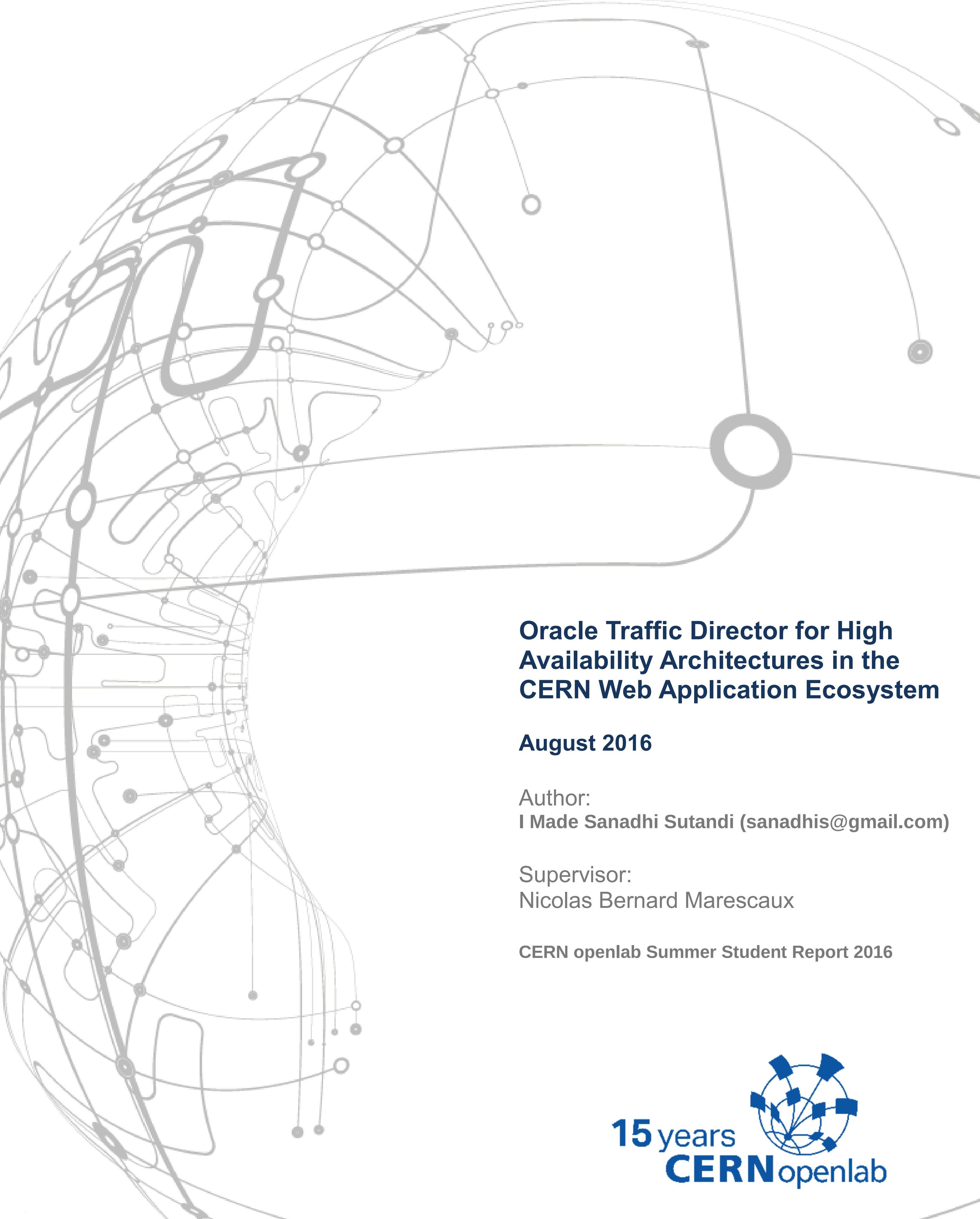


Oracle Traffic Director for High Availability Architectures in the CERN Web Application Ecosystem

August 2016

Author:
I Made Sanadhi Sutandi (sanadhis@gmail.com)

Supervisor:
Nicolas Bernard Marescaux

CERN openlab Summer Student Report 2016

Project Specification

Oracle Traffic Director (OTD) is an application layer (layer 7) load-balancer that works as the application delivery control to optimize communication between client and application servers. Starting from version 12.2.1(12c), OTD was released to all platforms and therefore it opens the possibility to implement the OTD without platform limitation and constraint.

The aim of this project is to evaluate the OTD as the front-tier application delivery control for the CERN's High Availability Web Application Ecosystem, elaborating the OTD's features and behaviours and its integration with WebLogic server.

Abstract

In the CERN's Web Application Ecosystem, front-tier or first tier applications, such as Apache and OHS, are used to distribute the traffic to second-tier application, e.g. Tomcat and WebLogic. CERN's IT-DB has planned to unify the front-tier application and to implement only one product as the front-tier application.

Oracle Traffic Director (OTD) is the middleware product from Oracle that acts as layer-7 software load balancer or application entry point. Since 12c version, OTD has been released to all platforms without the restriction of Oracle hardware or software. Therefore, it is interesting to evaluate the OTD and its features as there is a possibility to use OTD as the front-tier application for the High Availability Architectures in the CERN Web Application Ecosystem.

The major objective of this project is to evaluate the OTD's features and performances. The evaluation is conducted by reviewing from installation and configuration procedures until the deployment of the OTD's instances itself. The installation and configuration options are also reviewed to see if OTD can work well in CERN's Infrastructure and also to see the possibility of generating and automating these procedures. The OTD-WebLogic features integration is evaluated especially for the Dynamic Cluster and Domain Partitioning. In addition in this project, Coherence*Web is also implemented to see the benefits of coherence cluster's deployment tier regarding to Session replication mechanism.

Table of Contents

1 Introduction

1.1 Oracle Traffic Director

Oracle Traffic Director (OTD) is a layer 7 load-balancer and application delivery control which distributes the communication between clients and back-end servers. During the initial version, this product was only available for the Oracle Exalogic Cluster. Since 12c release, Oracle started to release Oracle Traffic Director for all platforms. Based on documentation, Oracle states that it is a High throughput, low latency HTTP(s), Web Socket, and TCP [1] load balancer which is fast, reliable, and scalable [2]. Figure 1.1. shows the example network topology with OTD.

