Design and implementation of a social network infrastructure for designers of Multi-Cloud applications

Christos Papoulas

Thesis submitted in partial fulfillment of the requirements for the

Masters' of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
Knossou Av., P.O. Box 2208, Heraklion, GR-71409, Greece

Thesis Advisors: Prof. *Kostas*, Dr. *Magoutis*

Optionally: This work was partially supported by **fill in sponsor name**

University of Crete
Computer Science Department

Your Title

Thesis submitted by
Author Name
in partial fulfillment of the requirements for the
Masters' of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Author Name

Committee approvals: _____
Name of first member
Assistant Professor, Thesis Supervisor

_____
Name of second member
Associate Professor, Committee Member

_____
Name of third member
Professor, Committee Member

Departmental approval: _____
Name of Director of Graduate Studies
Professor, Director of Graduate Studies

Heraklion, July 2015

Abstract

In this work ...

Περίληψη

Στην εργασία αυτή ...

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this work proposed the design and implementation of a model-driven social networking platform for designers of Multi-Cloud applications. In this target specific social networking platform, DevOps can benefit from other users experience and answer design questions such as which is the most cost-effectiveness deployment, which configuration fits their needs. This social networking platform joins together all social networking concepts such as personal messaging, groups, new feeds with modelling-driven concepts of application composition and deployment, integrating a repository of cloud applications and infrastructure description based on Cloud Application Modelling and Execution Language (CAMEL).

This repository means to several benefits, An integrated environment can enrich user interactions with structured references to applications and their components, execution data, and mined knowledge from real deployments. Mined knowledge can be combined with user activity and profiles to provide personalized suggestions and hints. An improved mode of user interaction is expected to result to stronger incentives for DevOps users to contribute information to the underlying repositories. More content should lead to better quality of mined knowledge, benefiting the DevOps community and providing further incentive for contributions. The social networking platform designed to be closely integrated with a set of information repositories satisfying the following requirements: (R1) handle entire applications rather than just software components; (R2) abstract application structure through software modeling; (R3) capture and analyze application runtime performance. Raising the level of abstraction from components to applications and from code-centric to model-centric is expected to facilitate interaction between DevOps professionals. The analysis of application execution data can provide answers to many interesting questions of the community and support discussions and arguments with hard data. These requirements can provide software developers with strong incentives to contribute, leading to the sustainability and growth of information and derived knowledge in the repository.

1.1 Background

1.2 Related Work

This section describes related work for other professional networks and their caching architecture.

1.2.1 Caching data

Facebook, the largest social network, serves billions of requests per second using memcached [1]. In this magnitude of scale Facebook has several pools of memcached servers (regional pools) along the globe. A single request for a page can produce hundred of requests to the back-end system. Memcached used to store not only key-value from MySQL queries but also pre-computed results from sophisticated algorithms. In order to achieve a near real time communication experience to the end user, memcached server have to be efficient, reducing latency.

The research question in such systems is when a particular key will be invalidated. This problem occurs according to [1] in two cases: (1) *stale sets* and (2) *thundering herds*. A stale set occurs when a web server sets a value to the memcached that does not reflect the real value of the database. Thundering herds occur when a specific key has a heavy read and write activity in the same time. Stale sets resolved by a N-bit token, bound to specific key, sent from memcached to web server that want to update the key when cache miss occurs. If a delete request received then the request for updating this value from that client is rejected. The thundering herbs solved by configuring memcached servers return a N-bit token only once every ten seconds per key.

Linkedin, the largest professional network, stores hundreds of terabytes of data to Project Voldemord [2], a key-value store, inspired by Amazon Dynamo [3]. Linkedin stores to Voldemort pre-computed offline data for example results of data mining applications such as features like "People You May Know" which running on hundreds of terabytes to make an estimation, using Hadoop as the computational component of those estimations. Voldemord compared with Dynamo has the same following requirements: (1) a simple *get/put* application interface (2) A *replication* factor, the number of replicas for each key-value tuble, implemented using vector clock (3) a *required read* factor to succeed a get request, (4) a *required write* factor to succeed a put request.

