kindred

Oracle Database high availability from the client perspective

Priit Piipuu

WHOAMI: Priit Piipuu

- Started as a DBA twenty years ago
- Currently database performance engineer at Kindred Group
- Main job is to help developers use Oracle technologies in best way possible
- Oracle Ace Associate
- Blog: https://priitp.wordpress.com, twitter: @PPiipuu

Agenda

- Threats against application high availability
- Why this presentation, isn't buying RAC enough?
- What tools are available?

- The acronyms: from TAF to UCP, Application continuity included

- How it all fits together
  - What application sees during failover event
  - How things work in real world

Availability

- Protection from data loss
  - For service to be available data has to be ”there”
  - Ways to keep data from disappearing
    - Backups
    - Recycle bin, flashback query, flashback database
    - Data Guard (read-only replicas)

- Service availability
  - Data needs to be accessible
  - Resource Manager
  - Oracle Real Application Cluster (RAC)
  - Data Guard

Accidents happen

- Node crashes, hardware failure, network and storage failure
- Software crashes
- Software bugs
- User errors (Bobby Droptables)
- Cosmic scale catastrophe that brings down the data centre

Sometimes it's planned

- Regular security patching, both OS and database!
- Database upgrades. It is ~~2019~~ 2023 and Oracle STILL CANNOT do it in rolling mode!
- Environment rebuilding:
  - Years pass, we all recognise some mistakes („legacy“) that require full environment rebuilds

Protection from data loss

- Data Guard
  - Real time copy of your database
  - Not writable but can be opened for reading (Active DG, extra £)
  - Can be synchronous, or not
  - Role transitions can be automated (Fast Start Failover), offline for clients
  - Database operations are suspended during the transition
  - Since 18c, copy can be updatable, hiding role transitions
  - Can help with database upgrades with Logical Transient Standby

Service availability

- RAC: Real Application Clusters
  - Single database serviced read/write by multiple instances
  - Strongly consistent
  - Can span over short distances (max 100km)
  - Application still connects to the specific instances
    - In case of failure application needs to reconnect
  - Adding and removing instances is still disruptive

- GDS: Global Data Services
  - Hides configuration topology complexity from applications

Why this presentation?

- We have implemented MAA, Oracle says we are now highly available.
  - Not true, Oracle gives you many tools to achieve availability,
  - You must use them properly.

- True high availability is not possible without a little help from the client
  - Here we will see how the client side tools provided by Oracle work internally
  - And how to avoid mistakes we made (and I think we have not been alone)

Why do we need failover in application layer

- In case of failure in persistence layer, database clears up its mess. But!
  - In case of service/instance/node failure networking resources might not be gc'ed
  - Resolving networking timeouts takes time, up to 127s
- Graceful failover in case of planned outages
  - Stale connections are retired gracefully
  - New connection attempts are sent to the surviving nodes
- Extra features: connection load balancing and connection affinity

Transparent Application Failover - TAF

- Driver in application will detect database failure and reconnect silently
- SELECT statements can be silently restarted
- Transactions are lost
- JDBC thin driver does not support TAF

Fast Application Notification (FAN)

- Mechanism that notifies other processes about service configuration and status changes
- Provides visibility into Oracle services, status messages can apply to services, instances or nodes
- FAN status message is published immediately when state change occurs

- Messages can be accessed either through
  - Oracle client (JDBC, UCP, OCI, ODP.NET),
  - FAN API
  - Server-side callouts
- Supported by RAC, DataGuard, GDS and Net Manager (netca)
  - Starting with 19c, in-band FAN messages for planned outages

Fast Application Notification (II)

- What FAN helps to avoid:
  - Waiting on TCP/IP timeouts if node fails to close the sockets or during subsequent connections when IP address is down
  - Attempts to connect service which is down
  - Not connecting when services resume
  - Processing the result in the client side when service goes down
  - Attempts to execute work on suboptimal nodes

Oracle Notification Service

- Propagates notifications between Oracle Clusterware and FAN client
- Clients can subscribe for specific event types and/or services
- Until UCP 12.2 had to be configured explicitly, can be massive configuration pain
- In modern Oracle versions ONS, FAN and FCF (Fast Connection Failover) are enabled automatically when appropriate