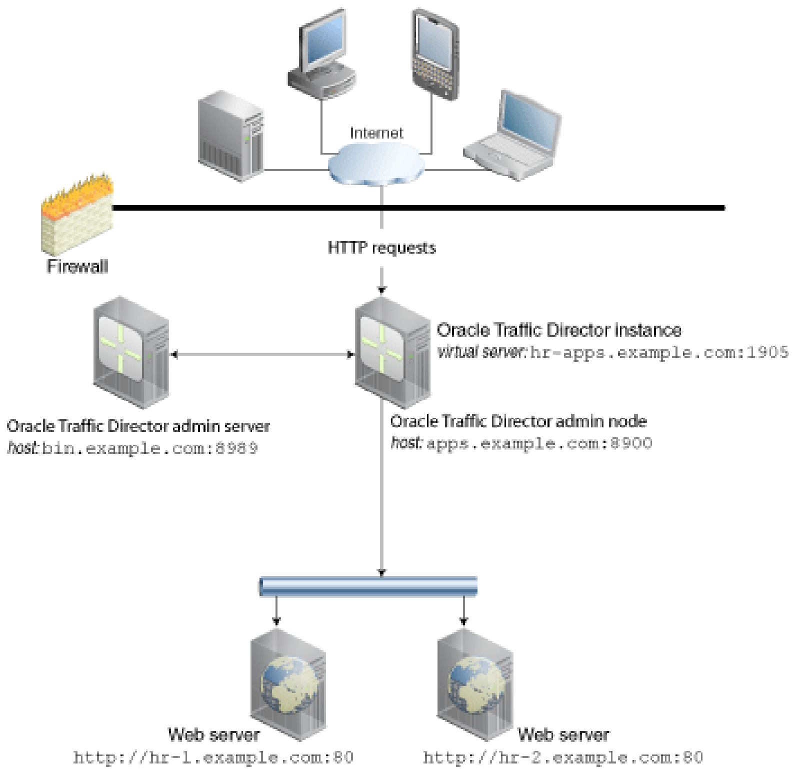

Internet
Firewall
HTTP requests
Oracle Traffic Director instance
virtual server: hr-apps.example.com:1905
Oracle Traffic Director admin server
host: bin.example.com:8989
Oracle Traffic Director admin node
host: apps.example.com:8900
Web server
http://hr-1.example.com:80
Web server
http://hr-2.example.com:80

Figure 1.1. Example Topology for OTD Deployment [3]

There are two types of OTD, which are Standalone OTD and Collocated OTD. **Standalone OTD** is the OTD managed independently from the WebLogic Server and **Collocated OTD** is the OTD managed through WebLogic Server.

1.2 Oracle WebLogic Server

Oracle WebLogic Server is a Java EE application server from Oracle. It is said to be scalable application server which support the deployment of many types of applications, e.g. distributed applications [4]. WebLogic Server supports the deployment of applications in a robust, secure and scalable environment. It provides web interface (through em/console), WLST, and RESTful services for managing the configurations and applications.

In this project, WebLogic Server is used to administer the Collocated type of OTD through the administration console that is come up as the part of Fusion Middleware Infrastructure.

1.3 CERN's Web Application Ecosystem

Presently, CERN uses multitier architecture for the Web Application Ecosystem. Overall, there are three tiers of application that form the CERN's Web Services network; they are front-tier, application tier, and database tier. The front-tier uses Apache Server and Oracle HTTP Server (OHS), and Application tier has Tomcat and WebLogic Server, meanwhile database tier implements Oracle database, PostgreSQL, MySQL, etc.

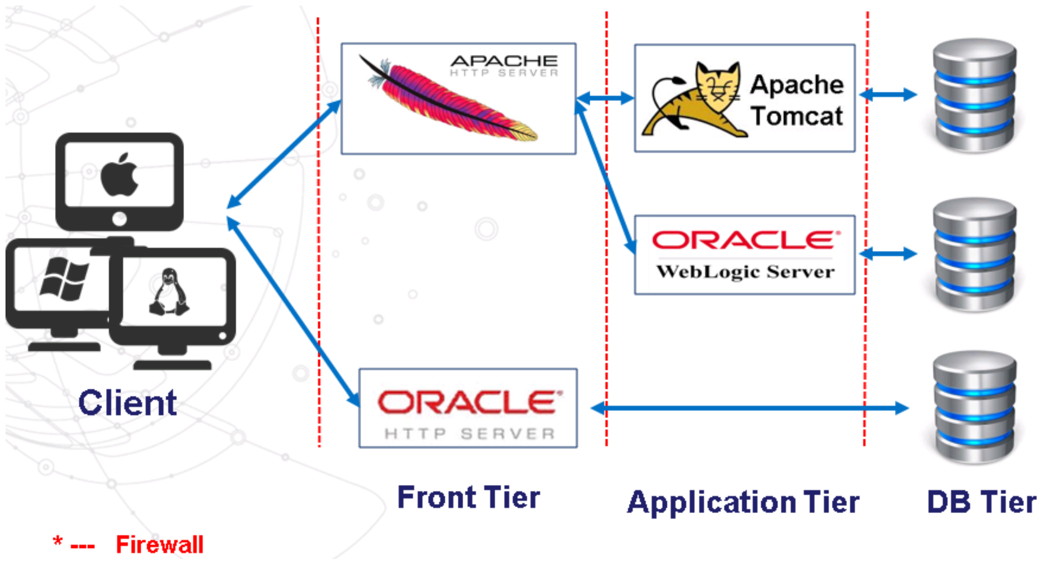


Figure 1.2. CERN Web Application's Multitier Architecture

In the beginning, OHS is being used in CERN's Web Application for the usage of mod_plsql to run procedural SQL in Oracle database. Since in OHS version 12.1.3, the plsql module becomes deprecated and then CERN's IT-DB group has decided to use ORDS with WebLogic Server as the solution of using mod_plsql. Therefore, OHS is not needed anymore and in the short future OHS will be removed from CERN Web Application to unify the configuration (only Apache) in the front-tier for easier management and troubleshoot. But, there is also a possibility for CERN to change the Apache into another front tier application which offers high availability and additional features to the CERN Web Application Ecosystem.

1.4 RPM

RPM (RPM Package Manager) is a command line software management system with the capability to install, uninstall, verify, query, and update the software package in Linux machine. RPM is involved in this project because CERN's IT-DB-IMS section uses RPM to install every application to their infrastructure.

1.5 Project Goals

The goals of this project include:

- Verifying that OTD can be installed in IT-DB's infrastructure and platform.
- Reviewing the installation and configuration process for the automation process.
- Automating the installation and configuration of the OTD.
- Evaluating the features of OTD and its integration with WebLogic features, especially for the dynamic capability of OTD-WebLogic integration.

2 Project Setup

Figure 2.1. describes the summary and comparison of deploying the Standalone OTD and the Collocated OTD.

