KTH Royal Institute of Technology

Highly Available and Robust Network Services in Under-served Areas

Jiannan Guo

Master's thesis

Supervisor: Prof. Björn Pehrson, KTH
Dr. Amos Nungu, DIT
Examiner: Prof. Markus Hidell, KTH

Stockholm, 1st November, 2014

Part I

Robust Network Infrastructure in Rural Areas of Tanzania

Chapter 1

Introduction

It has been proven that the uses of Information and Communication Technology (ICT) can effectively support poverty alleviation and livelihood enhancement[**?**]. Bringing ICT in developing counties and under-served areas requires multinational partnerships and unique research commitments. There exist many outstanding projects and business cases in the field, although not all of them is reproducible, especially when adapted to a different culture and work environment. Two districts have been selected to conduct a pilot project, namely Serengeti Broadband Network, aiming to enhance ICT penetration, build buying power and establish a foundation of rural ICT development. In this thesis report, we cover: 1)A brief history and background of project; 2)Current states and challenges; 3)Our approaches and implementation; 4)Future plan. This report stands as a comprehensive documentation of current states of SBN development.

1.1 Background

ICT4RD[1] is designed as a research and business development project, aiming at provisons of ICT services in under-served areas of Tanzania. The project is funded by Swedish International Development Agency (SIDA)[2], and coordinated by Tanzania Commission of Science and Technology (COSTECH)[3]; Dar es Salaam Institute of Technology (DIT)[4] Tanzania; and the Royal Institute of Technology (KTH)[5], Sweden.Two pilot projects are created under ICT4RD, respectively Serengeti Broadband Network development (SBN-development) and Wami project. In this report, we only focus on SBN project.

[1] www.ict4rd.ne.tz
[2] www.sida.se
[3] www.costech.or.tz
[4] www.dit.or.tz
[5] www.kth.se

SBN development aims at building a self-sustained Local Access Network (LAN) convering two districts[**?**]. It hosts services locally, such as VoIP, mails. When upper link exists at any site of the LAN, other parts can also be online. By interconnecting schools, dispensaries and governments, a variety of applications are proposed and implemented over the network, such as e-learning, e-governing and e-health.

It is the seventh year of SBN project. During these years, enormous research effort has been contributed from partners to establish a robust, scalable and sustainable broadband network island.

1.2 Related Work

There are many successful rural ICT projects which stand as good references. Several of them are presented here, mainly focusing on technical issues while excluding business development. **Macha Project Nipel Wireless South Africa**

1.3 Problem Statement

ICT development in rural areas is a challenging task which requires innovations on both technical and managerial sides. We identify following main obstacles that limit technology deployment:

- Low affordability. In rural development, expenditure has alwasy been a critical factor, not only during the phase of procurement, but also maintainance and refurbishment. Necessary trade-off needs to be made between capacity and cost.
- Poor supply chain, especially power shortage. The lack of electricity is always a limitation in rural development. In Tanzaina, only 14% of the country is electrified and the figure reduces to 2% in the case of rural area[**?**]. Innovative power source such as solar is highly desired. Even those sites along the power line are also challenged by frequent power outage and voltage spikes.
- Harsh environment. In rural sites, temperature is considerably higher than recommanded operational level, especially at those sites where equipments boxes are mounted outdoor.

1.4 Approach

Special requirements need to be treated differently with innovative approaches. In SBN development project, we apply iterative approaches to test new solutions. Different components are gradually replaced with newly developed

technologies. we carefully select off-the-shell hardware and open source software to reduce the cost and enhance rebustness. To attack the problem of power supply, we run our equipments over heterogeneous power source, including sink device such as battery or super capacitor. By reducing the power consumption, we minimize discharge cycles and prolong battery life.

1.5 outline

Remaining content is organized as following: Chapter 2 provides an overview of previous projects of SBN. It then explains current network topology and administration; Chapter 3 demostrate the design of low power router. We discuss about the testing result in this chapter as well. In chapter 4, we draw our conclusion and propose improvements that could be done in the future.

Chapter 2

Network Administration

In rural ICT development, technologies should be affordable and easy to use.

2.1 An Overview of SBN

Technology selection in rural ICT development is highly affected by physical environment and the demand of services. Conditions are often different from one site to another, hence not replicable in its entirety. As introduced in section **??**, Macha project represents a typical setting of rural ICT envrionment, while Nipel project serves as a good example of distribution of network. In this section, Four communication technologies are presented and evaluated against SBN environment. **Optical fiber links** are always desired due to its capacity and durability. Although deployment and maintainence require special tools and skills, and civil work involved is immense. Building a fiber line in rural area demands innovative cooperation to distibute risk and cost, as well as sharing the benefit. In the case of SBN development, during the time that Tanzania set up the power line between two districts, a 140km optical fiber link was also established along the power transmission line and owned by Tanzanian power company, TANESCO[1]. The fiber was donated to ICT4RD in exchange for network connection. To distribute fiber backbone network, Low power routers that support both optical fiber and copper links are developed. More details can be found in section **?? Terrestrial Wireless** is an optimum approach in last-mile delivery. It is easy to deploy and highly customizable to adapt to different landform. Among various wireless technologies, IEEE 802.11 family (more commonly known as WiFi) running in license-free 2.4GHz or 5GHz offers satisfactory bandwidth while eliminating the cost of frequency registration **Very Small Aperture Terminals Delay Tolerent Network**

[1] http://www.tanesco.co.tz/

2.2 Network Topology and Administration

2.3 Key Challenges

Chapter 3

Toward Low Power

3.1 System Design

3.2 Deployment and Testing

Chapter 4

Conclusion and Future Work

4.1 Conclusion

4.2 Future Work

Part II

Highly Available Web Services

Chapter 5

Introduction

In this chapter, we introduce the motivation to build a highly available web service in rural area. We illustrate the result of observation and identify the key problems.

In chapter 6, we start by formulating the problem into a multi-master system, and then investigate several available technologies that address this challenge. We examine them against our unique requirement and select one as our basis. We then propose and explain our approach on top of it.

In chapter 7, we illustrate the idea of using ARM-based hardware to achieve low-power consumption web service. To study the capacity of hardware, we benchmark the performance of each components of web service on different platform. According to the result, we propose a cluster than can be easily scaled out to serve more users.

In chapter 8, we draw our conclusion and shed light on future work.