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28468@oc1n14.example.com:0:1010440:1541760309;creationTime:1541760309;clusterId:dbInstance_oc1n13.example.com_6200;clusterName:dbInstance_oc1n13.example.com_6200;instanceId:dbInstance_oc1n13.example.com_6200;instanceName:dbInstance_oc1n13.example.com_6200;LocalOnly:false;numberOfProperties:11;Content-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

  - Cluster id
  - Cluster name
  - Instance id
  - Instance name

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedCom
eneratingComponent:database/rac/service;generatingPro
unibet.com;generatingNode:oc1n13.swe1.unibet.com;eve
unibet.com:0:1010440:1541760309;creationTime:154176
oc1n13.swe1.unibet.com_6200;clusterName:dbInsta
6200;instanceId:dbInstance_oc1n13.swe1.unib
ce_oc1n13.swe1.unibet.com_6200;Loca
Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.pte.unibet.com, db_domain=pte.unibet.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

Event type can be one of
- Service
- Service member
- Node
- Instance
- Database
- Service metrics

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedCompo…s:;generatingComponent:database/rac/service;generatingProces…e1.unibet.com;generatingNode:oc1n13.swe1.unibet.com;eventI…e1.unibet.com:0:1010440:1541760309;creationTime:15417603…ce_oc1n13.swe1.unibet.com_6200;clusterName:dbInstance…m_6200;instanceId:dbInstance_oc1n13.swe1.unibet…tance_oc1n13.swe1.unibet.com_6200;Local…ise;numberOf…nt-Length:210;SubscriberID:1;{reason=USER, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER, timezone=+01:00, service=fan_test.pte.unibet.com, db_domain=pte.unibet.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

Reason can be one of
- USER
- FAILURE
- member_leave
- public_nw_down
- BOOT

FAN messages: example

- (event origin:00000000.00000000.00000000.0a45800e!6200;stamp:5b9b8241.000f6b2d;Version:1.0;eventType:database/event/service;affectedComponents:;affectedNodes:;generatingComponent:database/rac/service;generatingProcess:7239@oc1n13.example.com;generatingNode:oc1n13.example.com;eventId:28…ple.com:0:1010440:1541760309;creationTime:1541760309;clus…c1n13.example.com_6200;clusterName:dbInstance_oc1n13.exam…anceId:dbInstance_oc1n13.example.com_6200;instanceNam…example.com_6200;LocalOnly:false;numberOfPropertie…nter…Length:210;SubscriberID:1;{reason=U…, database=ptecdb2, instance=ptecdb24, event_type=SERVICEMEMBER…timezone=+01:00, service=fan_test.example.com, db_domain=pte.example.com, host=oc1n12, timestamp=2018-11-09 11:45:09, status=down})

Status can be one of
- up
- down
- nodedown
- not_restarting

Fast Connection Failover

- Preconfigured FAN client-side integration, opaque for the application code
- FCF client can automatically subscribe and act on FAN events
- Uses FAN connection signature to match database sessions with events
- Acts based on event type, status and reason
  - status=down, reason=FAILURE – failure at database server, FCF aborts related connection immediately. Dead connections are removed from pooled clients
  - status=down, reason=PLANNED – FCF client drains the sessions ahead of planned maintenance
  - status=up – posted when service starts first time or resumes. FCF reallocates sessions so that load is balanced across the cluster
  - Load % -- enables runtime load balancing and QoS features

Connection pooling 101

- Oracle is optimised to work with long running sessions
  - To put it mildly
  - Creating a new connection/session is expensive, should not be done for every incoming web request
- Workaround: create a pool of database connections on application server
  - Initially, set of available connections are created
  - Incoming web request borrows one of the available connections
  - Application returns connection to the pool when done
  - Sessions get reused a lot

Connection pooling 101

- Problems for high availability:
  - When some connections are terminated, how should CP know about it?
  - Testing connections before lending it out could add significant overhead
  - Some connections could go to the overloaded instances

