WHITE PAPER

Managing FIX Sessions in a Dynamic Infrastructure

MAXIM VASIN, PHD

Senior Software Engineer

MARK BISKER

Senior VP of Banking & Finance Solutions, Capital Markets Competency Center

IOURI JARYI

Software Engineering Manager

<epam>

Contents

Introduction

Brokerage houses, investment banks, wealth management firms, exchanges and clearing houses operate their systems in complex distributed computing environments.

Many of them are considering or in the process of migrating applications and data to the cloud to take advantage of its low cost and elasticity. Applications which use libraries for FIX protocol communications or application servers dedicated to FIX message routing and transformation (FIX servers) are a critical part of those systems. In the twenty years since FIX protocol was introduced, it has become a global standard for electronic trade communications. Companies operate hundreds of thousands of FIX sessions, processing millions of messages a day. Migration of FIX connectivity to a dynamic infrastructure requires rethinking the way the FIX layer is constructed, which we will describe in this white paper.

History of High Availability Solutions for FIXEdge and FIXAntenna-based Applications in Linux

Historically, to service multiple FIX sessions reliably, we used to deploy FIXAntenna (FA) and FIXEdge (FE) in a Linux HA cluster with two or three nodes and shared storage. The cluster solution utilizes Corosync and Pacemaker – tools that facilitate the use of an Linux HA cluster for applications that do not have native support for clustering. Health checks are used to figure out whether a node or an application that runs on the node is operating properly. Node health checks are implemented by monitoring an FE PID file.

The cluster can be configured as active-active or active-passive. For example, two nodes constitute a cluster, and at any moment, only one of them runs FE (active-passive cluster). When the failure of an active FE node is detected, a shared FIX message file system storage is unmounted on that node and mounted on the second node, then FE is started on the second node. All sessions are handled by the one active server and will be started on another node in case of failure. This approach leaves no opportunity to balance the load between the cluster nodes. FIXEdge start-up time grows with the number of the session it serves.

We had experience setting up the active-active configuration on the Linux HA cluster to partition FIX sessions and allocate them between nodes. The practical implementation of this mode, though, is quite complex and typically needs to be customized per customer-specific environment. Most of the complexity arises from the need to manage a FIX session's state on multiple storage devices. Additionally, reconfiguration of statically defined partitions is not a trivial task and requires the involvement of a highly qualified Linux HA system administrator.

Read more on this topic at https://kb.b2bits.com/display/B2BITS/FIXEdge+Failover+Cluster+installation

This approach gives a working, highly available solution and works well for active-passive clusters with a few nodes and static FIX session configuration.

New Generation Ecosystem for FIX Layer

While designing the FIX layer, we considered the following business challenges:

- Continuing increase in number of connections
- Continuing increase in number of messages
- Growing requirements for services uptime
- Reduced time for onboarding of new connections
- Ability to access markets globally
- Ability to operate multiple versions of FIX protocol and middleware integration platforms simultaneously

and the following main use cases:

- FIX server as Order Entry Gateway
- FIX server as Smart Order Router/Message Router
- FIX server as Market Data Server
- FIX server as Trade Capture/Drop Copy Server

KEY ARCHITECTURAL CONSIDERATIONS

We envision the FIX layer as a set of computing and storage nodes provisioned on demand. The FIX traffic is distributed to these nodes based on certain algorithms to meet the SLA. We have defined a new generation ecosystem (NGE) for FIX layer operations as a set of services managing the FIX sessions across multiple instances of FIX servers. A single FIX session is a minimal unit of work in NGE. The allocation and reallocation of a FIX session to an instance of a FIX server is performed dynamically.

The following architecture concerns are addressed in the NGE:

- Scalability
- State persistence
- State transfer
- Service discovery
- Central/stateless configuration
- Load balancing
- Ability to monitor quality of the service
- Control throughput and latency
- Elasticity
- Authentication and authorization
- Auditing
- Data safety

The NGE is implemented as a set of microservices to support FIX session life cycle management. It also leverages the latest achievements in virtualization/containerization technology, load balancing and process orchestration available in open source projects. EPAM has built a FIX Load Balancing Cluster, enabling its products — FIXEdge and applications using FIX Antenna — to operate in NGE.