5.1 Background

Affordable, yet stable web services are highly desired in rural area, as a mean to alleviate digital divide and improve life quality. When it comes to under-developed regions in Africa, requirements and conditions need to be carefully assessed and analyzed, for that challenges could be unique and dramatically different than metropolitan.

5.2 E-learning for Open University of Tanzania

Open University of Tanzania (OUT) [1] is the first university of East Africa Region to provide open and distance learning programmes. To distribute course content through the whole country, Moodle has been chosen as underlying digital resource management platform. Moodle[2] is an open-source industrial-level online learning platform and resource management system. As a typical data-driven web service, Moodle runs over an underlying database

and assemble its webpages on-the-fly based on user requests. It is written in PHP and heavily tested against Apache, Nginx and MySQL. At present, the platform is running as a standalone web service in a central server and mainly serve static content such as PDF, Text and Slides. Although OUT has the vision to introduce multi-media materials to enhance education quality. OUT also establishes learning centers in major cities and towns all over the whole country and is ambitious to extend to a larger scale. An emerging obstacle is to provide services in remote areas with poor network connection and bandwidth.

5.3 Problem Identification

As part of the project, we investigated local conditions and needs within the scale of Serengeti Broadband Network, especially in areas with evident demands of services and lack of infrastructures. And we were able to identify following challenges:

5.3.1 Power Outages

Power grid in rural areas of Tanzania is so unreliable that UPS for critical device is almost a must. While people are gradually adapting to mobile platforms, such as smart phones and tablets, backbone infrastructures are also required to be more persistent. Equipments powered up by solar and battery are highly desired due to cost-efficiency. Although power consumption need to be optimized in this circumstance in order to prolong battery life and improve reliability.

5.3.2 Poor network quality and frequent failure

Although local network is operated by ICT4RD project and can be fairly reliable, uplink is still depending on national-wide ISP and is somewhat unpredictable according to our observation. Network failure could occur anytime and can last for random period (several minutes to several days). Those web services that depend on a central server are apparently not accessible during the failure. On the other hand, the uplink can be very narrow due to poor infrastructure and limited budget. It could be difficult to squeeze multimedia services into such bandwidth.

To better illustrate this problem, suppose a typical setting in Figure 5.1. Major backbone components in this LAN are interconnected through fiber-optical lines, and network is distributed to users through WiFi or Ethernet. The LAN is linked to the Internet through ISP distribution link and central server resides on the otherside of the Internet.

Due to limited budget and ISP capacity, the upper link is equipped with an average bandwidth of 2 4Mbps which is shared among all users in

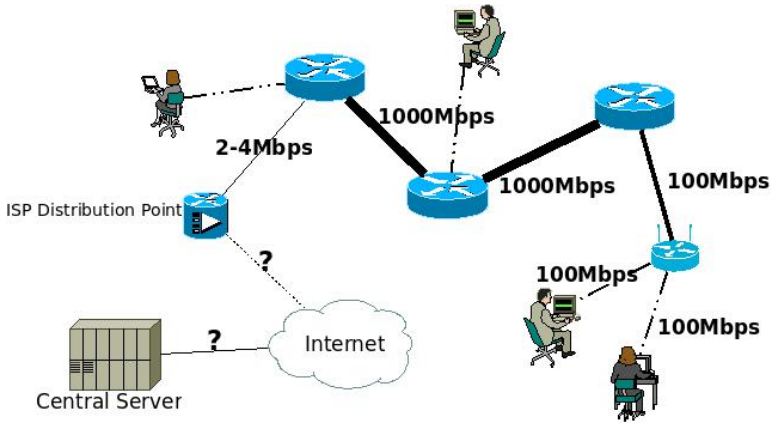


Figure 5.1: A Typical Setting of Rural Local Access Network (LAN)

LAN. While a minimum bandwidth of 1.5Mbps is recommended for video streaming, it is difficult for users to get decent service from central server. To worsen the situation, the upper link is somewhat unpreditable, which leads to the isolation of LAN, as shown in 5.2.

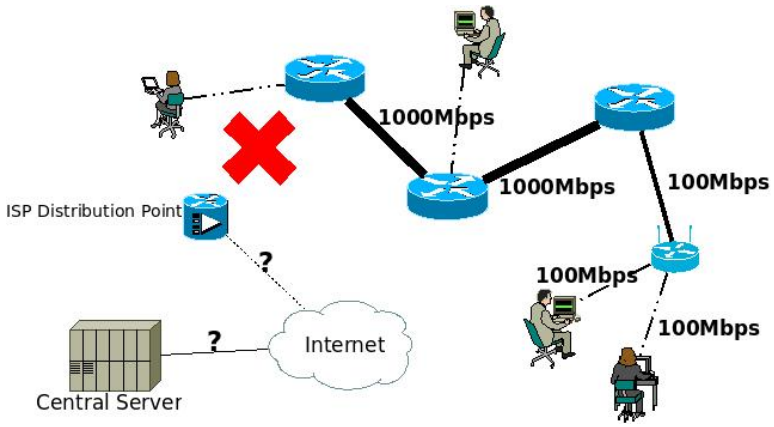


Figure 5.2: Uplink Failure leads to the isolation of LAN

On the other hand, components in the LAN can also break down which leads to network separation, see Figure 5.3. In the first case, multi-media content can hardly reach end users. And in other two cases, users cannot get service at all.

5.3.3 Limited budget

Cost is an essential factor during rural ICT development. Given relatively smaller user base and weaker demand, equipments need to be chosen wisely. Although future maintainence and development also need to be considered.

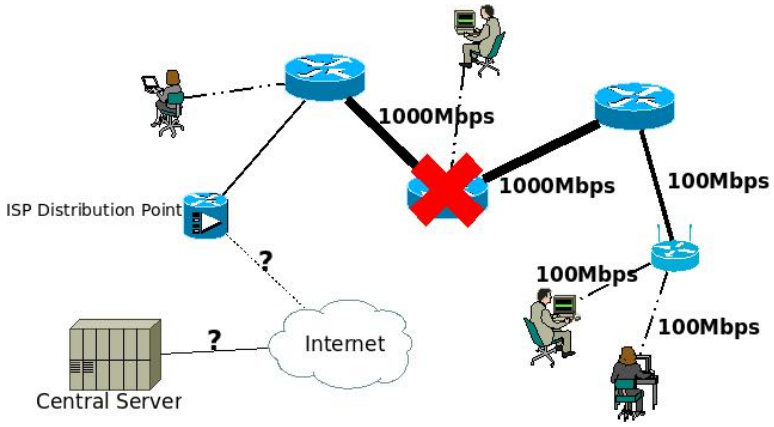


Figure 5.3: Network Separation due to Component Failure

Chapter 6

Adapt to Frequent Network Failure and Limited Bandwidth

In this chapter, problems are further decomposed and analyzed. To prevent reinventing-the-wheel, a variety of possible solutions are proposed and investigated, whereas focus has been put into our unique requirements.

