Free Open-Source High – Availability Solution for Java Web Application Using Tomcat And MySQL

Dhanny[1], Sandi Badiwibowo Atim[2]

[1,2] PT. Danwin Cipta Niaga, Jakarta Selatan 12430, Indonesia

Article Information

Received:
Accepted:
Published:
DOI: 10.33555/ejaict.v…

Corresponding Author:

Dhanny
Email: dhanny@gmail.com

ISSN 2355-1771

ABSTRACT

With the growth of the internet, the number of web applications is also growing. Many web applications are becoming more important to the stakeholders that they cannot afford downtime which can cause loss of revenue, loss of productivity, etc. In the past, only big organizations with deep pocket could afford implement high-availability for their web application, but nowadays there are free open-source software programs that support high-availability feature available to everyone. This research studied the feasibility of implementing high-availability for Java web application system without using commercial software. This research compared the capability of proprietary and free open-source high-availability solution for Java web application based on a simple high-availability design, where a test Java web application was deployed into the environment based on proprietary and free open-source solutions, and tested how well each solution perform when problem occurs. The result showed that the free open-source high-availability solution worked, but not as well as proprietary one. However, the proprietary high-availability solution for database did not perform well, and neither did the open-source one. This research concludes that the free open-source high-availability solution works and thus made high-availability become much more affordable, especially for individual or small organizations with budget constraints.

Keywords: *High Availability, Java, Web Application, Architecture*

1. Introduction

Along with the growth of the internet, the number of web applications has also been growing rapidly. The web applications can vary in size, complexity, number of users, working hour, etc.

These web applications serve different purposes to their users. There are web applications for commercial as well as non-commercial purposes. The businesses of all size already understand the power of the internet, and leverage the internet to grow their businesses. In fact, many businesses built their business foundation on web applications. For these organizations, it is very important to ensure their web applications continuously operate without any or minimum disruption. There are also web applications built for non-commercial purposes and community websites providing online services for its member and many others. The availability of these web applications is also as important as the commercial ones.

It can be seen that in most situations if not all, it is very desirable that these web applications can always operate without any disruption, or in other words, implement high-availability.

1.1. Technology

The early version of the web started as static web pages which visitor can access to view their content [1]. When a visitor accessed the website, the server serves the page based on the static information in the file system, such as documents and images [2]. However, these static websites quickly evolve into dynamic websites.

Web application is defined as an application that is accessed via the Web browser over a network such as the Internet or an intranet [3]. In this document, we also refer to web application as website that is powered by server-side program scripts to provide dynamic behaviour. Thus, in general this document does not refer to website with static contents as web application.

1.2. Stateless and Stateful Web

Nowadays, when visitor visits a website, typically he would expect the website to remember his credentials, preference, locations, etc., even if it is only temporary. However, the core HTTP protocol used by the Internet communication is stateless. Stateless means that all information about a request must be stored in the request itself [4]. To achieve a useful user interaction, it would be necessary to be able to keep information longer than the duration of a single request-response cycle. Java has the concept of session which represents a series of request-response exchanges between a user and a web application.

1.3. Multi-tier Web Application

With proper sizing, one can always setup a complete web application server on a single computer. In many scenarios, this approach is acceptable.

According to Java Enterprise Edition blueprint, the functionality of an application can be separated into isolated functional areas called tiers [5]. The following are the description of the tiers:

1. The client tier consists of application clients that access the server. It is usually located on a different machine from the server.
2. The web tier consists of components that handle interaction between clients and the business tier.
3. The business tier consists of components that provide business logic for an application.
4. The enterprise information systems (EIS) tier consists of database servers, enterprise resource planning systems, and other legacy data sources, like mainframes.

1.4. High Availability Definition

A simple way of understanding high-availability is to think that a system must be always functioning at all time. However, this definition does not provide sufficient detail for proper management of the system. The simplistic definition above is also a very idealistic one. In practice, high-availability always comes with a cost. Thus, the benefits of having high-availability system must justify its cost. Some formal definitions related to high-availability: Availability is the ability of an IT service or other configuration item to perform its agreed function when required [6].

Downtime is the time when an IT service or other configuration item is not available during its agreed service time [6].