NGE LOAD BALANCING CLUSTER

The operation of the NGE Load Balancing cluster is backed by the following services and tools:

1. L4 Load Balancer (LB)
2. Configuration Service
3. FIXEdge Configuration Tool
4. Scheduler Service
5. Shared Configuration Storage
6. Shared FIX Message Storage
7. Shared Key-Value Storage for Session Endpoint State Management
8. Service Discovery Provider
9. Health Checks Provider
10. Log Collector
11. Authentication

Let's dig deeper into each of these services.

L4 LOAD BALANCER

The L4 Load Balancer redirects incoming connections to one of the FIXEdge nodes. Several nodes constitute a cluster, and at any moment at least one node is required for the cluster to be working. FIXEdge nodes can be dynamically added to and removed from a cluster without the need to restart the whole cluster.

The unit of work that is balanced over the cluster's nodes is a FIX session. The sessions are distributed between all nodes of the cluster where FIXEdge runs.

FIX acceptor sessions that were handled by the failed/removed node will be forwarded to other nodes upon the client's reconnect by load balancer. FIX initiator sessions will be moved to other nodes when health checks indicate a problem with the node/sessions that must be active.

L4 LOAD BALANCER (CONTINUED)

The duration of the failover process is significantly reduced compared to an active-passive cluster configuration. Once a FIX session is established, it is handled by the same node until logout or TCP connection termination.

Automatic FIX session balancing is performed by the L4 Load Balancer (the actual balancing strategy is configurable). The load balancer can be configured to forward all incoming FIX sessions to the same node, creating a pure active-passive high-availability cluster setup — a special case for a more general load balancing configuration. When the automatic balancing strategy is not efficient, manual pinning of the sessions to specific nodes is supported.

CONFIGURATION SERVICE

Configuration Service provides access to FIX sessions and schedules configuration via REST API. It stores the configuration in a relational database and benefits from native DBMS high-availability features. Configuration Service is a stateless service and can be transparently run on multiple nodes to avoid introducing a single point of failure into the cluster. Configuration Service will also be extended to support versioning of configuration.

FIXEDGE CONFIGURATION TOOL

FIXEdge Configuration Tool is an administrator's interface to configure the cluster. It provides commands to add, change and remove sessions and schedules, list all configured sessions, schedules, and currently established sessions, and indicate which nodes handle each of the sessions. These commands are handled via REST requests to Configuration Service and Scheduler Service.

SCHEDULER SERVICE

Scheduler Service is responsible for the initiation and termination of sessions according to a pre-defined schedule. To accomplish this task, Scheduler Service updates the session endpoint state in Consul and sends commands to the FIXEdge nodes. Upon starting, Scheduler Service loads the configuration from Configuration Service. It then receives updates from Configuration Service and FIXEdge Configuration Tool via the REST API during runtime.

Scheduler Service can be run on several nodes in the cluster to avoid being the single point of failure. To choose a primary Scheduler Service, the leader election technique is leveraged. Health checks are used to verify whether the Scheduler Service is alive and initiate a new round of the leader election when necessary.

SHARED CONFIGURATION STORAGE

Shared Configuration Storage is an RDBMS with all FIX sessions configuration information managed on the cluster.

SHARED FIX MESSAGE STORAGE

Shared FIX Message Storage is used to store FIX messages and sessions' last incoming and outgoing sequence numbers. FIXEdge reads the session state from the shared storage when required.

SHARED KEY-VALUE STORAGE FOR SESSION ENDPOINT STATE MANAGEMENT

Shared Key-Value Storage is used for storing the endpoint state of the sessions (connected, ready to connect, terminated) and locking/binding sessions to the FIXEdge nodes.

SERVICE DISCOVERY PROVIDER

All services, including FIXEdge nodes, could run on any node. Each service registers itself in the Service Discovery Provider using a well-known name. All processes use these names to locate the services they need.

HEALTH CHECKS PROVIDER

Service health is monitored by a pair of checks. The first verifies whether the service is working. For FIXEdge, it checks that the engine is able to establish new sessions as well as receive and send messages. For Configuration Service and Scheduler Service, the check tests specific REST API. The second verifies the network availability of the node. Combined, these checks provide a reliable way to verify service health.

Health checks are initiated by a Health Check Provider — a service that is a single authority for health status. The Health Checks Provider gathers the results of the actual checks and publishes them for consumers. All other services in the cluster use the health status published by the Health Checks Provider.

LOG COLLECTOR

FIXEdge features sending of the application log messages to a TCP collector such as Splunk.

PERFORMANCE

Cluster performance is determined by the number of nodes and the performance of individual nodes. FIXEdge performance is limited by the type of FIX message persistence storage.

