PostgreSQL Compatible “Smart” Drivers for YugabyteDB

Neeraj Kumar

Friday, Feb/04/2022

YFTT
YUGABYTEDB
FRIDAY
TECH TALKS

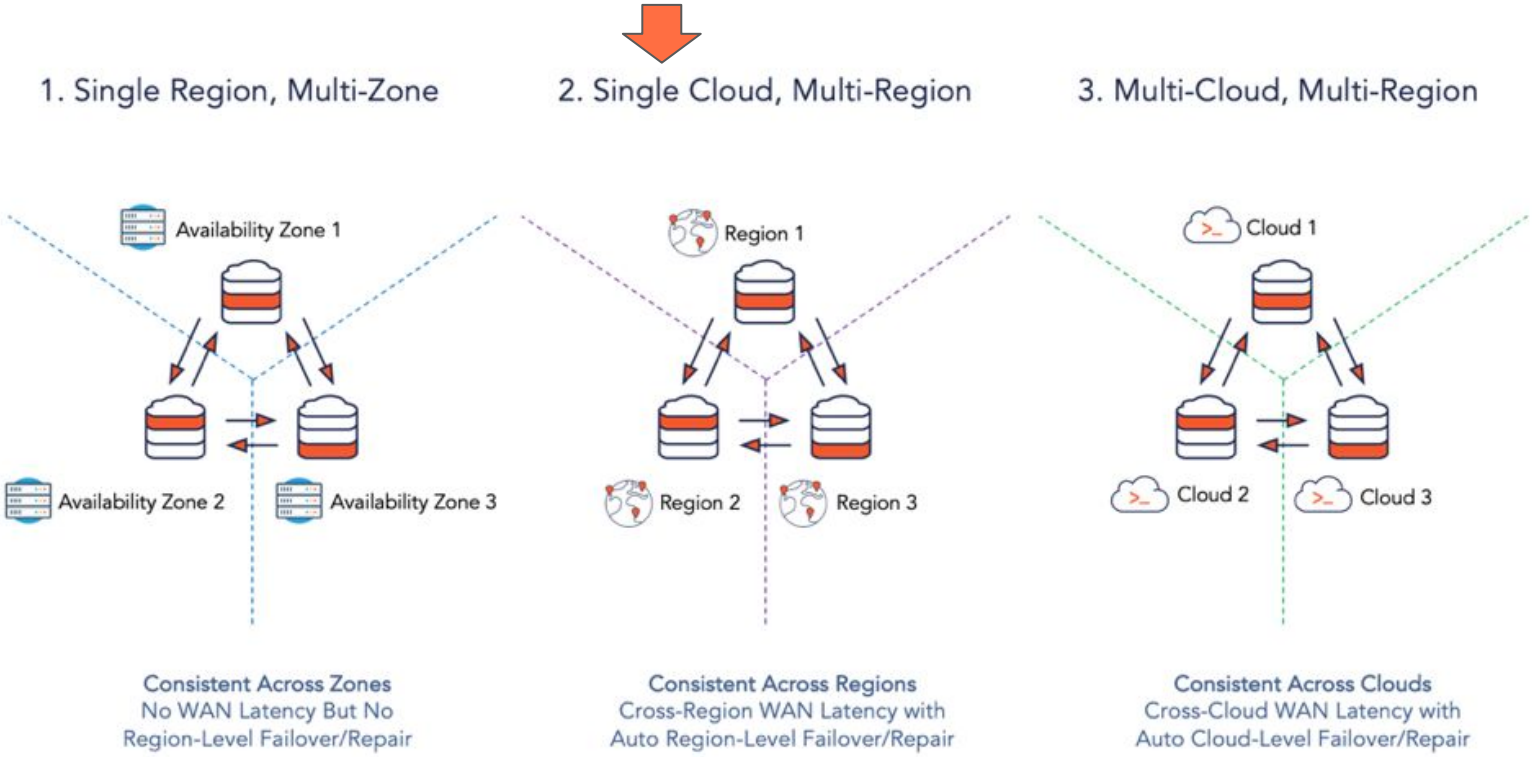
