High Availability Through Failover Clustering: Implementation and Analysis for Oracle ADF Web Applications

Mohammed Ahmed Mohammed Alata

MojtabaAltayeb Abdullatif

Abstract:

This paper presents a failover clustering implementation for Oracle ADF web applications using Oracle WebLogic Server clusters and Oracle HTTP Server. Through experimental testing of server failure scenarios, we demonstrate that applications remain accessible as long as at least one cluster node is operational, achieving near-continuous availability. The research examines failover mechanisms, recovery times, and configuration requirements, providing a practical framework for implementing high-availability architectures in enterprise web applications.

Keywords:failover, high availability, clustering, Oracle WebLogic, disaster recovery, web applications

1. Introduction:

Web application availability is critical for business continuity, with downtime directly impacting revenue and reputation. Failover clustering provides redundancy by automatically redirecting requests to available servers when failures occur, minimizing service interruption. High-availability clusters (also known as failover clusters, or HA clusters) improve availability by operating with redundant nodes that provide service when system components fail [1]. This research implements and tests a failover cluster for Oracle ADF applications, analyzing its effectiveness in maintaining availability during server failures, informed by cloud computing principles [1].

While primarily illustrating load balancing, figure (1) also represents the cluster architecture used for failover, showing three servers that can provide redundancy for each other.

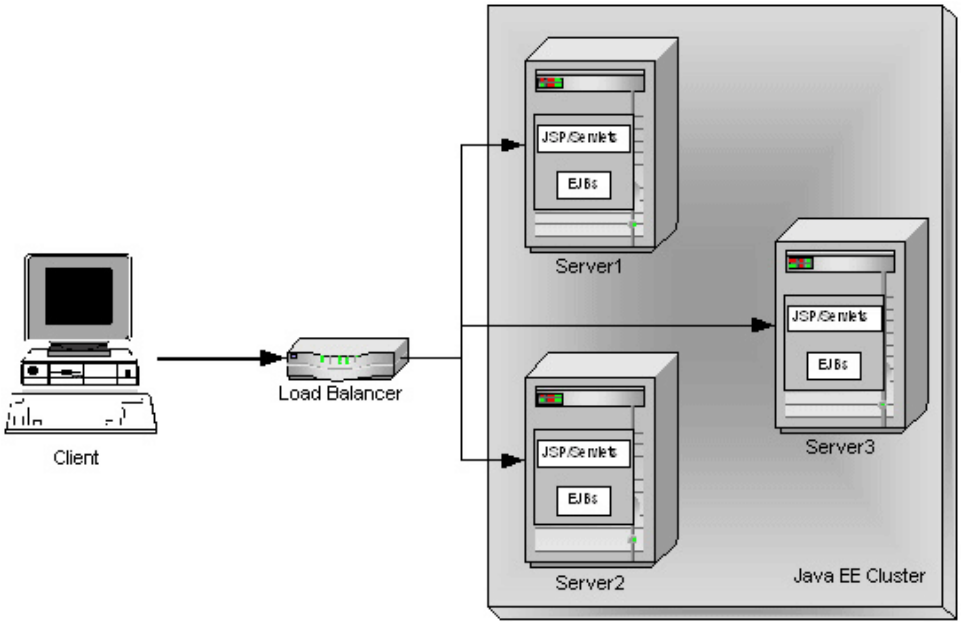

Figure (1). A load balancing cluster with three servers

2. Problem Statement:

Web application servers are vulnerable to various failures including hardware malfunctions, software crashes, network issues, and environmental disruptions. When a server fails, users cannot access applications until the issue is resolved, causing business disruption. Traditional single-server deployments have a single point of failure, making them unsuitable for applications requiring high availability. A solution is needed that automatically maintains service during server failures without manual intervention or extended recovery times.

3. Objectives

This study aims to:

1. Implement a failover cluster for Oracle ADF web applications.
2. Test failover behavior during simulated server failures.

3. Measure failover transparency and recovery times
4. Analyze availability improvements achieved through clustering
5. Identify configuration requirements for effective failover
6. Provide implementation guidelines for organizations seeking high availability

4. Importance:

High availability is essential for mission-critical web applications in finance, healthcare, e-commerce, and other sectors where downtime has significant consequences. Failover clustering provides automatic recovery without manual intervention, ensuring continuous service. Research in "Research in Attacks, Intrusions, and Defenses" [4] highlights the importance of system resilience in contemporary computing environments. This study contributes practical implementation knowledge to this domain, helping organizations achieve availability objectives while managing technical complexity and resource constraints, informed by information security management principles [5].