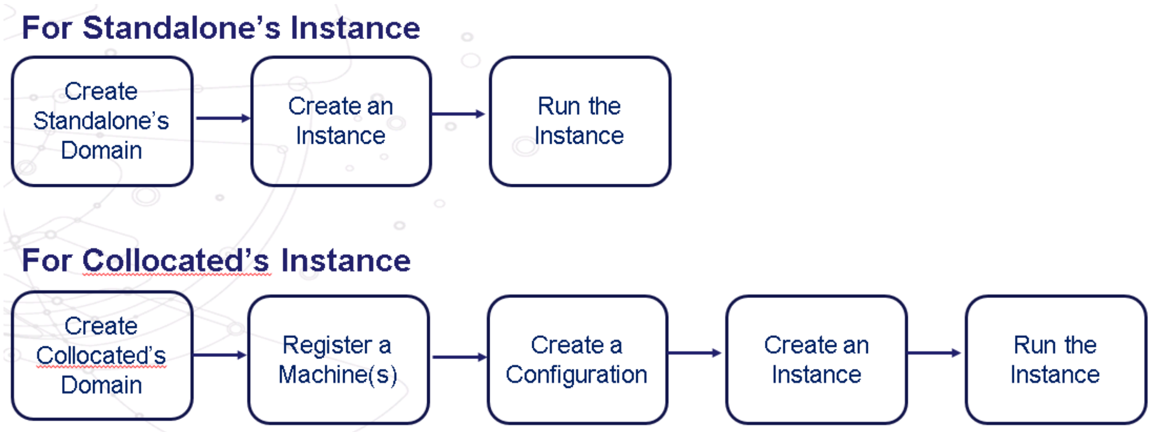


Figure 2.1. Summary and Comparison of Standalone OTD and Collocated OTD

2.1 Scope

In this project, the installation procedures and configurations are conducted in single Machine (local machine). The machine's specifications are:

Operating System	:	CentOS Linux Release 7.2.1511 (Core)
CPU	:	Intel® Core™2 Duo CPU E8400 @3.00 GHz
Memory	:	8 GB DDR2
Disk's Capacity	:	250GB

The installation of Collocated OTD requires Java SE Development Kit 8, Oracle Fusion Middleware Infrastructure (FMW), and WebLogic Server (comes as a part of the Fusion Middleware). Therefore, in this project the version of software that was being used are as follow:

Java JDK 8	:	1.8.0_92
Fusion Middleware	:	12.2.1.1.0
WebLogic Server	:	12.2.1.1.0 (12c)
OTD's Version	:	12.2.1

The project doing has been validated using IT-DB's VM Infrastructure with the specification as follow:

VM	: OpenStack VM
Operating System	: RHEL 6.7

In addition, there are also several conditions in this project:

- The configuration and results about dynamic cluster, domain partition, and coherence cluster are obtained in the development domain of Collocated OTD.
- Failover modes and failover group cannot be implemented in CERN IT-DB's infrastructure because CERN does not implement Floating IP feature of OpenStack. This prevents the usage of Virtual IP which is needed for Failover Modes of OTD.

2.2 Terminologies

2.2.1 Standalone & Collocated Terminologies

- **WLST**: WLST is a WebLogic Scripting Tools, a scripting language that supports the use of Jython and it can be used to administer the WebLogic server as the replacement of Administration Console. Moreover, Jython is a Java Implementation of Python [5].
- **Administration Console**: Administration console is the web interface that provided by Fusion Middleware Infrastructure's installation bundle. Administration console function as the web application to administer the WebLogic setup and configuration.

2.2.2 Standalone Terminologies

- **Server name**: the server hostname where OTD is being deployed.
- **Listener-port**: the corresponding listen port number of the OTD's running instance
- **Origin-servers**: the back-end servers (the servers which run the applications and web services)

2.2.3 Collocated Terminologies

- **Instance**: An OTD's instance is the OTD configuration that runs on the specific machine
- **Virtual Server**: Is the running instance of OTD that serves as the entry point for the client request.

- **Listener**: listener is an associate target with an address and a port. Clients/users can access the origin servers by accessing the listener address and port.
- **Origin Server:** Origin server is the server(s) that run services in back-end and accept request from OTD's instance and return the corresponding responses to it.

2.3 Installation

As written in Section 1.1., there are two types of OTD, which are Standalone OTD and Collocated OTD. Both of the OTD's modes have some perquisites for the installation. These perquisites are required for the installation of OTD in this project:

- ksh
- compat-libcap1-1.10
- libaio-devel.x86_64
- compat-libstdc++-33.x86_64

OTD comes up with two installation modes:

- Graphical installation (through the Graphical Wizard)
- Silent installation (via Command Line)

It is relatively easy to install OTD either by graphical or silent mode. For the graphical mode, there are on-screen instruction to follow. Meanwhile for the silent installation, it requires several parameters to be passed on while executing the installation files and the parameters are different for the standalone and collocated type of OTD.

The parameters needed to install OTD are not clearly specified in the Oracle documentation it is discovered that the parameters needed are:

For Standalone OTD:

- ORACLE_HOME
- DECLINE_SECURITY_UPDATES
- INSTALL_TYPE
- MYORACLESUPPORT_USERNAME
- SECURITY_UPDATES_VIA_MYORACLESUPPORT

For Collocated OTD:

- ORACLE_HOME
- DECLINE_SECURITY_UPDATES
- INSTALL_TYPE

2.3.1 Prerequisites for Collocated OTD Installation

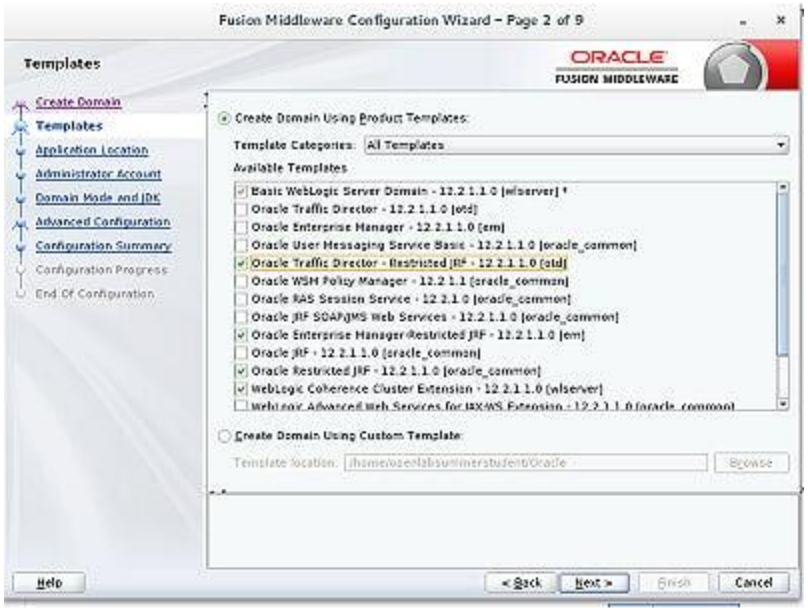
In addition, installation of the Collocated OTD requires:

- Java SE Development Kit 8
- Oracle Fusion Middleware Infrastructure (includes the WebLogic Server)

2.4 Domain Creation

Every configuration and instance runs on specific WebLogic domain and it is a logical grouping of JVM. For both of Standalone and Collocated OTD, a domain can be created either by:

- Configuration wizard
- or WLST


Fusion Middleware Configuration Wizard – Page 2 of 9

Templates

Create Domain
Templates
Application Location
Administrator Account
Domain Mode and JDK
Advanced Configuration
Configuration Summary
Configuration Progress
End Of Configuration