yugabyteDB

Agenda

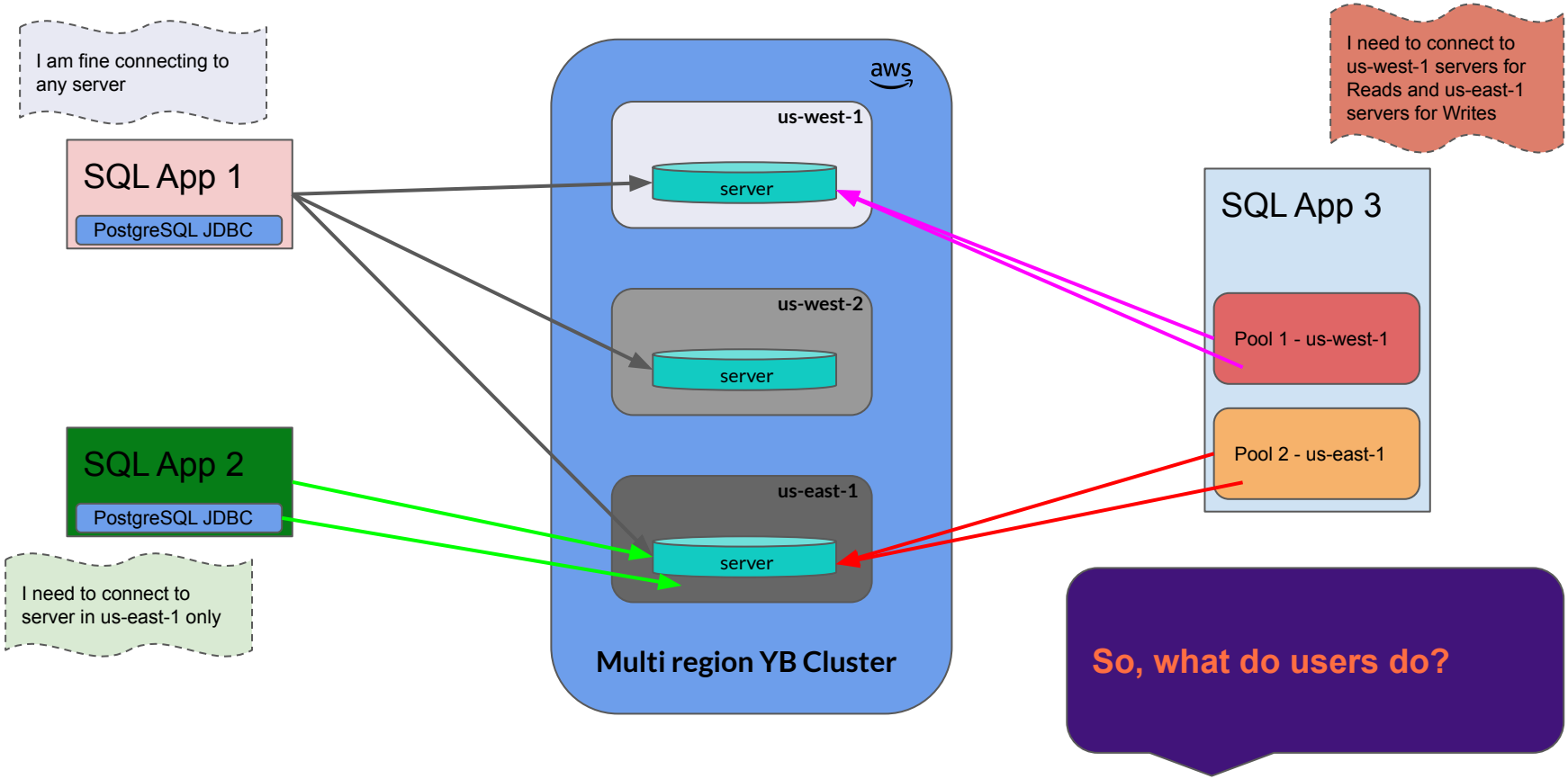
- Understanding the “Smart” Drivers with use cases
- Walking through some “Code Snippets”
- Demonstrations

