TECHNOLOGY OVERVIEW DOCUMENT

Version 1.0

	Prepared By/Last Updated By	Reviewed By	Approved By
Name	Rajkumar Sithavan	Brian T Fujito	Brian T Fujito
Role	Business Technology Analyst	Chief Technology Officer	Chief Technology Officer
Signature	Raj	Brian	Brian
Date	12/11/2009	12/11/2009	12/11/2009

TECHNOLOGY OVERVIEW

Purpose

The purpose of this document is to give an overview about the setup of our system from a technical standpoint.

Key Components

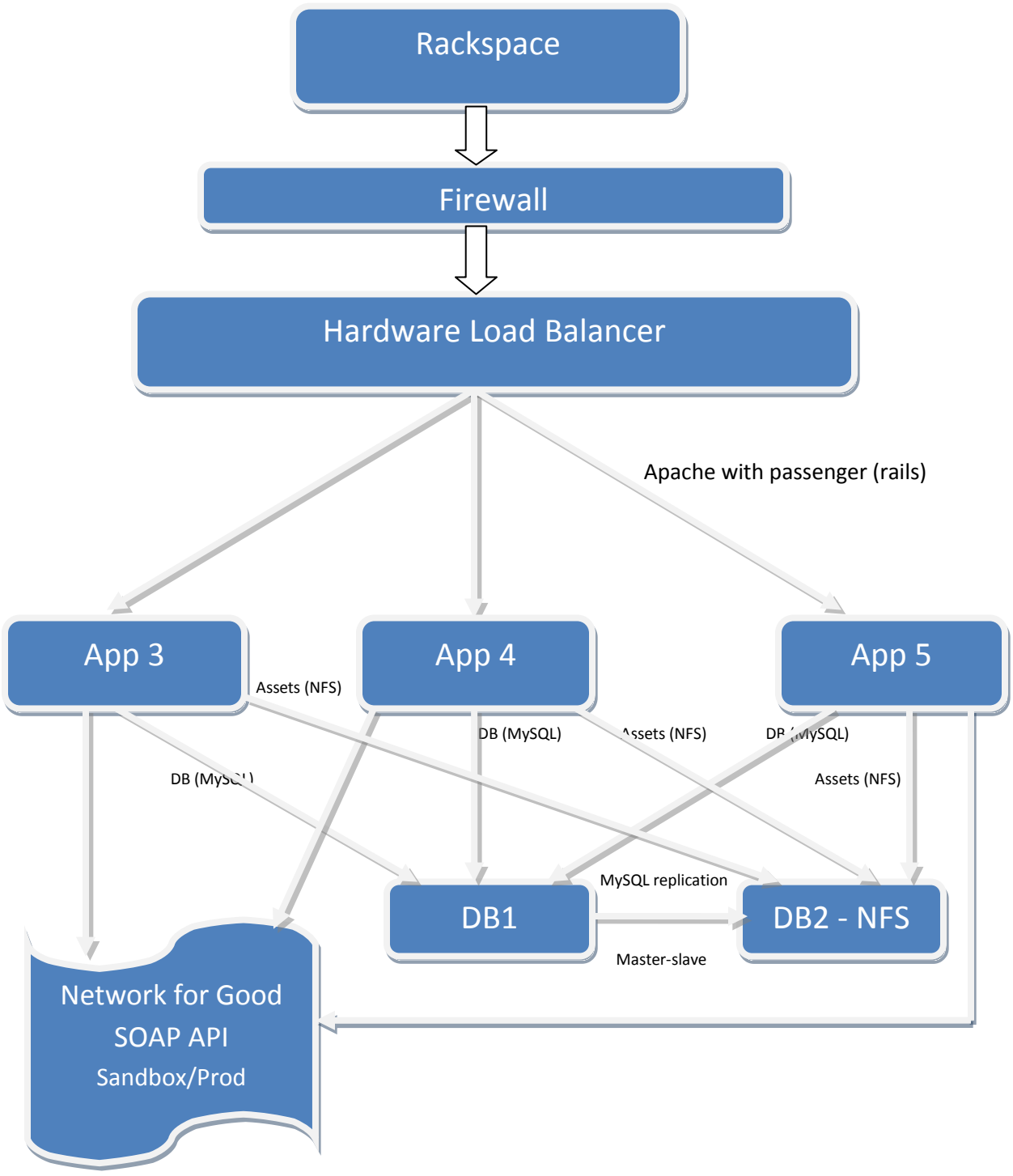
- Rackspace
- Firewall
- Load balancer
- App servers
- Database (DB1) – Database Master; Sphinx Search
- Database (DB2) – Database Slave; Network File Server

System Setup

Razoo is hosted by Rackspace, one of the top leaders in hosting.

The diagram below would help understand how the components are mapped together.

The firewall is dedicated, and permits direct SSH access only to our staff (via IP), and then: 80 (HTTP) and: 443 (SSL) traffic only to our front-end application servers (app3, app4, app5).


Rackspace
Firewall
Hardware Load Balancer
Apache with passenger (rails)
App 3
App 4
App 5
Assets (NFS)
DB (MySQL)
Assets (NFS)
Assets (NFS)
MySQL replication
DB1
DB2 - NFS
Master-slave
Network for Good
SOAP API
Sandbox/Prod

There are three front-end application servers (app 3, app 4 and app 5) running Apache with passenger on Rails cloud. Both production and stage environment are currently in each of these App servers. The hardware load balancer is used to basically split network load across these three servers. So any requests that come in are routed to individual servers by the hardware load balancer. The routing is based on source IP only – we do not enforce sticky sessions as that's not needed.

The main reason for using the load balancer is to increase the capacity of the system, improve performance and to provide resilience.

All the App servers are connected to a database (DB1) which is a MySQL database. All the data gets stored in this database. Whenever there is a request to the App servers, it will fetch/store data from/on DB1. The DB2 server is configured to mirror MySQL from DB1 via a master-slave configuration. This is currently purely for redundancy/failover purposes – i.e. we are not load-balancing database access.

In addition, DB2 serves as a Network File server. The user generated assets (images) in the website are stored in DB2. So basically the App servers have to communicate with DB1 and DB2 in parallel to load a page.

Finally, we also implement a search functionality using Sphinx which resides on DB1. Periodic cron jobs on DB1 update the Sphinx search index.

There are also some backend and background processes that facilitate our work. BackgroundRB runs on all the servers to allow off-loading of long-running methods (e.g. admin reports). Cron jobs are used to run search re-indexing, stale session clearing, and processing of recurring donations through NFG.

Interaction with Network for Good

Currently, the company "Network for Good" (NFG) is the credit card processing partner for Razoo. So any donation made needs to be communicated to NFG for processing. In order to facilitate this process, the system uses a SOAP API. NFG has both a sandbox and a production environment. So any request sent from Razoo dev/test/stage environment reaches only NFG's sandbox environment. This is mainly used for testing purposes.

The SOAP API gets the NPO details, creates card on file, makes donation and stores the token and not the card. Behind the scenes, NFG communicates with Guidestar for nonprofit data, and PayflowPro to handle the actual card authentication, storage and charging.

Content Delivery Network

We utilize EdgeCast.com's Content Delivery Network service (CDN) to optimize delivery of our various small-file assets (e.g. images, JS files, CSS files). The CDN essentially acts as a web proxy cache, which detects expiration based on URLs that we provide (e.g. with asset timestamps). We do not actively push files up to their caching servers – rather, their caching servers simple contact our server environment in Rackspace if they do not have the assets that were requested (in short, we use a "pull" mechanism, with our servers acting as the origin server).