Availability is calculated or measured as the percentage of time that a system operates during its intended duty cycle [7]. In the context of high-availability, the availability metric is often specified by the number of 9 (nines).

1.5. High Availability Design

There are several design techniques relevant to achieving high-availability: Load balancing, Mirroring, Clustering. “Server load balancing deals with distributing the load across multiple servers to scale beyond the capacity of one server and to tolerate a server failure" [8]. With respect to web application, a load balancer is usually placed between the clients and several web servers.

1.6. Cause of Downtime

A study stated that planned downtime accounted for 80% of all downtime, while less than 20% was unplanned downtime [9]. The causes of planned downtime include activities such as backup, recovery, hardware upgrade, software upgrade, system administration, production test, and other activities that plan ahead a system downtime.

1.7. Proprietary Solution for High-availability

A survey by New Relic shows that for Java users, the most commonly deployed proprietary application servers are WebSphere and Oracle [10]. For proprietary database, Oracle Database was the top market share leader, followed by Microsoft SQL Server [11].

1.8. Free Open-source Software

"Free software" means software that respects users' freedom and community. Roughly, the users have the freedom to run, copy, distribute, study, change and improve the software. With these freedoms, the users (both individually and collectively) control the program and what it does for them" [12].

Open-source software comply with the criteria: free redistribution, inclusion of source code and compiled source code, allowing derivative works, maintaining integrity of the author's source code, and so on [13].

Commercial high-availability solutions have been around for long time. However, nowadays there are also free open-source high-availability solutions. Being free, it usually means they are free to acquire, but many times professional services and trained resources are required to implement and maintain such solutions.

According to JRebel technology report [14], Tomcat appeared to be the most commonly deployed free open-source application server, and WebLogic is the second most popular one.

The popular free open-source database are MySQL, PostgreSQL, Firebird, and many others. A survey by Jelastic showed that MySQL was still hold the highest market share among free open-source database servers [15].

2. Research Design and Experiment

The methodology used to achieve the intended objectives were by doing literature study, conceptual high-availability design, developing a simple test Java web application, design and implementation using proprietary and free open-source solution.

2.1. Setup

This research proposed the high-availability design that is relatively simple to setup based on the nature of Java-based web application. The experiment was setup and conducted in a single virtual machine instance, in which multiple instances of the web application server and database server were configured; and excluded the virtual machine failure scenario, as this would mean every configured web application server and database server instances within it would not be available. The virtual machine instance was installed Windows 2008 R2 operating system.

2.2. Java-based Online Shop

The web application was a simplified online shop, where user can browse the product catalogue and place their orders. Therefore, the functionality of the web made application was made simple, but yet still sufficient to represent the user experience in the aspect of unavailability.

2.3. Behavior in Non-High-Availability Deployment

In a deployment without high-availability design, the sample web application was deployed with Java Web /Application Server and Database Server. The sample web application was compiled and packaged into a WAR file. The database was setup with relevant tables and prepopulated data. The web application uses JDBC to connect to the database for the read and write operation.

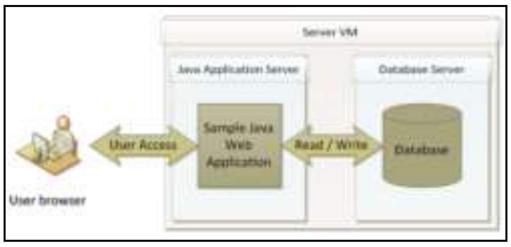

Figure 1. Single Server Deployment

In the event of failure at the Java application server, the user will not be able to access the web application at all, and thus will be getting error on the internet browser. In the event of failure at the database server, the user will still be able to access the web application partially. Certain parts of the web application that involve access to the database would encounter error.

2.4. Automated Web Application Test

Apache JMeter Version 2.7 r1342410 was used to simulate the user accessing the website. A JMeter test plan was created to simulate user performing the following steps using web browser. The test plan was configured such that it simulates if a user encountered an error, he stops. Therefore, the number of submitted orders in the database indicated how many users encountered error.

2.5. High-Availability Deployment

In order to achieve high-availability of the web application, the following conceptual design was proposed.

