**National Informatics Centre**

**Department of Information Technology**

**Ministry of Communications & Information Technology**

**Government of India**

**Open Government Platform**

***A Joint Initiative  
 of India & US Government***

OGPL: Architecture Document

**OGPL-Architecture**

**Version 0.3**

**11th January 2012**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Description of Change** | **Author** | **Date** |
| 0.1 | Initial Draft | National Informatics Centre | 13-Dec-11 |
| 0.2 | Updated H/W deployment | National Informatics Centre | 29-Dec-11 |
| 0.3 | Fine tuned the architecture and associated descriptions and H/W requirement | National Informatics Centre | 11-Jan-12 |
|  |  |  |  |

TABLE OF CONTENT

1 INTRODUCTION 4

1.1 Purpose 4

1.2 Scope 4

1.3 References 4

1.5 Assumptions and Constraints 4

1.6 Definitions, Acronyms and Abbreviations 5

2 ARCHITECTURAL BLOCKS 6

3 DEPLOYMENT ARCHITECTURE 7

4 SECURITY 9

4.1 Securing Admin URL 9

5 SCALABILITY AND AVAILABILITY 9

5.1 Frontend Servers 9

5.2 Database 9

6 DISASTER RECOVERY 9

7 MULTI-TENANCY 10

8 HARDWARE REQUIREMENTS 11

**LIST OF FIGURES**

[Figure 1: Architectural Blocks 6](#_Toc312961803)

[Figure 2: Deployment Architecture Blocks 7](#_Toc312961804)

[Figure 3: Deployment Hardware 7](#_Toc312961805)

# INTRODUCTION

Open Government Platform (OGPL) is envisioned to be a platform for creating and hosting Open Data Websites. The website is intended to be used by Governments and their agencies to publish datasets, documents, services, tools and applications collected by them for public use. It enhances the transparency in the functioning of Government and shall also open avenues for many more innovative uses of Government Data to give different perspective of the situation.

## Purpose

The purpose of this document is to define the architecture of OGPL platform and document the rationale behind architectural decisions.

## Scope

This document describes the functional and deployment architecture of OGPL platform.

## References

Functional requirements document of ODP.

Various links related to Drupal scaling.

<http://london2011.drupal.org/sites/default/files/DC%20London%20Presentation.pdf>

<http://buytaert.net/drupal-performance>

<http://www.johnandcailin.com/blog/john/scaling-drupal-step-four-database-segmentation-using-mysql-proxy>

<http://lists.drupal.org/pipermail/support/2009-July/012599.html>

<http://drupal.org/node/121209>

## Assumptions and Constraints

### Assumptions

Following assumptions have been made towards the architecture.

* W-W-CMS/DMS/VRM will be integrated into one system
* Mysql replication will be use for high availability

### Constraints

Following constraints have been considered towards the requirements for OGPL.

* Drupal have constraint on database scalability, so we have considered on scaling on web servers and use caching mechanism to cope with the load on the database.

## Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| **Acronym** | **Full Form** |
| W-CMS | Website Content Management System |
| DMS | Data Management System |
| IA | Information Architecture |
| LDAP | Lightweight Directory Access Protocol |
| NIC | National Informatics Centre |
| OGPL | Open Government Platform |
| PMO | Program Management Office |
| POC | Point of Contact |
| VRM | Visitor Relation Management |
| CAS | Central Authentication Services |
| OpenId | Open Id Authentication |

# ARCHITECTURAL BLOCKS

**OGPL Components**

VRM

W-CMS

Database

Internet/Browser

**OGPL Web server**

DMS

Apache

SOLR

Mail Gateway

Charts Library

Other Vendors Api

Add this

Third party integration

###### Figure 1: Architectural Blocks

Following are salient points about architectural blocks of OGPL.

* All the functional components of the system though developed separately, will be closely integrated in single instance of Drupal.
* This will be responsible for creation of Dataset Catalogs, Managing feedbacks and W-CMS for Website management. This system will be accessible to the W-CMS/DMS/ VRM users over internet with using CAS/OpenID or LDAP authentication protocol as proposed in DMS development.
* Website functionality will be generated through its W-CMS using same Drupal instance. This will be accessible to public without any user authentication.
* W-CMS/DMS/VRM will be responsible for managing the content to be published on the Website and managing feedback coming from users of the Website.
* As per DMS implementation sessions are maintained in database, this would also be applicable for all the backend modules W-CMS, DMS and VRM.
* Frontend site does not have user login/registration, so there won't be explicit session management for this site.
* All security aspects would be taken care in compliance with the OWASP guidelines.

