Quick Guide to Using the SQream Dynamic Workload Manager (DWLM)

Overview

Each SQream instance has its own hardware resources (GPU, RAM etc.) as defined in its parameters setting. SQream distributes work among its instances in such a way as to maximize hardware utilization. By default, this distribution is done in an equal manner.

- You can change this setting and optimize hardware utilization by using SQream Dynamic Workload Manager and subscribing each SQream instance to specific service/s, and connecting different sessions to these services.

Managing workloads

You can define the specific hardware resources to be used by each SQream instance in its configuration settings. This setting defines the ID of the GPU and its internal memory capacity, CPU memory and more. See more information regarding this here <add a link to the installation guide on the json parameter setting>).

The DWLM allows you to define which SQream instance (by instance_id) will provide resources to which services and assign specific sessions to those services. This can be used as a tool to manage the general priority of connections/statements in resource allocation.

With this feature, you can subscribe 1 service to multiple SQream instances, and/or, 1 SQream instance to multiple services.

NOTE: If you do not specify any service to any instance, then all resources will be shared by all requests equally using the default service "sqream".

Examples:

Case 1:

The following configuration allocates resources as follows:

- 1 resource for ETL
- 3 resources for queries
- all resources for sessions connected with service management

Service/Instance	Instace1	Instance2	Instance3	Instance4
ETL service	✓	-	-	-
Query service	-	✓	✓	✓
Management service	✓	✓	✓	✓

With this configuration, ETL will not compete for hardware resources with the common query service statements.

Statements coming from management service will be able to use any hardware resource from any instance.

Case 2:

The following configuration allocates resources as follows:

- 1 resource for ETL
- 2 resources for queries
- 1 resource for sessions connected with service management

Service/Instance	Instace1	Instance2	Instance3	Instance4
ETL service	✓	-	-	-
Query service	-	✓	✓	-
Management service	-	-	-	✓

With this configuration, ETL/queries/management services will not compete on hardware resources and each will have its own pool or pools of instances.

Managing services

Creating a service

You do not need to manually create a service. When an instance is subscribed to a service that does not exist, that service is created automatically.

Subscribing and unsubscribing services

There are 2 methods for subscribing or unsubscribing a service:

- Initial configuration: insert or delete the service name in the initial configuration file. The service will be subscribed or unsubscribed only after a system restart.
- Online: use the appropriate utility function to subscribe or unsubscribe a service while the system is running. The change takes place immediately for the current session.

  Since the utility function affects only the current session, you must also edit the initial configuration in order to permanently modify the service subscription status.

NOTE: If no service name is specified in any command, the default 'sqream' service is assumed.

Subscribing a service

When a SQream instance is subscribed to a service/s it can serve the connections of the service/s.

Initial configuration:

Define the parameter ***initialSubscribedServices*** in the instance configuration flag.

The parameter can be defined with multiple services.

For example:

```
"runtimeGlobalFlags": {
    "initialSubscribedServices" : "etl,management"
}
```

Online subscription:

Use the utility function ***subscribe_service***() with the relevant instance_id and service name. The instance ID can either be found in the instance configuration file, or by using the utility function ***show_subscribed_instances()*** (see Monitoring services).

For example:

```
Select subscribe_service('node_11','etl');
```

Unsubscribing a Service

To disable the service/instance connectivity and stop using a specific instance for a specific service, use the utility function ***unsubscribe_service()*** with the relevant instance_id and service name.

By unsubscribing a service from the instance, SQream will stop directing new requests from this service workload to the instance.

Note the following:

- Existing executing operations will not be stopped. .
- The last instance from an existing service with working/waiting statements in its queue cannot be unsubscribed.
- To stop a working or waiting query , use the utility function ***stop_statement()***

Initial configuration:

Delete the service name from the parameter initialSubscribedServices in the instance configuration flag.

For example:

In the Service Subscribe example above, we subscribed both ETL and management. To unsubscribe ETL, delete it from the instance configuration flag:

```
"runtimeGlobalFlags": {
    "initialSubscribedServices" : "management"
}
```

Online un-subscription:

Use the utility function ***unsubscribe_service()*** with the relevant instance_id and service name. The instance ID can either be found in the instance configuration file, or by using the utility function ***show_subscribed_instances()*** (see Monitoring services).

For example:

```
Select unsubscribe_service('node_11','etl');
```

Monitoring services

To see which services are being used by which instances, use the utility function ***show_subscribed_instances()***.

For example:

```
Select show_subscribed_instances();
```

To see the current statements queue with the allocations for each service/statement, use the utility function ***show_server_status().***

For example:

```
Select show_server_status();
```

Connecting to a service

To connect a specific service, use the attribute `SERVICE` in the connect string.

If no service is specified, SQream will connect the default service 'sqream'.