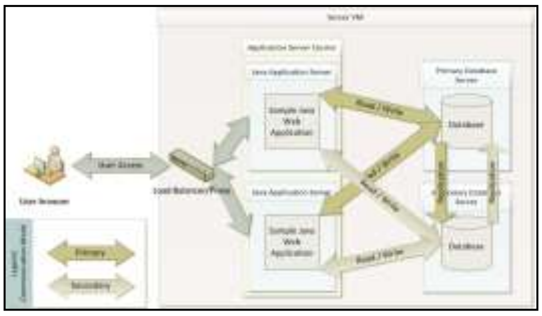

Figure 2. Proposed High-Availability Deployment

Two database servers were setup, with one as the master and the other as the slave. The master was the main one that was used for normal read-write operation, whereas the slave was the standby backup. Under the normal the web application in each instance application server instance accesses the primary database. In the event of failure of the secondary database server, it could be fixed without impacting the usage of the web application.

2.6. Application Server down Scenario

JMeter test plan was executed to simulate 10 users accessing the web application and each user repeated the action 5 times. In a non-error scenario, there were 50 orders submitted, each with 3 order lines. Once the JMeter test plan started, the application server log files or console was monitored. The one showing activity was shutdown 20 seconds later to simulate server crash. The WebLogic app server was shut down by issuing a force shutdown command from its admin console, whereas the Tomcat app server was shut down by executing the shutdown script. This scenario was repeated 5 times, the results were captured, and each time, the database records was reset to initial state and the servers were restarted.

2.7. Database Server down Scenario

The failover from primary database to secondary database was initiated manually using Oracle's command-line utility program. This simulated the unavailability of the primary server. The manual failover was done 15 seconds after the JMeter test plan started.

The preliminary test of the proprietary solution showed that the failover process required more than 1 minute to complete. Therefore, the JMeter was executed to simulate 10 users accessing the web application, and each user repeated the action 25 times. In a non-error scenario, there were 250 orders submitted, each with 3 order lines.

Based on the configuration free open-source solution, mysql_proxy was expected to perform the important role of routing the connection from the application to the primary database server, and route to secondary server if the primary is not available.

3. Results and Discussion

3.1. Session Replication and Failover

WebLogic Application Server.
The user always accesses the web application via the proxy at port 8080 of the Web Server. This proxy server routes the user requests to the appropriate server, which is either ManagedServer1 at port 7051 or ManagedServer2 at port 7052. In this research, it was configured to use "round-robin-affinity" algorithm. Using the example illustrated in Figure 8, if ManagedServer1 is down, User 1 will not be able to access it anymore, and the proxy will actually detect that ManagedServer1 is no longer available. Since there is a replica of User 1's session in ManagedServer2, User 1 can continue to use the web application without any disruption. And the User 1 session in ManagedServer2 becomes the primary session. This process is known as "session failover".

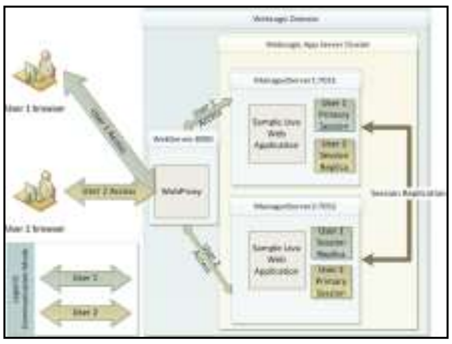

Figure 3. WebLogic Session Replication

Figure 4. WebLogic Failover Monitoring

The WebLogic Admin console can be accessed to monitor the status of the sessions in the cluster as shown in Figure .

Tomcat Application Server.

In this research, the mod_jk was configured to load-balance with sticky-session. The user always accesses the web application via the mod_jk proxy, which routes the user requests to either Tomcat1 or Tomcat2 server appropriately. Essentially, this worked in the same manner as the round-robin-affinity algorithm of the WebLogic cluster.