Let’s See What Applications Need

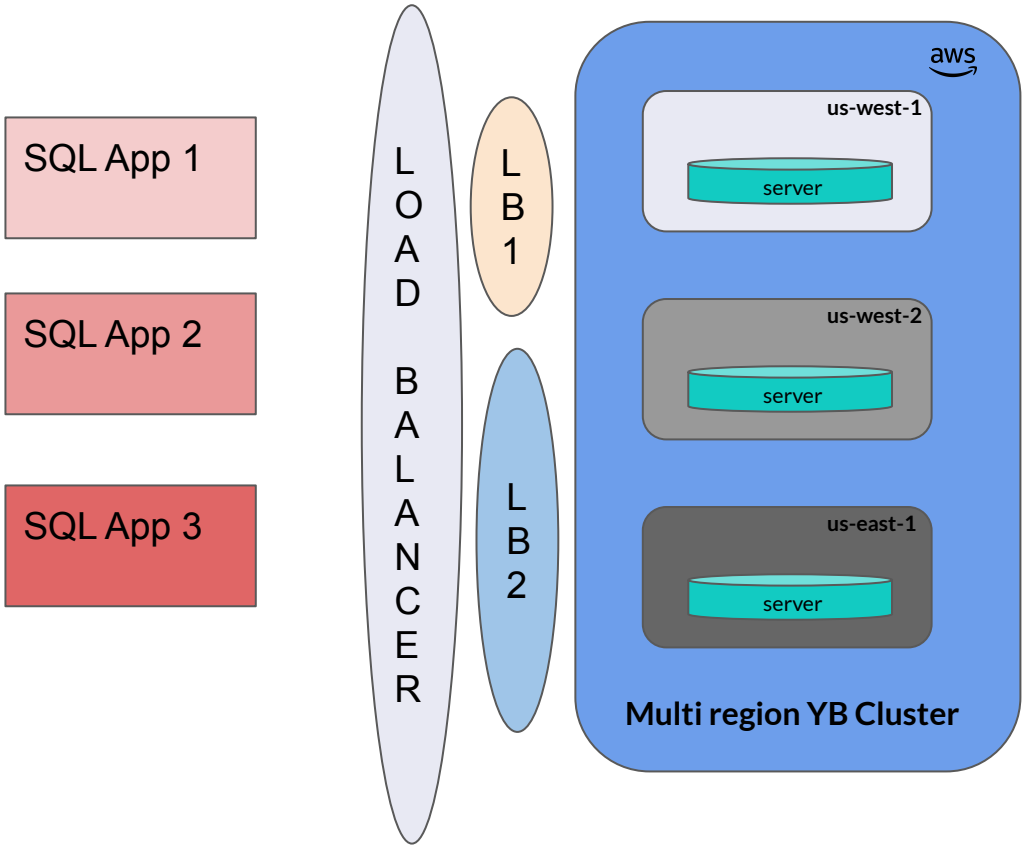
3 typical deployments



Different application requirements



Use load balancing solutions



Issues with LBs

Maintain and manage one or more LBs.

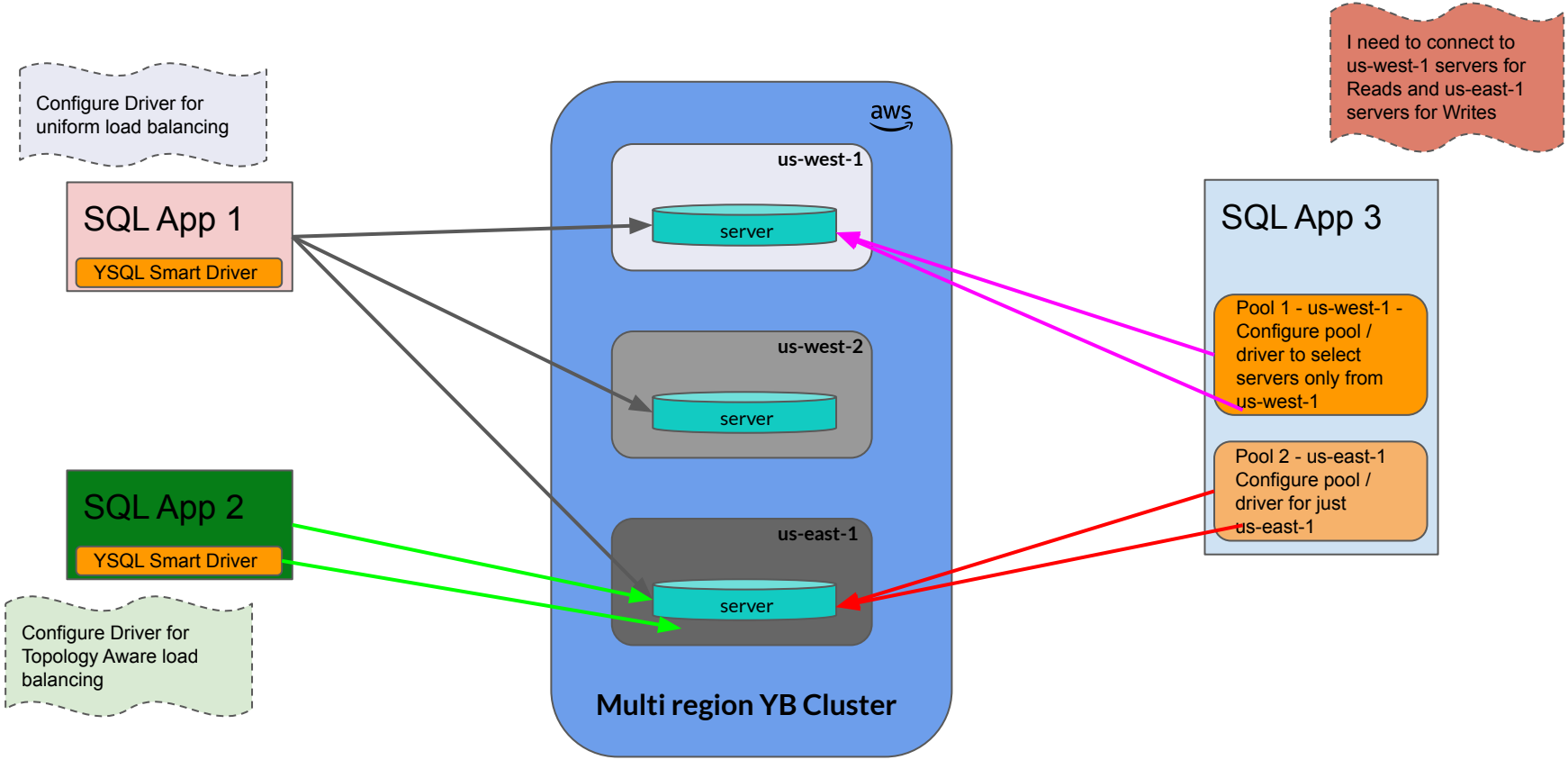
Configurations complex because of different needs.

