Cassandra Browser Tool

(Using Cassandra and Python)

Group Number 13

Nitin Agarwal Prajit Patil Pushyami Srikanth R

Phase 2 Submission

Brief description/Overview

Create a Cassandra Browser tool using python which gives interface for the user to access Cassandra Database server with the following **Functionalities**.

1. **Keyspace Operations** : A section for creating a keyspace, dropping keyspace and retrieving information about keyspaces.

2. **Column Family Operations** : A section for creating a column family, indexing a column family, get the description of a column family, dropping column families.

3. **Rows and Columns Operations** : A section for inserting rows and columns, deleting rows and columns and altering rows and columns.

4. **Managing Cluster** : A section that will show the backend details such as cluster, machine added to cluster, load on each machine, a section for user to add a machine just by giving IP address, etc.

5. **Understanding Application** : A section that will provide the documentation to the user explaining the architecture and working of our application and how to use.

Main Goals of our Project :

- Create a web application that interacts with Cassandra (Database server)
- Web Base, so can access using any browser.
- Managing and monitoring backend system(cassandra nodes) very easily.
- Help user insert into database without knowing anything about cassandra query language.
- It should be easy for user(with less cassandra knowledge) to use our application

Extra Functionalities :

- A Login section so that only authorized user has access to modify the data.
- User can migrating the data when require.
- Load balancing