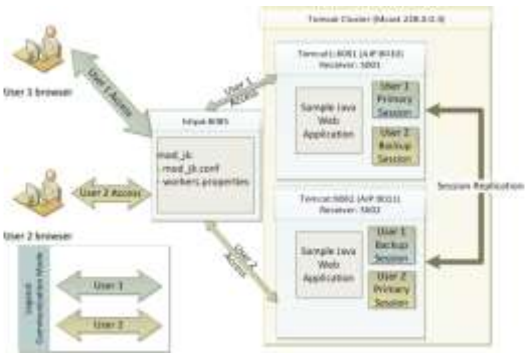

Figure 5. Tomcat Session Replication

Sessions Administration for /store

Sessions Administration for /store

Figure 6. Tomcat1 and Tomcat2 Manager Console

However, Tomcat did not have centralized Admin console. It only had Admin console on each instance of the server. Figure and Figure 6 shows the session monitoring page available in the console of each Tomcat instances.

The results showed that the application servers in both the proprietary and free open-source solutions did work properly providing high-availability in term of session failover. Setting up WebLogic was relatively easier as it could be done from the web-based Admin console. Although setting up the Tomcat cluster and Apache Web Server with mod_jk module was rather manual and technical, it could be done by following the guides provided in their websites as well as other related articles in the internet. Thus, one can consider using such free open-source solution as an alternative to proprietary solution at the application server layer.

3.2. Database Replication and Failover

Oracle Database Server

The Oracle database instances were configured for Data Guard. The first database instance called "Prod" was configured as the Data Guard primary Database. Then a Data Guard Physical Standby database, called "Stdby1" was created in a rather manual way. The configuration and control files for the secondary database were created using command line based on the primary database, and the database files were manually duplicated from primary to secondary. Then the secondary database configuration files were altered to become suit the appropriate configuration as secondary database. Lastly, Oracle listener was configured for the secondary database.

According to Oracle documentation, the optional Active Data Guard feature introduced in Oracle Database 11g enables the Physical Backup database to function in read-only mode. Prior to this version, the Physical Backup can only be in "mounted" state, in which query cannot be executed against it. In order to facilitate database failover process, Data Guard Broker was configured for the two instances using DGMGRL. The Data Guard Broker had to be activated in both instances by executing "alter system set dg_broker_start=true" while connected to each instance using SQLPLUS. Then the data guard configuration was creating a configuration using DGMGRL to register the Prod instance as the Primary database, and the Stdby1 instance as the Physical standby database. Manual failover was manually initiated by connecting DGMGRL to the Stdby1 instance and executing "failover to stdby1".

MySQL Database Server

Two MySQL instances were configured in Master-Slave replication scheme. Two database instances, called Mysql1 and Mysql2 were configured to listen at port 3307 and 3308 respectively. The Mysql1 instance was configured to perform binary logging of selected database. The Mysql2 instance was configured to become the slave for Mysql1 instance. The replication was also verified when the sample application schema created only in the Mysql1 instance also appeared in the Mysql2 instance.

MySQL did not actually have failover mechanism. But with Master-Slave configuration as this, if the Master server fails, the application can immediately switch to access the Slave database, and resume its operation. The results showed that the database servers in both proprietary and free open-source solutions did have the replication mechanisms that work properly.

Server-Agnostic Java Web Application

In this research, the sample web application was implemented without using any vendor specific feature in the program code. However, there were some difference as described below:

1. When connecting to Oracle database, the web application used the data source at the WebLogic server layer which had been configured to use Oracle JDBC driver. When connecting to MySQL database, the web application used the MySQL Connector/J driver packaged within the web application WAR file. Because the sample web application used the Spring framework, only the data source definition in the Spring configuration file needed to be changed when the database changed.
2. For WebLogic deployment, the web application WAR file included weblogic.xml file that contains WebLogic specific directive.

3.3. Application Server down Scenario Results

Table 1. Application Server down Scenario Results

	Proprietary Solution			**Free Open-source Solution**		
Iteration	**F**	**PF**	**T (s)**	**F**	**PF**	**T (s)**
1	6	0	93.38	3	0	59.46
2	5	1	66.47	2	0	61.41
3	0	0	86.07	8	0	59.58
4	0	0	81.41	8	0	55.73
5	1	0	83.29	0	0	63.21
Average	**2.4**	**0.2**	**82.12**	**14.6**	**0**	**59.88**

F = Failure, where order did not get submitted at all.

PF = Partial failure, where order was submitted, but there were missing order lines.

T = Time required to complete one test iteration.

The results showed that the proprietary solution had an average of 2.4 failures, compared to 14.6 failures of the free open-source solution. However, the time taken by the proprietary solution to complete the test was 1.37 times longer than the free open-source one.