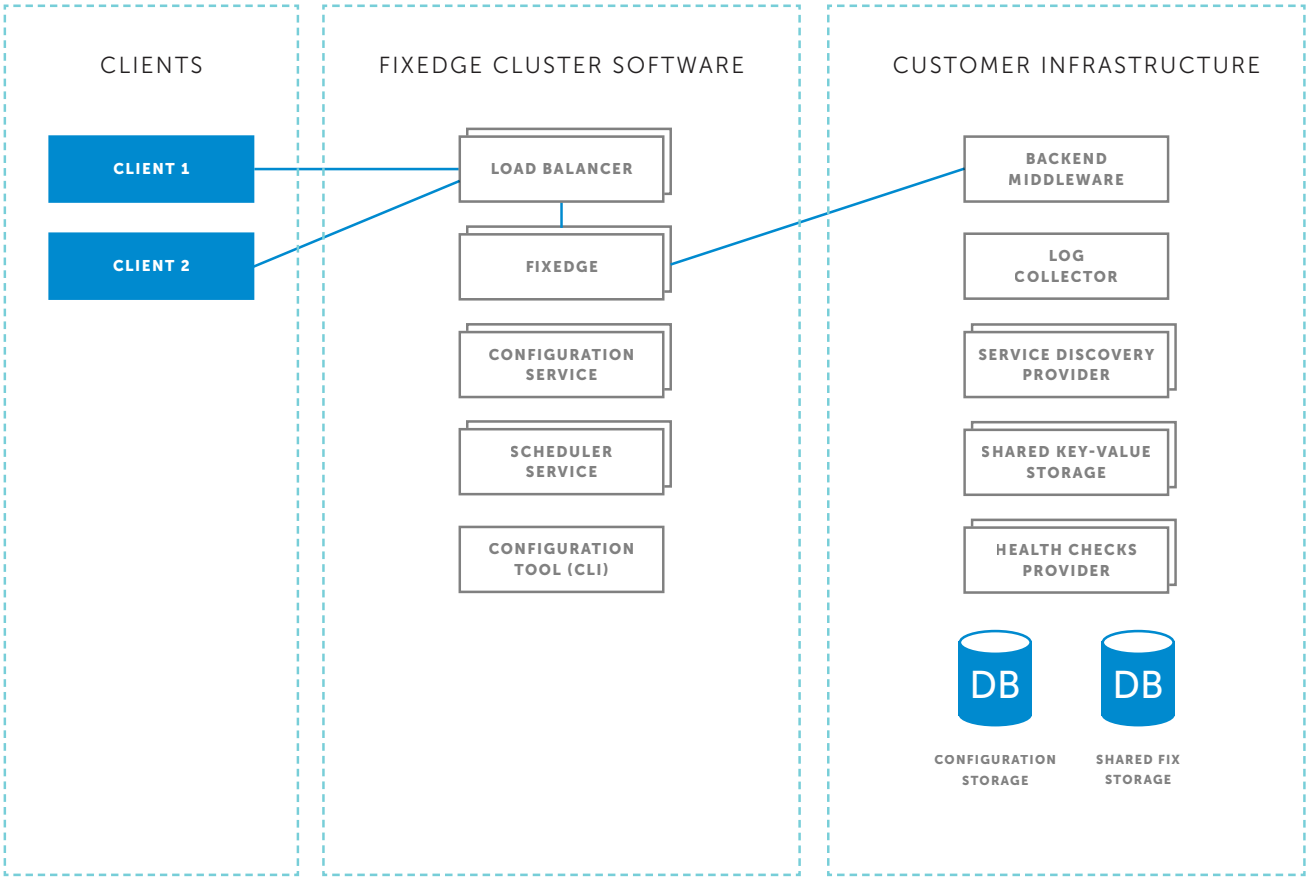
Earlier versions of the cluster worked on the flat files on SAN which provide the highest performance. SAN provides a block device which is mounted to one of the FIXEdge nodes at a time. This approach is suitable for high-availability setups only. For a load-balancing solution, we use a shared storage (Oracle Database).

PERFORMANCE (CONTINUED)

When we used Oracle instead of flat files, we started to worry about performance. Some clients want the uncompromised solution's performance. So, we plan to use clustered storage with better scalability and throughput (e.g. Cassandra), add support for other RDBMSes (e.g. PostgreSQL, MS SQL) to meet clients' existing infrastructure, and add replicated storage.

REFERENCE IMPLEMENTATION

The architecture diagram of the FIXEdge cluster reference implementation is presented in the figure below.



