TRN5515 Industrial-strength Microservice Architecture with Next-Generation Oracle Database

Wei Hu, Vice President of Development
Anil Madan, Director of Development
Dominic Giles, Master Product Manager

Oracle Database Development

October 22, 2018
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.
Program Agenda

1. Microservice Architecture – Promise & Challenges
2. Data Management for Microservices
3. Customer Case Studies
4. Transactional Messaging System using Advanced Queuing
5. Wrap Up
Thought Experiment: True or False?

Microservices Require...

1. Running your application in Docker containers
2. Orchestrating your application using Kubernetes
3. Using Kafka for event processing
4. Using NoSQL
5. All of the above?
6. None of the above?
Microservices

- Organize application as set of well-defined, loosely-coupled services
  - Similar in some form to SOA development
- Makes development and the resultant application more modular
  - Improves the parallelization of development for an application
  - Makes the application easier to understand
  - Allows teams to deploy their services independently of other teams work
  - Able to leverage the best of breed solutions to develop/update the application
  - Works well in a continuous delivery model
- **Basically is the best thing ever... Until the next best thing**
Microservice Architecture

- Architecture Pattern, not just putting code into Docker containers
- Each microservice can run in a container
  - Private database and data model for each microservice
- Microservices communicate using asynchronous messaging via some event queuing system
  - Decoupled for maximum resiliency and scalability
  - Services do not talk to each other directly – only via event queuing service
  - Insulated from slowdown or failure of other microservices
Microservice Challenges

• Each service gets its own database that needs to be maintained, patched, upgraded, made HA, secured ...

• Hard to share and query data across services
  – Analytics requires the federation of data from various services
  – Even within the same application!

• Need for transactional event queuing / messaging system
  – Microservices are loosely coupled and communicate via Events
  – Reliable event queuing system (at-most-once, at-least-once, exactly-once, ...)

• Transactions that span multiple microservices
  – Microservice A calls B that calls C. Workflow only commits when C completes
  – How to undo A & B should C fail
  – Approach of using SAGAs requires creation of compensating transactions
Program Agenda

1. Microservice Architecture – Promise & Challenges
2. Data Management for Microservices
3. Customer Case Studies
4. Transactional Messaging System using Advanced Queuing
5. Wrap Up
Really nice to have’s for Microservices Data Architecture

An ACID compliant enterprise class database

Support for different data models i.e. Relational, Document, Graph, Key Value

A transactionally consistent messaging service to support Microservices events

Materialized views to support caching of data from other services to support cross-service analytics
Multitenant Database for Microservices
Reduce the cost and complexity of data management for new apps

- Microservices can use PDBs
  - Each microservice allocates its own dataset (can be a PDB, Schema, or a subset)
  - Each microservice has private data model
- Multitenant database containers deliver
  - Manage many databases as one
  - Secure separation of data
  - Easy sharing and querying of data across PDBs
- Database views can also isolate what each microservice sees
PDB Sharding for Microservices

Scalability, fault isolation and geo-distribution

- Want centralized database (CDB) with ultra-high availability and scalability – Exadata is great for this
- 19c also supports PDB Sharding
  - Each PDB can be sharded individually across multiple CDBs
- Provides fault isolation and geo-distribution for microservices
  - Loss of an entire CDB makes only part of a PDB unavailable
- Also allows each microservice to scale its PDB individually
  - More efficient use of resources compared to scaling a monolithic application (CDB).
Importance of Messaging

• The term ‘microservice’ may imply that you should look at the services first

• In fact, the best way to design is to think of the messages first

• A microservices-based architecture is described by the interaction of messages. This provides the abstraction that allows each microservice to be developed and evolved independently
  – Provided the messages remain the same, you can replace a service by one or more other services transparently. This gives you resiliency and scalability

• The messaging system also simplifies the architecture. Instead of figuring out which microservice talks to which other microservice, they all use messaging to publish/subscribe to messages/events
Oracle Advanced Queuing

**Transactional Event Queuing System for Microservices**

- Microservices send/receive events using Java Message Service (JMS) over Advanced Queuing (AQ)
  - JMS+AQ provide reliable and rich async messaging within Oracle DB
  - REST is not as good because of tight coupling between sender/receiver

- Provides all the resiliency and scalability of messaging and event processing, **AND**

- Unified transaction across messages and tables without two-phase commit

- Powerful SQL queries over messages
  - Important for debugging enterprise-scale microservices

- High performance: 12c Sharded Queues rearchitected for 10x higher throughput
Program Agenda