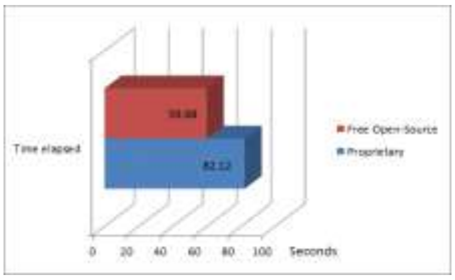

Figure 7. Average Time Elapsed in Seconds

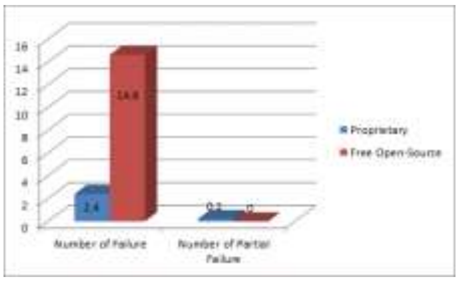

Figure 8. Average Number of Failures When One Application Server Was Down

Based on the result obtained, the availability measure can be estimated.

Table 2. Availability Estimation

	Proprietary Solution	Free Open-source Solution
Average Elapsed Time (s)	82.12	59.88
Average Time Per User (s)	1.6424	1.1976
Estimated Downtime (s)	3.94176	17.48496
Estimated Uptime (s)	78.17824	42.39504
Availability (%)	**95.20**	**70.80**

The results indicated that the free open-source solution performed better in term of speed; but did not perform as well as the proprietary solution in term of high-availability.

As observed in the experiment, in both solutions shutting down one of the application servers did not completely disrupt the users. Thus, one could purposely shutdown one server for maintenance or application upgrade without causing major disruption to user.

3.4. Database Server down Scenario Result

For the proprietary solution, the failover was performed 5 times and the result was shown below.

Table 2. Proprietary Database Server down Scenario Results

Iteration	**Failover Time (s)**	**Failures**
1	48.54	218
2	55.61	244
3	54.43	241
4	55.42	240
5	57.34	245
Average	54.27	237.6

The Oracle primary database on average required 54.27 seconds to complete. Compared to the application server down scenario, the downtime was much longer failure happened to the database server. As the result showed, the number of failures was very high. This was because, during failover, the web application encountered error, and no order could be submitted. Similar test scenario was not performed for the free-open source solution. Because if the primary server was shutdown, no further order can be submitted until the application was changed to point to the secondary database manually. However, it was also important to note that the secondary MySQL server was always up.

In terms high-availability feature by database replication, both the Oracle and MySQL databases did work as expected. The MySQL Master-Server replication scheme showed better availability as both instances were always up (hot standby), MySQL database definitely can be an alternative to proprietary solution such as Oracle Database. With some application logic or load-balancer or proxy that could control which database instances the application should access, one could potentially orchestrate the start-up and shutdown of the MySQL instances to allow maintenance or schema upgrade with minimal downtime.

This research observed that it was not possible to orchestrate such database start-up and shutdown of the Oracle database instances to allow maintenance or schema upgrade with minimal downtime; because at the same time, only one database instances can be open for read and write operation.

3.5. Cost Comparison

The free open-source solution proposed in this research was completely free to use, including for commercial purposes. Optionally, some of these free open-source solutions had third party consultants if the in-house personnel require extra support. The proprietary solution proposed in this research had price tags in many places. The Oracle products such as WebLogic Application Server and Oracle Database were licensed either per user or per processor [16].

Both proprietary and free open-source solution require hardware and operating system. In terms of monetary cost, it is obvious that the proprietary solution cost much more than the free open-source solution.