6.1 A closer look at the problem

As introduced in section 5.2, Moodle is deployed as underlying course management system for OUT E-learning platform. Moodle is an open sourse project written in PHP and well-documented[3][4]. Similiar to other web applications, it can be deployed in a typical LAMP or LNMP stack. In this chapter, we mainly focus on possible solutions for two problems stated previously, and leave the choice of actual server to chapter 7

Moodle is a typical database-driven web application where all the pages are generated on-the-fly based on user request. The whole application is composed of three main components:

- PHP source code, typically in `/var/www/moodle/`
- A database to store data or metadata including site configuration, student information, course details, events, etc. There exist volatile tables in Moodle database which store sessions and temparory information.
- A directory to store materials and resources, as well as cache and temparory files. Typically it is named as `moodledata/`

The problem addressed previously can be simplified and modelised as following, see Figure 6.1. Each node in the model denotes a local server/proxy and has a certain amount of users associated with it.

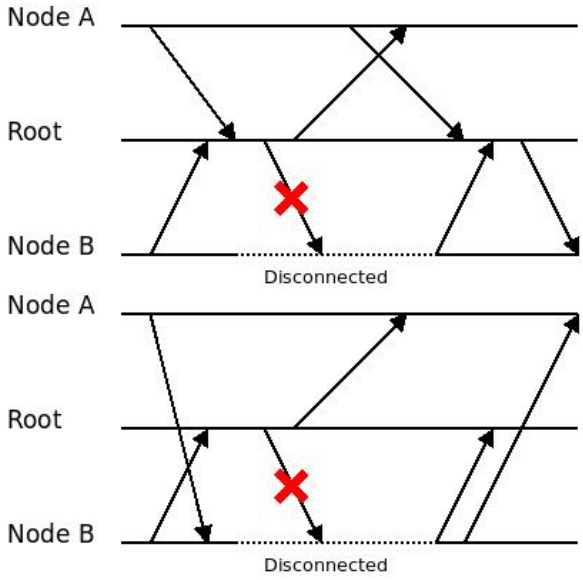


Figure 6.1: Web Delivery Model

As simple as the model might be, components in it could be vastly heterogenous if mapped to different techniques. Content stored in a node can either be web objects, SQL replies, codes or even entire databases. Communication in between can also be based on a variety of protocols.

As an online learning platform, users do not only passivly accept information, but also interact with Moodle through forum, personal blogs and quiz. All the changes made by users must be stored and seen everywhere. Thus, the system should not be read-only under any circumstance.

Moodle has been in service and adding new services should affect existing structure as less as possible. Also, steps of adapting changes should be properly designed to avoid crushing the service.

To maintain consistency and serve up-to-date content, a reasonable amount of communication overhead is necessary and is normally positive proportional to the extent of consistency. Although, due to the presumption of poor network connection and narrow bandwidth, different nodes in the system are preferably decoupled and autonomous.

The autonomy is also closely correlated to the ability of performing offline operation. Many distributed systems have the ability to detect and recover from network partitioning, although it normally leads to a compromise of consistensy and content freshness. When a user request a page, Moodle loads all previliges of the user, generate pages accordingly and log the session. This results in uncacheble content and interaction-must logins. It has been proven that consistency, high availability and partition tolerence are impossible to be achieved at same time[5][6], necessary trade-off has to be made according

to the condition and needs.

While the majority of web caching and content distribution techniques aim at better performance and delivery efficiency, we prioritize the ability of performing basic functionalities during network failure. We tolerate a relatively loose consistency while ensuring eventual convergence.

Lastly, to realize affordability, we mainly focus on open source techniques and free ware. Thankfully, many successful projects and tools have been made open source and publicly available. In the following sections of this chapter, we evaluate a variety of techniques against the criteria stated above and propose our solution based upon the conclusion. Several of potential solution are also tried out.

6.2 Push Web Service to Edge

6.2.1 Content Delivery Network

Content Delivery Network overlaps with Web Cache Proxy at the concept of pushing web content to users. A Content Delivery Network is a collaborative set of surrogate servers spanning the network, where web contents are mirrored[7]. Users will perceive a smaller latency while fetching content from a nearby CDN surrogate server rather than original web server. The essence of CDN is illustrated in Figure 6.2.

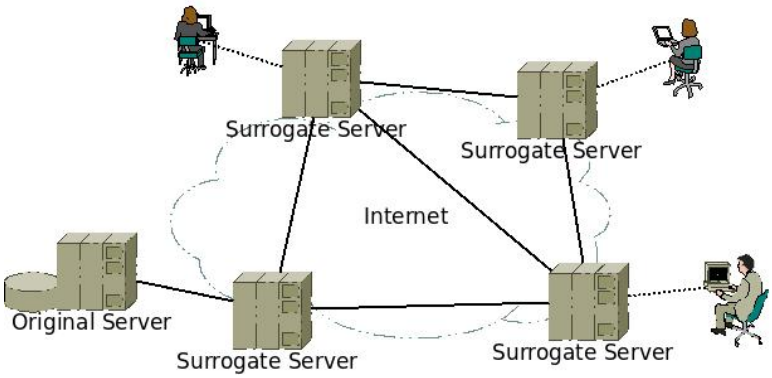


Figure 6.2: Content Distribution Network

Since more and more web services are evolving to provide dynamic content, CDN also takes advantages of cachebility hints when dealing with dynamic contents[8].

6.2.2 Simple Web Caching

An intuitive and common solution for the problem of limited bandwidth is to cache popular web content locally, as illustrated in Figure 6.3. A client-side

web cache proxy is typically deployed in user local network, requesting web servers on behalf of users, and cache web objects for further references. Web caching has been proven to be an effective approach to reduce bandwidth usage, user-perceived latency and loads on original server[9].