1. Microservice Architecture – Promise & Challenges
2. Data Management for Microservices
3. Customer Case Studies
4. Transactional Messaging System using Advanced Queuing
5. Wrap Up
Challenges

• State Grid is biggest electrical utility in the world – serves 1.1 billion people
  – #2 in Fortune Global 500 (2nd largest company globally by revenue)

• State Grid is adopting Microservice architecture for the growing business
  – Microservice architecture provides resiliency through loose coupling and flexibility to meet growing and changing business needs

• Microservice architecture in turn requires a highly scalable, loosely-coupled, and ultra-reliable distributed database

• State Grid wanted to prove that their business-critical ordering service system can leverage Oracle Sharding to meet these requirements
  – Ordering service currently has 200-300 million users distributed across 26 subsidiaries. Goal is to build centralized ordering service
Case Study: Oracle Identity Cloud Service (IDCS)
Largest production microservices-based application in Oracle
Why Identity Cloud Service uses AQ

1. **Stateless microservices:** *Keeping* state of messages in centralized location (DB) helps microservices to be truly stateless

2. **No message loss:** Dequeued messages are retained until explicit commit. Microservices can send events asynchronously without worrying about lost actions

3. **Local or distributed Transaction:** Commit/rollback helps to handle data and message in same transaction locally or across multiple microservices. Hard with non-DB based pub/sub system.

4. **Handles large message backlogs:** Database messaging helps smooth out spiky loads and tolerates services that are temporarily down
Program Agenda

1. Microservice Architecture – Promise & Challenges
2. Data Management for Microservices
3. Customer Case Studies
4. Transactional Messaging System using Advanced Queuing
5. Wrap Up
Transactional Messaging System using Advanced Queuing

- Introduction to Database Advanced Queuing (AQ)
- How AQ helps build resilient Microservices
- How two critical Oracle Cloud Services use Advanced Queue
- What’s new in Advanced Queue in 18c and 19c?
Advanced Queuing Overview

- Provides a reliable and robust messaging infrastructure within Oracle DB
- Stores messages in abstract storage units called queues
- Two Queues – Classic Queues and Sharded Queues (12c Onwards)
How AQ helps build resilient Microservices

- Allows reliable communication without coupling
- Allows stateless microservices
- Transacted session guarantees no loss of message and exactly once delivery
- Supports local and distributed transactions
- Supports a change notification mechanism on DMLs
- Local atomic transactions involving queue operations and DML
How AQ helps build Microservices Contd..

- Provides transparency
  - Works for any data
  - Works for PDBs and non-CDBs
- Support schema versioning using Edition Based Redefinition
- Handles large message backlogs
- Provides Queue level access privileges
- High availability and disaster recovery support (Logical Standby and Oracle GoldenGate)
/* Update the status of the order in the order status table: */
CREATE OR REPLACE PROCEDURE update_status(
    new_status IN VARCHAR2,
    order_msg IN BOLADM.ORDER_TYP)
BEGIN
    /* Query old status from the table: */
    SELECT st.status INTO old_status FROM order_status_table st
        WHERE st.customer_order.orderno = order_msg.orderno;

    IF new_status = 'SHIPPED_ORDER' THEN
        IF old_status = 'BILLED_ORDER' THEN
            return;       /* message about a previous state */
        END IF;
    END IF;

    /* Update the order status */
    UPDATE order_status_table st
    SET st.customer_order = order_msg, st.status = new_status;

    /* Enqueue the message into Shipping Queue */
    dbms_aq.enqueue('SHIPPING_QUEUE', enqopt, msgprop, order_msg, enq_msgid);

    COMMIT;
    ..........
END;
Identity Cloud Service based on AQ

IDCS usage of AQ:-
- Persist Audit published by various Microservices
- Manage notifications (Email, SMS at al.)
- Provision applications based on user/group activity

Q1_A – Subscription for Q1, Audit handler
Q1_N – Subscription for Q1, Notification handler
Total connections = 8 (2 Queues x 2 Worker groups x 2 Handlers)
ADWC & PDBCS asynchronous job execution framework based on AQ

Worker Pool for Job Type 1

Worker 1
Worker 2
... (up to Worker K)

Round Robin router

POD API
POD 1
CDB Cloud Scripts

POD API
POD N
CDB Cloud Scripts

Worker Pool for Job Type N

Worker 1
Worker 2
... (up to Worker K)

Round Robin router

Central CDB

Job Queue 1
Job Queue 2
... (up to Job Queue N)