An HA Proxy is used as an L4 Load Balancer. An Oracle Database serves as a shared configuration and FIX message storage. Shared FIX storage can be implemented as flat files while these files are accessible to all cluster nodes (e.g. via NFS). HashiCorp Consul is used for Shared Key-Value Storage, Service Discovery and Service Health Checks.

FIXEdge, Configuration Service and Scheduler Service are designed to be integrated into a client's existing infrastructure, and dependencies on the services are replaceable.

SERVICES START AND REGISTRATION

At the start, each service registers itself in Consul and provides information on the service location (host and port) and a set of health checks.

FIXEDGE/FA APPLICATION START AND REGISTRATION

Upon FIXEdge startup, it loads configuration from Configuration Service. The location of the Configuration Service is received from the Consul Service Discovery. Then, it registers itself in the Consul Service Discovery, providing host and ports of FIX, FIXSSL and REST APIs and a set of health checks.

ACCEPTOR SESSION LIFECYCLE

Now, let's consider what happens when a client tries to connect to the cluster. On the cluster side, a suitable acceptor session should exist. The client establishes a connection to the HA Proxy node, which in turn redirects the connection to one of the FIXEdge nodes. Afterward, all session traffic is handled by the same FIXEdge node.

The client sends a Logon (MsgType = A) message to FIXEdge. FIXEdge reads the session endpoint state from Consul and rejects the session if the endpoint state does not allow a session to be established.

The sequence number of the Logon (MsgType = A) message should be checked against the last sequence number used during the previous connection of the same session, which may have been handled by another node of the cluster.

FIXEdge uses Oracle Database to store FIX messages and the last sequence numbers of each session. On the session negotiation, FIXEdge requests the last sequence numbers from the database and checks those against the Logon (MsgType = A) message. Based on this sequence number check, the session is either accepted or rejected.

When sequence numbers are validated, FIXEdge locks the session using its identifier (sender, target, qualifier) in Consul. If the session cannot be locked, it means that it was already locked by another FIXedge node. In this case, the session is rejected.

ACCEPTOR SESSION LIFECYCLE (CONTINUED)

After successful session negotiation, the client and FIXEdge start processing application FIX messages. Each FIX message and its sequence number is written to the database. At any time, the client can request to resend some or all FIX messages that were sent in the session. To fulfill the request, FIXEdge queries the messages from the database and sends them back to the client. Use of shared FIX message storage enables FIXEdge to respond with all messages of the session, regardless of the node they were initially sent to.

When the session is terminated, FIXEdge unlocks it in Consul to allow the next connection to be handled by another node. When a node's health check fails, all sessions that were locked by the node are released.

Consider balancing four sessions using a round-robin strategy over two FIXEdge nodes. Client 1 connects to the LB. The LB, according to the balancing strategy, forwards the connection to the FIXEdge1 node to handle all traffic from that connection. Client 2 connects to the LB and gets forwarded to the FIXEdge2 node. Client 3 connects to the load balancer and gets forwarded to the FIXEdge1 node. Client 4 connects to LB and gets forwarded to the FIXEdge2 node.

Consul periodically checks the health of FIXEdge on each node and tracks their current state. This state is used to adjust the set of nodes that HA Proxy redirects incoming connections to. Also, Configuration Service uses this state to forward configuration updates.

INITIATOR SESSION LIFECYCLE

Consider what happens when the cluster has to establish a connection to another system. Scheduler Service monitors the state of the initiator sessions and verifies it against the schedule. When it finds a session with a state not matching that required by the schedule, it issues a command to fix the situation.

To start an initiator session, Scheduler Service requests from Configuration Service what node the session should run on. The Configuration Service can provide either a specific node or an indication that any node can be used. In the case of a specific node, the Scheduler Service sends a REST API request to that node. In case of any node response, Scheduler Service sends a REST API request to one of the FIXEdge nodes via the HA Proxy. The HA Proxy then balances the request over available nodes. To stop the session, Scheduler Service sends a REST API request to all FIXEdge nodes.

When an initiator session is started, it begins actively connecting to the target system. Once the connection is established, the session reads the last sequence numbers from the database and uses them to create the Logon (MsgType = A) message. Afterward, the lifecycle of an initiator session is the same as it is for an acceptor session.

HEALTH CHECKS

To check whether a node or an application that runs on the node is operating properly, a health check procedure is run.