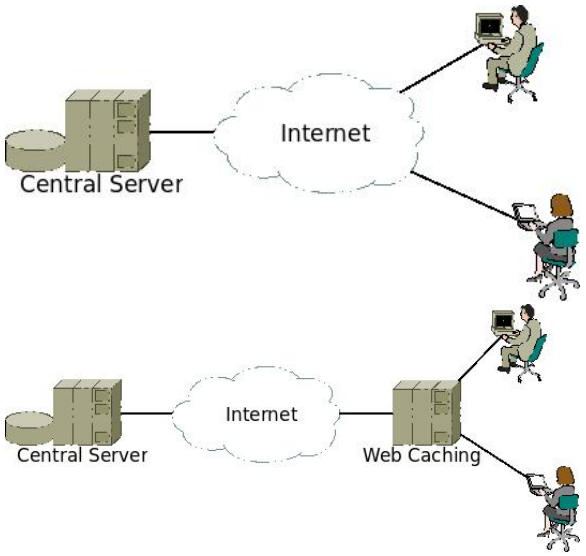


Figure 6.3: Serve users without and with a Gateway Cache

Web cache is greatly advantageous in our scenario that it does not require modification on web application, except that some TCP optimizations could be done between web cache proxy and web server frontend. Web cache proxy can continue serving requests with offline mode enabled, which also meets our requirements. The traffic between cache proxy and authentic server is standard HTTP request and response. The overhead and frequency of communication mainly depends on cache hit / cache miss ratio, expiration time and cache directory size.

Although, this solution encounters two major constraints in our case:

- Moodle pages are generated on-the-fly based on user information. Web cache cannot operate on its own during network failures, except for serving previously requested pages.
- Users are required to log in to browse. Thus, user-server interactions are always necessary, which also contradict with our purposes.

6.2.3 Page generation on Edge Server

To address the issue of dynamic content generation and client-server interaction, an intuitive and brute-force solution is to generate user-specific page at the edge. A comprehensive study of edge servers can be found in [7].

Four strategies are presented: (a) edge computing (b) content-aware caching (CAC) (c) content-blind caching (CBC) (d) data replication. In each of the strategy, the edge server attempts to reply user request on the behalf of original server with the information that is locally available. Edge computing still heavily relies on central database, hence out of our consideration. While CAC and CBC store partial database at the edge, data replication stores a complete copy of the database.

If the size and complexity of database permit, data replication is desired since it outperforms other strategies in both response speed and offline operation. Although, it adds another layer of complexity to perform transaction processing and maintain the consistency through mutliple distributed databases. Techniques to achieve a consistent distributed database system are presented in section 6.3.

Application code rarely changes in our case and is always one-way synchronized from original server, thus consistency can be relatively easy to achieve by periodically utilizing tools like rsync[10].

6.3 Multi-Master Database Synchronization

As addressed previously, consistency, availability and partitioned-network cannot be accomplished at the same time, according to CAP theorem**??**. Necessary trade-off has to be made to adapt to particular circumstance. Based on objectives defined previously, we prioritize availability and partitioned-network, while allowing loose consistency, as long as eventual consistency is guaranteed**??**. Pessimistic replication always guarantees consistent content perceived by users, thus widely adopted in distributed database system to enhance availability and performance. Although optimistic replication permits diverged databases, which is more suitable in our case. An exhaustive survey on optimistic replication is presented in **??**.

Comprehensive theoretical researches have been made available although we find limited open source implementations that serve our purposes. We investigate several data replication technologies in this section based on following criteria:

- **Deployable over WAN** Most of distributed database system assumes a LAN environment, where delay and bandwidth do not evidently affect exchange of data among servers. Although in our case, databases are deployed over WAN, which is highly unreliable and network latency is not neglectable.
- **Serializibility** To achieve eventual consistency, concurrent changes made at different edge servers need to be serialized. Hence, an order of operations is extracted based on relations among them.
- **Conflict detection and reconsiliation**

Several data replication technologies are investigated in this section in order to find a tool that requires minimum modifications to fit into our problem.

Distributed database system is widely adopted in server clusters and workstations, as a mean to enhance performance and redundancy. Although LAN is normally assumed by most of database replication technologies.

6.3.1 Database Cluster

CouchDB[12] is an open source distributed database system developed in Apache. The most attractive feature of CouchDB is that it natively supports bi-directional synchronization among multiple database replicas and offline operations. When one replica is disconnected from the network, it retains autonomy and continues as a fully functional database from user point of view. Although, it fails to be our candidate since it is NoSQL database, whereas Moodle heavily relies on SQL calls and it will a significant task to modify Moodle to use NoSQL database.

MySQL Native Replication is shipped with most of MySQL standard distributions that provide built-in functionalities to replicate databases. The synchronization is uni-directional that databases on master node are replicated to slave nodes. ALthough bidirectional synchronization can be achieved by creating loop in the topology. A simplest 2-nodes example can be found in Figure X. Node A and node B synchronize with each other by maintaining mutual dominance. The figure also shows a more complicated topology where three nodes comprise a loop. Although, the synchronization can be easily broken by conflict and the reconciliations always require human intervention. To avoid insertion conflicts of auto-incremental keys, MySQL provides an option to increase incremental step. This feature is further discussed in section 6.3.4

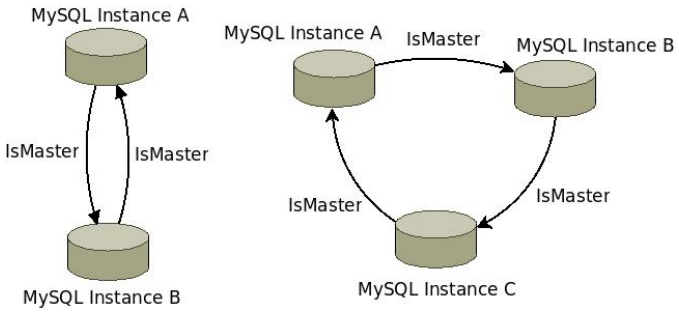


Figure 6.4: MySQL Replication

MySQL Cluster[13] is an open source distributed database system based on MySQL. It supports database sharding and duplicating. A typical use case of MySQL Cluster is shown in Figure 6.5. Although, redundent copy

of database can only be accessed with the presence of management master, and cannot be updated during network failures. Furthermore, database nodes are closely coupled with the assumption of LAN (low latency and high bandwidth).