# DEPLOYMENT ARCHITECTURE

**File System**

Load Balancer

Drupal 1

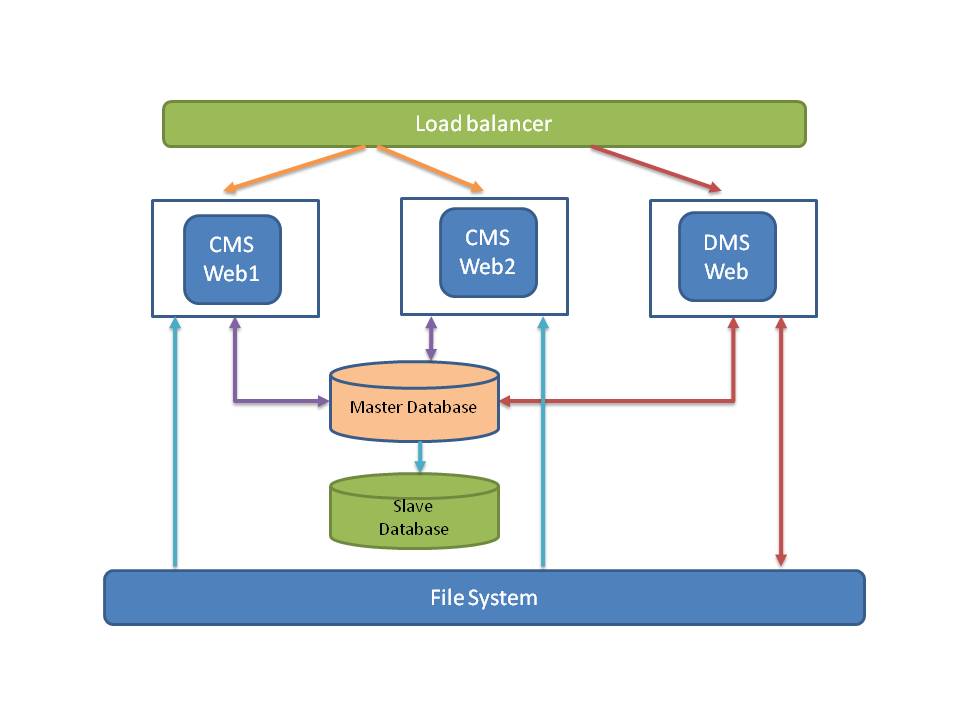
Drupal 2

Drupal N

**Master Database**

**Slave Database**

###### Figure 2: Deployment Architecture Blocks

Figure 3: Deployment Hardware

Following are the salient points about Deployment Architecture of OGPL.

* An instance of OGPL will be an array of frontend servers with Drupal containing all the functional blocks as described in the previous section.
* It will contain a load balancer to balance the load between the frontend servers. We can have either hardware load balancer or load balancer software for distribution.
* There will be another load balancer deployed on one of web server (W-CMS Web1) for failover of load balancer. This load balancer is optional hence not depicted in diagram.
* All Frontend servers will access data from the database server. They will have restricted write privileges to the database server and will be able to write to specific tables for which the write privileges are provided.
* Database server will be replicated for the purpose of high availability. In the event of failure of master database server, slave will be promoted as the master.
* W-CMS/DMS/VRM will be deployed on a single server. The load on backend modules such as W-CMS/DMS/VRM will be limited and a single web server is sufficient to take load on the OGPL backend modules.
* W-CMS/DMS/VRM file system will be NAS and mounted on all web servers. This will ensure that there are no data integrity issues in the file system to which these modules write data.
* W-CMS/DMS/VRM web server will have read/write permission to file system. Other web servers will have only read permission to file system.
* All these servers could be either virtual machines or physical machines as per the requirement.
* Load balancer will be configured to make sure DMS traffic is routed to one server (DMS web).