Configurations may not be ideal to app needs.

Cluster can scale-up, scale down. LBs to be updated accordingly, Nodes may become unavailable.

Can we avoid LBs and make the drivers smarter?

Make Drivers smart – configure them as per APP's need



YSQL “Smart” Drivers

- ➔ YSQL “Smart” Drivers = Modified PostgreSQL Drivers for “smart” features
- ➔ Current “smart” features are 2 flavours of inbuilt Load Balancing
  - ◆ **Cluster Aware Load Balancing** (aka Uniform Load Balancing)
  - ◆ **Topology Aware Load Balancing** (can load balance across one or more regions instead of entire cluster)
- ➔ YugabyteDB conn url: **jdbc:yugabytedb://**

  YugabyteDB Driver: **com.yugabyte.Driver**

  YugabyteDB DataSource: **YBClusterAwareDataSource**
- ➔ Configure the Driver / DataSource with just a single endpoint
- ➔ Automatically discovers all the nodes of the “cluster”
- ➔ Transparently refreshes “cluster” information and uses only healthy servers for new connections

State of “Smart” Drivers

Current

- ➔ Currently available in only Java (JDBC) driver
  
  https://github.com/yugabyte/pgjdbc.git

```
<dependency>
   <groupId>com.yugabyte</groupId>
   <artifactId>jdbc-yugabytedb</artifactId>
   <version>42.3.3</version>
</dependency>
```

MAVEN CENTRAL COORDINATE

- ➔ Go, C# and Python Drivers to be available in less than a month
- ➔ C/C++ Driver is being worked on

Future

More features in future apart from Load Balancing

- ➔ Data Location Aware Query / DML routing
- ➔ Auto retry on Failures whenever possible

Java/JDBC Code Snippets & Examples

Connection url for “uniform” and “topology” aware load balancing

```
String url  = "jdbc:yugabytedb://" + host + ":" + port +
       "/yugabyte?user=yugabyte&password=yugabyte&load-balance=true";
```

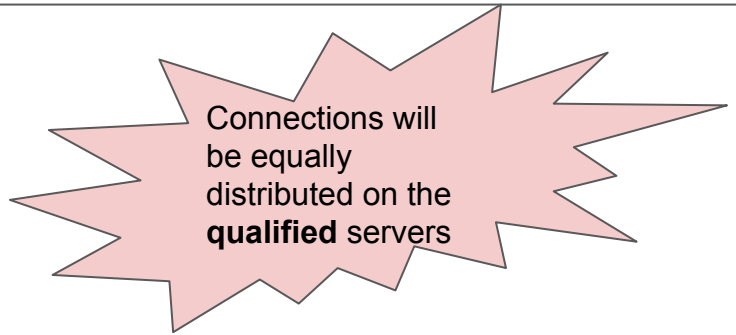
URL for Uniform Load Balancing

```
String url  = "jdbc:yugabytedb://" + host + ":" + port +
     "/yugabyte?user=yugabyte&password=yugabyte&” +
load-balance=true&topology-keys=cloud1.region1.zone1,cloud2.region2.zone1";
```