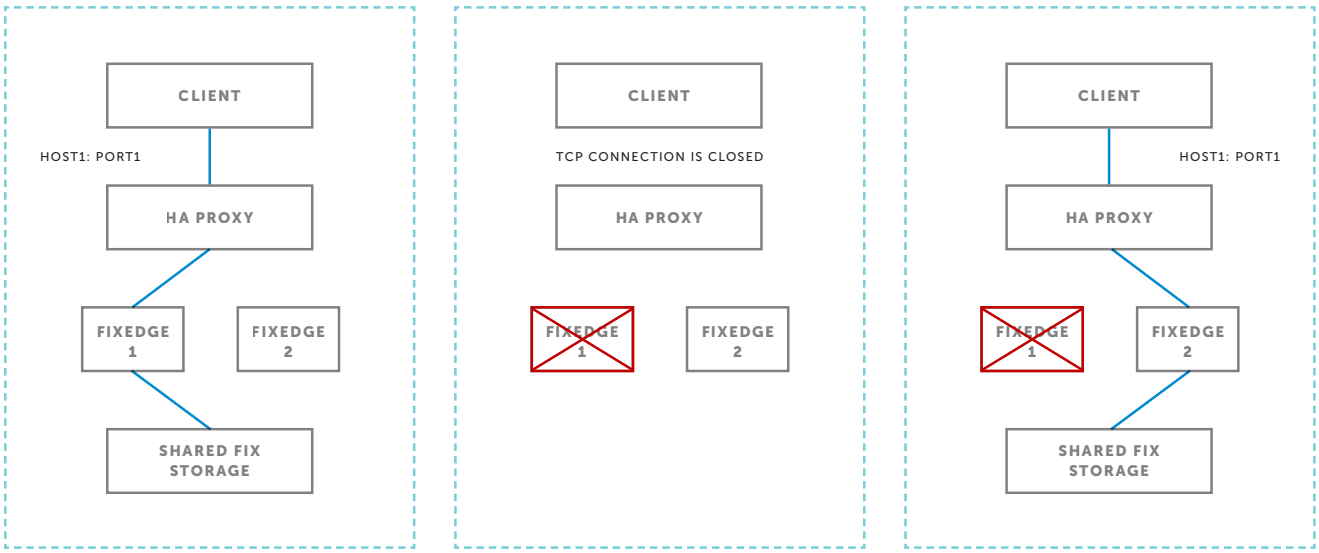
HEALTH CHECKS (CONTINUED)

Health checks of the FIXEdge nodes are performed on two levels. The first one is run on the same machine as FIXEdge and checks the state of FIXEdge itself, its ability to accept incoming connections, establish FIX sessions and process messages. This check is run periodically by Consul Agent. The second check is performed by Consul Agents between cluster nodes. The result of the health check is determined by a combination of these two checks.

Consul initiates all health checks and gathers their results. FIXEdge, Configuration Service, Scheduler Service and FIXEdge Configuration Tools read the health status of the services from Consul on demand. HA Proxy receives health status from Consul using Consul-Template — a special daemon running on the same nodes as HA Proxy.