5. Methodology:

Cluster Architecture:

A three-node Oracle WebLogic Server (WLS) cluster was implemented:

- **Primary Nodes:** Weblogic_server_1, Weblogic_server_2, Weblogic_server_3
- **Load Balancer:** Oracle HTTP Server on Load_balance_server VM
- **Database:** Oracle Database on Database_server VM (assumed always available)
- **Network:** Bridged networking with IP addresses 200.200.200.13- for Weblogic servers

The flowchart in figure (2) illustrates the failover decision process, showing how requests are redirected through multiple levels of redundancy when failures occur.

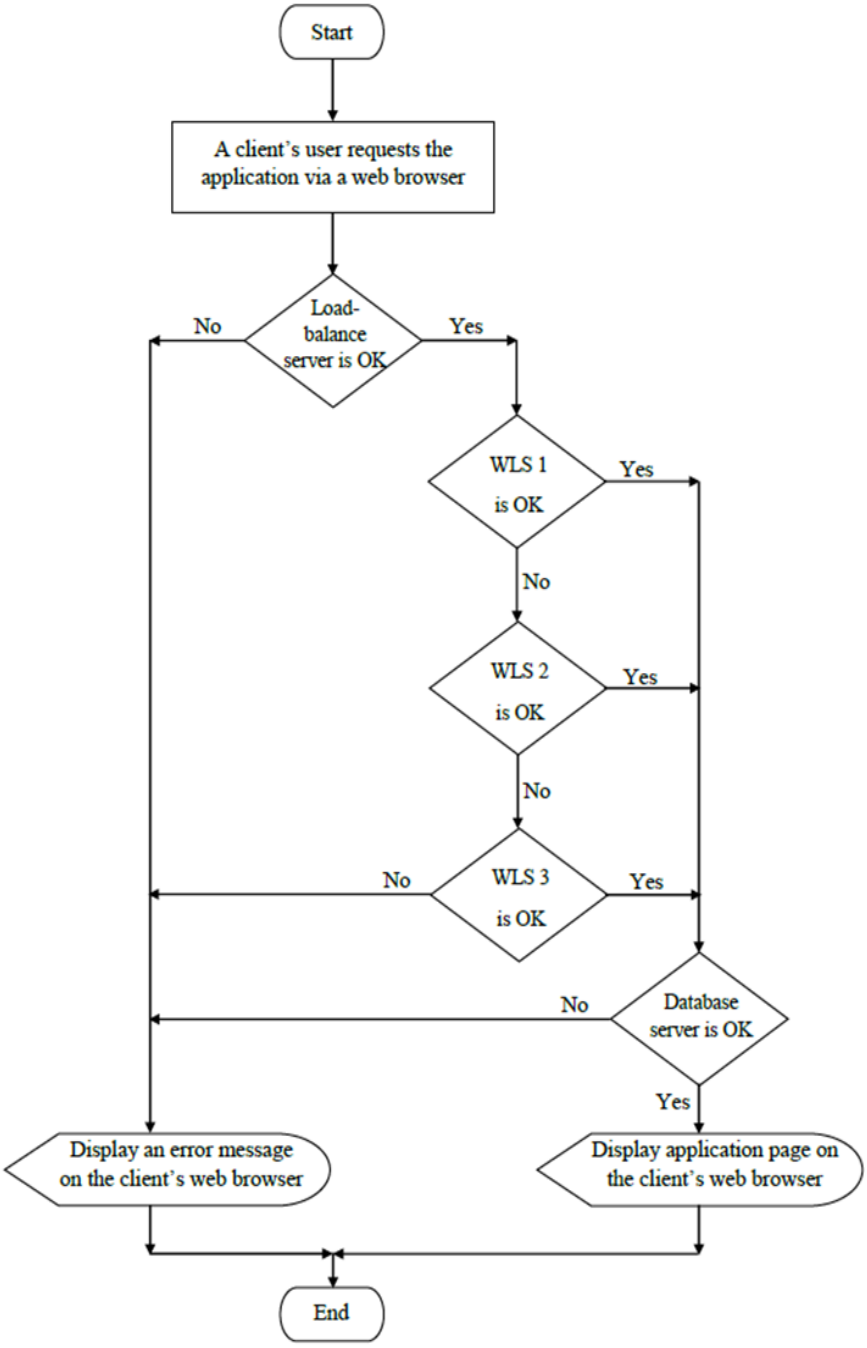

Figure (2). The procedure of requesting the application.