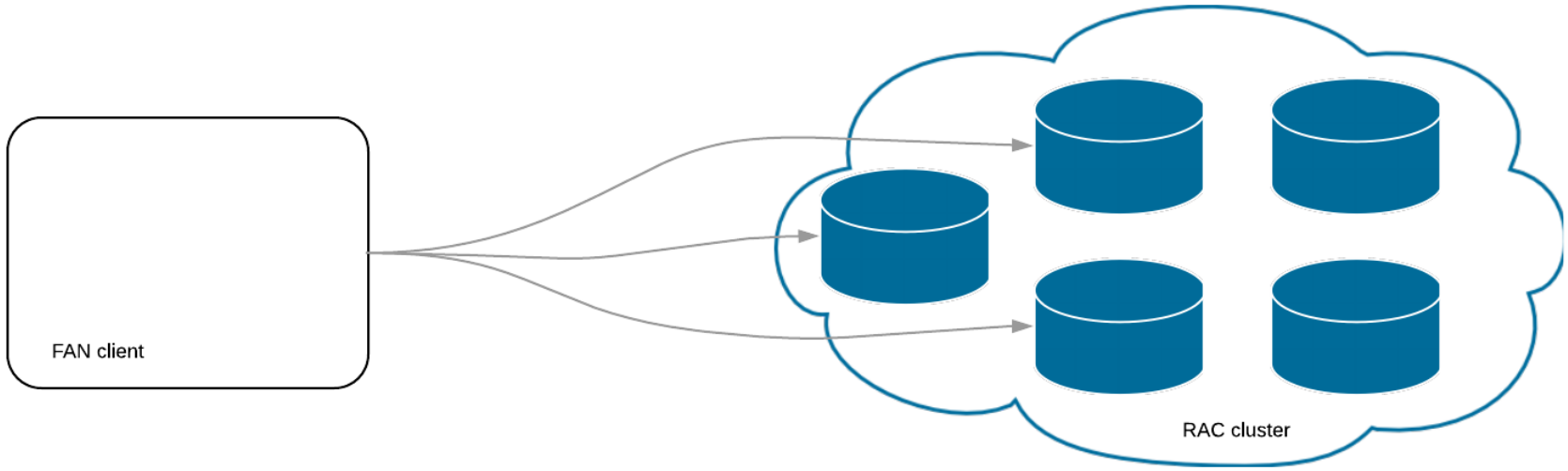
- Connections can be tested before lending them out
  - By sending commit or “select 1 from dual” and checking the result
  - If test fails, this single connection is removed from pool

- Maximum connection lifetime for planned maintenance
  - Stop services on an instance
  - Wait for a max connection life time to pass
  - Instance should be empty then of client connections

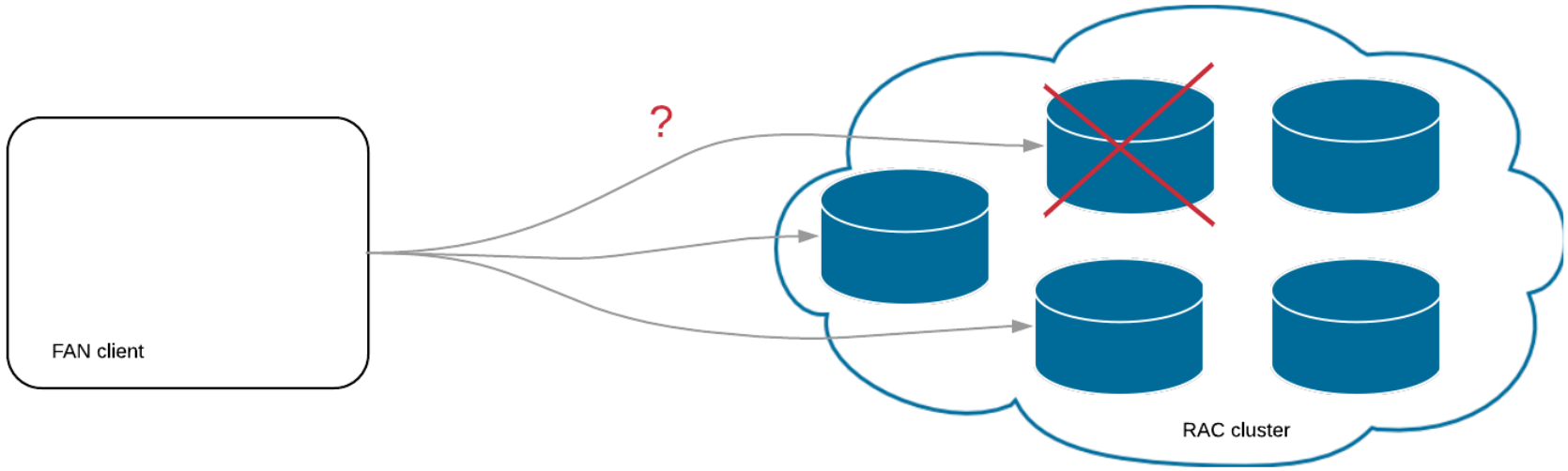
Case study: Universal Connection Pool

- Oracle provided JDBC connection pool implementing most of the High Availability features
- Universal: supposed to work with all behaving drivers
- Free to use
- Supports many cool features
  - Sharding!
  - Support for Application Continuity
  - Can send empty Net8 packet to the server for testing the connection
- Acts as client for Fast Application Notification
  - Feature is called Fast Connection Failover (FCF)

