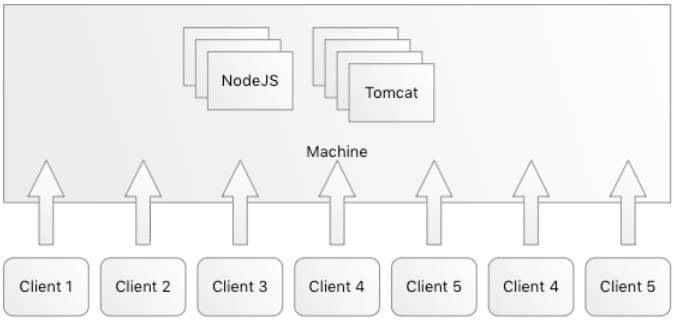
Name: Bernardo Breder
Email: bernardobreder@gmail.com
Skype: bernardobreder
Linkedin: https://www.linkedin.com/in/bbreder
Graduate: Universidade Federal Fluminense (UFF) - Monograph: https://goo.gl/lG2yyH
Master: Universidade Federal Fluminense (UFF) - Dissertation: https://goo.gl/esrozB

Project Proposal

Introduction

A cheap implementation of Web Application Server (WAS) for different clients with isolated environment, can done by sharing differents Application Servers like NodeJS, Tomcat, PHP on a single physical machine.

Web Application Server (WAS) works on one or more servers, which hosts projects from different clients simultaneously on the same or some machines. According to each company's politics, you can use a single server to serve multiple websites.



There are several ways to implement a Shared Web Application Server (SWAS), where all of them simulate an Application Web Server Private for customers. For example, SWAS can be implemented by allocating different server ports, referenced by domain DNS, for each client. Or can be implemented using the same static IP and through the request header, it can be directed to the appropriate client.

Difficulties

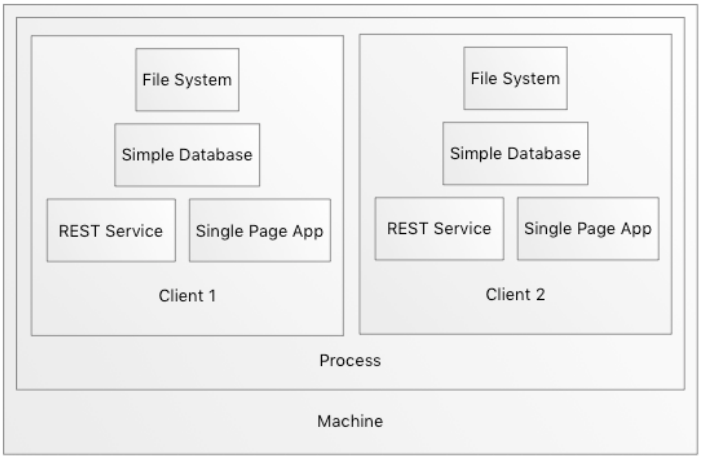
The Shared Web Application Server (SWAS) has some features such as:

- You need to make some adjustments to the Application Server to redirect requests to client projects by simulating a Private Web Application Server. These adaptations are not so simple to implement and require a certain management both to maintain and to create.
- Popular Application servers such as NodeJS, Tomcat and PHP with their corresponding Programming Languages are typically built for private environments, not implementing important requirements when they are installed on a shared infrastructure. These requirements not implemented are adapted by generating an overhead on the machine for each client using the machine.
- Depending on the application server you want to share, you must allocate an Application Server Process for each client to ensure isolation between clients. With this, the number of clients will determine the number of Processes in the Operating System, limiting the resources of the machine in function of the overhead of each process.
- Likewise, depending on the shared database installed available to clients, you need to allocate resources to each client in order to isolate the databases of the clients. In addition, external Database Process from Application Server can cost many resources for the machine.

Proposal

The main goal of the project is to build a Programming Language (LP), a simplified Database, and a Shared Web Application Server (SWAS) that better manage client projects and machine resources without having to make adaptations to isolate and reference the projects of different clients.

Ideally, this project aims at a single machine to have a single process in the operating system for all projects and database of all clients, simulating a Private Web Application Server Environment. In addition, all files are being managed by SWAS, virtually, with automatic backups, encrypted and moved from one disk to another according to storage needs. When the server goes into overload, a set of clients will be moved to another machine automatically with the goal of balancing each of each server.



Creating a SWAS allows you to implement some functionality for managing machine resources such as:

- The SWAS of this project will use a single Process in the Operating System to respond to all client project requests. This requirement also applies to database sharing for different clients, where this single Process will be used to serve all REST and Database requests from different clients.
- A Simplified Database will be built to store project data to reduce the cost of large databases such as MySql, Oracle and Postgre on the machine.
- With this SWAS you can create the Service Modularization feature where clients can use services from other clients as a code reuse. This modularisation is a very light way of sharing code and service. This functionality can enable service sales from one customer to another, generating a ServiceStore market.
- You can create a load balancing system for SWAS to balance request peaks. When a machine is overloaded with many requests, SWAS can transfer the execution context of some clients to another machine. This feature allows sites that receive

many requests to be automatically transferred to other servers in order to balance server load.

Creating a new programming language (LP) allows you to implement some improvements to the host and client projects such as:

- The client project implemented in the LP will be compiled into a Single Page Application allowing the reduction of network traffic, where the responsibility of the host is to respond only to REST requests and not HTML, CSS, JS and Images, as is usually done . All HTML, CSS, JS and Images will be stored offline by the browser through the Html5 feature in order to reduce data traffic. This feature will be available for projects using this LP. This reduction in traffic will decrease server demand, thus increasing the number of clients per machine.
- Creating a new File System Library for client applications has the following advantages:
    - The library will automatically back up according to the operations written by the applications. With this, the host company will no longer manage the backup of the data anymore.
    - The library will allocate the files to the clients virtually, invisible to the client the location of the files on the disk. This feature saves disk space and when the client wants more space, reallocation can occur any time.
    - Encryption of client data on the disk will occur when data is read and written, protecting the client from unauthorized access.