Calculate the remaining life of household appliances

System architecture of Group 9 - Scalable computing

Sensor layer

- Data will be created and streamed continuously from sensors. Each sensor is a different type of household appliance.
- There will be several instances of the same type of appliance.
- Each sensor may send different types of data (temperature readings, electrical consumption, uptime, images etc.). These will be narrowed down later.
- Sensors will be simulated by lightweight web servers (Go, Flask, Node JS etc.).
- Different data-generation functions will be employed (exponential decay etc.) to get some patterns in.

Data ingestion layer

- Data streamed from sensors will pass through a security layer to ensure the origin of the data.
- Data streamed from sensors will be aggregated using an ingestion service (Apache Kafka).
- A message queue (RabbitMQ) will listen to packages from the ingestion service and deliver them to an HTTP server (Node JS).
- The HTTP server provides an authenticated interface to the database. The server authenticates itself and inserts the data in the database.
- **Lambda architecture Kappa architecture**

Database layer

- Since we are dealing with fast sensor data, CassandraDB will be the database of choice, due to its implementation of SSTables with consistent hashing.
- The database will contain data of the form {key, value}, where the key will constitute a timestamp of the given value.
- Each row in the database relates to an appliance.
- Replication strategy between cassandra nodes to ensure reliability and fault tolerance across node instances.
- **Write results back to some database**

Backend API / Computational layer

- A RESTful API (Flask / Node JS) will be created to serve the user requests related to the status of each household appliance.
- The backend will read from the database, run some function on the data, and respond with the result.
- Authentication could also be implemented through JWT with users stored in MongoDB.
- **Map/reduce; spark; algorithm that works on top of map/reduce (something more interesting than word count). CLUSTERING**

Frontend layer

- We assume that the application should be in the form of a website through which a user can login (or just land on) and query the status of their appliances.
- The frontend will be done in an SPA VueJS application with Buefy and SASS.
- Data shown live through graphs (Chart JS) / raw.
- The website will be served by NGINX.

Container layer

- Every service will be containerized using Docker.
- Different services will run on different machines (our laptops or potentially AWS/**GCP**), coordinated and managed with Kubernetes.