URL for Topology Aware Load Balancing

```
ArrayList<Connection> connections = new ArrayList<>();
for (int i=0; i<N; i++ {
  Connection conn = DriverManager.getConnection(url);
  connections.add(conn);
}
```

JDBC API call to get a connection using DB URL

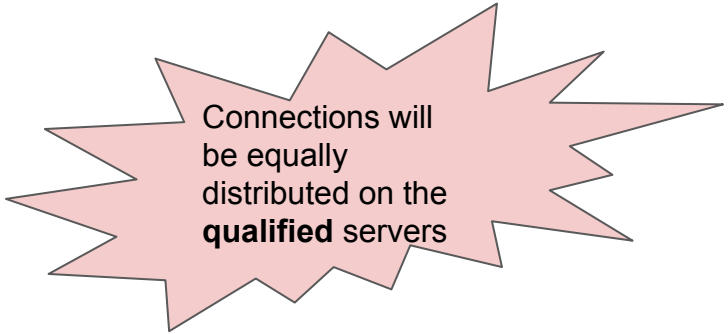


Configuring YBClusterAwareDataSource

```
ArrayList<Connection> connections = new ArrayList<>();
YBClusterAwareDataSource ds = new YBClusterAwareDataSource();
ds.setServerName(host);
ds.setUser("yugabyte");
ds.setPassword("yugabyte");
ds.setTopologyKeys("cloud1.region1.zone1"); // for using only servers in this location

for (int i=0; i<N; i++ {

   Connection conn = ds.getConnection();
   connections.add(conn);
}
```



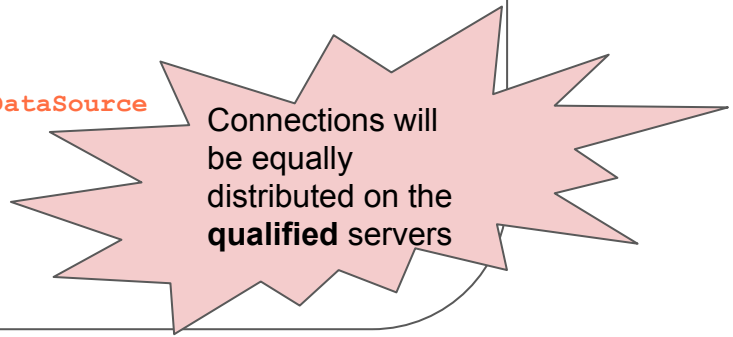
Configuring HikariPool with YBClusterAwareDataSource

```
// Create a property bag for the Hikari Connection Pool
Properties poolProps = new Properties();
poolProps.setProperty("poolName", "yb_lb_demo");
poolProps.setProperty("dataSourceClassName", "com.yugabyte.ysql.YBClusterAwareDataSource");
poolProps.setProperty("maximumPoolSize", String.valueOf(N));
poolProps.setProperty("autoCommit", "true");
poolProps.setProperty("dataSource.serverName", host);
poolProps.setProperty("dataSource.databaseName", "yugabyte");
poolProps.setProperty("dataSource.user", "yugabyte");
poolProps.setProperty("dataSource.password", "yugabyte");

poolProps.setProperty("dataSource.topologyKeys", "cloud1.region1.zone1");

// Create the Hikari Pool config
HikariConfig hikariConfig = new HikariConfig(poolProps);
hikariConfig.validate();

ArrayList<Connection> connections = new ArrayList<>();
// Create the Hikari Data Source configured with YBClusterAwareDataSource
HikariDataSource hds = new HikariDataSource(hikariConfig);
for (int i=0; i<N; i++) {
        Connection conn = hds.getConnection();
        connections.add(conn);
}
```



Demos

Demos

Demo 1 - LB in a 3 Region Cluster

- AWS Cloud
- 3 Regions, us-west-1, us-west-2, us-east-1
- 1 server in each region

App1 - Creates 10 connections to the cluster

App2 - Creates 10 connections to only us-west* regions

Demo 2 - Changing Cluster

- Run the Packaged examples to demonstrate automatic refresh of changing cluster nodes

Thank You

Join us on Slack: yugabyte.com/slack (#yftt channel)

Star us on Github: github.com/yugabyte/yugabyte-db

YFTT YUGABYTEDB FRIDAY TECH TALKS

yugabyteDB