1.2.2 Professional Networks

IT professionals use a variety of online sources as aids in their daily tasks. Developers typically prefer community-moderated forums over vendor-moderated sites.Social networks focusing on software technology in particular provide developers with the opportunity to leverage the knowledge and expertise of their peers.

One of the most popular such platforms is GitHub [4], a collaborative revision control platform for developers launched in April 2008, and arguably the largest code-hosting site in the world. GitHub provides social networking functionality such as feeds, followers, wikis and a social network graph that captures how developers work on versions of their repositories, which version is newest, etc. Gitter [5] is a related service that facilitates discussions between members of GitHub communities by providing a long-term chat integrated with code and issues. Sourceforge [6] was the first code-hosting platform offered to open-source projects. It was launched in 1999 and offered IT professionals the ability to develop, download, review, and publish open-source software. Sourceforge is similar to GitHub in its support for social features. Other similar code-hosting platforms are Google Code [7] and Microsoft CodePlex [8]. None of those platforms collect, analyze, or use information from executions of application deployments to improve the level of technical discussion between users or abstract code structure through modelling or enhance user interactions through the use of analytics over application execution histories.

StackOverflow [9] advances on earlier Q&A sites in which users ask and answer questions. Users can vote up or down questions and answers and earn *reputation points* and *badges* in return for their active participation. Although StackOverflow and GitHub address different aspects of software development (StackOverflow is not a code-hosting platform) there is a synergy and correlation between the two [10]. The proposed social network platform extends StackOverflow through the use of social networking features that enable users interested in reasoning about application deployments to use and share knowledge drawn from analyses of information repositories.

IBM's BlueMix [11] is a development and support platform for communities of DevOps users wishing to compose distributed applications out of components drawn from libraries and deploy them at IBM-provided and supported cloud infrastructure. BlueMix is a key component of IBM's DevOps best practices [12] for achieving rapid prototyping, automated deployment, and continuous testing of software. BlueMix encourages its users to ask their questions to StackOverflow but also includes a community forum [11] with rating of answers contributes to eventually building a basic knowledge base, similar to traditional approaches such as StackOverflow. The proposed social network platform system differs from BlueMix in its support for expressing applications as models (CloudML, CAMEL) and its use of two information repositories, the PaaSage repository of models and execution histories and Chef supermarket, and the use of analytics over past executions to enable users to reason about application deployments. A common feature between the proposed social network platform and BlueMix is support for deployment of distributed applications.

LinkedIn widely adopted across a range of professional communities due to its robust set of social features (and to some extent due to its use of extensive analytics over collected information [13]), LinkedIn provides no specific support for software engineering activities and thus more closely resembles traditional social networking platforms such as Facebook.

The above systems can be further classified based on whether they use a repository to store software-related information (code, models, configuration, or execution histories) and whether this information is shared and raised through crowsourcing [14]. GitHub, GoogleCode, CodePlex, SourceForge, BlueMix, Chef Supermarket, and our platform store at least one type of software-related information and all systems but BlueMix are raising shared content in their software-related repositories via crowsourcing. Our professional network is the only solution that analyzes information in its software-related repositories to assist users with suggestions and hints.

Table 1.1: Feature comparison.

	Interaction Between Users				Repository						
	Social features [a]	Groups	Q & A	Personal messaging	Software code	Software models	Software config	Execution histories	Crowd sourced	Repo assisted hints [b]	Application deployment
GitHub	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗
Sourceforge	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗
GoogleCode	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗
CodePlex	✓	✗	✓	✓	✓	✗	✗	✗	✓	✗	✗
StackOverflow	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
BlueMix	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓
Chef Supermarket	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗
LinkedIn	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗
bettercodes											
PaaSage SN	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓

[a] Features: follow and news feed

[b] User assistance based on data analysis of the repository

Chapter 2

Social Network User Interface

General discussion ...

2.1 AA

2.2 BB

2.3 CC

2.4 DD

Chapter 3

Implementation

This section describes the implementation of social network site and how the system scales.

