metadata technology

White Paper

Fusion Registry® 9 Enterprise Edition

A Scalable High Availability Reference Architecture

Fusion Registry 9 Enterprise Edition is designed for mission-critical statistical data collection, storage and dissemination. In this White Paper we examine a reference architecture for high availability and scalability which is suitable for both virtual cloud and on-premises deployment.

Fusion Registry 9 is a complete and fully integrated statistical data and metadata management system for SDMX.

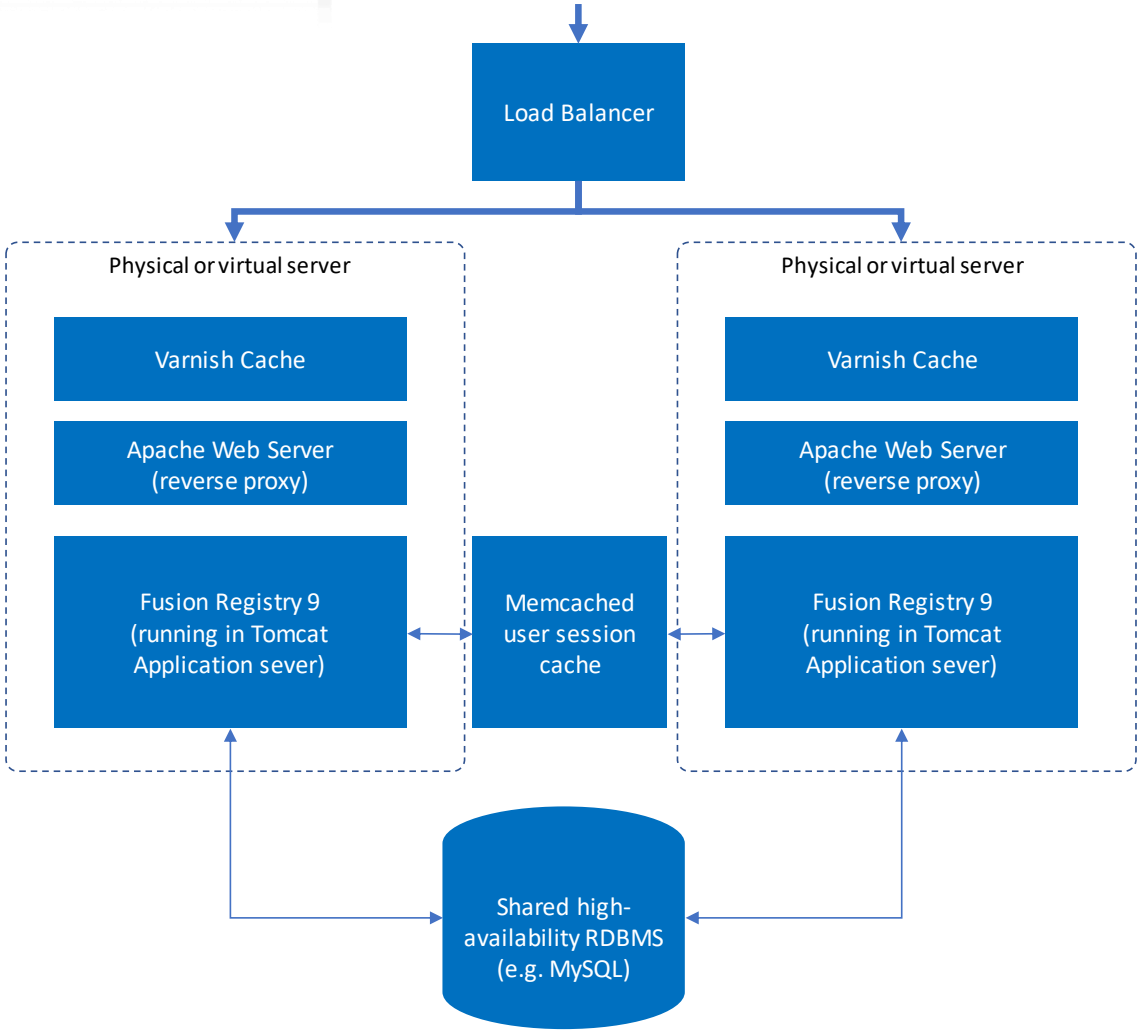

Figure 1 Scalable high availability architecture

Overview

Multiple instances of Fusion Registry 9 Enterprise Edition can be deployed in a load balanced configuration to meet high availability and capacity requirements.

Its web user interface and REST web services API are accessed over standard HTTP/s protocols meaning that standard web server load balancing tools and methods can be used.