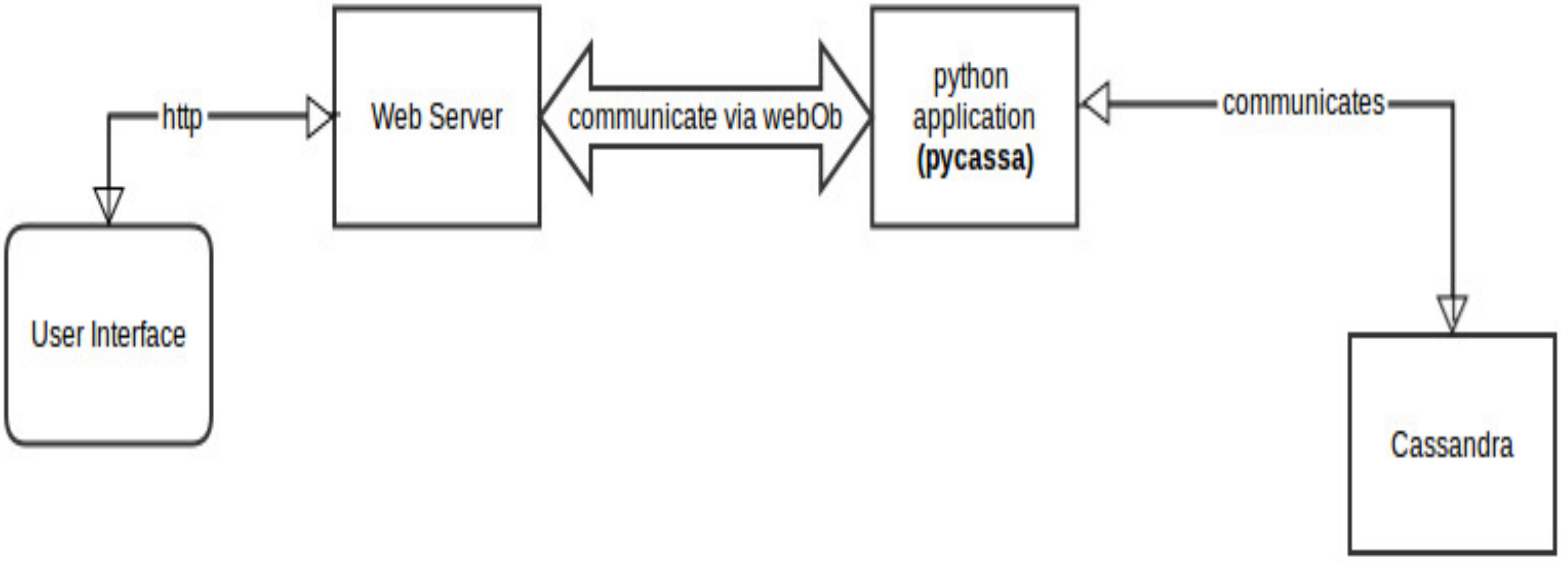
Architecture

Back end

- Creating cassandra cluster with two to three nodes.
- Writing a python application that will execute queries on cassandra (use of pycassa library in python for communication with cassandra).
- Use of WebOb library to create a request and response object which will be use as interface communication between web servers and our python applications.
- A python web application which will get request from user forward it to python application, get the result from it, then generate html pages using Jinja 2 Templates which will be send back to user.

Front end

- Listing out all the functionalities of our application.
- Designing a layout of our web application.



- Appropriately displaying the information retrieved from the server.
- Installation of Jinja2 templates and WebOb.
- Understanding the working of WebOb framework.
- Documentation on how to use our web application.

Layout
User interface will be divided into following sections:

- **Dashboard** : Get all machines information, which are connected by Cassandra Alerts and show status of machines, disk health, network latency.

- **Machine View** : View of machines which are connected, by pictorial representation so the user can directly go to machines by clicking on UI and access the Keyspace and column family.

- **Data Model** : This section lists underlying structure/properties of keyspaces, cluster and column family related to keyspaces.
  Functionality handled for example
    - Keyspace
        - Replication factors for given keyspace
        - List other details related to key space
    - Column family
        - List details related to column family
    - Cluster
        - List all nodes connected to it.
        - Other properties such as max_overflow,pool_size, pool_timeout,max_retries etc

- **Data view** : This shows the information of the key value, Indexes and Column family.

  Functionality handled for example
    - Cluster
        - Add a node to cluster
        - Remove a node from cluster
        - Migration of data if required
    - Keyspace
        - List all key space

    - Create a key space
    - Delete a keyspace
- Column family
    - List all column families details
    - Insert column families in given keyspace
    - Delete all column families from given keyspace
    - Alter all column families in given keyspace
- Indexes
    - List all Indexes details
    - Insert Index to given keyspace
    - Delete all Indexes from given keyspace
    - Alter all Indexes in given keyspaces.

Tools/Libraries Used

1. **Cassandra 2.0.1** : The Apache Cassandra database provides high scalability and high availability without compromising performance. Linear scalability and prove fault tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's data model offers the convenience of column indexes with the performance of log structured updates, strong support for denormalization and materialized views, and powerful built in caching.
   Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure.
   Basically this will provide as a platform for the database management server.

2. **Python Programming Language**

3. **Pycassa** : Python client API with features such as connection pooling, SuperColumn support, and a method to map existing classes to Cassandra column families.

4. **Webob** : WebOb is a library to create a request and response object. It's centered around the WSGI model. Requests are wrappers around the environment.
   WSGI more specifically is made up of an *application* and a *server*. The application is a function that receives the request and produces the response. The server is the thing that calls the application function.

5. **Jinja2 Templates** : Jinja2 is one of the most used template engines for Python. We will be using to develop our User frontend interface.

Explanation of the work done so far

- Understanding the working of cassandra.
- Installing cassandra on single node and performing basic operations on it using cassandra CLI.
- Performing operation on cassandra using pycassa library (code attach below).
- Requirement gathering that includes functionality we should have in our application.
- Working on layout of our application.

Issues Encountered/Foreseen

- When user requests for contents of column family, which is huge, we need to handle:
    - How many rows we should retrieve from the server (If we fetch too many rows, the Cassandra server might over-allocate memory and fail).
    - How do we make sure user is able to randomly access any content.
- How do we display data specially if there are many columns related to keys.
- Not able to figure out or catch the exceptions generated if the PyCassa library function fails because of improper input.

Link to a public code repository/file sharing tool that contains your code written so far (remember to keep it public/available to anyone with the link)

Our code for the project can be checked out at github link which is public : https://github.com/CloudProjectWork/CassandraBrowserTool