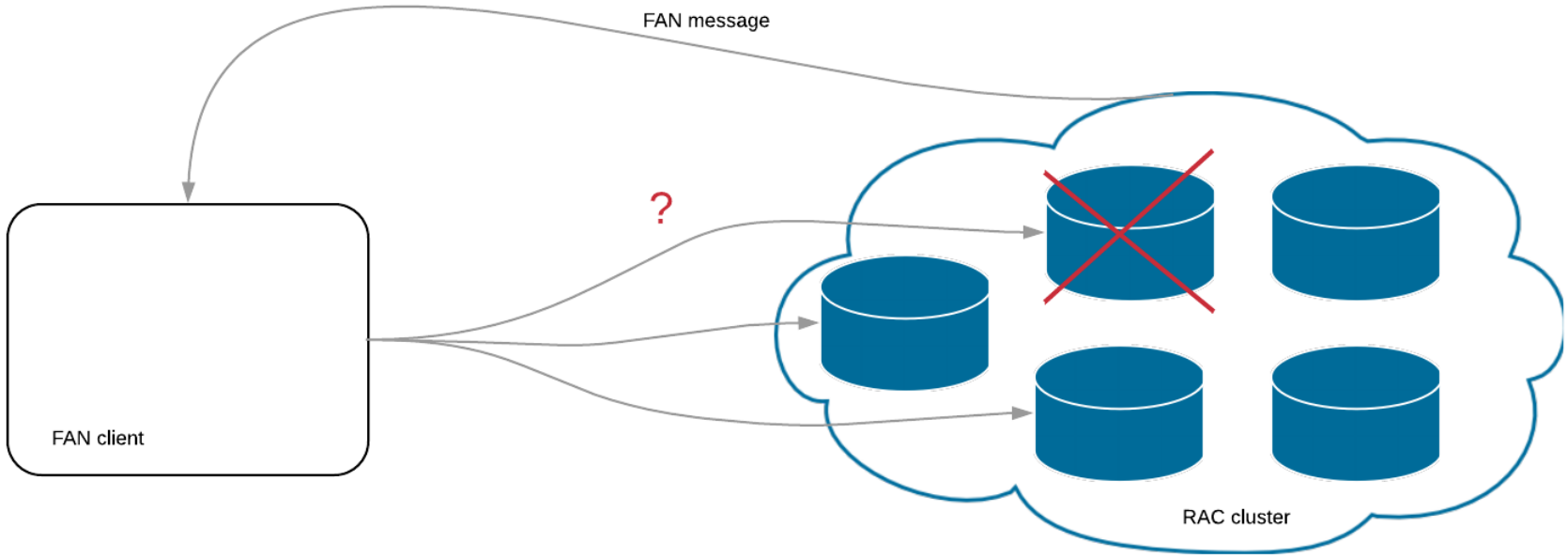
UCP: how failover works



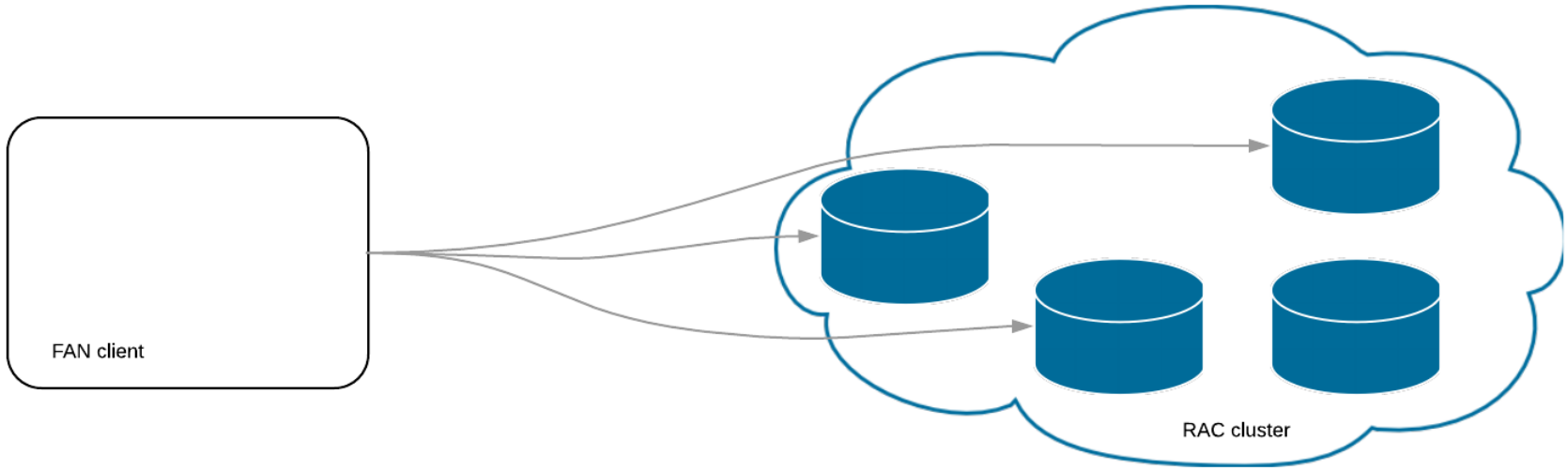
UCP: how failover works



UCP: how failover works



UCP: how failover works



UCP: handling the outages

- Unplanned outages
  - Finds stale connection that do not have service available on any instance in cluster due to service/down or node/down events
  - Operates both on borrowed and available connections, kills physical connections
  - In case of borrowed connection, application receives exception
  - Connections are not replaced in the pool, application is supposed to retry

- Planned outages
  - Stale borrowed connections are marked and will be aborted when returned to the pool
  - Does not affect any ongoing transactions
  - Main difference with unplanned outage is how borrowed connections are handled
  - There is graceful connection draining period
    - oracle.ucp.PlannedDrainingPeriod

UCP: load balancing and connection affinity

- Runtime connection rebalancing
  - Routes connections to the instance that has best performance available
  - Adjusts distribution of work based on different backend capabilities (CPU capacity or response time)
  - Reacts quickly to cluster reconfiguration, overworked nodes or hangs
  - Configured in server-side

- Connection affinity
  - Allows connection pool to select connections that are directed to specific instance.
  - Transaction based affinity
  - Web session affinity
  - Needs callback to be created (callback as in Java class that implements ConnectionAffinityCallback interface)

Road to availability: singleton services

- Good:
  - Performance: no cluster waits

- Bad:
  - Service relocation creates DOWN events for SERVICE and SERVICEMEMBER
  - UCP gets the message and does hard failover
  - All the sessions are terminated, and everything is dead until corresponding UP messages