The system is composed by the following components, as shown in figure 3.1: At the first layer lives (1) the Social Networking engine, which runs all PHP scripts and described in section 3.1. At the second layer lives (2) the Memcached caching system, which described in section 3.3. At the third layer lives (3) the Social Network MySQL database, and (4) the CDO server - client components and the CDO repository.

Achieving the scalability of the system, two system architectures are examined at two layers of the system: (1) We added more than one Social Network engine at the first layer of the system. In this implementation, in order to keep the file system in consistent mode we integraded Apache Zookeeper[15]. (2) We added more than one memcached machines at the second layer in order to add more cpu capacity and improve the system response time.

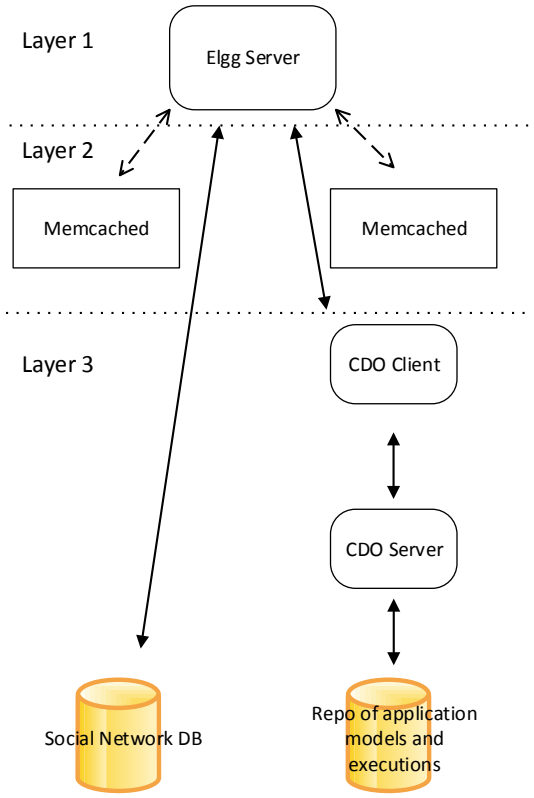
3.1 Implementation of Social Network

The social networking platform is implemented over the extensible Elgg social network framework[16]. Elgg is open source software written in PHP, uses MySQL for data persistence and supports jQuery [17] for client-side scripting. The architecture of Elgg Social Network shown in figure 3.3. The Model of the framework is structured around the following key concepts as shown in figure 3.2

- *Entities*, classes capturing social networking concepts: users, communities, application models. Elgg Core comes with four basic objects: ElggObject, ElggUser, ElggGroup, ElggSite, ElggSession, ElggCache and a lot of other classes necessary for the proper engine operation.

- *Metadata* describing and extending entities (e.g., a response to a question, a review of an application model, etc.).

Figure 3.1: The overall architecture of Social Network.

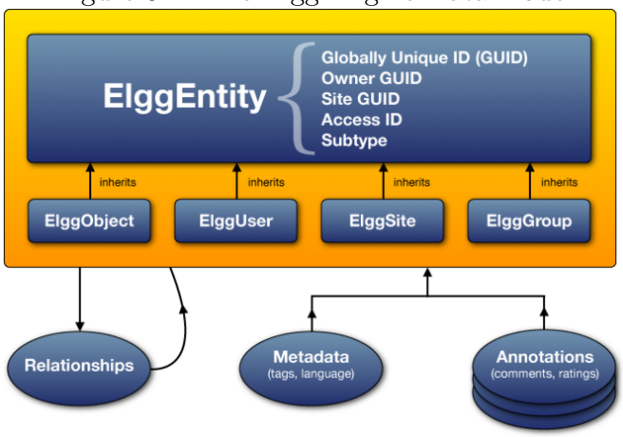


- *Relationships* connect two entities (e.g., user A is a friend of user B, user C is a contributor to an application model, etc.) and are persisted in the Social Network DB.
- *Annotations* are pieces of simple data attached to an entity that allow users to leave ratings, or other relevant feedback.