Shared RDBMS

All Fusion Registry instances need access to a common shared RDBMS which we recommend should be configured for high-availability to avoid being a single point of failure. Fusion Registry supports MySQL, SQL Server and Oracle. The RDBMS principally stores the SDMX structural metadata, and also stores any data that has been loaded into Fusion Registry's local data repository.

Data doesn't have to be physically loaded and stored into Fusion Registry's local database but can also be integrated from multiple external sources on demand. In the case where all data comes from external sources, the RDBMS stores only the structural metadata - Fusion 'Hub-and-Node' architecture uses this principle. See the Metadata Technology White Paper 'Distributed Data Collection using Fusion Hub-and-Node'.

Tomcat Application Server

Fusion Registry 9 is a platform agnostic Java application which should run on any platform with a Java runtime environment (Java 1.7 and 1.8 are currently supported), and a Java application server. Apache Tomcat is recommended for that purpose.

Load Balancer

Incoming requests are handled in the first instance by a front-end load balancer such as Amazon Web Services (AWS) Elastic Load Balancer if deploying on the cloud, or BigIP for on premises deployments.

The load balancer should be configured to monitor the health of the Fusion Registry instances under its control and only route requests to healthy instances. Fusion Registry provides a specific JSON web service that load balancers can poll as part of their health-check routines.

Varnish Cache and Maintaining Cache Consistency

Fusion Registry supports the Varnish HTTP Cache (https://varnish-cache.org/). Again, to avoid single points of failure, each instance has its own Varnish cache installation which needs to be kept consistent when changes are made to the data or metadata. Cache consistency is maintained by each instance of Fusion Registry periodically polling the shared database for changes, and selectively purging the cache if changes are found.

The polling process simply checks the database for changes in GUIDs which is not onerous and does not impose an appreciable load on either the application or the database. (GUIDs are Global Unique Identifiers that are modified when data or metadata is changed).

Sticky Sessions

The load balancer should be configured for 'sticky sessions'. Sticky sessions, also known as 'session affinity' or 'session persistence' is a standard feature of most load balancers and ensures a GUI user's requests will always be routed to the same Fusion Registry instance. Like most user interfaces, the Fusion Registry GUI is stateful, so it's important that a user interacts with the same instance for the duration for their session.

The REST web service API is entirely stateless, so does not benefit directly from sticky sessions. Nevertheless, there are caching performance benefits to be gained from the load balancer directing all REST requests from a user or source application to the same Fusion Registry instance.

Session Failover and Session Caching

Under normal circumstances, sticky sessions guarantee that a user's requests are always routed to the same instance. If a Fusion Registry instance fails in service, the load balancer should be configured to redirect user sessions to other healthy instances – most load balancers will do this 'session failover' by default.

However, an additional mechanism is required to ensure users active on the failed instance do not have their sessions interrupted. This is done by the use of a 'memcached' service (https://memcached.org/) which allows session state to be shared between instances. The application server of a running Fusion Instance (usually Apache Tomcat) saves active session state information to memcached. When a session is moved to a new instance, that instance can retrieve the session state from memcached allowing the user session to continue.

It's important to note that session failover may not be completely instantaneous. Load balancers will generally only move sessions when they recognise the instance as being unhealthy, and that depends on how they conduct their health checks. If instance health is assessed every 60 seconds by polling, users with active sessions may need to wait for up to a minute for their session to resume. This interval can be reduced by increasing the health check polling frequency within reason.

Scalability

Fusion Registry is designed to be horizontally scaled.

The reference architecture in Figure 1 shows two instances which help to meet high availability requirements by improving resilience. In practice though, the number of Fusion Registry instances required behind the load balancer is usually determined by the volume of data queries expected.

Tests reveal Fusion Registry is able to deliver excellent performance and load tolerance by using a range of methods including an SDMX optimised in-memory database (Fusion Data Store) and caching. Independent load tests conducted on a four-instance load balanced cluster show the system is capable of servicing an average of 46.6 dissemination query requests per second. This service response was consistent up to the test limit of 500 concurrent users. At 50 concurrent users, average query response times were below 1 second increasing linearly with demand. The results indicate that a load balanced cluster of between two and four Fusion Registry instances should have sufficient capacity to support many dissemination use cases.

