Parameter handling using multi-part messages

Denis Bertini[1]
[1]Scientific Computing ,GSI, Darmstadt, Germany

Introduction

The FairDbMQ library provides a solution to adapt any parameter initialisation scheme to Message Queue distributed FairMQ tasks [1].

Design

FairDbMQ implements a simple Actor Model for concurrency using ZeroMQ [2] to cope with a large number of distributed tasks requesting asynchroneously parameter initialisation from a RDMS or file system based store. The challenge is then to be able to control the number of processes connecting the database or reading the file.
To solve this problem FairDbMQ separates the *frontend request client* from the *backend reply worker* adding a non-blocking worker queueing daemon in between acting as a proxy (Figure [1]).

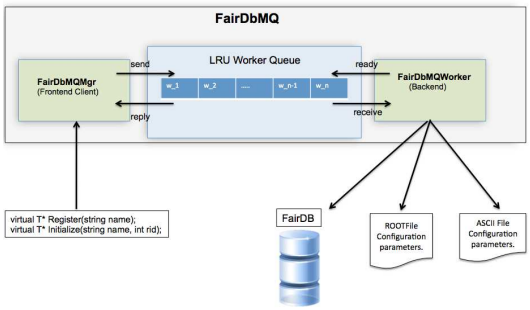

Figure 1: FairDbMQ components

Routing and protocol

The Worker Queue Daemon is continuously listening for connections from the *frontend sockets* and the *backend sockets* via event polling. It also uses ROUTER sockets to perform messages routing, the clients using REQ synchronous socket to connect.
When a initialisation request is sent, the socket identifier is kept as a frame content within the multi part message giving the possibility for the proxy to know to which client it should send back the processed reply (Figure [2]).

Scalability and load balancing

The previous single cluster architecture can be scaled to more than one cluster by swapping a REP client socket with a DEALER socket. The Reply-Request mechanism goes then full asynchronous and request clients as well as worker can talk to more than one proxy (Figure [3]).
Of course, other implementations of a multiple clusters compatible with FairDbMQ design exists like for example connecting 2 proxies together using a ROUTER to ROUTER combination.

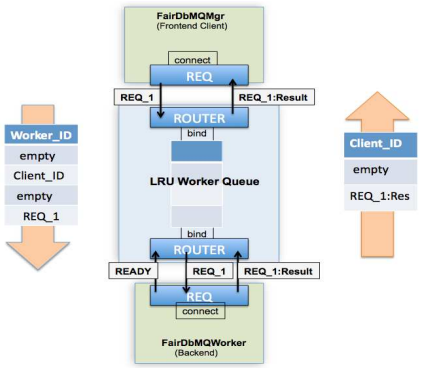

Figure 2: Routing is implemented by a dedicated Protocol using Mutli-part Messages.

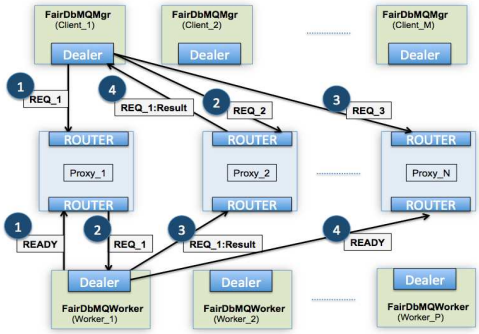

Figure 3: Multiple Cluster Design

Conclusion

The FairDbMQ library implements a custom queueing system to provide a scalable and fault-tolerant request-reply architecture for FairMQ tasks parameter initialisation.

References

[1] The FAIR simulation and analysis framework 2008 J. Phys.: Conf. Ser. 119 032011

[2] ZeroMQ, Martin Sustrik, The architecture of open source application volume 2.

DOI:10.15120/GR-2015-1-INFRASTRUCTURE-IT-08