Assignment 19.3

Apache HBase is the Hadoop database, a distributed, scalable and a Big Data store. Apache HBase can be used when a random, real-time read/write access to your Big Data is required.

HBase provides many methods for interacting with it. Since HBase is built in Java and the Java API is most widely used so it provides the most number of functionalities.

In case you would like to use HBase without Java, HBase provides you with two options:

Thrift Interface – This is much more lightweight and hence faster of the two options.

REST Interface – This uses HTTP verbs to perform an action. By using HTTP, a REST interface offers a much wider array of languages and programs that can access the interface.

In this post, we will learn about the concept of Thrift and how to install it on Centos, code sample of Python for accessing the Hbase.

You must have the Hadoop cluster with Hbase installed in it to implement the concepts explained further in the blogs.

So, what is Apache Thrift?

Apache Thrift is a software framework for scalable cross-language services development, which combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages.

When to use Thrift?

Thrift can used when developing a web service that uses a service developed in one language access that is in another language. For example: HBase is built in Java and if there is a web application in Python, then HBase can be accessed through Python through Thrift API.

HBase Thrift

In the context of HBase, Java is the only language which can access HBase directly. The Thrift interface acts as bridge which allows other languages to access HBase using a Thrift server that interacts with the Java client.

For Thrift to work, another HBase daemon needs to be running to handle these requests. This daemon comes with HBase, but we need to install some additional dependencies along with it. The below diagram shows how Thrift and REST are placed in the cluster.

HBase Rest:

An obvious choice is Representational State Transfer (REST), which is based on existing webbased technologies.

The actual transport is typically HTTP—which is the standard protocol for web applications.

This makes REST ideal for communicating between heterogeneous systems: the protocol

layer takes care of transporting the data in an interoperable format.

Internally, all these servers (REST, Thrift and Avro) use the common HTable-based client API

to access the tables.

They are started on top of the region server processes, sharing the same physical machine.

There is no one true recommendation for how to place the gateway servers.

You may want to collocate them, or have them on dedicated machines.

- The REST server also comes with a comprehensive Java client API.
- It is located in the org.apache.hadoop.hbase.rest.client package.
- The central classes are RemoteHTable and RemoteAdmin.
- Steps to access a remote HBase table

Note:

The Thrift and REST client hosts usually don't run any other services (such as DataNodes or RegionServers) to keep the overhead low and responsiveness high for REST or Thrift interactions.

Make sure to install and start these daemons on nodes that have access to both the Hadoop cluster and the application that needs access to HBase.

The downside to Thrift is that it's much more difficult to set up than REST. You will need to compile Thrift and generate the language-specific bindings. These bindings are nice because they give you code for the language you are working. Meaning, there's no need to parse XML or JSON like in REST. Rather, the Thrift interface gives you direct access to the row data. Another nice feature is that the Thrift protocol has native binary transport; you will not need to base64 encode and decode data.

To start using the Thrift interface, you need to figure out which port it's running on. The default port is 9090