(●) Create Domain Using Product Templates:

Template Categories: All Templates

Available Templates

- [✓] Basic WebLogic Server Domain - 12.2.1.1.0 [wlserver] *
- [ ] Oracle Traffic Director - 12.2.1.1.0 [otd]
- [ ] Oracle Enterprise Manager - 12.2.1.1.0 [em]
- [ ] Oracle User Messaging Service Basic - 12.2.1.1.0 [oracle_common]
- [✓] Oracle Traffic Director - Restricted JRF - 12.2.1.1.0 [otd]
- [ ] Oracle WSM Policy Manager - 12.2.1.1 [oracle_common]
- [ ] Oracle RAS Session Service - 12.2.1.0 [oracle_common]
- [ ] Oracle JRF SOAP/JMS Web Services - 12.2.1.1.0 [oracle_common]
- [✓] Oracle Enterprise Manager-Restricted JRF - 12.2.1.1.0 [em]
- [ ] Oracle JRF - 12.2.1.1.0 [oracle_common]
- [✓] Oracle Restricted JRF - 12.2.1.1.0 [oracle_common]
- [✓] WebLogic Coherence Cluster Extension - 12.2.1.1.0 [wlserver]
- [ ] WebLogic Advanced Web Services for JAX-WS Extension - 12.2.1.1.0 [oracle_common]

( ) Create Domain Using Custom Template:

Template location: /home/openlabsummerstudent/Oracle

Figure 2.2. Process of Creating Domain using Configuration Wizard

In creating domain either for WebLogic Domain or OTD Domain, a template must be specified. Below is the specification of templates that are used to create domains:

- For Standalone Domain : Oracle Traffic Director Standalone
- For Collocated Domain : Oracle Traffic Director – Restricted JRF

For both WebLogic Domain and OTD Domain (Collocated), there are two modes of domain, Development Mode and Production Mode. Both domain have the same features and configuration, but the security configuration is much more stringent in Production Mode. Every retry to make a configuration shall consider the “Lock & Edit”, to lock session and prevent other accounts from making changes. Also after making changes in production domain, the changes are not being activated immediately. Instead, user has to perform manual “active changes” command, e.g. in Administration Console by clicking active changes button or in WLST by inputting "active" command.

2.5 Configuration in the Domain

Specific configuration such as Machine, Configuration, Virtual Server, etc. are NOT EXIST in the standalone domain. Custom configurations are only available for the Collocated OTD where the Oracle Fusion Middleware is also available. There are two ways to conduct configuration in Collocated OTD:

- **Using Administration Console**
- **Using WLST**

2.6 Instance Creation

An instance is an Oracle Traffic Director configuration that runs as a server in an administration node or the administration server. An instance acts as the virtual server and can listen to request from client.

2.6.1 Standalone Instance

In the Standalone OTD, there is no possibility to reconfiguration or re-edit the instance. This happens because WebLogic is needed to administer the configuration and instance on the domain and it is not available in the Standalone OTD. A Standalone OTD instance can only be created by WLST and user must recreate the instance to edit the deployment. Standalone OTD's instance does not start at the time it has been created and it has to be started manually by invoking the startserv file in the directory given:

/DOMAIN_HOME/config/fmwconfig/components/OTD/instances/$INSTANCE_NAME/bin/star
tserv

2.6.2 Collocated Instance

In contrast with the Standalone Instance, Collocated instance will be up automatically as soon as WLST creates it. The Collocated instance, which has been created through Fusion Middleware Control, does not automatically start neither. But, it can be modified through the Fusion Middleware Control or WLST. Furthermore, it has more features integration such as cluster (normal cluster/dynamic), URI routing, support with TCP proxies, logs and security management, etc.

3 Automation of Project Setup

3.1 Automation for Installation and Configuration

The idea of automation process is to use python scripts to invoke shell command (terminal) and WLST. Also, it will be more applicable and easier to automate the configuration with the help of WLST.

1) For automating the installation process, python scripts write shell command and execute it to install OTD using the silent mode of installation. Python program reads saved configuration from txt file and passes the parameters that is needed for the silent installation and execute the installer file.

   The created python scripts for the installation are:

   1) For installation process: otd_installationSetUp.py
   2) Module for code reduction and modularity: otdModule.py
   3) Configuration file : otdConfiguration.txt

2) For automating the process of creating configuration and instance, python script writes the WLST command needed for the configuration to the .py file and then execute the code with WLST interpreter. Python program reads saved configuration from txt file and pass the parameters needed for creating configuration in WLST commands.

   The created python scripts for the configuration process are:

   1) Creating domain : otd_createDomain.py
   2) Creating machine : otd_createMachine.py
   3) Creating configuration : otd_createConfiguration.py
   4) Creating instance : otd_createInstance.py
   5) File to store the WLST command (later invoked by WLST) : otd_wlstScript.py
   6) Module for code reduction : otdModule.py
   7) Configuration file : otdConfiguration.txt

Each code can be executed independently with the *python <filename>* command.

To perform automation process from installation until instance deployment, execute:

```
$ python otd_Automate.py
```

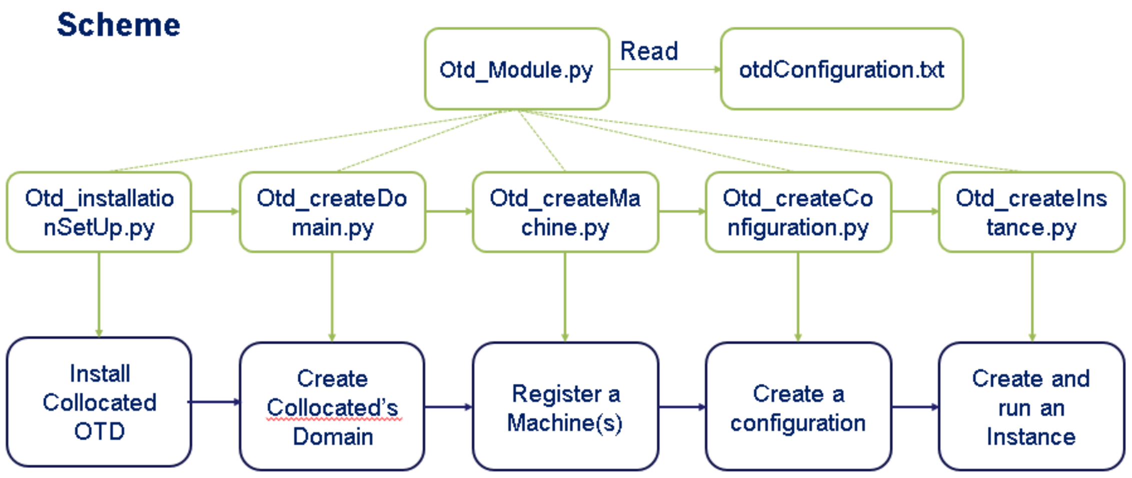


Figure 3.1. The Flow of OTD's Automation using Python Script

3.2 RPM Integration

RPM for the Standalone OTD installation-only (without configuration) is created using rpmbuild command. The idea of .spec file is to install Standalone OTD in the BUILDROOT directory and then when executed through rpm –i command, it will copy all the files that have been installed to /ORACLE_HOME directory.