ClientCmd command line:

Add the argument ***service=<service_name>***

For example:

Connect the ETL service

```
./bin/ClientCmd --user=sqream --password=sqream --database=master
--host=127.0.0.1 --port=3108 --service=etl --clustered
```

JDBC driver

Add the argument ***service=<service_name>*** to the JDBC URL

For example:

Connect the service ETL

```
jdbc:Sqream://hostname:3108/master;user=sqream;password=mypassword;service=etl;clus
ter=true;ssl=true;
```

For more information, visit SQream JDBC documentation.

Java connector

Add the argument ***service="service_name"*** to the Java connection handler:

For example:

Connect the ETL service

```
ConnectionHandle Client = new ConnectionHandle ("hostname", 5000, "master",
"sqream", "mypassword","etl");
Client = Client.connect();
```

Note that currently the SQream java connector must connect to SQream directly, and not via the load balancer.

For more information, visit SQream Java connector documentation.

ODBC driver

Add the argument ***service*** to the ODBC DSN.

- In Linux, add the argument in the *odbc.ini* file.

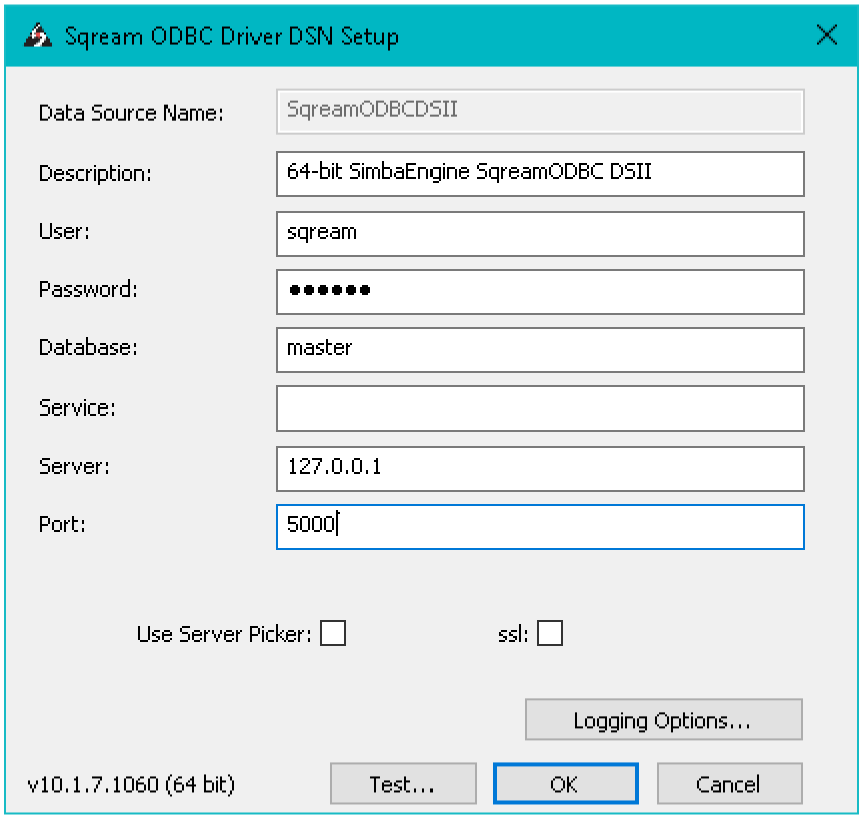
  For example:

```
[mydsn]
Driver=/home/sqream/ODBC/sqream_odbc.so
Description=sqream64
Server=hostname
Port=3108
Database=master
Service=etl
User=sqream
Password=mypassword
Cluster=true
ssl=false
```

  For more information, visit SQream ODBC for Linux documentation.

- In Windows, add the argument at the ODBC application, at the specific DSN

For example:



Data Source Name:	SqreamODBCDSII
Description:	64-bit SimbaEngine SqreamODBC DSII
User:	sqream
Password:	●●●●●●
Database:	master
Service:	
Server:	127.0.0.1
Port:	5000

Use Server Picker: ☐ ssl: ☐

v10.1.7.1060 (64 bit)

For more information, visit SQream ODBC for Windows documentation.

PYTHON

Add the service to the connection settings.

For example:

```
con = SQream_python_connector.Connector()
sqream_connection_params = '127.0.0.1', 3108, 'master', 'sqream', 'mypassword',
True, 30, 'etl'
```

For more information, visit SQream python documentation.

.NET

Add the service to the connection settings.

For example:

```
var connectionString = "Data
Source=127.0.0.1,3109;User=sqream;Password=mypassword;Initial
Catalog=master;Integrated Security=true;Service=etl";
var connection = new SqreamConnection(connectionString);
connection.Open();
```

For more information, visit SQream .NET documentation.

NodeJS

Add the service to the connection settings.

For example:

```
const Connection = require('SQream-nodejs-connector');
const config = {
  host: 'hostname',
  port: 3108,
  username: 'sqream',
  password: 'mypassword',
  connectDatabase: 'master',
  cluster: 'true',
  service: 'etl'
  };
```

For more information, visit SQream NodeJS documentation.