data crunching.
- Setup web API interface (Node, Express).