# SECURITY

# Securing Admin URL

Following are options to make admin URL more secure:

* Renaming the admin URL using custom re-write rules

<http://fresh2l.com/en/blog/2010/12/28/change-admin-url-drupal-enhance-security>

<http://www.anjalytics.com/www/node/32>

* Having SSL certificates on the important backend URL

http://drupalscout.com/knowledge-base/drupal-and-ssl-multiple-recipes-possible-solutions-https

http://drupal.org/project/securepages\_prevent\_hijack

* Login security module can be used to block an account/IP address based on number of failed login attempt.
* Port 443 at the firewall (only let in the IP addresses that are allowed to log in)
* HTTP authentication

<http://www.chapterthree.com/blog/josh_koenig/howo_use_drupal_http_authentication>

# SCALABILITY AND AVAILABILITY

# Frontend Servers

* Frontend servers will be scaled using a sticky load balancer. Load balancer can be Apache mod\_proxy, haproxy or a commercial hardware based load balancer.
* High availability for load balancer will be implemented by using a redundant load balancer. Heartbeat will be used to monitor primary load balancer and switch over to the secondary load balancer in case of failure of primary load balancer.

# Database

* Drupal has limitations on database scalability. It doesn’t have support for sharding unlike other W-CMS frameworks such as Wordpress.
* The traffic foreseen on the database is not expected to hit the database most of the times as most of the content such as Datasets will be served through the cache.
* Use of Drupal cache as well as external caching mechanisms such as “memcache” or “Boost” module to reduce the load on the database significantly. In case of memcache it can be configured on web server or exclusive server may be dedicated for enabling memcache.
* Database high availability will be achieved through replication of database on master and slave mode. In case of DB fail the slave can be promoted to master.

# DISASTER RECOVERY

* A complete DR setup will be created; this will be a replica of the active setup. This will be hosted at a separate physical location than the active setup.
* DR setup will be activated in the event of any disaster at the active setup and traffic diverted to the same using DNS.
* Incremental backup of database will be maintained on a daily basis that will be used for recovering data in the event of failure of database server or its file system.

# MULTI-TENANCY

It is preferred to replicate the entire setup for implementing multi-tenancy over Drupal’s intrinsic multi-tenancy feature. Following is the reasoning behind the same.

|  |  |
| --- | --- |
| **Drupal’s Multi-tenancy** | **Cloud** |
| Drupal instance will remain the same for all the tenants. This will result in heavy load on the frontend server and impact performance as the number of tenants increase. Load on one tenant can have an impact on all the tenants. | Each deployment becomes a separate instance of Drupal with a separate database instance. All tenants (countries in this case) will remain completely isolated. Load on the server and database will be specific to that tenant. Scalability of solution is much better than the other model. |
| Creation of a tenant will be handled through Drupal and no installation required. Users and roles will have to be created for each tenant. | Extremely easy to install. An image of raw installation can be created. The basic installation and configuration/branding can happen in a matter of hours after which the system can be ready for data entry. No additional configuration required on the users and roles. |

# HARDWARE REQUIREMENTS

Following are hardware minimum requirements for OGPL setup.

* Load Balancer
  + - One primary load balancer. This can be software such as HAProxy or a commercial hardware solution.
    - One fallback load balancer. This will be activated in case primary load balancer fails. This can be deployed on one of the web servers.
    - *Hardware specifications for one Load balancer*

CPU: Intel Xeon 2GHz

RAM: 4GB

HDD: 100 GB

* Web Servers
  + - Three web servers will be used. One for W-CMS/DMS/VRM and two for rendering website.
    - Number of backend servers can be scaled up based on the load.
    - Hardware specifications for web servers

CPU: Intel Xeon 2GHz

RAM: 4GB

Storage: As per requirement, but initial allocation should be around 200 GB

* Database servers
  + - Two database servers
    - Master server to which all the Drupal instances will connect for read and write operations.
    - Replication slave server for the purpose of high availability.
    - Hardware specifications for database servers

CPU: Intel Xeon 2GHz

RAM: 8GB

Storage: As per requirement, but initial allocation should be around 500 GB.

* File system server
  + - This will be used store File system of W-CMS/DMS/VRM. This file system will be mounted on all web servers.
    - *Suggested NAS Configuration*

NAS can be of any make, vendor should provide iSCSI card preferably 4GPS / 8 Gbps, we will require 4 cards one each on 3 web Servers and one for NAS box. SAS HDD of 200 GB for NAS storage for file system.

* + - *Other option*

*In case NAS is not available we can use server similar to web server configuration with 200 GB disk space.*

* Summary of hardware specifications.
  + - Load balancer hardware -1 server
    - Web servers – 3 servers
    - Database server – 2 servers
    - File system – 1 storage server