4 Evaluation of Oracle Traffic Director-WebLogic Integration

The Oracle Traffic director is evaluated in context of the dynamic capability, to find out its unique features with WebLogic and investigate whether or not it suits in the High Availability Architectures context. This chapter depicts the components for integration and also the corresponding result.

In addition, the implementation and evaluation of coherence cluster is also described in this chapter.

4.1 Integration of WebLogic's Dynamic Cluster and Domain Partition with OTD

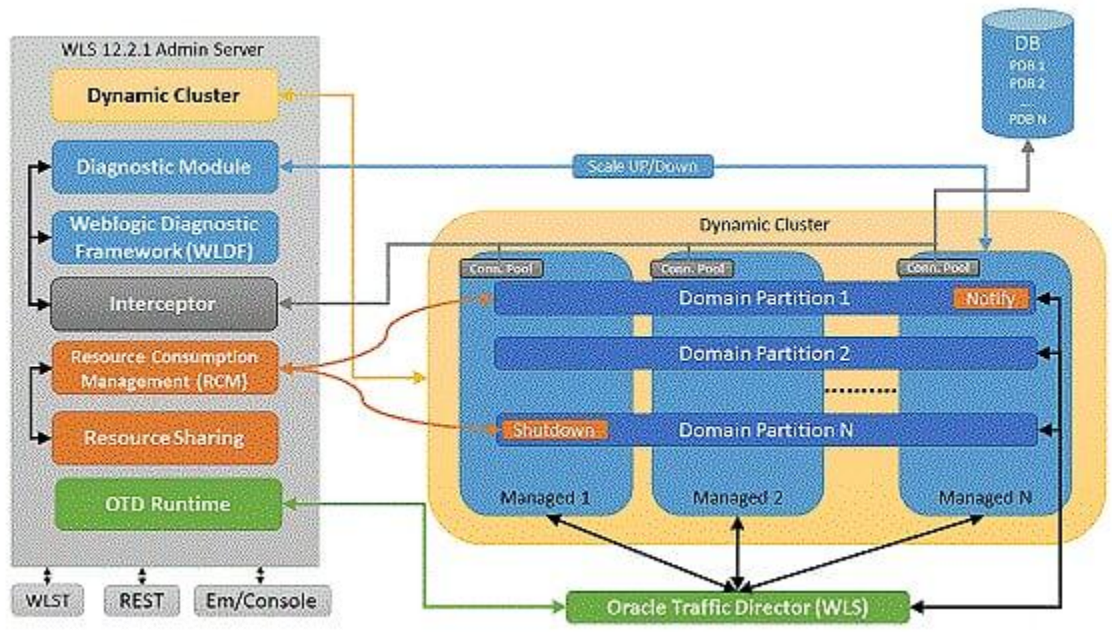


Figure 4.1. The System Architecture for OTD-WebLogic's Features Integration[6]

The components needed to achieve this integration are listed below:

1. WebLogic Domain and OTD

   It is a better approach to separate the Application Domain (for managing Dynamic Cluster, Domain Partition, and Application Deployment) and OTD domain (for managing load balancing/entry point). Therefore two domains with associate template are created:

   - WebLogic Domain (Oracle Enterprise Manager-Restricted JRF)
   - OTD Domain (Oracle Traffic Director - Restricted JRF)

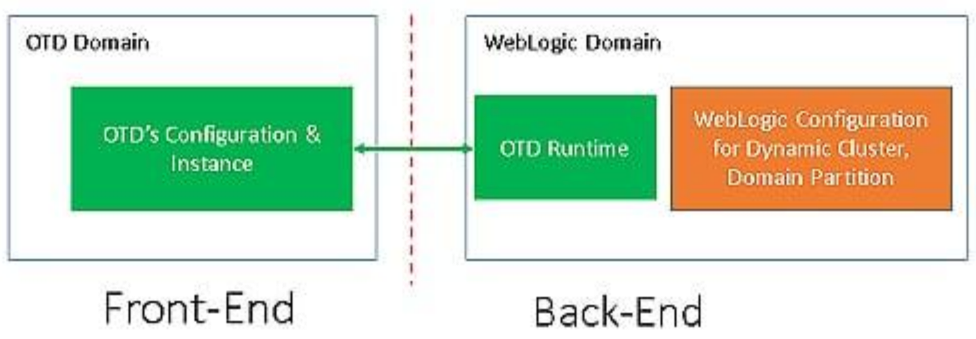

OTD Domain
OTD's Configuration & Instance
WebLogic Domain
OTD Runtime
WebLogic Configuration for Dynamic Cluster, Domain Partition
Front-End
Back-End

Figure 4.2. Separation of OTD Domain and WebLogic Domain

2. Dynamic Cluster

   Dynamic cluster is a cluster which can easily be expanded and reduced by scale Up/Down action. Dynamic Cluster has a sequential naming by default, for example Dyn-server-1, Dyn-server-2. This easier management command contrast dynamic cluster from common cluster although both of them can be enlarged in size. All servers in the cluster have the same behaviours, same application, and same services. It is impossible to modify only specific server behaviour. If there is a change, the change must be implemented to the cluster instead.

3. Virtual Target for the Dynamic Cluster

   In this context, virtual target is an entity that represents the dynamic cluster and it defines an URI as access point to the Resource Group of Dynamic Cluster, both at the domain level and in a domain partition.

4. OTD Configuration

   The OTD configuration's runtime is registered in the WebLogic Domain and it will be configured automatically based on the corresponding WebLogic domain settings. This will create dynamicity since there is no need to configure or set OTD manually for listing the origin-servers.

5. LifeCycle Manager and OTD Runtime

   OTD configuration is registered in WebLogic Domain as an OTD Runtime to receive the settings and configurations from the WebLogic Domain. LifeCycle Manager is the part of the LifeCycle Management Tools (LCM) and it is needed for access management to OTD configuration in separated domain of WebLogic.

6. Domain Partition

   Domain partition is the way of separating a single domain into a number of independent domains that have their own resources (Heap Memory, CPU) and configuration (Resource Group, Virtual Target, etc). Domain partition has its own Virtual Target that is transferred to the OTD. With this domain partition-OTD integration, OTD is able to recognize the change in dynamic cluster.

7. Application Deployment

   Application in this integration is the cluster application which has the capability of using WebLogic Session to replicate the session data to every node server in the cluster.