All Elgg objects inherit from ElggEntity, which provides the general attributes of an object. Elgg core comes with the following basic entities: ElggObject, ElggUser, ElggGroup, ElggSite, ElggSession, ElggCache, as well as other classes necessary for the operation of the engine.

Elgg comprises a core system that can be extended through plugins (examples are the Cart system or the handling of Application Models). Plugins add new functionality, can customize aspects of the Elgg engine, or change the representation of pages. A plugin can create new objects (e.g., ApplicationObject) characterized (through inheritance of ElggEntity) by a numeric globally unique identifier (GUID), owner GUID, Access ID. Access ID encodes permissions ensuring that when a page

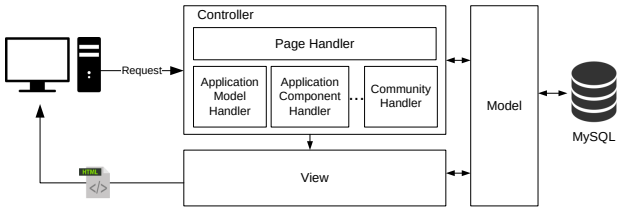
Figure 3.2: The Elgg Engine Data model.



requests data it does not touch data the current user does not have permissions on.

Figure3.3 shows the model, view, and control parts of Elgg's architecture. In a typical scenario, a web client requests an HTML page (e.g., the description of an application model). The request arrives at the *Controller*, which confirms that the application exists and instructs *Model* to increase the view counter on the application model object. The controller dispatches the request to the appropriate handler (e.g., application model, component handler, community handler) which then turns the request to the view system. View pulls the information about the application model and creates the HTML page returned to the web client.

Figure 3.3: Architecture of the Elgg Social Networking engine.

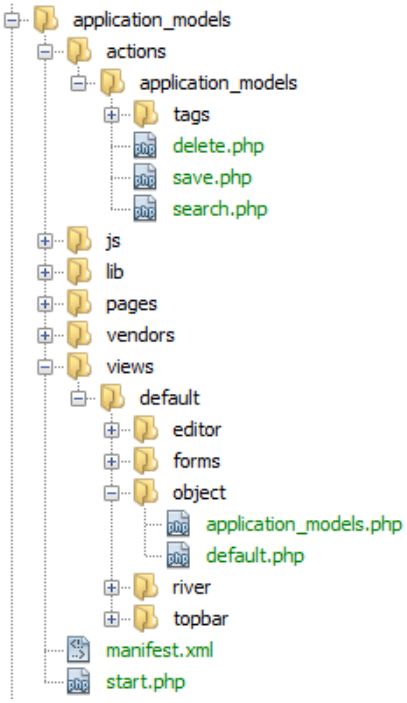


The extensibility of Elgg can be established not by modifying the core system but by introducing new plug-ins which follow the MVC model. A new plug-in can create a new entity. Thus, each entity is characterized by a numeric Globally Unique Identifier and Access ID. The Access ID determines the permissions that other users have. Thus, when a page requests data, it never touches those data that the current user does not have permission to see. All plug-ins share a common structure of folders and PHP files, following the MVC model of figure 3.3.

The hierarchy of a plug-in is shown in figure 3.4. Folder *actions* includes the actions applied on application models. Every active participation by the user is

performed via an action. Logging in, creating, updating or deleting content are all generic categories of actions. The *views* folder contains the *php* forms applied on application models, *river* events (Elgg terminology for live feeds). Viewss are responsible for creating the output for the client browser. Generally, this will be HTML, but it can be also JSON or other format. *Pages* overrides elements of core Elgg pages and can be from chunks of presentation output (like sidebars) down to individual html code. The *js* and *lib* folder provides javascript and *php* library functions. Finally, the *vendors* folders include third-party frameworks such as Twitter's bootstrap front-end [18]. The most important file of a plug-in is the *start.php* script, which contains the *page handler*. Page handler is a function manages the plug-in pages enabling custom url redirect to a specific page. The plug-in initialization is also defined in the start.php and registers actions, events and determines the views.

Figure 3.4: The structure of the application description plug-in.