HA SCENARIO

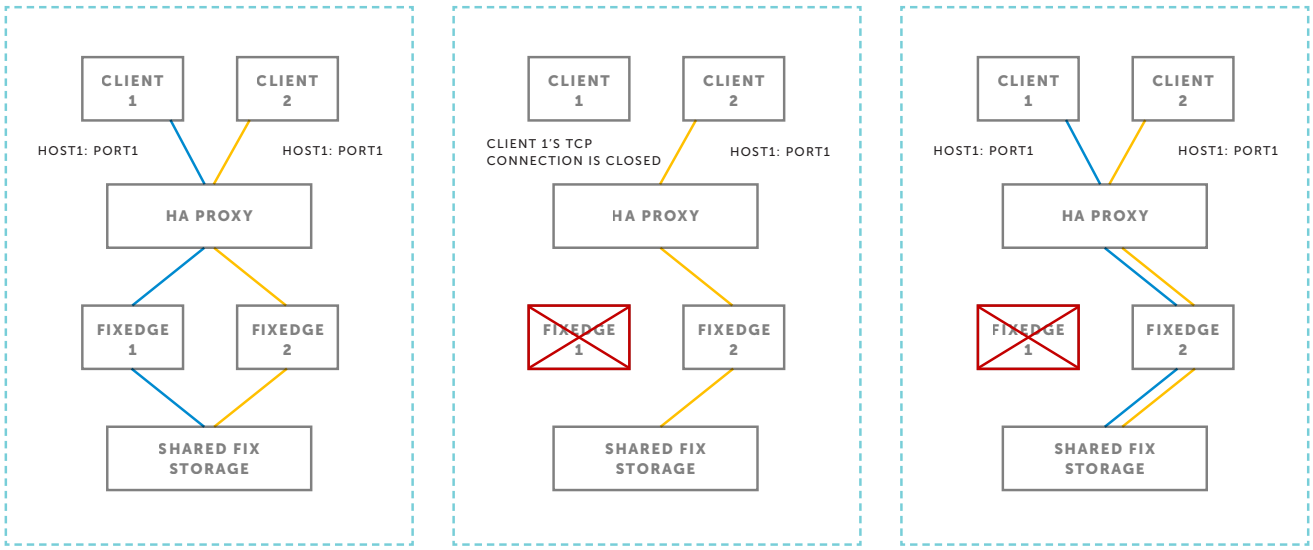
New Feature: High Availability



Consider a situation where the client is connected to the cluster and is handled by the FIXEdge 1 node. The messages that are passed to and from the client are stored in the Oracle database (Shared FIX Storage). After some time, the FIXEdge 1 node crashes, and the client's TCP connection is terminated. The client then reconnects to the cluster and the connection is forwarded to the FIXEdge 2 node (as FIXEdge 1 node is available). FIXEdge 2 fetches session information from the Oracle Database and continues the client's FIX session. The client always connects to the same address, which is bound to the HA Proxy. All changes in the working nodes in the cluster are completely transparent to the client.

LB SCENARIO

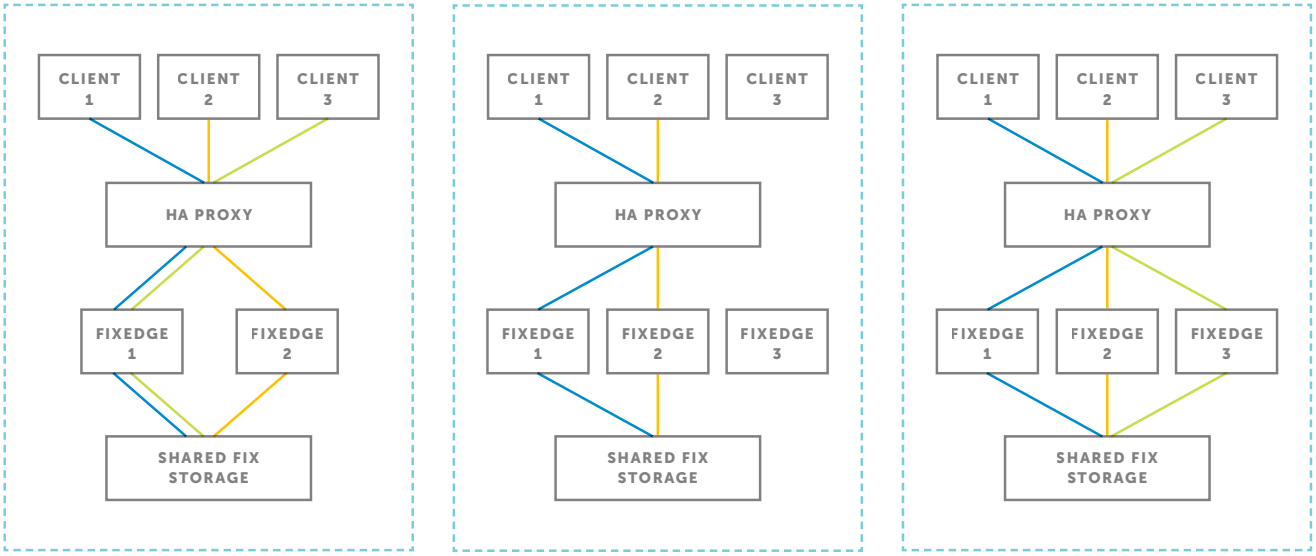
New Feature: Load Balancing



Consider a situation where two clients are connected to the cluster. Client 1 is handled by FIXEdge 1 while Client 2 is handled by FIXEdge 2. The messages that are passed to and from the clients are stored in the Oracle Database (Shared FIX Storage). After some time, the FIXEdge 1 node crashes, and Client 1's TCP connection is terminated. The client then reconnects to the cluster, and its connection is forwarded to the FIXEdge 2 node (as FIXEdge 1 node is available). FIXEdge 2 fetches session information via Configuration Service and continues the client's FIX session. The client always connects to the same address, which is bound to HA Proxy. All changes in the working nodes in the cluster are completely transparent to the client.

ADDING NODE TO THE CLUSTER AND REDISTRIBUTING FIX SESSIONS' LOAD

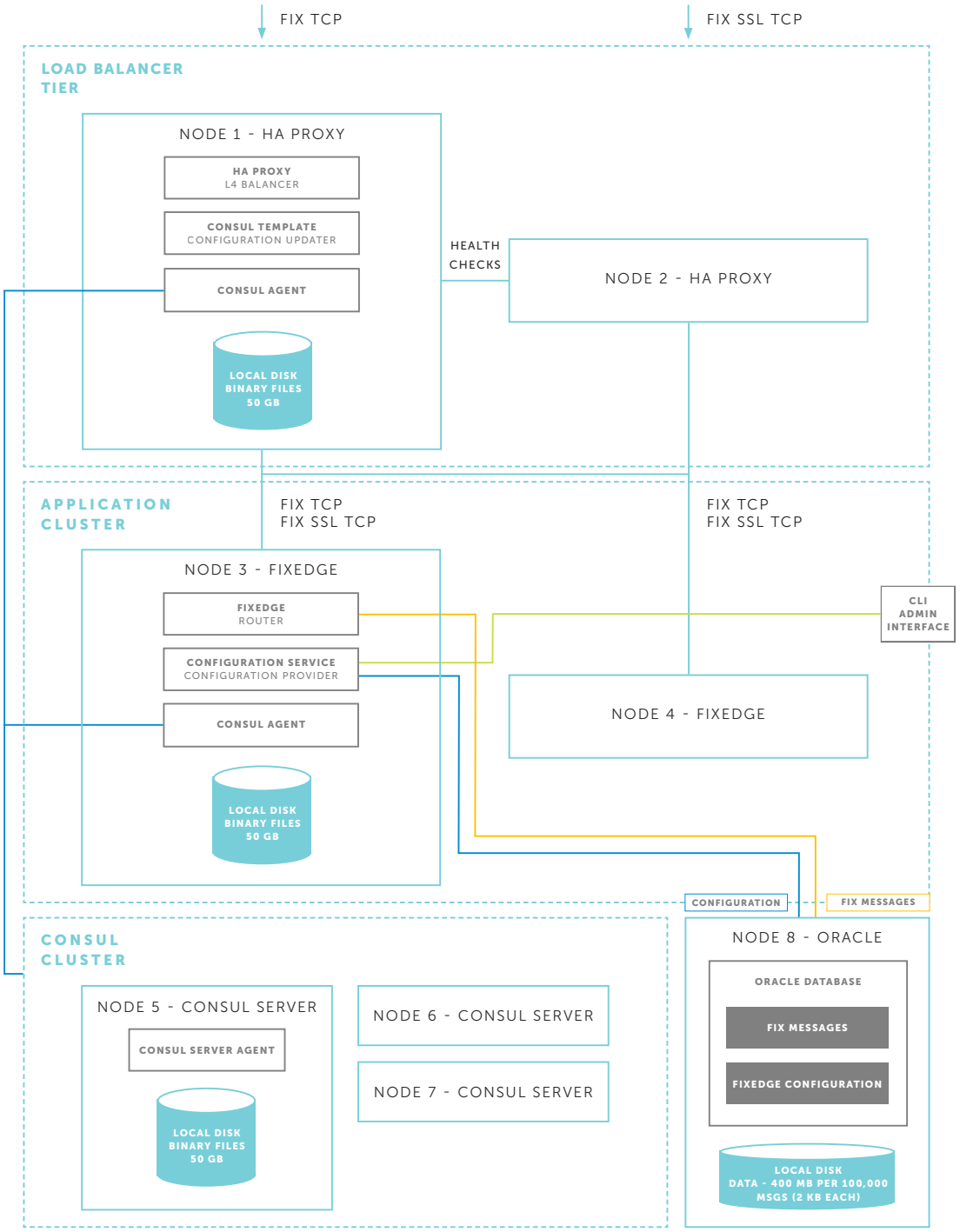
New Feature: Redistribute to New Node



Consider a situation where three clients are connected to the cluster. Clients 1 and 3 are handled by FIXEdge 1 while Client 2 is handled by FIXEdge 2. The messages that are passed to and from the clients are stored in the Oracle Database (Shared FIX Storage). After some time, a new node is added to the cluster. The administrator disconnects the Client 3 session to force the reconnect and rebalance the load. Client 3 reconnects to the cluster and is forwarded to FIXEdge 3. As result, the load on the FIXEdge 1 node was reduced.

