hive—high Avaliable

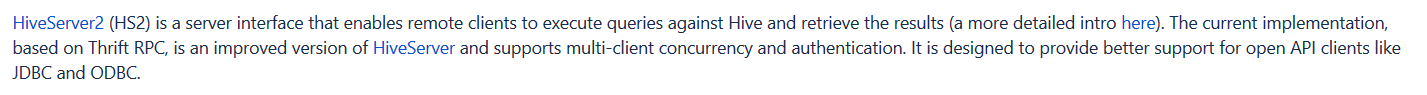
hive的搭建方式有三种，分别是

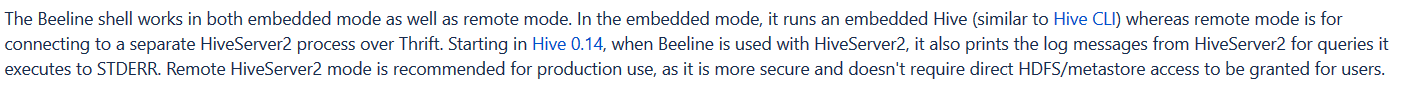
1、Local/Embedded Metastore Database (Derby)

2、Remote Metastore Database

3、Remote Metastore Server

一般情况下，我们在学习的时候直接使用hive –service metastore的方式启动服务端，使用hive的方式直接访问登录客户端，除了这种方式之外，hive提供了hiveserver2的服务端启动方式，提供了beeline和jdbc的支持，并且官网也提出，一般在生产环境中，使用hiveserver2的方式比较多，如图：





使用hiveserver2的优点如下：

1、在应用端不需要部署hadoop和hive的客户端

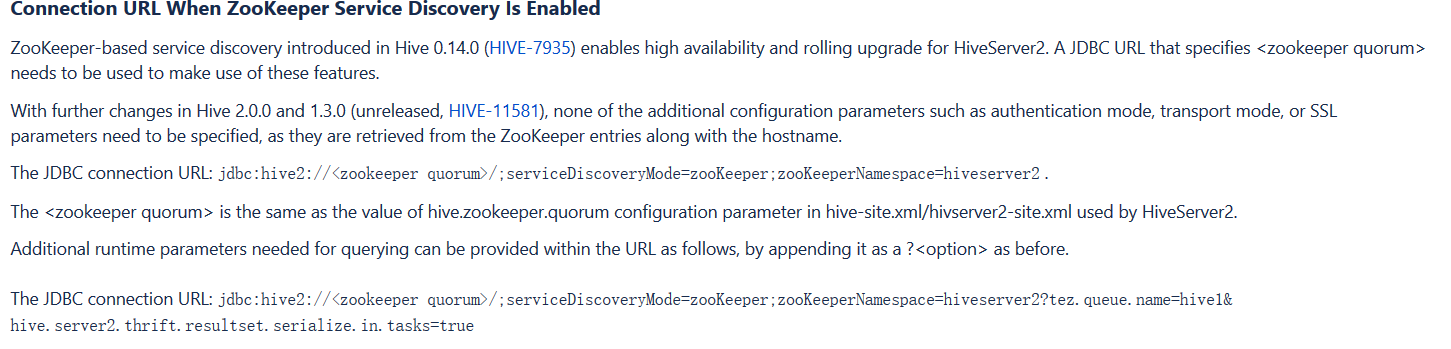
2、hiveserver2不用直接将hdfs和metastore暴露给用户

3、有HA机制，解决应用端的并发和负载问题

4、jdbc的连接方式，可以使用任何语言，方便与应用进行数据交互

本文档主要介绍如何进行hive的HA的搭建：

如何进行搭建，参照之前hadoop的HA，使用zookeeper完成HA



1、环境如下:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Node1 | Node2 | Node3 | Node4 |
| Namenode | 1 | 1 |  |  |
| Journalnode | 1 | 1 | 1 |  |
| Datanode |  | 1 | 1 | 1 |
| Zkfc | 1 | 1 |  |  |
| zookeeper |  | 1 | 1 | 1 |
| resourcemanager |  |  | 1 | 1 |
| nodemanager |  | 1 | 1 | 1 |
| Hiveserver2 |  | 1 |  | 1 |
| beeline |  |  | 1 |  |

#之前都是 3，由元数据s，配了hs2，，4 配了hive-cli，，2是（s 和 c）。

#之前可以再4，启动是因为，，4到时候找3

2、node2—hive-site.xml

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

</property>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://node1:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123</value>

</property>

<property>

<name>hive.server2.support.dynamic.service.discovery</name>

<value>true</value>

</property>

<property>

<name>hive.server2.zookeeper.namespace</name>

<value>hiveserver2\_zk</value>

</property>

<property>

<name>hive.zookeeper.quorum</name>

<value>node1:2181,node2:2181,node3:2181</value>

</property>

<property>

<name>hive.zookeeper.client.port</name>

<value>2181</value>

</property>

<property>

<name>hive.server2.thrift.bind.host</name>

<value>node2</value>

</property>

<property>

<name>hive.server2.thrift.port</name>

<value>10001</value>

</property>

3、node4—hive-site.xml

<property>

<name>hive.metastore.warehouse.dir</name>

<value>/user/hive/warehouse</value>

</property>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://node1:3306/hive?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>123</value>

</property>

<property>

<name>hive.server2.support.dynamic.service.discovery</name>

<value>true</value>

</property>

<property>

<name>hive.server2.zookeeper.namespace</name>

<value>hiveserver2\_zk</value>

</property>

<property>

<name>hive.zookeeper.quorum</name>

<value>node1:2181,node2:2181,node3:2181</value>

</property>

<property>

<name>hive.zookeeper.client.port</name>

<value>2181</value>

</property>

<property>

<name>hive.server2.thrift.bind.host</name>

<value>node4</value>

</property>

<property>

<name>hive.server2.thrift.port</name>

<value>10001</value>

</property>

4、使用jdbc或者beeline两种方式进行访问

1） beeline

!connect jdbc:hive2://node1,node2,node3/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2\_zk root 123

2）jdbc

**public** **class** HiveJdbcClient2 {

**private** **static** String *driverName* = "org.apache.hive.jdbc.HiveDriver";

**public** **static** **void** main(String[] args) **throws** SQLException {

**try** {

Class.*forName*(*driverName*);

} **catch** (ClassNotFoundException e) {

e.printStackTrace();

}

Connection conn = DriverManager.*getConnection*("jdbc:hive2://node1,node2,node3/default;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2\_zk", "root", "");

Statement stmt = conn.createStatement();

String sql = "select \* from tbl";

ResultSet res = stmt.executeQuery(sql);

**while** (res.next()) {

System.***out***.println(res.getString(1));

}

}

}