The execution history of deployments of application models and the description of those models is stored in the CAMEL information repository, which is implemented as an Eclipse CDO server. The exchange of information between Elgg and the CAMEL information repository is going through CDO Client who retrieves the information from CDO server and sent it to the Elgg over sockets.

3.2 User interface

The design of User interface of Social Network based on 102 mock-ups designed by HCI experts. In order to support those look & feel and the functionality of those mock-ups 25K lines of php, js and css code is written. ...

3.3 Memcache

This section describes the experience gained by using memcached[19]. Memcached is an open source, high-performance, distributed memory object caching system. We choose memcached, because is a generic simple in-memory key-value store. It has a powerful API available for PHP. After memcached integration the system increase the response time and performance.

Memcached stores all entities of Social Network, applications, components, users, group discussions and most important the executions of applications. Storing the executions of applications at Memcached the response time of the system increased because the PHP modules do not need to go through the heavy CDO client but get directly the executions of applications from Memcached.

The apache jmeter[20] was used to measure the responce time of the system and the sysstat tool[21] was used to measure the cpu usage. Section 4.1 shows the performance results of this implementation.

Chapter 4

Evaluation

This chapter describes the evaluation of the two different implementation of the system. (1) By introducing more than one memcached instances at layer 2 as figure 3.1 shows and (2) by introducing more than one Social Network engines at layer 1.

4.1 Improving Performance with memcached

4.2 Improving Performance with engine

Chapter 5

Comparison

Compare your work ...

5.1 AA

5.2 BB

Chapter 6

Conclusions and Future Work

Bibliography

[1] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab *et al.*, "Scaling memcache at facebook." in *nsdi*, vol. 13, 2013, pp. 385–398.

[2] R. Sumbaly, J. Kreps, L. Gao, A. Feinberg, C. Soman, and S. Shah, "Serving large-scale batch computed data with project voldemort," in *Proceedings of the 10th USENIX conference on File and Storage Technologies*. USENIX Association, 2012, pp. 18–18.

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," in *ACM SIGOPS Operating Systems Review*, vol. 41, no. 6. ACM, 2007, pp. 205–220.

[4] "Github," http://github.com/, [Online; accessed 24-May-2015].

[5] "Gitter: The chat for github," http://gitter.im/, [Online; accessed 24-May-2015].

[6] "Sourceforge," http://sourceforge.net, [Online; accessed 24-May-2015].

[7] "Google code," https://code.google.com, [Online; accessed 24-May-2015].

[8] "Codeplex: Project hosting for open source software," https://www.codeplex.com, [Online; accessed 24-May-2015].

[9] "Stackoverflow," http://stackoverflow.com/, [Online; accessed 24-May-2015].

[10] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *Social Computing (SocialCom), 2013 International Conference on*, Sept 2013, pp. 188–195.

[11] "Ibm blue mix for developers," http://https://developer.ibm.com/bluemix/, [Online; accessed 24-May-2015].

[12] "Ibm devops best practices," http://www.ibm.com/developerworks/devops/practices.html, [Online; accessed 24-May-2015].

[13] R. Sumbaly, J. Kreps, and S. Shah, "The big data ecosystem at linkedin," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. ACM, 2013, pp. 1125–1134.

[14] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.

[15] A. Zookeeper, "Apache zookeeper," https://zookeeper.apache.org/, 2015, [Online; accessed 20-May-2015].

[16] E. S. N. Engine, "Elgg social networking engine," http://elgg.org/, 2015, [Online; accessed 19-May-2015].

[17] jQuery, "jquery," https://jquery.com/, 2015, [Online; accessed 19-May-2015].

[18] "Bootstrap front-end framework," http://getbootstrap.com/2.3.2/, [Online; accessed 24-May-2015].

[19] Memcache, "Memcache," http://memcached.org/, 2015, [Online; accessed 18-May-2015].

[20] A. jMeter, "Apache jmeter," http://jmeter.apache.org/, 2015, [Online; accessed 19-May-2015].

[21] sysstat, "sysstat," https://github.com/sysstat/sysstat, 2015, [Online; accessed 19-May-2015].