DEPLOYMENT

The recommended deployment is presented in this diagram:



DEPLOYMENT (CONTINUED)

The solution is implemented in the microservice paradigm. Such an application is impractical to deploy manually; thus, we provide a set of scripts to automate the roll-out and CLI tool for administrators to configure.

Client connections go to the Load Balancing Tier, which consists of two nodes with HA Proxy load balancers. Clients connect to a floating IP that is assigned to one of the HA Proxy nodes. Load balancers forward connections to the two FIXEdge nodes. Each FIXEdge node hosts a FIXEdge server and Configuration Service. All Configuration Service instances are registered in Consul service discovery, and all connections to Configuration Service are established via Consul service discovery to ensure that a crash of one Configuration Service instance does not lead to failure of cluster configurability.

NGE is environment-agnostic. Currently, it is running on Linux, but the only component that requires Linux is the HA Proxy. All other components could be run under Windows. The cluster can be deployed on physical servers and virtual machines as well as in the cloud. The deployment of the cluster is automated and does not require manual deployment or setup of each of the components.

COMMAND LINE INTERFACE

To manage the cluster, a command line tool is provided. The configuration is applied to the cluster as a whole instead of configuring individual nodes. Currently, fectl covers the configuration of sessions and schedules. For example, to add a session to a cluster, the user runs the command:

```
fectl add-session --cluster-name=FIXEdge1 --file ./session.json
```

Here we specify two required options for the fectl add-session command: the cluster name and file with session definition. The command expects to have a local Consul agent running. The session definition file must contain a JSON object, e.g.:

```
{
  “Role”:                    “ACCEPTOR”,
  “ConfiguredName”:          “FIXAcceptor1”,
  “SenderCompID”:            “FIXEDGE”,
  “TargetCompID”:            “FIXCLIENT1”,
  “Qualifier”:               “”,
  “DefaultApplicationProtocol”:  “FIX44”,
  “Version”:                 “FIX44”,
  “StorageType”:             “oracle”,
  “InSeqNum”:                1,
  “OutSeqNum”:               1,
  “SenderLocationID”:        “SLocationId”,
  “TargetLocationID”:        “TLocationId”,
  “Username”:                “FIXAcceptorUser”,
  “Password”:                “AcceptorPassword”,
  “Description”:             “Testing Acceptor session”,
  “IntradayLogoutToleranceMode”: true,
  “ForceSeqNumReset”:        “OFF”,
  “SocketOpPriority”:        “EVEN”,
  “HandleSeqNumAtLogon”:     false,
  “RecreateOnLogout”:        false
}
```

Data is passed in JSON format, which makes it easy to use this tool in automation frameworks.

SUMMARY

We created a scalable platform to host FIX layer which takes advantage of the elasticity of dynamic infrastructure. By leveraging shared storage for preserving session states, the non-stop management of FIX sessions is achieved. Client session recovery by switching to another node takes only seconds.

Multiple instances of each service could be run; thus, no single point of failure exists in the system. The NGE provides a set of APIs to collect information about the platform state and performance, set parameters and control services. Main FIX Layer use cases such as Order Entry Gateway, Smart Order Router/Message Router, Market Data and Trade Capture/Drop Copy Servers can now be implemented in NGE.

ABOUT EPAM SYSTEMS

Since 1993, EPAM Systems, Inc. (NYSE: EPAM), has leveraged its core engineering expertise to become a leading global product development and digital platform engineering services company. Through its 'Engineering DNA' and innovative strategy, consulting, and design capabilities, EPAM works in collaboration with its customers to deliver innovative solutions that turn complex business challenges into real business opportunities. EPAM's global teams serve customers in over 25 countries across North America, Europe, Asia and Australia. EPAM is a recognized market leader among independent research agencies and was ranked #8 in FORBES 25 Fastest Growing Public Tech Companies, as a top information technology services company on FORTUNE'S 100 Fastest Growing Companies, and as a top UK Digital Design & Build Agency. Learn more at www.epam.com and follow us on Twitter @EPAMSYSTEMS and LinkedIn.

GLOBAL

41 University Drive,
Suite 202
Newtown, PA 18940, USA

P: +1-267-759-9000

F: +1-267-759-8989

EPAM_B267_V5 (09/19)

<epam>