**Solution Summary**

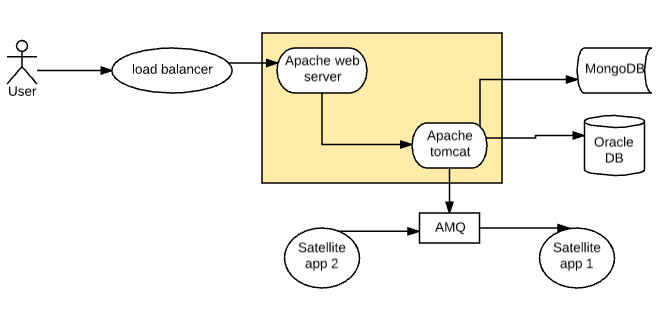
1. **Problem Overview**

Migrating a monolithic java application stack, consisting of Apache Web Server, Apache tomcat, Apache Active MQ, Oracle and mongoDB to AWS cloud platform.

Maintain high availability and load balanced architecture for system.

1. **System Context**

Below diagram illustrates overview of current services/platform in discussion:



User request comes via load balancer to Apache Web Server, Load balancer supports reverse proxy and TLS offloading, will forward the request to Apache Web server listening on port 8080, which serves static content and redirect the request to tomcat server running on port 7080, tomcat connects to mongoDB on port 27017, and serves read requests for faster performance, write operations are performed on Oracle, listening on port 1521.

AMQ is used for transmitting/exchanging messages between satellite applications, runs on a separate server, admin console is running on port 8080, and messaging port is tcp 61616, there is a single master node, and no slave node to support failover.

1. **Proposed Solution:**
   1. **Using SQS and DynamoDB:**

In this approach, we shift from existing tool stack to AWS managed services.

1. Apache Web Server and apache tomcat will be deployed on an EC2 instance as a part of LC-ASG-ELB setup spanning across more than one AZ’s.

This would ensure HA w.r.t unavailability of Availability Zones.

Amazon’s S3 bucket storage will be used for storing static content being served from server.

The Security group rules of the webservers would replicate existing n/w policies, similar ports would be opened here to allow HTTP, Oracle, NOSQL and SQS connectivity.

1. Mongo DB will be replaced by DynamoDB, AWS provides excellent support and easy manageability for DynamoDB, there is no direct performance benefit here other than easier manageability, however will required application level code changes to support DynamoDB.
2. AMQ will be replaced with Amazon’s SQS (Simple Queue Service), SQS is again easier to manage and provided as AWS managed service, notably unlike ActiveMQ, SQS is a polling based mechanism and not push based mechanism.

Application code changes in all satellite apps as well as system under discussion will be required to extend support for SQS.

1. Oracle: Oracle is offered as a managed service from AWS as part of the AWS RDS service offering, server will support no external access, only traffic from Tomcat EC2 instance will be allowed, as architecture remains same from DB perspective, no application level changes will be required.
2. Elastic load balancer will replace existing LB in system, Auto scaling feature will be used to make system fault tolerant, highly available and scalable.
3. Existing DNS will be replaced with Route 53 service, DNS failover feature will be used to enhance availability of application running behind ELB.
   1. **Using MongoDB and AMQ:**

In this approach, we will be using EC2, S3, RDS, ELB, Route53 as in above.

1. For AMQ: a new EC2 instance will be provisioned, Apache AMQ will be installed on this server, this will allow us to use same tech stack for messaging, meaning no code changes will be required, tradeoff will be. There is no tradeoff here except additional requirement of an EC2 instance and a storage.
2. For MongoDB: a new EC2 instance with S3 storage will be provisioned, MongoDB will need to be installed and configured.

**Conclusion**: 3.2 is the preferred approach for this migration.

Data migration approach will remain same for both the cases, Amazon’s DMS (Database migration Service) will be used to migrate the Data from Oracle to RDS, and MongoDB to either DynamoDB or MongoDB (running on EC2).

1. **Risks, Assumptions and dependencies:**
   1. **Risks:**

Introduction of new Services: EC2, ELB, Route53 and S3.

Mitigation Action: Transition session to be held for supporting teams.

* 1. **Assumptions:**
     1. DMS will not require oracle Downtime, at the point snapshot/GRP is taken for Oracle, no new data will be keyed in, thus maintaining integrity and no data loss once services go live on RDS.
     2. Data migration for both Oracle, MongoDB is in scope
     3. Services will use same port post migration, AMQ will run on 61616, oracle Db on 1521, MongoDB on 27017 etc.
  2. **Dependencies:**

Database credentials used by application to connect with Oracle and RDS will need to be considered, can be kept same, and user/password can be validated upon migration, if different will need to be synced before signing off the Data migration.