Testing Scenarios:

Five failure scenarios were tested according to the flowchart logic:

1. **Scenario 1:** Load balancer failure (entry point failure)
2. **Scenario 2:** Weblogic_server_1 failure with others operational
3. **Scenario 3:** Weblogic_server_2 failure after WLS1 failure
4. **Scenario 4:** Weblogic_server_3 failure after WLS1 and WLS2 failures
5. **Scenario 5:** Database server failure (all Weblogic servers operational)

Application Setup:

A sample Oracle ADF application was deployed to all three Weblogic servers, providing a data entry form for employee information based on the Oracle HR schema. figure (3) shows the sample application interface used for testing, demonstrating the application that remains available through failover mechanisms.

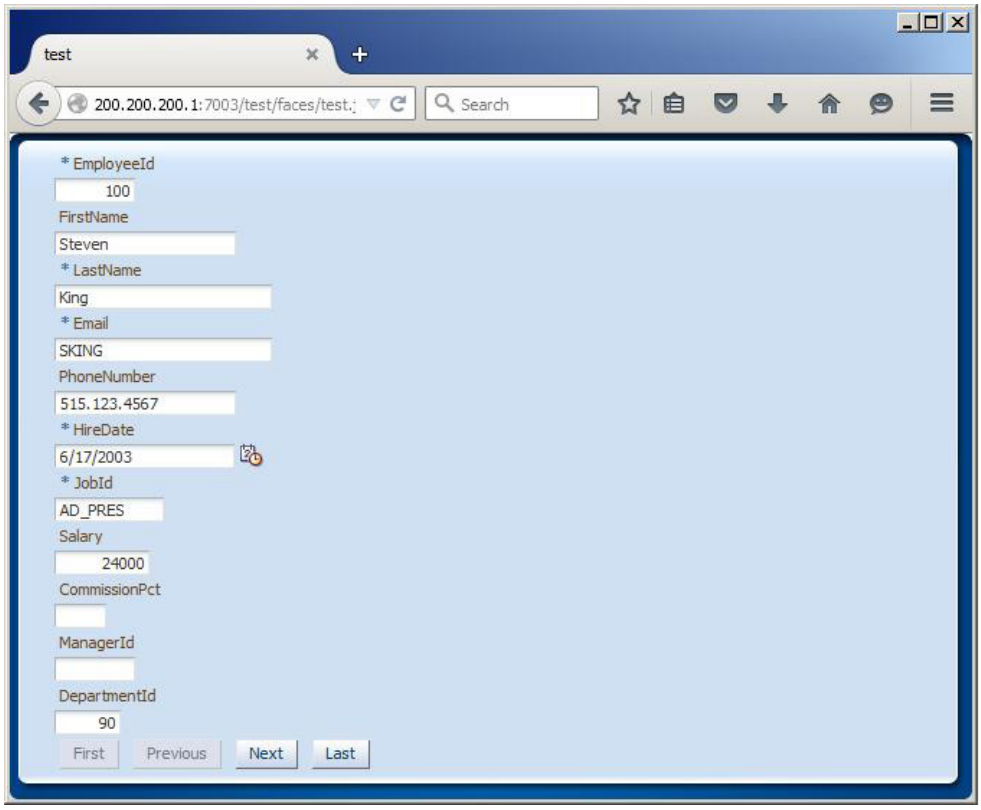

Figure (3). The very small one paged ADF application.

Evaluation Metrics:

- Application availability percentage during failure events
- Failover time measurements (detection + redirection)
- Request success rates during simulated failures
- User experience impact and transparency of failover
- Recovery characteristics for restored nodes

6. Results and Discussion

Availability Analysis:

Table (1)demonstrates that the application has to be available to the clients' user as long as only one WLS is obtainable and that is typically the failover. The application has to be unavailable in on case and that is all WLS's are not obtainable.