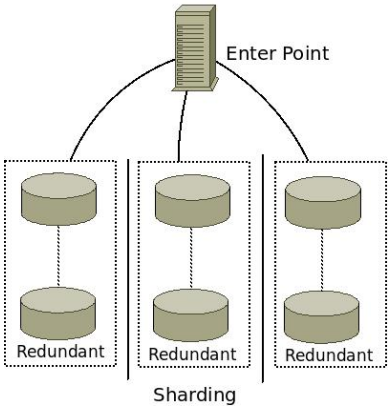


Figure 6.5: MySQL Cluster

6.3.2 Middleware-based Repliation System

Several studies also proposed the approaches to solve database synchronization at middleware-level[14][15][16]. C-JDBC [14] is an Java implementation of RAIDb[17], aiming at a framework to manage heterogenous databases. With built-in functionality of scheduling transaction processes, C-JDBC is perceived by users as a single virtual database. Although, the system is still centrally managed and could not handle partitioned network. Ganymed middleware system[16], inspired by C-JDBC, achieves consistency by serializing update/write requests at master and propogating changes to replicas in a lazy fashion. Users see a consistent data state (snapshot isolation), even though stale might it be. The limitation of these two middleware is also clear, that no write can be served during network failures.

6.3.3 Multi-Master Sycnhronization System

Similiar to MySQL Cluster Multi-Master setup, we found three state-of-the-art open source tools to the similiar end.

- **Galera**[18] is an open source synchronous database replication software developed to scale web application and provision high availability. It achieves consistency by optimistic locks and group communication. Galera is quorum-based and handles network partitioning by sacrificing the minority. Hence, Galera lacks the essencial features that we are seeking for and is out of consideration.

- **Tungsten**[19] replicates database asynchronously and allows loose consistency. Transactions are commited locally, and then propogated across all other nodes. Tungsten features offline operation and automatical recovery although it leave the responsibility of conflict avoidance completely to the application. Conflicts may disturbe replication and hard to trace.

- **SymmetricDS**[20] is another open source asynchronous database replication management tool. It has several built-in rules to detect and reconsile conflicts. It can be configured flexibly to meet different needs, for example, have different courses store in different sites. Also, one appealing feature is that SymmetricDS support file synchronization as well. Figure 6.6 birefly illustrates how SymmetricDS works and more details in terms of conflicts detection and reconsiliation are discussed in section 6.3.4

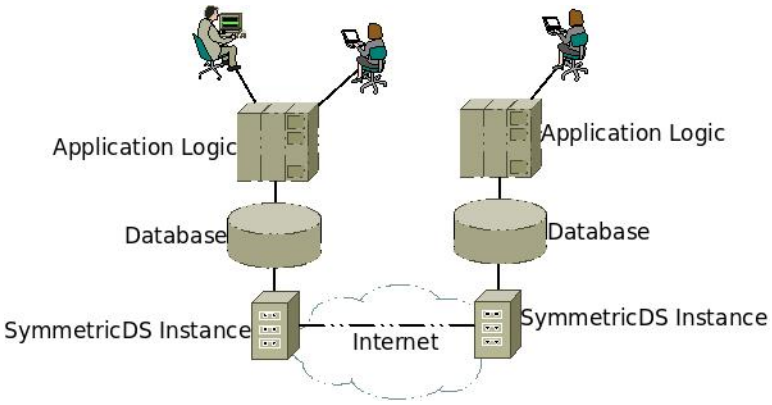


Figure 6.6: SymmetricDS

We can conclude from above comparison that SymmetricDS is closest to our critaria and can be properly extended to meet our requirements. In the following section, we explore SymmetricDS in details and propose an approach to extend it.

6.3.4 Conflicts Avoidance and Reconciliation

Before we start illustrating our approaches for conflict detection and resolution, several reasonable application-specific assumptions need to be made:

- No user will attempt to login into two different server simutaneously, thus the content belong to this user will not be modified at two different sites.

- Administration-related content is always synchronized through top-down approach, that is, user creation, deletion and modification are always synchronized in one-way.
- No local user will be able to reply to a post that is not propogated to this edge server yet.
- Volatile entries are not synchronized, such as cache and session.
- Deletion wins. And events that causally follow the deletion also get deleted. If users reply to a post on a remote server whereas the post gets deleted on central server, all replies will get deleted once converged.

As introduced in section 6.3.1, insertion conflicts of primary keys can be avoided by setting different incremental steps, as long as the primary key is auto-incremented integer, see Table 6.1.

Primary Key	Node A	Node B	Node C
1	inseted by A		
2		inserted by B	
3			inserted by C
4	inserted by A		
5		inserted by B	

Table 6.1: MySQL instances with different auto-incremental steps

Thankfully, all Moodle database tables are designed to use auto-incremental keys as primary key. By configuring incremental step and SymmetricDS, we are able to achieve a naive mutually consistent system. Configuring SymmetricDS is nontrivial and requires constant observation. Appendix A describes a work flow of configuring SymmetricDS.

Although, there are several major drawbacks of this design:

- It is not scalable. The incremental step limits the max number of servers in the system.
- It is blind to application. It does not check logical correctness of modification according to application, e.g. replies to a previously deleted thread can still be inserted into database.

To attack these problems, we examine the way SymmetricDS handles conflicts and propose an extension that is more flexible and tunable. SymmetricDS applies the concept of CVS (Concurrent Versions System) to detect conflicts at a granularity of table entry level. When a node propogates a change on one table entry, it sends the before-state of that entry along with the change. Upon receipt, remote node compares the current state of this

entry and the history. If they differ, a conflict is detected. Even though conflict resolution is highly application-specific, SymmetricDS provides limited rules to resolve known types of conflicts.

A SymmetricDS node can be configured to either actively pull from other nodes, or passively waiting for changes being pushed. While optimistic replications normally assume a negotiation phase while attempting a resolution, push and pull in SymmetricDS are independent from each other. Thus, conflicts resolution can be sometime confusing. For example, suppose a scenario in Figure 6.7. Slave node is configured to push data to master while pulling changes from it. Since two operations are independent from each other, two replies to the same post are replaced by each other at two nodes, whereas the intention is to simply stack them.

