AMQP (RabbitMQ) scaling by broadcasting databroadcast messages

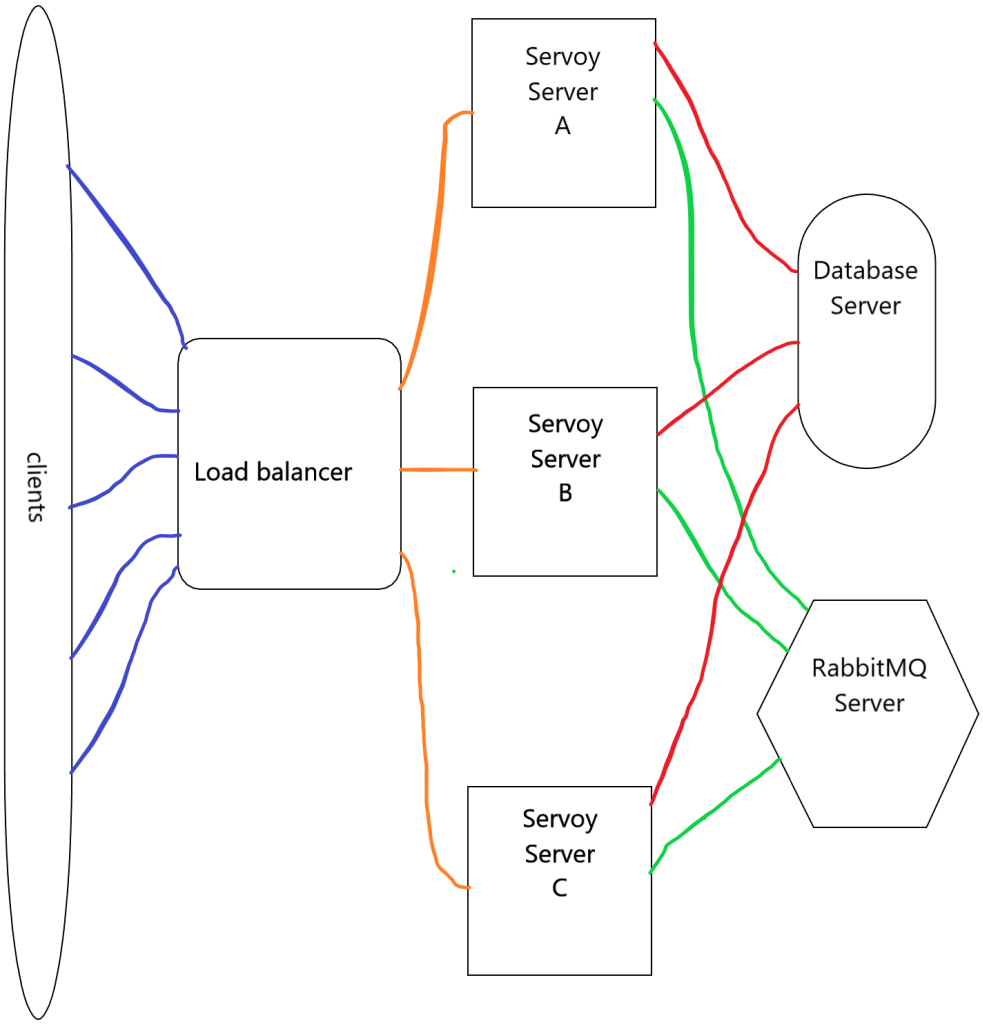
With 8.2(.1) Servoy supports the ability to broadcast our own broadcast messages through an AMQP 0.9.1 message bus. Or any kind of other system by using the new pure server api of IServerAcces through the IDataNotifyService and IDataNotifyListener that the broadcaster plugin is using.

Currently the amqp broadcaster plugin is written for the AMQP 0.9.1 protocol (What RabbitMQ supports) which doesn't support AMQP 1.0 protocol, so servers based on that protocol can't be used.

The default "broadcaster" plugin that Servoy ships with has currently multiple options, of which one is mandatory: setting the hostname of a RabbitMQ installation on all of the installed Servoy servers (WAR deployments) that want to have data broadcasting enabled between them. That RabbitMQ installation hostname can be specified via property ' amqpbroadcaster.hostname ' in servoy.properties. The complete list of settings is:

- **amqpbroadcaster.hostname** Set the hostname of the AMQP (RabbitMQ) server where to connect to (this is mandatory field)
- **amqpbroadcaster.username** Set the username of the AMQP (RabbitMQ) server where to connect to (default value is guest)
- **amqpbroadcaster.password** Set the password of the AMQP (RabbitMQ) server where to connect to (default value is guest)
- **amqpbroadcaster.virtualhost** Set the virtual host of the AMQP (RabbitMQ) server where to connect to (default value is /)
- **amqpbroadcaster.port** Set the port of the AMQP (RabbitMQ) server where to connect to (default value is 5671 for SSL connection and 5672 for default connection)
- **amqpbroadcaster.connectiontimeout** Set the connection timeout of the AMQP (RabbitMQ) connection (default value 60000 - 60 seconds)
- **amqpbroadcaster.handshaketimeout** Set the handshake timeout of the AMQP (RabbitMQ) connection (default value 10000 - 10 seconds)
- **amqpbroadcaster.shutdowntimeout** Set the shutdown timeout of the AMQP (RabbitMQ) connection (default value 10000 - 10 seconds)
- **amqpbroadcaster.rpctimeout** Set the rpc continuation timeout of the AMQP (RabbitMQ) channel (default value 10 minutes)

Below there is a setup of 3 servoy servers talking to the same database and using a RabbitMQ server to synchronize the databroadcast messages.



In this setup it is better that every Servoy server does have its own repository database, this way you can upgrade one without touching anything else.

Things that are not supported in this setup are:

- Pure Servoy record locks (this has to be a database lock (servoy.record.lock.lockInDB must be true and supported by the database)
- Servoy sequences.

With the above approach you have scaling and also zero downtime deployments because you can put one of the servers in maintenance mode and wait until all the clients are gone from that one, then upgrade that system by deploying a new WAR and turn it back on out of maintenance mode.

The load balancer should understand the maintenance mode by catching 503 (SC_SERVICE_UNAVAILABLE) response message and redirect it then to another.

For just zero downtime deployments you can use a simpler setup by just using the Tomcat Parallel deployment. Then you only have 1 tomcat server with 1 WAR talking to a RabbitMQ server. Then when upgrading make a new WAR with a higher version and deploy that on the same tomcat. Then tomcat will make sure that the new sessions end up in the latest version and older clients still work on the previous version. The MQ server will make sure both versions are seeing all the Servoy databroadcast notifications.

Tomcat or the Loadbalancer know that an incoming request is meant for Server C or for an older version in parallel deployment because of the http session that is created (by NGClient and WebClient). So the LoadBalancer should always be configured to use sticky sessions, request coming from 1 browser should always go to the same server. There can be multily NGClients (tabs in the browser) for 1 http session, but if all NGClients are killed then the http session is right away invalidated, (its will neer invaldate other wise). Then if in maintenance mode the initial call to the "index.html" page for a ngclient will already return a 504 error response code, if there is already an NGClient for that session it will pass and will or reuse the same client (if it was a page refresh) or it will then report to the user that the server is in maintenance mode. So if 504 happens the load balancer should redirect to another server.

A user needs to stop all the ngclient browser tabs (all ngclients on the server for that user needs to terminate) before that user will be moved to the a new version or another server.

A Servoy developer can use the new clientmanager plugin to see which client uses what httpsession id, this information is stored in the client info that starts with "httpsessionid:", this info is also shown on the admin page.