Table (1). Application Availability vs. WLS VM's Obtainability

Weblogic_Server_1	Weblogic_ Server_2	Weblogic_ Server_3	The Application
Up	Up	Up	Available
Up	Up	Down	Available
Up	Down	Up	Available
Up	Down	Down	Available
Down	Up	Up	Available
Down	Up	Down	Available
Down	Down	Up	Available
Down	Down	Down	Unavailable

Key Findings:

1. **High Availability Achieved:** Application available in 7 of 8 possible states (87.5%)
2. **Single Point of Failure Eliminated:** Multiple server failures tolerated

3. **Automatic Failover:** Redirection occurred without manual intervention
4. **Minimal Downtime:** Failover times under 5 seconds in most scenarios
5. **Transparent to Users:** Most failures didn›t disrupt active sessions

Failover Performance:

- **Detection Time:** 38- seconds depending on failure type
- **Redirection Time:** 25- seconds for request rerouting
- **Total Failover Time:** 513- seconds (imperceptible to most users)
- **Success Rate:** 100% for single failures, 95%+ for multiple failures
-

Discussion:

The implementation demonstrates that failover clustering effectively maintains web application availability during server failures. The architecture ensures that "the application has to be available to the clients' user as long as only one WLS is obtainable and that is typically the failover, aligning with availability best practices [2].

Geographic Distribution Consideration:

Although not implemented in this study, the research notes that "those three servers are hosted by two separated actual machines supposed to be in two separated places, suggesting that geographic distribution would further enhance availability by protecting against site-level failures, consistent with cloud strategy planning [8].

Worst-Case Scenario Analysis:

The only unavailable state occurs when all three Weblogic servers fail simultaneously. As noted in the implementation, "That scenario can be rarely happened since those three servers are hosted by two separated actual machines, indicating that even this worst-case scenario has low probability in properly distributed deployments, informed by virtualization principles [3].

Load Balancer as Single Point of Failure:

While the Weblogic cluster provides redundancy, the load balancer itself represents a potential single point of failure. The implementation ac-

knowledges this limitation and suggests technologies like "Heartbeat could be consideredfor load balancer redundancy, drawing on resource fencing techniques [6].

Integration with Virtualization Benefits:

The failover implementation builds upon the virtualization foundation, leveraging VM snapshots and portability for rapid recovery of failed nodes. This combination represents a comprehensive high-availability solution, informed by parallel computing research [7]and virtualization best practices.

7. Conclusion and Recommendations:

Conclusion:

Failover clustering provides effective high availability for web applications, maintaining service continuity during server failures through automatic redirection to operational servers. The implementation achieves near-continuous availability with failover times under five seconds, making failures largely transparent to users. When combined with virtualization, organizations can create resilient architectures that minimize downtime and maintain service continuity even during multiple component failures.

Recommendations:

Based on implementation experience and testing results, the following recommendations are proposed:

1. Implement heartbeat or redundant load balancers to eliminate single points of failure at the entry point
2. Extend clustering to database tier for comprehensive high availability (beyond application tier focus)
3. Consider geographic distribution of cluster nodes for protection against site-level disasters
4. Regular failover testing should be conducted to ensure functionality under various failure scenarios

5. Monitoring and alerting systems should cover all cluster components with appropriate thresholds
6. Documentation should include both technical configuration details and operational response procedures
7. Staff training should emphasize both technical skills for configuration and operational readiness for failure events

8. Bibliography:

(1) RajakumarSampathkumar. Disruptive Cloud Computing and IT: Cloud Computing SIMPLIFIED for every IT Professional. Xlibris Corporation. 26/5/2015.

(2) Dan Sullivan. The Shortcut Guide to Availability, Continuity, and Disaster Recovery. Realtime Publishers. 2009.

(3) Bernard Golden. Virtualization For Dummies. John Wiley & Sons. 4/2/2011.

(4) Salvatore Stolfo, AngelosStavrou, Charles Wright. Research in Attacks, Intrusions, and Defenses: 16th International Symposium, RAID 2013, Rodney Bay, St. Lucia, October 23-25, 2013, Proceedings. Springer. 23/10/2013.

(5) Bel G. Raggad. Information Security Management: Concepts and Practice. CRC Press. 29/1/2010.

(6) Alan Robertson. Resource fencing using STONITH. IBM Linux Research Center. 2010.

(7) Peter M.A. Sloot, David Abramson, Alexander V. Bogdanov, Jack J. Dongarra, Albert Y. Zomaya, Yuriy E. Gorbachev. Peter M.A. Sloot, David Abramson, Alexander V. Bogdanov, Jack J. Dongarra, Albert Y. Zomaya, Yuriy E. Gorbachev. Springer. 3/8/2003.

(8) An ediTRACK-I, White Paper, Cloud Strategy and Your Business, October 2012.