8. OTD-WebLogic Integration's Test

   After the application is ready, every node server available in the dynamic cluster's node member is accessed through:

   http://<Server_Address>:<Listen_Port>/<Virtual_Target_Prefix/<Application>

   For example, the cluster member's node is on local machine with port address 7101, Virtual Target URI is app1, and application that has been deployed is clusterJSP:

   http://localhost:7101/app1/clusterjsp

   Without configuring OTD in advance, it is able to perform the load-balancing and automatically recognize the dynamic cluster's node server and serve request by accessing:

http://<OTD_Address>:<OTD_Port>/<Virtual_Target_Prefix/<Application>

4.2 Dynamic Capability of OTD

Dynamicity of OTD is checked by doing the Scale up/down action:

- Scale down action or force shut down action is performed to dynamic cluster. Then the result shows that OTD is able to recognize the down in back-end server immediately and stop distributing request to the corresponding node.
- Scale up action is performed to dynamic cluster. The result is as soon as the new server state changes to up, OTD is able to recognize the new server in dynamic cluster and start distributing the request to the corresponding node.

The dynamicity of this OTD-WebLogic integration can be described in figure 4.3-4.5:

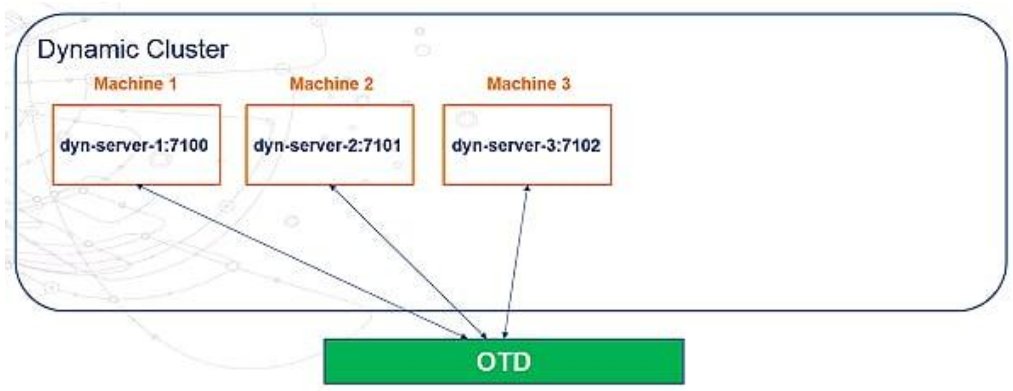


Figure 4.3. Initial State

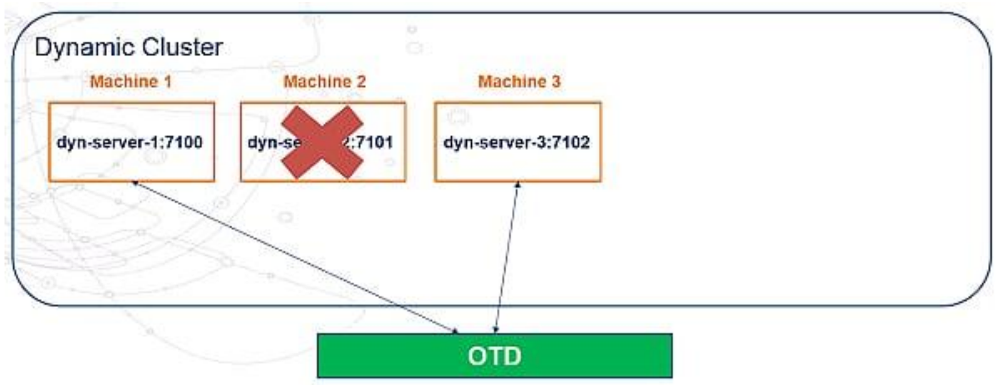


Figure 4.4. OTD Recognizes Down Server

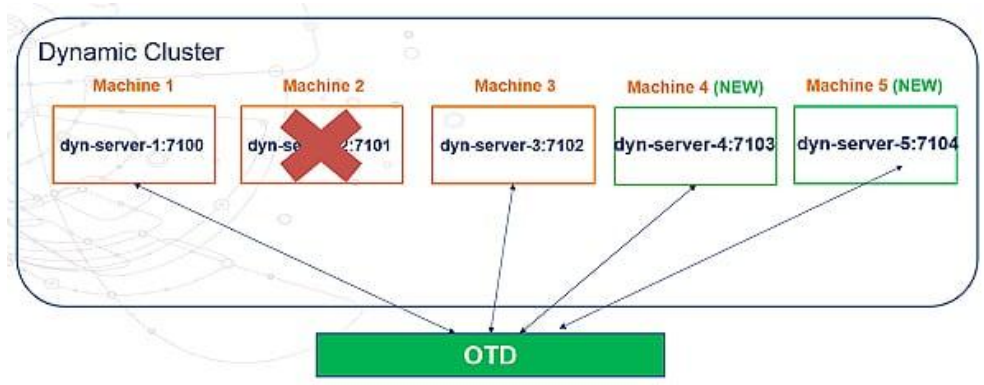


Figure 4.5. OTD Detects New Server Immediately

4.3 Diagnostic Module

Another useful integration of OTD-WebLogic is the diagnostic module. Diagnostic module allows the creation of a module for automatic scale up /down action based on given parameters such as Throughput. The figure 4.6. shows the result of monitoring diagnostic module behaviour using administration console.

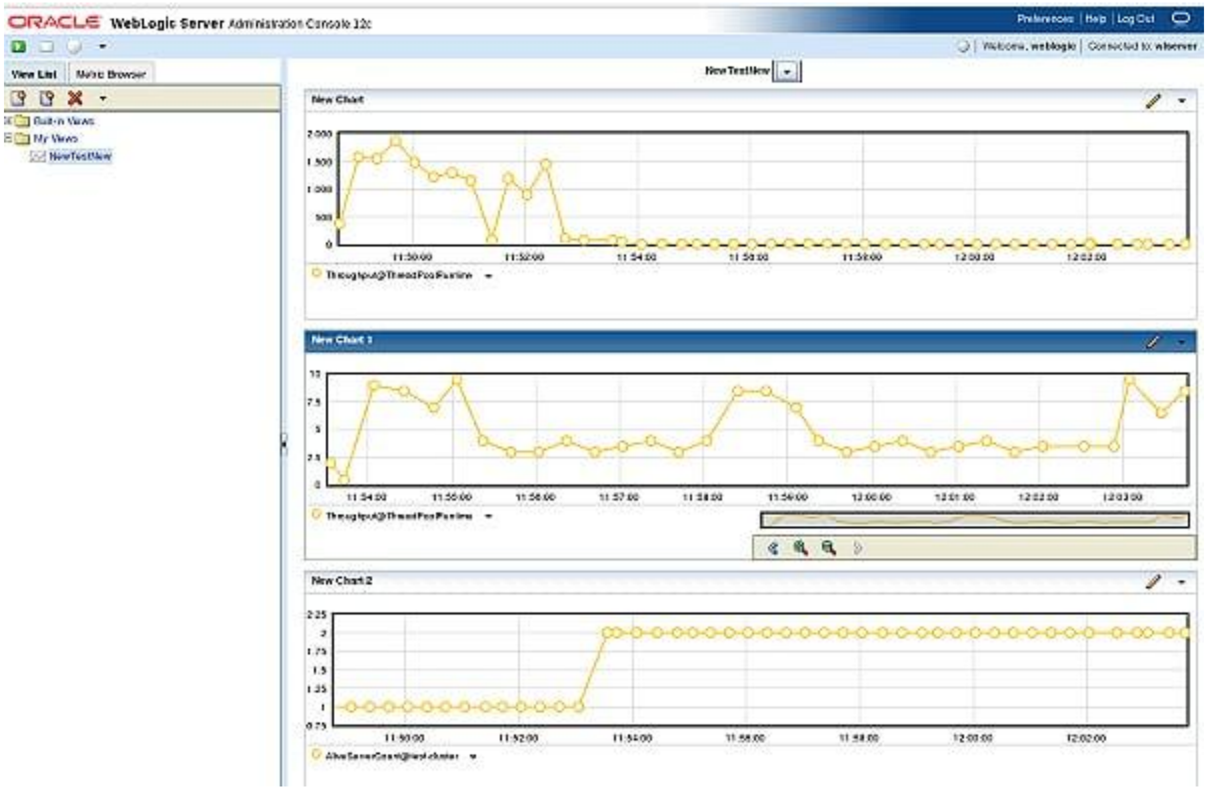

ORACLE WebLogic Server Administration Console 12c
Preferences | Help | Log Out
Welcome, weblogic | Connected to: wlserver
View List
Metric Browser
Built-in Views
My Views
NewTestView
New Chart
Throughput@ThreadPoolRuntime
New Chart 1
New Chart 2
AliveServerCount@testcluster

Figure 4.6. Diagnostic Module Automatically Starts the New Server

As seen in the figure above, when there was only 1 node in the dynamic cluster and it faced by high number of request, the diagnostic module started the second node as it was continuing to retrieve the client request.

4.4 Coherence Cluster

Coherence cluster is one of the included software in Oracle Fusion Middleware Infrastructure and it can be administered using WebLogic Server. Coherence clusters is a cluster that consist of multiple server instances that distribute and replicate cache, session, or even application data through the use of grid system. Coherence cluster can be used to increase the application scalability, availability, and performance. In the context of the CERN's Web Application Ecosystem, coherence cluster can be used to replicate the HTTP session data to prevent the data loss if the web application servers are down. One coherence cluster has 3 kinds of cluster tier inside them, they are data tier cluster, application tier cluster, and proxy tier cluster as shown in figure 4.7.

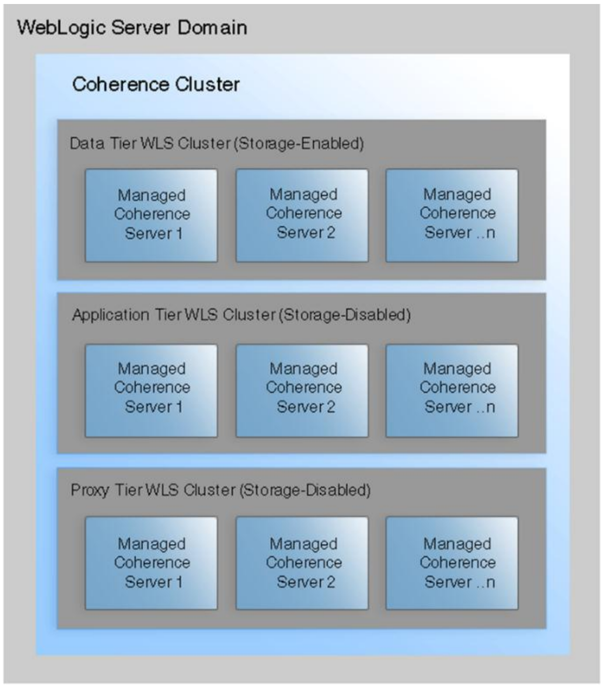

WebLogic Server Domain
Coherence Cluster
Data Tier WLS Cluster (Storage-Enabled)
Managed Coherence Server 1
Managed Coherence Server 2
Managed Coherence Server ..n
Application Tier WLS Cluster (Storage-Disabled)
Managed Coherence Server 1
Managed Coherence Server 2
Managed Coherence Server ..n
Proxy Tier WLS Cluster (Storage-Disabled)
Managed Coherence Server 1
Managed Coherence Server 2
Managed Coherence Server ..n

Figure 4.7. Coherence Cluster Overview [7]

The components, which are needed for the coherence cluster in the context for deploying a replicating HTTP session application, are:

1. Coherence Cluster

   Basically, Coherence cluster is a collection of JVM processes with distributed grid system. Every machine that runs coherence's JVM process is automatically joining cluster and later it is called as cluster member or coherence managed server [8]. Coherence cluster uses Tangosol Cluster Management Protocol (TCMP) to communicate between cluster member for multicast and unicast mechanism.
2. Data Tier Cluster

   Data Tier Cluster is a cluster that has a number of storage-enabled coherence managed server to store and distribute the data (application, cache, session) among the cluster members.
3. Application Tier Cluster

   In contrast to Data Tier Cluster, all of application tier cluster members are storage-disable managed server and they are providing the application services (application host).
4. Proxy Tier Cluster

Proxy Tier Cluster's members have functionality to allow Coherence*Extend clients to use Coherence caches without being cluster members [7]. The Proxy Tier Cluster has not being used during this project.

5. Managed Servers for Coherence Cluster

   Managed servers in coherence cluster are WebLogic's managed server that joining the coherence cluster as either Data Tier Cluster members, Application Tier Cluster members, or Proxy Tier Cluster members. Every cluster in WebLogic domain can be expanded by adding a new machine or existing machine as a member in the cluster. Every machine in same deployment tiers (data/application/proxy) will share the same behaviours even for the new cluster member.

6. Coherence*Web

   Coherence*Web is an HTTP session management module dedicated to managing session state in the cluster environment [9]. In this project, Coherence*Web allows application to store HTTP sessions in a Coherence Cluster. This approach duplicates the HTTP sessions data into several coherence managed servers in data tier cluster and increase the reliability and availability of the application. A WLS deployment descriptor (weblogic.xml) has been made to enable the Coherence*Web sessions and it has been save into <APPLICATION_DIR>/clusterjsp-ear/clusterjsp-war/WEB-INF/ directory.

4.5 Reliability and Data Availability of Coherence Cluster Deployments

In addition to these project goals, the test was conducted to the application deployed in the coherence cluster. This is an additional goal regarding to project to analyse the expected behaviours of coherence cluster deployment tier. The scenario is as follow:

- The Coherence Cluster Deployment tier does not contain Proxy Tier Cluster as it is not needed during trial.
- Deployed application is a Cluster JSP application that is being provided by Oracle and it has function to store data across the Data Tier Cluster. It is also capable to delete the HTTP sessions data. The application is available in the following link: https://blogs.oracle.com/OracleCoherence/resource/files/clusterjsp-ear.zip
- Data Tier Cluster and Application Tier Cluster at least contain two Coherence Managed Servers.