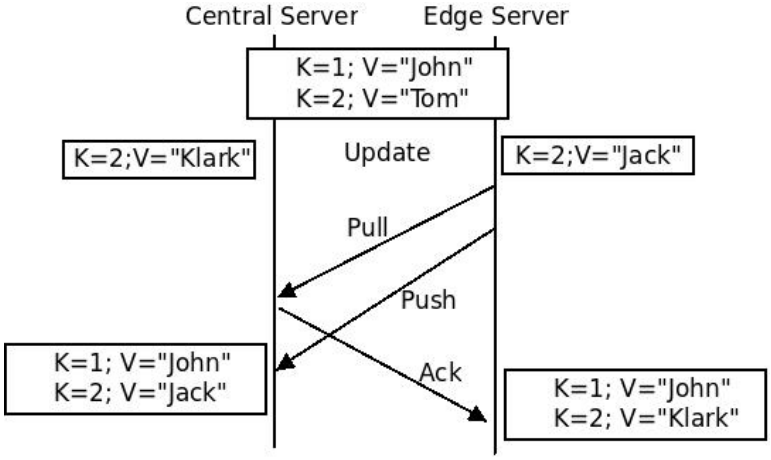


Figure 6.7: Misbahavior of SymmetricDS

To attain serializibility, several repliation systems applies a centralized algorithm that serialize all changes at master node and then propogate to slaves. Although, it often requires a closely connected system and can suffer from latency and disconnection. Inspired by Git and Operational Transformation [21], we propose a distributed work flow in Figure X where conflicts are resolved at edge server, similiar to the mechanism used in Dropbox[22]. Edge node always pull the master first before commiting any changes.

Chapter 7

Low Power, yet Powerful

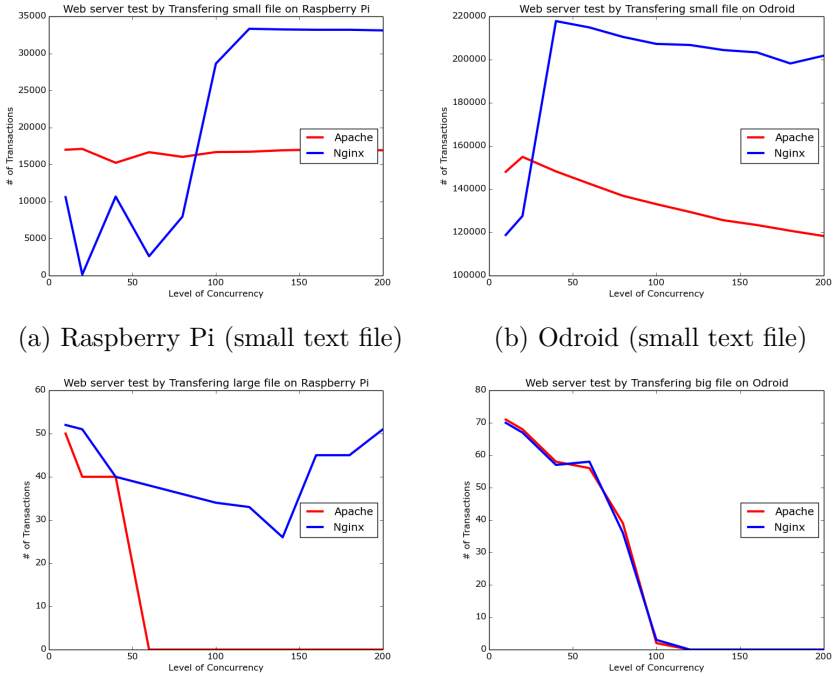
The ARM-based processors have received great attention for its characteristic of low power consumption and energy efficiency, especially in smart phone and portable device industry, where power consumption is one the most critical specifications. Furthermore, there is trend in server industry to shift to ARM-based architecture in order to cut off the bill of electricity. ARM is also ambitious in this area and about to publicize processors capable of virtualization. When it comes to rural development, power shortage has been forcing researchers and engineers to seek for alternative power sources. Lower power consumption of the equipments implies a bigger potential of surviving severe environment. Currently, we are exploring the possibilities of two platforms, namely Raspberry Pi and Odroid. The specifications of these two platform can be found in Table X In the following sections, we benchmark a variety of attributes of Odroid and Raspberry Pi. The objective is to clarify the capacity of these two platforms and an optimum form to run the web service. We test every component of a complete Moodle installation to identify the bottleneck of the application. Based on the results, we propose the optimum cluster solution that can be easily scaled out to serve more users.

7.1 Benchmark of Web Components

To test components of a complete Moodle installation, we design a expirement in Figure X. Each part is respectively substituted with either Odroid or Raspberry Pi, and remaining parts are running on a high-end machine which is much more powerful than these two platforms, in order to put enough siege on the testing target. Furthermore, simulated requests are generated from high-end machine as well.

7.1.1 Web Frontend

Most of the websites today are powered by Apache due to its long history and abundant extensions. Although Apache relies on a processed-based manner to handle new connection, which has limited the scalability and concurrency. Nginx is an event-based reverse proxy that handles request asynchronizily. It addresses C10K problem from the beginning and focus on scalability. To determine which server runs better on a resource constraint platform, we benchmark Nginx and Apache on both platform to evaluate the throughput, level of concurrency and response delay. We use siege to generate workload and compare the performance of web server with two different type of object: small text file and large JPG file. We siege the server in following setting:



(a) Raspberry Pi (small text file)

(b) Odroid (small text file)

(c) Raspberry Pi (big jpg file)

(d) Odroid (big jpg file)

Figure 7.1: A Benchmark of static file request on both platforms

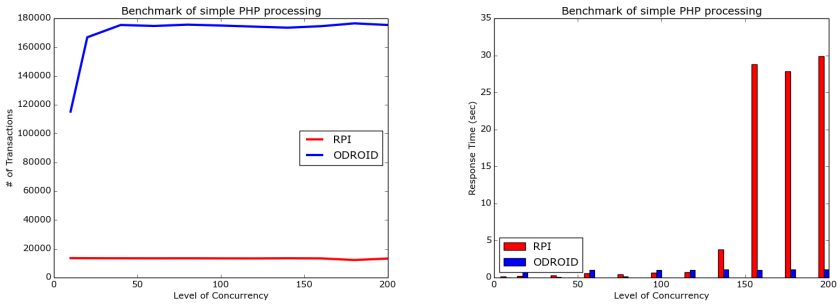
As shown if Figure 7.1, Nginx and Apache achieve comparable performance under a low level of concurrency, although Nginx outperforms Apache notebly when concurrency level increases. To be noticed, both web servers perform indistinguishably while serving large file. We observe fully occupied CPU in this specific test, which is due to intensive OS kernel processing of socket manipulation and packet tranfer. The impact of web server on the CPU is neglectable in this circumstance. Thus, expirement result in Figure

7.1(d) does not denote same performance of web servers.