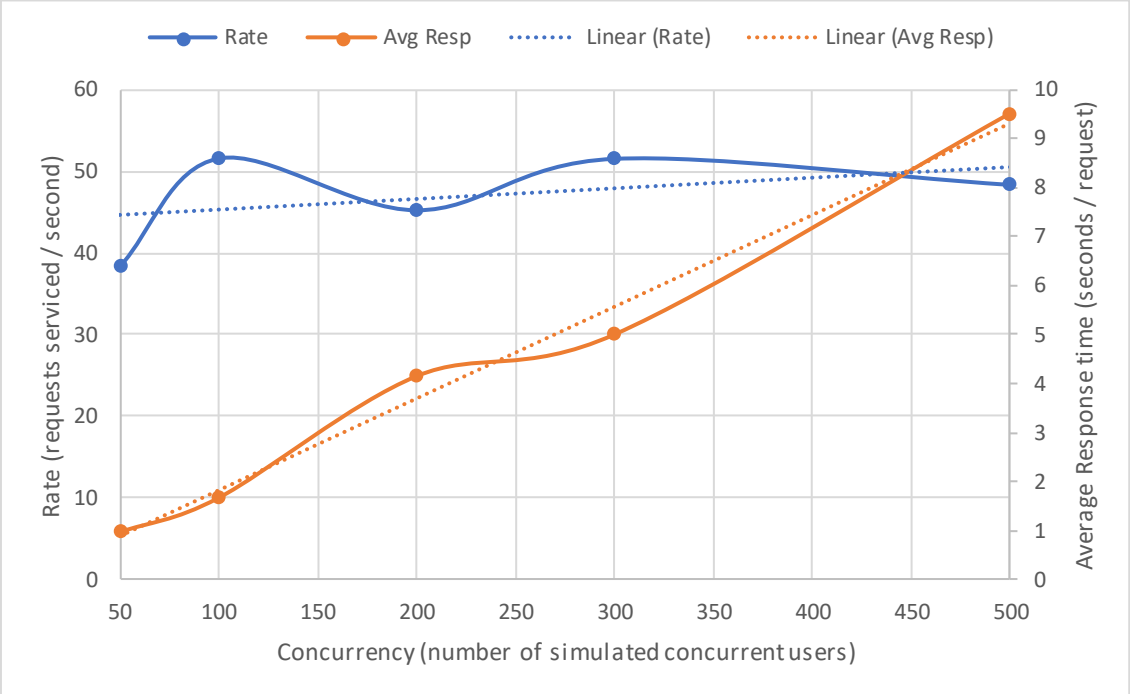

Figure 2 Four-instance load balanced cluster load test results 28 February 2018

Note that where high volumes of data need to be disseminated to a global audience, a distributed multi-region architecture may be more appropriate. This can be achieved by deploying Fusion Node edge-servers to a number of different geographic locations. The Metadata Technology White Paper 'Global Dissemination Architectures using Fusion Node' discusses this subject in more detail.

Changing Capacity

System capacity can be easily changed in service by adding or removing instances behind the load balancer. Each instance operates largely independently of its peers, so no special configuration procedures are required.

To increase capacity, start a new Fusion Registry instance with access to the shared RDBMS and session cache. When ready to service requests, instruct the load balancer to add the new instance to the cluster.

Reducing capacity requires one or more Fusion Registry instances to be removed from the cluster. The load balancer should be instructed to stop sending new user requests to the unneeded instances which should be left quiescent for a period of time to allow existing sessions to drain. The load balancer should then be instructed to remove the redundant instances from the cluster after which they can be stopped.

Rolling Upgrade

The architecture allows for 'rolling upgrade' where new versions of Fusion Registry (and upgrades to other components in the stack) are deployed without any service downtime.

Rolling upgrade can be implemented using procedures similar to those for changing capacity which are outlined above.

The basic process is to remove one instance at a time from the cluster, upgrade it and add it back in. If using cloud services or virtual servers, it may be more practical to start new instances with the upgraded software, add those to the cluster before removing the old. This approach avoids reducing the overall service capacity while the upgrade is in progress.

AWS Auto Scaling

Auto Scaling is a useful feature for AWS users that automatically replaces failed instances by shutting down the offending virtual server and deploying a new one. It can also be configured to dynamically increase or reduce the size of the load-balanced cluster in response to demand.

Appendix A – Resources

Amazon AWS	https://aws.amazon.com/
Amazon Elastic Load Balancer	https://aws.amazon.com/elasticloadbalancing/
Amazon RDS Database Service	https://aws.amazon.com/rds/
Apache Tomcat	http://tomcat.apache.org/
Apache Web Server	https://httpd.apache.org/
Big-IP	https://f5.com/products/big-ip
Fusion Registry 9	https://metadatatechnology.com/software/FR9.php
Memcached	https://memcached.org/
MySQL High Availability	https://www.mysql.com/products/enterprise/high_availability.html
Varnish Cache	https://varnish-cache.org/

metadata technology

+44 1483 418 058

info@metadatatechnology.com

metadatatechnology.com

Metadata Technology Ltd
Floor 2 Solly's Mill
Mill Lane
Godalming
GU7 1EY
United Kingdom