- Coherence*Web is set to enable HTTP sessions management module as described in section 4.4.

First the browser, directly or by front-end application, accessed the web application available in each of the Application Tier Cluster Members. Then the user input the HTTP sessions data to the application and it was being replicated in Application Tier Cluster. The browser was used to access the application in other managed servers in cluster and the application showed that the HTTP sessions data had been replicated. Then the test was conducted to the application by shutting down all of the managed servers in the Application Tier Cluster. As soon as the Application Tier Cluster started again, the application was still having the HTTP sessions data and it has been proed that application data had not changed even after the application died. The result of Coherence Cluster Deployment can be seen in figure 4.8. and 4.9.

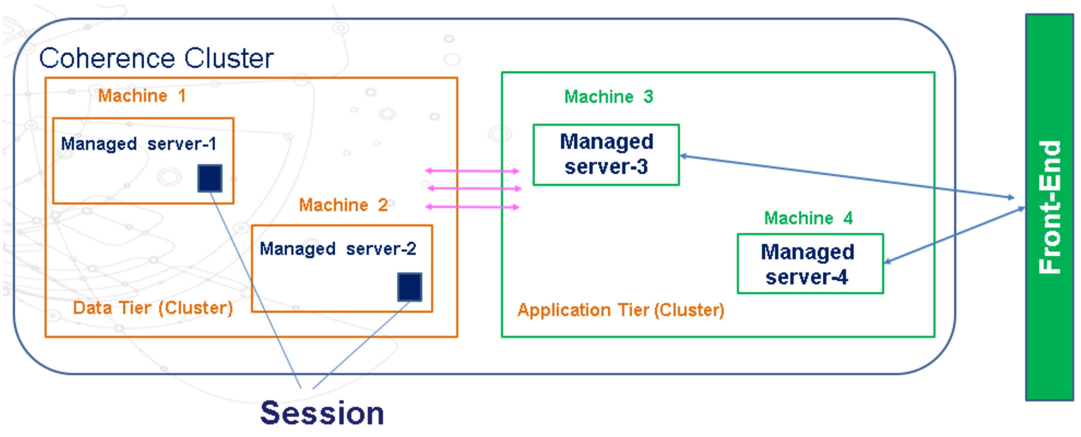


Figure 4.8. The initial Condition, session data is replicated to every server in data tier cluster

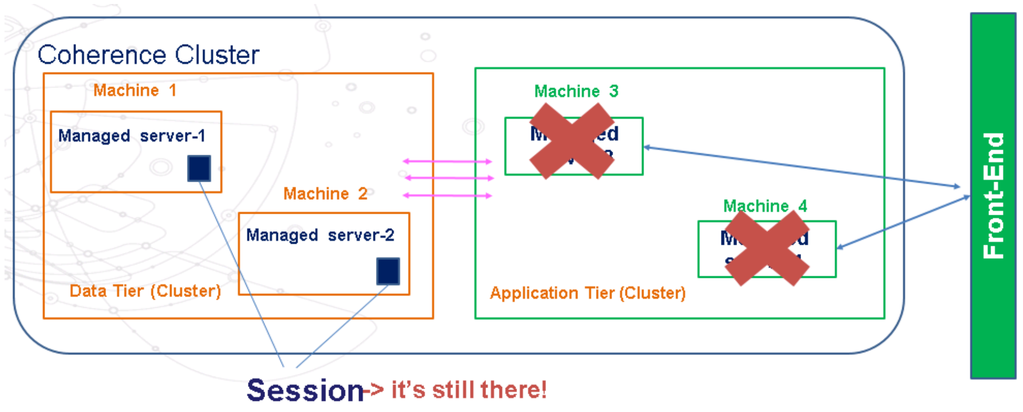


Figure 4.9. The Coherence Cluster can prevent the session data loss even if the application tier cluster goes down

It has been discovered also that it is possible to use coherence cluster deployment tier to the WebLogic’s Dynamic Cluster and Domain Partition alongside OTD as the front-end to the application. This integration will create the dynamicity, availability, reliability, scalability, and performance of the application.

5 Conclusion

The evaluation of Oracle Traffic Director from its installation until integration with WebLogic features has been made through this project and it can be concluded as follow:

- Oracle Traffic Director (OTD) comes up with two installation types (Standalone OTD and Collocated) and it works fine with the CERN’s IT-DB infrastructure for installation, configuration, and instance deployment.
- With OTD silent mode installation and WLST-support, it is possible to automate the installation procedure and configuration.
- The OTD can be managed as RPM package for installation Standalone OTD, with the help of python scripts that have been created during this project.
- Integration of OTD-WebLogic’s Dynamic Cluster and Domain Partition creates dynamicity for OTD as the front-end application to recognize changes in back-end servers and be able to start/stop request to the new/down servers.
- Apart from this project goals, evaluation of Coherence Cluster shows that the deployment tier of Coherence Cluster can be used to prevent data loss, which is HTTP sessions data in this case. This implies that users can maintain their data even though the application servers are all down.

6 Future Scopes

Currently, CERN has not implemented OpenStack's Floating IP features and this prevents the usage of Virtual IP. This causes the disability to discover failover modes of OTD which needs the Virtual IP to implement. Therefore, if CERN decides to use OTD as the front-tier replacement, CERN shall start to implement the OpenStack's Floating IP and inversitage the OTD's failover modes as it can increase the High Availability Architectures for CERN's Web Application Ecosystem.

This project results are obtained from the working in single machine or few numbers of machine (office PC and CERN IT-DB's VM Infrastructures). Consequently, the bigger or complex testing should be conducted to verify the project result for more complex system and architecture.

There are a number of features from OTD and WebLogic that need to be discovered and integrated each other to create another benefits to Web Application Ecosystem. The security features of OTD and WebLogic should be considered also.

In addition, the scripts, that has been created during this project, need to be improved for the scope of OTD configuration that they are covering. Moreover, the scripts and spec file for RPM need to be upgraded to support the creation of RPM for Collocated OTD.

7 References

[1] http://www.oracle.com/technetwork/middleware/otd/overview/index.html

[2] https://docs.oracle.com/middleware/1221/otd/admin/get_started.htm#OTADG102

[3] https://docs.oracle.com/middleware/1221/otd/admin/img/tdgsg_simple_topology.gif

[4] http://docs.oracle.com/middleware/1221/wls/INTRO/intro.htm#INTRO124

[5] https://wiki.python.org/jython/FrontPage

[6] http://www.fabriziomarini.com/2015/12/weblogic-1221-multitenancy-example-of.html

[7] https://docs.oracle.com/middleware/1212/wls/CLUST/coherence.htm#CLUST661

[8] https://docs.oracle.com/cd/E18686_01/coh.37/e18677/cluster_overview.htm#COHDG5163

[9] https://docs.oracle.com/cd/E24290_01/coh.371/e22620/start.htm#COHCW109