7.1.2 PHP Processing

As a large PHP application, Moodle requires significant computational resources for PHP processing. Thus, a testbed is formed to benchmark PHP processing capacity of two platforms, see Figure X The inputs are two different PHP scripts:

- a php script simply echoes `Hello, world!`
- Moodle index page which requires heavy php processing.



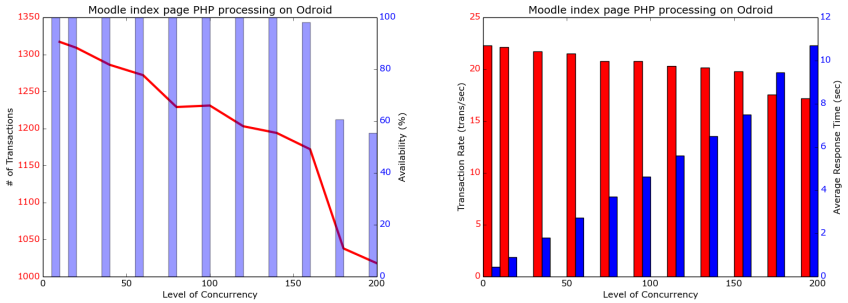
(a) $\sum trans$ (b) Longest Response Time

Figure 7.2: A Benchmark of simple PHP processing on both platforms

Figure 7.2(a) shows that Odroid can handle much more PHP request than Raspberry Pi. Overall response time is shorter for Odroid. Although, half of the CPU usage is taken by OS kernel during these two test and the difference satisfactorily convincing. The gap between two platforms is more evident when tested with heavier PHP processing, as shown in Figure 7.3.[1]

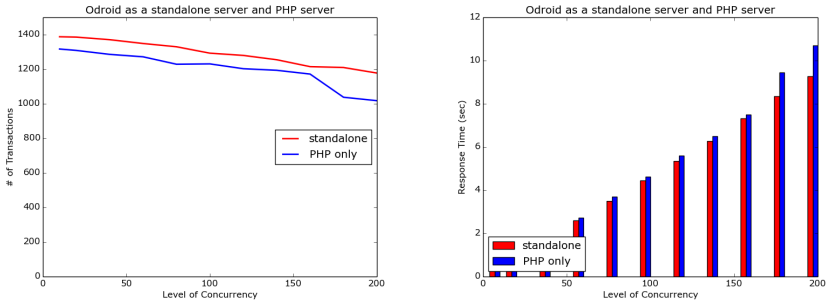
As shown in Figure7.3(a), transaction rate and availability drops under 180 concurrent requests occur. The latency is beyond tolerable under 50 concurrent requests, according to a study of tolerable waiting time of website**??**. As the total transaction number in this test is significantly lower than the previous one in Figure 7.2, we suspect that PHP processing power is the bottleneck in a moodle installation rather than database query and web frontend. Thus, we test the capacity of a standalone installation of Moodle, in which all components are in one box (Odroid), shown in Figure 7.4.

[1]We observe a 6 7 seconds delay to process index page of Moodle on Raspberry Pi and it can barely tolerate the mildest siege. Thus, the test result of Raspberry Pi is left out in this test.



(a) $\sum trans$ & Availability (b) $Rate_{trans}$ & T_{resp}

Figure 7.3: Moodle index processing on Odroid



(a) $\sum trans$ & Availability (b) T_{resp}

Figure 7.4: Odroid as a standalone server and PHP server

Impact of adding web frontend does not evidently influence transaction rate, which implies that PHP processing capacity is the limiting factor in one Moodle installation.

7.1.3 Database

We apply the same method to test database although fail to exhaust the resource on Odroid before running out CPU and memory on our testbed high-end machine, hence we conclud that database query does not limit a Moodle installation before expanding PHP capacity.

7.2 Scale Out to Eliminate Bottleneck

To benefit most from currently available hardwares, we propose to form a small scale cluster than can be easily scaled out to cope with an increase of users. Furthermore, we investigate multiple different formations to find out the most economical setting. An intuitive solution is to put different

services in physically separated hardwares, as a structure shown in Figure 7.5. Nginx server in Rapsberry Pi features both web server and load balancer with upstream PHP servers running in Odroid. To cope with the increase of users, this setting can be easily scaled out by simply adding more hardware for PHP processing. Although, as we compare the Siege test result with a

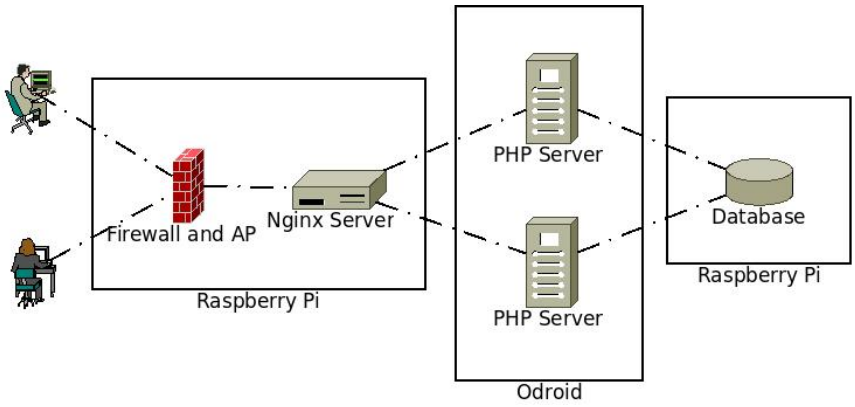


Figure 7.5: A naive form of cluster

standalone installation where all services run in one Odroid, we encounter a slightly lower performance, which is even worsen after we add one more PHP server, see Figure 7.6.