Job Table

Broker

RESTFUL Requests

Copyright © 2018, Oracle and/or its affiliates. All rights reserved.
Sharded Queues

• Sharded Queues support was introduced in 12.1.0.2
  – A single logical queue with many “shards”
  – “Sharding” obtains higher concurrency and throughput via horizontal partitioning.
  – Automatic management of session affinity to shards
  – Automatic management of table partitions to avoid contention
  – Automatic management of partition instance affinity
  – Integrated with the database to optimize performance
  – 12.1.0.2 only supports JMS interface
Sharded Queues
Architecture for Scalability and Performance

Single logical queue model

Physical Queue: partitions mapped to shards with instance affinity on RAC
Sharded Queues

Key functionalities

- Automatic Management of Shards and Partitions
- Oracle Real Application Cluster Optimizations
- Message Cache
- Native JMS Support from 12.1 onwards
- Enqueue / Dequeue Optimizations
- Rules Engine
- Subscriber Management
- Event-based Listener
Sharded Queues: Contd...

Key functionalities

- PL/SQL support for Sharded Queues
  - PL/SQL Enqueue, Dequeue and Notification
  - JMS, ADT and RAW message payloads
  - Propagation between sharded queues
  - Message delay
  - Exception queue
- Message cache advisor for Sharded Queues
- Event-based JMS listener for Sharded Queues
- Long Identifiers for Classic and Sharded Queues
Sharded Queues

Existing AQ applications don’t need to be changed

PROCEDURE CREATE_SHARDED_QUEUE (  
    queue_name IN VARCHAR2,  
    storage_clause IN VARCHAR2 DEFAULT NULL,  
    multiple_consumers IN BOOLEAN DEFAULT FALSE,  
    max_retries IN NUMBER DEFAULT NULL,  
    comment IN VARCHAR2 DEFAULT NULL);
Sharded Queues

Key benefits

- Higher throughput
- Less system resource consumption
- Large number of subscribers
- Event-based listener with fewer database connections
- Many concurrent enqueuers and dequeuers across multiple RAC instances
- Backwards Compatible for Standard JMS and AQ PL/SQL based applications
  - just recreate the AQ in the database
Advanced Queue: Contd...

Key functionalities

- AQ-OGG support for Classic Queue
  - except Sharded Queues, Propagation and Array interface.
- Logical Standby support for Sharded Queues
- PL/SQL and JMS Array interface for Sharded Queues
- Python interface for Classic Queues (Raw Queues)
- Key based messaging (Coming in 19c)
- REST APIs (Admin, Monitoring coming in 19c)
## Sharded Queues

<table>
<thead>
<tr>
<th>Restrictions</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>No message retention</td>
<td>JDBC Thin Driver only</td>
</tr>
<tr>
<td>No Transformation</td>
<td>No ODP .NET support</td>
</tr>
<tr>
<td>No Enterprise Manager support</td>
<td>No OGG procedural replication support</td>
</tr>
<tr>
<td>No transactional grouping</td>
<td>No recipients support</td>
</tr>
<tr>
<td>No JMS-AQ extensions</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** Classic Queue supports all these features
Program Agenda

1. Microservice Architecture – Promise & Challenges
2. Data Management for Microservices
3. Customer Case Studies
4. Transactional Messaging System using Advanced Queuing
5. Wrap Up
Oracle Database as a platform for Microservices

- Multitenant Architecture allows microservices to have separate databases (PDBs) but still manage them as one (CDB)
- Oracle Advanced Queuing simplifies microservices design by providing transactional messaging for events
- Oracle Advanced Queuing also simplifies SAGA design by allowing row and messaging commits as a single transaction
- Oracle Database provides a true Multi Model engine supporting Relational, OLAP, JSON Document, XML, Graph, Spatial
- Oracle Materialized Views, multi-shard and cross-PDB queries support real time analysis
- Industry leading security, scalability, analytics, and high availability
Oracle Database simplifies deployment of Microservices

Sharded Database for Ultra High Availability
Other Microservices Sessions of Interest

TIP4071: Microservices, Containers, Databases, and Persistence Models
Kuassi Mensah, Director, Product Management, Oracle
Christopher Jones, Senior Product Manager, Oracle
Paul Parkinson, Consulting Member of Technical Staff, Oracle
10/23/2018 5:45pm-6:30pm Moscone West 3010

TIP4175: Data Management in a Microservices World
Gerald Venzl, Senior Principal Product Manager, Oracle
Monday 10/22 5:45pm-6:30pm Moscone West 3003