Road to availability: uniform distribution

- Bad:
  - Not-so-good for performance (cluster waits)
- Good:
  - Planned outages do not cause major interruption
- Ugly:
  - Is UCP receiving FAN messages? Is it taking action?
  - Logging in UCP needs extra work
  - Badly behaving applications

UCP in a real world

- Only new connections are balanced, so set the timeouts
  - InactiveConnectionTimeout: how long available connections can be idle before being removed from the connection pool
  - MaxConnectionReuseTime: how long connection can be used before being gracefully removed from the pool
- Logging of FAN events is fixed in UCP 21c
- How to monitor what's going on?
  - Does Simplefan work?
  - FANwatcher for the rescue?

Transaction Guard and Application Continuity

- Transaction Guard provides transaction idempotence
  - Transactions are tagged with Logical Transaction Identifier (LTXID)
  - This can be used by application after the failure to check the outcome
  - Has API for client, and a server-side API (DBMS_APP_CONT package)
  - Application Continuity for Java uses client-side API for session recovery and SQL replay

Application Continuity (I)

- General purpose recovery solution for the application
  - In case of crash status of in-flight transactions are checked
  - Uncommitted transaction will be replayed
- Implemented through separate data source
- Database service must be configured for the Application Continuity

Application Continuity (II)

- Transparent Application Continuity
  - Tracks and records session and transaction state
  - No need for code changes

- Mutable values can be retained
  - Applies to sequence.NextVal, SYSTIMESTAMP, SYS_GUID, etc
  - User needs KEEP privileges for that

- Does it work for availability?
  - Consider how long brownout in RAC or role transition with DataGuard FSFO will take