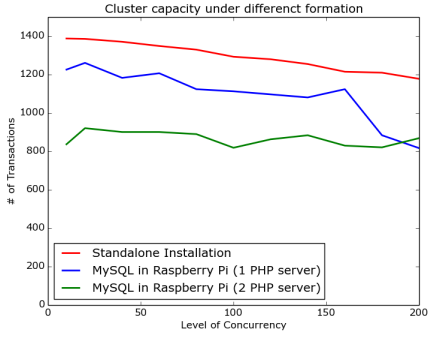
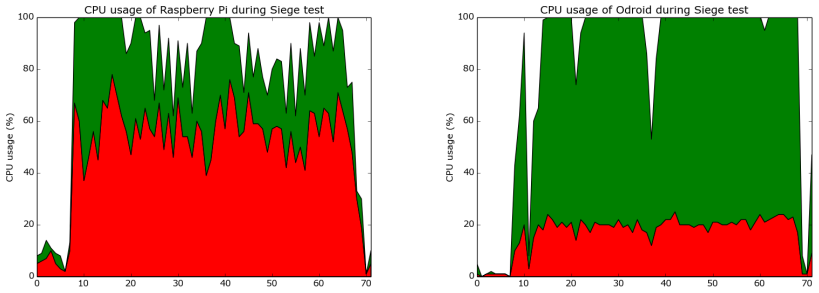


Figure 7.6: The Performance of running Database in Raspberry Pi

To further identify the cause, we siege the cluster with 100 simulated concurrent clietns and observe CPU usage of servers (CPU of Nginx server is far from full load hence left out in the result), the result is shown in Figure 7.7. We deduce that most of CPU resource in database server is consumed by OS kernel to handle concurrent sockets. Hence, database needs to be more closely coupled with PHP server, which results in the cluster formation in Figure 7.8. And the result meets our assumption that server capacity improves by adding extra PHP server, see Figure 7.9



(a) database (b) PHP

Figure 7.7: CPU usage of servers

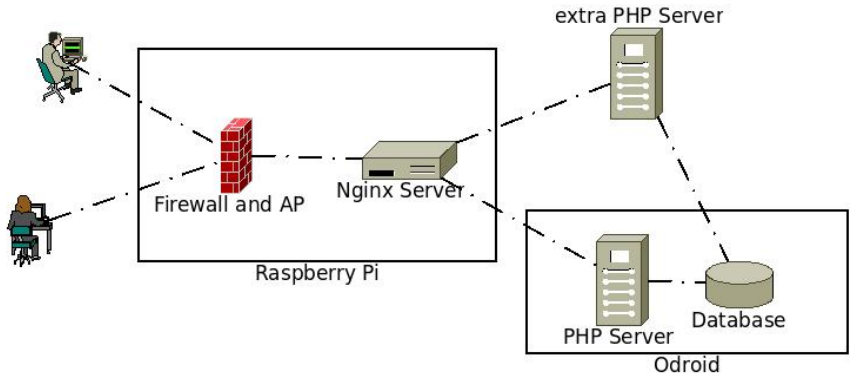


Figure 7.8: A better form of cluster

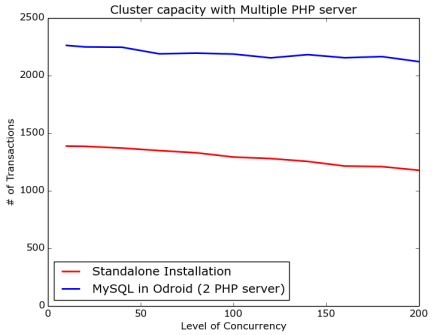


Figure 7.9: Performance of cluster with multiple PHP servers

To further boost web performance, we configure Nginx to cache frequently requested page hence reducing the load on PHP and database server.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

Starting from the user case, by applying the concept of divide-and-conquer, we are able to formulate the problem and identify the key challenges. Given the impossibility of achieving consistency and availability at the same time, we propose a multi-master system with reasonable assumptions and compromises. We surveyed a variety of existing technologies and investigate the potential to map them into our case. With extensions and proper configurations, SymmetricDS is tuned to serve our purpose. To achieve low-power consumption and affordability, we test the capacity and proposed a cluster which can be easily scaled out. We build Moodle web service on the cluster with integrated synchronization functionalities. We conclude that this prototype has the potential to serve in rural areas where highly available web services are desired and number of users is limited.

8.2 Future Work

At the beginning, the plan was to implement the system and test it in production environment. Although enormous effort has been put into exploring the possibilities of a variety of technologies. Also the overhead to adapt to local work culture unexpectedly disturbs original plan. Thus, the next step will be testing and debugging the system in a real production service. Also, the system can be generalized to serve other database-driven applications although proper configuration and modification are required. This potential should be further investigated.

Bibliography

[1] http://www.out.ac.tz/. [Online]. Available: http://www.out.ac.tz/

[2] https://moodle.org/. [Online]. Available: https://moodle.org/

[3] http://www.aosabook.org/en/moodle.html. [Online]. Available: http://www.aosabook.org/en/moodle.html

[4] https://docs.moodle.org/. [Online]. Available: https://docs.moodle.org

[5] E. A. Brewer, "Towards robust distributed systems," in *PODC*, 2000, p. 7.

[6] S. Gilbert and N. Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," *ACM SIGACT News*, vol. 33, no. 2, pp. 51–59, 2002.

[7] M. Pathan, R. Buyya, and A. Vakali, "Content delivery networks: State of the art, insights, and imperatives," in *Content Delivery Networks*. Springer, 2008, pp. 3–32.

[8] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *Internet Computing, IEEE*, vol. 6, no. 5, pp. 50–58, 2002.

[9] B. D. Davison, "A web caching primer," *Internet Computing, IEEE*, vol. 5, no. 4, pp. 38–45, 2001.

[10] A. Tridgell, *Efficient algorithms for sorting and synchronization*. Australian National University Canberra, 1999.

[11] W. Vogels, "Eventually consistent," *Communications of the ACM*, vol. 52, no. 1, pp. 40–44, 2009.

[12] http://couchdb.apache.org/. [Online]. Available: http://couchdb.apache.org/

[13] http://www.mysql.com/products/cluster/. [Online]. Available: http://www.mysql.com/products/cluster/

[14] E. Cecchet, "C-jdbc: a middleware framework for database clustering."

[15] C. Amza, A. L. Cox, and W. Zwaenepoel, "Conflict-aware scheduling for dynamic content applications."

[16] C. Plattner and G. Alonso, "Ganymed: Scalable replication for transactional web applications," in *Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. Springer-Verlag New York, Inc., 2004, pp. 155–174.

[17] E. Cecchet, "Raidb: Redundant array of inexpensive databases," in *Parallel and Distributed Processing and Applications*. Springer, 2005, pp. 115–125.

[18] http://galeracluster.com/. [Online]. Available: http://galeracluster.com/

[19] http://www.continuent.com/solutions/clustering. [Online]. Available: http://www.continuent.com/solutions/clustering

[20] http://www.symmetricds.org/. [Online]. Available: http://www.symmetricds.org/

[21] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *ACM SIGMOD Record*, vol. 18, no. 2. ACM, 1989, pp. 399–407.

[22] "Dropbox conflicts resolution," https://www.dropbox.com/developers/blog/48/how-the-datastore-api-handles-conflicts-part-1-basics-of-offline-conflict-handling.