4. Conclusion and Future Works

4.1. Conclusion

This research attempted to compare proprietary and free-open source high-availability solution for Java web application using Tomcat and MySQL from capability point of view, and drew the following conclusion:

1. For application server, Tomcat server and Apache Web Server with mod_jk – as part of free open-source solution, can provide high-availability by session replication and failover in cluster like WebLogic – as part of proprietary solution. But it may not perform as well as WebLogic in this aspect.
2. In terms of complexity, both types of solution are equally complex. Some free open-source solutions are also supported professionally and commercially.
3. For the database server, this research concluded that both proprietary and free open-source databases such Oracle and MySQL respectively can provide high availability by data replication.
4. In general, when using free open-source software such as Tomcat and MySQL regardless of the purpose, there are risk involved as they usually do not have guaranteed support. This risk increases as the complexity increases in the high-availability solution.
5. It is very apparent that using Tomcat and MySQL as free open-source high availability solution cost less than proprietary ones in terms of software cost.
6. The availability of the free open-source high availability solution for Java based solution using Tomcat and MySQL has made it much more affordable. However, one should be aware of the risks when using free open-source solution and decide if it is acceptable.

4.2. Future Works

Due to the limited time and resource, this research focused its attention to the application server and database server components of a web application. Some other aspects of the high-availability solutions for Java web application could be further researched. The following are some recommendations for future research:

1. Redundancy for other components of the system, such as the load-balancer and proxy should be studied as well.

2. In addition to standard HTTP traffic, future research should consider high-availability solution for Java web application involving HTTPS traffic.
3. Use separate servers for the system components as well as the test client. This research contained all the software in one virtual machine. Installing the components in separate servers brings the experiment closer to actual production environment, even if the servers are virtual machines.

5. References

[1] "The birth of the Web | CERN." https://home.cern/science/computing/birth-web (accessed Oct. 01, 2020).

[2] D. Helic, "Server-side Technologies CGI, PHP, Java Servlets, JSP," 2004. http://coronet.iicm.edu/lectures/mmis/material/slides_serverside_main.pdf (accessed Nov. 30, 2015).

[3] "Strategic Guide on Web Designing | Learn How to Design Your Own Website." http://wspgweb.com/web-application-development.php (accessed Oct. 03, 2020).

[4] B. Goetz, "Java theory and practice: Are all stateful Web applications broken?," 2008. https://www.ibm.com/developerworks/library/j-jtp09238/index.html (accessed Oct. 05, 2020).

[5] E. Armstrong *et al.*, "Distributed Multitiered Applications," 2005. https://docs.oracle.com/javaee/1.4/tutorial/doc/Overview2.html (accessed Oct. 05, 2020).

[6] Joe Hertvik, "Service Availability: Calculations and Metrics, Five 9s, and Best Practices – BMC Blogs," 2020. https://www.bmc.com/blogs/service-availability-calculation-metrics/ (accessed Oct. 05, 2020).

[7] W. J. Bender and A. Joshi, "High Availability Technical Primer Availability , High Availability , and Fault Tolerance : What do these terms mean ?," pp. 1–13, 2004.

[8] C. Kopparapu, *Load Balancing Servers , Firewalls , and Caches Chandra Kopparapu*. 2002.

[9] P. Soila and N. Priya, "Causes of Failure in Web Applications (CMU-PDL-05-109)," *Parallel Data Lab.*, 2005, [Online]. Available: http://repository.cmu.edu/pdl/48.

[10] Newrelicblog, "The Death of WebSphere and WebLogic App Servers? New Infographic shows the Rise of OSS Java - New Relic Blog," 2012. https://blog.newrelic.com/product-news/infographic-oss-java-wins-in-the-cloud-era/ (accessed Sep. 15, 2020).

[11] SolidIT, "DB-Engines Ranking - popularity ranking of database management systems," 2020. https://db-engines.com/en/ranking (accessed Nov. 01, 2020).

[12] I. Free Software Foundation, "What is free software? - GNU Project - Free Software Foundation," 2019. http://www.gnu.org/philosophy/free-sw.html.

[13] Opensource.org, "The Open Source Definition (Annotated) | Open Source Initiative Version 1.9," 2007. https://opensource.org/osd-annotated (accessed Oct. 20, 2020).

[14] JRebel, "2020 Java Technology Report | Rebel," 2020. https://www.jrebel.com/blog/2020-java-technology-report (accessed Sep. 15, 2020).

[15] M. Sprava, "Open source database market share within Jelastic: February 2012 | Jelastic," 2012. https://jelastic.com/blog/open-source-database-market-share-within-jelastic-february-2012/ (accessed Oct. 20, 2020).

[16] Oracle, "Oracle Technology Global Price List Software Investment Guide," pp. 1–13, 2012, [Online]. Available: http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf.