Implementation and Performance Evaluation of A CMIS Server for Open Source PHP Based WCMS

Yan Zhang

A Dissertation submitted to the University of Dublin, Trinity College

in fulfillment of the requirements for the degree of

Master of Computer Science in Computer Science

(Networks and Distributed Systems)

August 2013

Declaration

Me, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

Yan Zhang

Dated: August 29, 2013

Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this Dissertation upon request.

Yan Zhang

Dated: August 29, 2013

Acknowledgements

I would like to thank my supervisor David Lewis for his guidance and help on this disseration. And I am also grateful to Leroy Finn for his suggestions.Thanks to all the MSc programme staff for their assistance in the past one year. Finally, I would like to express my thanks to all my classmates and friends for their encouragement and help.

Yan Zhang

University of Dublin, Trinity College

August 2013

Abstract

Nowadays, most organizations use content management systems (CMS) to manage large volume of content which are created in their daily work. For many large organizations and enterprises, different CMS might be used for various reasons. Although the demand of exchanging content crosses different CMS is imperative, the interoperability is difficult to realize due to proprietary APIs. The Content Management Interoperability Service is an open standard which defines a domain model and some services to improve the interoperability for CMS. Despite many enterprise content management systems have now implemented CMIS, it has received relatively less attention in the important class open source web content management system (WCMS). This research aims to find how the CMIS standard can be implemented to support open source PHP based WCMS. To achieve the expected goal, a CMIS server is implemented for WordPress and an evaluation tool is implemented to investigate the feasibility and performance issues of the server implementation compared to a Java ECM implementation. By using the server implementation, organizations are able to get content from the WordPress and execute query operations. Although the performance of the server implementation needs to be improved, the server implementation does not have a significant detrimental impact on the computer's performance. Furthermore, the server implementation provides a potential way for CMIS to support other WCMS.

Contents

Acknowledgements **iv**

Abstract **iv**

List of Figures **vii**

Chapter 1 Introduction **1**

1.1 Background 1

1.2 Motivation 2

1.3 Subject 3

1.4 Approach 3

1.5 Outline 4

Chapter 2 State of Art **5**

2.1 Web Content Management System 5

2.2 ECMS 8

2.3 Content Management Interoperability Service 9

2.3.1 CMIS Specification 12

2.3.2 Current CMIS Implementation 14

2.3.3 Use Cases 15

2.3.4 Industry Response 16
2.4 Related Standards 17
2.4.1 Web Distributed Authoring and Versioning 17
2.4.2 Content Repository API for Java 17
2.4.3 PHP Content Repository 18
2.4.4 Comparison with CMIS 18
2.5 Summary 19

Chapter 3 Design **21**
3.1 Stakeholders 21
3.2 Repository Choice 22
3.3 Requirements 23
3.3.1 Functional Requirements 23
3.3.2 Non-functional Requirements 25
3.4 Design 25
3.5 Summary 28

Chapter 4 Implementation **30**
4.1 Plug-in CMIS Server for WordPress 30
4.2 CMIS Server for WordPress 31
4.2.1 Architecture 31
4.2.2 Data Model 34
4.2.3 CMIS Service 36
4.3 Evaluation Tool 40
4.3.1 Architecture 40
4.3.2 CMIS Client Features 42
4.3.3 Simulator 44

4.4 Technical Setup 46
4.5 Summary 47

Chapter 5 Evaluation **48**
5.1 Features Evaluation 48
5.2 Features Evaluation 49
5.3 Evaluation Metrics 50
5.4 Simulation 52
5.4.1 Browse Simulation 52
5.4.2 Query Simulation 56
5.4.3 Workload Simulation 60
5.4.4 Network Simulation 64
5.5 Improvement 64
5.6 Summary 65

Chapter 6 Conclusion **66**

Chapter 7 Future Work **67**

Bibliography **68**

List of Figures

2.1 Benefit of CMIS 12

3.1 CMIS Plug-in Server 26
3.2 CMIS Server for WordPress 27
3.3 Evaluation Design 28

4.1 OpenCMIS Server Framework 32
4.2 OpenCMIS framework Sequence Graph 33
4.3 CMIS Server for WordPress Structure 34
4.4 Query Sequence Graph 39
4.5 Evaluation Tool Structure 41
4.6 Evaluation Tool Structure 42
4.7 Evaluation Process 46

5.1 Browse Simulation Test Suite 52
5.2 Browse Simulation Result 53
5.3 Process or Time of Browse Simulation 54
5.4 Available MBytes of Browse Simulation 55
5.5 Page Per Sec of Browse Simulation 55
5.6 Query Simulation Test Suite 56

5.7 Query Simulation Result 57
5.8 Processor Time of Query Simulation 58
5.9 Available MBytes of Query Simulation 59
5.10 Page Per Sec of Query Simulation 60
5.11 Workload Simulation Test Suite 61
5.12 Workload Simulation Result 61
5.13 Process or Time of Workload Simulation 62
5.14 Available MBytes of Workload Simulation 63
5.15 Page Per Sec of Workload Simulation 63

Chapter 1

Introduction

1.1 Background

In todays interconnected world, most enterprises and organizations are dealing with a high volume of content, such as project documentation and marketing material. Since manage this large volume of content is more and more critical for enterprises and organizations, the requirement of some kinds of content management is imperative.

Content Management System (CMS) is a system that can be used to manage and organize text, movies, pictures, links and many other types of content. There are different types of CMS, such as WCMS and ECMS. Typically, a Web Content Management System (WCMS) is employed by both technical and non-technical people to create, edit, print and manage online content of a website. Drupal and WordPress are the most typical examples of WCMS. Enterprise content management system (ECMS) is employed by enterprise to organize and store its documents, and other content that pertains to the organization's processes. For instance, Alfresco

and SharePoint are the most common ECMS.

1.2 Motivation

Usually, CMS from different vendors can be different in many ways such as programming language, performance and features. For many large organizations and enterprises, different CMS must be used in order to achieve different purposes. For example, the development department may use a document CMS to manage the documentations of a project while the propaganda department may use a WCMS to manage the website for the organization. As a result, those organizations have to frequently maintain content cross several CMS.

Interoperability can be defined as the ability of two or more systems or components to exchange information and to use the information that has been exchanged. [7] However, realizing interoperability is difficult in reality since different CMS has its own specific APIs. While both organizations and enterprises have significant needs of managing content across multiple content management systems, the cost to achieve that is high without a standard interface.

The Content Management Interoperability Service is an open standard aim to improve the interoperability between different Content Management Systems which approved by the Organization for the Advancement of Structured Information Standards (OASIS) which is a web standards consortium. While many enterprise content management systems, like Alfresco and Nuxeo, have now implemented CMIS, it has received relatively less attention in the important class open source web content management systems. While WCMS has a significant effect on the

daily work of organizations and individuals, the interoperability of Web CMS also needs to be improved.

1.3 Subject

This research aims to find how the CMIS standard can be implemented to support open source PHP based Web Content Management System. To achieve the expected goal, the following steps should be realized. At first, a CMIS server for a specified open source PHP based WCMS should be implemented. Afterwards, evaluation should be undertaken to investigate the feasibility and performance issues of the implementation. Additionally, the implementation ought to be compared to an existing Java based CMIS server while evaluation process.

1.4 Approach

A. Research state of art in the relevant territories to grasp the requirements of stakeholders.
B. Based on the requirements, design the architecture of the CMIS server and relevant evaluation tool.
C. While the implementing stage, it will indicate which part is difficult to be implemented. The structure of the prototype CMIS server will indicate how easy it can be extended to other open source WCMS.
D. Performance evaluation of the prototype CMIS server will examine its performance compare to the existing Java CMIS server and indicate whether or not it can be run in a realistic situation.

1.5 Outline

Chapter 2 presents the relevant topics in WCMS and CMIS.

Chapter 3 describes the requirements for CMIS server and evaluation, as well as the optional designs based on those requirements.

Chapter 4 presents the implementations, including the CMIS server for WordPress and an evaluation tool.

Chapter 5 discusses the performance of the prototype CMIS server compare to Alfresco.

Chapter 6 presents the conclusion of this project.

Chapter 7 discusses the potential future works based on this project.

Chapter 2

State of Art

In this chapter, the status and impact of WCMS are first discussed. In particular, the benefits of WCMS are examined. Then CMIS is introduced in details which show its significant impact on CMS. Afterwards, the specification of CMIS is presented which includes a data model, services, current implementations and use cases. The most relevant use case in this project is the A2R, which the implemented CMIS server is acting as the CMIS compliant server and the evaluation tool acting as the client application. In the last, the comparison of CMIS with some similar standards is discussed to show the advantage of CMIS.

2.1 Web Content Management System

Since the computer and the network are popularized these days, everyone wants to create a website or a blog of their own. In traditional way of building websites, people build every page of the website as a static page and upload them to a web server by using an FTP tool. Although this may work fine with some simple websites, in fact it has many limitations. The first main disadvantage is that the

owner must have the technical knowledge of how to build and maintain a website which few people do. Secondly, in a static website, codes and contents are mixed within different static page which means the contents in the static website cannot be easily added or modified. Thus, the website can only be maintained by the developer who is familiar with the structure. Moreover, you have to write a new page or update the page carefully and upload it again to the web server which cost a lot of time. And usually the static website contains many duplicated information which means the owner has to handle the consistency problem on his website. No wonder having websites in those days were expensive and keeping them updated was definitely a challenging task. [8, 16]

In contrast, by using a WCMS, the imperfections of static website are solved. A WCMS separates the content from presentation which allows developers to create templates for displaying various types of content, such as page title, images, body. As a result, the administrator of the website can focus on managing the structure of the site while the editor can focus on the creation of content. For example, a writer can use the pre-defined templates to create and update a novel on his website, while the WCMS can build and manage the structure of the site. And no doubt that a WCMS provides a user friendly client for users to easily control their site so that they do not need know technical knowledge. Managing their content is easy as working on a simple application like word and the content will be published automatically by WCMS. [8, 12]

The separation of content and presentation also bring the benefit of reusable since each WCMS provides its proprietary APIs for developers to get the contents and their metadata. It is the key difference between a WCMS based website and a

static website. With the APIs, the content of one website can be pulled out and reused in another site. Additionally, the APIs provide a way for developers to build plug-ins and widgets for various purposes. Plug-in and the widget will be a great help, if the basic features of the WCMS cannot satisfy the user. Users can always easily find different kinds of free widgets and plug-ins which developed by other programmers in the open source community. For example, the Post Profile plug-in for WordPress allows the user to view how many people view the post recently. But with a static website, you cannot even image this. The only way that users want to add more features to their static website is developing it manually which is time consuming and expensive and maybe need a programmer to be involved. [2, 6, 18]

Not only for individuals, WCMS is also meaningful for companies. Firstly, WCMS helps companies improve the multichannel customer experience and localize content with elements target at specific local markets. Secondly, WCMS helps them generate consistent and up-to-date information for customer which is most important for them. [20]

Not only help organizations and individuals get free from hand-coding web pages and publishing them to the web, WCMS is also a useful marketing tool. WCMS enables enterprise to effectively create fresh content and push it to the web, and more importantly, it enables them to monitor how users interact with a web presence. This interaction is essential for website to customize the pages in a way of meeting the customers expectations. Moreover, popular of social website and mobile is also forcing people to rethink their WCMS. Both social and mobile provide a way for website to get more details about a users preferences and interests which

will be useful to decide the way of content delivery and presence. No longer be a stand-alone, these emerging issues call for the need for WCMS integration. WCMS not only need to be integrated with different tools, but also need to collaborate with other kinds CMS. [11]

Nowadays, users can get more and more functionality and features at a lower and lower price due to the improvements of technology. A few years ago, WCMS might be expensive for common users, but with the introduction of open source WCMS the cost has become affordable for most users. In fact, almost all of the open source WCMS are free which give people a great way to build a website for themselves. [22] Presented in Effective Web Content Management: Empowering the Business User, [6] the number of websites created by WCMS shows an increasing trend and according to 2011 Open Source CMS Market Share Report [21] which based on a survey of more than 2500 CMS users and additional research, open source PHP based WCMS, such as WordPress, Joomla and Drupal, continue dominating the web content management market. So PHP based WCMS can be the representative of open source WCMS.

2.2 ECMS

Normally these days, the enterprises will generate and maintain plenty of content and many of them are generated by their customers. The categories of generated content can be various, such as project documentation, personnel file, marketing material or other content generated in the daily activities of the enterprises. Since the global economy, many enterprises expand their business to different countries around the world which means they have to deploy Enterprise Content Manage-

ment System (ECMS) in different locations and may need different ECMS also. And sometimes, different departments within a company will also choose different ECMS according to the advantage of each ECMS. Having multiple ECMS will cause a challenge that they have to deal with content deployed in different ECMS. And enterprises want to access and manage the content without regard which ECMS they are using and organize the content in the way most suit them. For example, some employees in the company want to get the entire customer and product documents stored in different ECMS, so the company wants the client application can retrieve the content across different ECMS and organize them in one single interface automatically.

Although the fundamental features of ECMS are similar, and ECMS tend to public their proprietary APIs to integrate. As a result, the interoperability of ECMS is different to realize. The traditional ways to do that is get connectors cross two different ECMS. But with this design, there are two limitations need to be fixed. Firstly, the company must have connectors between any two different ECMS they have. If the company owns several ECMS, it has to manage and maintain many connectors which are complicated and not convenient. Secondly, most of the connectors tend to be the single purpose focus while enterprise usually wants to do multiple tasks.

2.3 Content Management Interoperability Service

Historically content management systems were purchased for specific application uses and this led to islands of incompatible systems. The lack of a standard in-

terface to content management systems made it difficult to integrate content from multiple repositories into a single application such as a portal, CRM system, or office desktop. It also made it difficult for Independent Software Vendors (ISVs) and integrators to build applications that supported multiple content management systems consistently or easily. [3]

This statement is declared by OASIS Content Management Interoperability Services Technical Committee to describe the interoperability problem. To define a standard interface for ECMS and make it easier to integrate content across multiple repositories, the CMIS is proposed and developed. *The Content Management Interoperability Services (CMIS) standard defines a domain model and Web Services, Restful AtomPub and browser (JSON) bindings that can be used by applications to work with one or more Content Management repositories/systems.* [3]

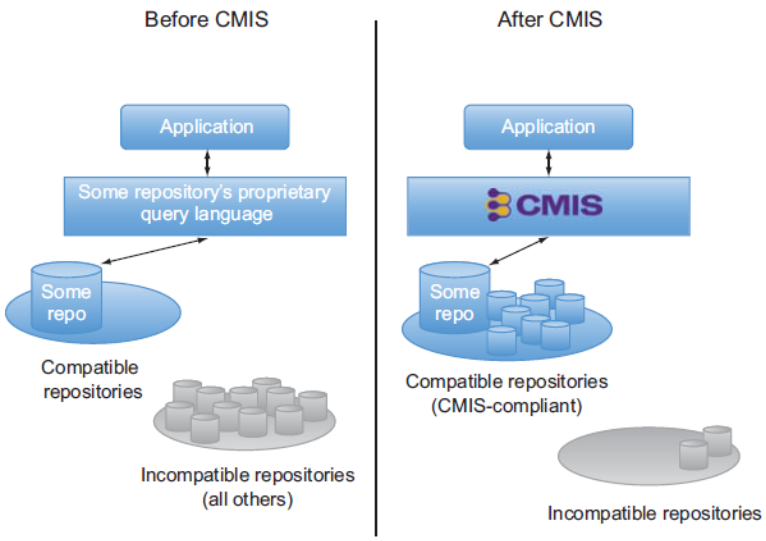
On May 1, 2010, the Organization for the Advancement of Structured Information Standards (OASIS), a web standards consortium, approved CMIS as an OASIS Specification. On December 12, 2012, CMIS 1.1 has been approved as the latest OASIS specification. In this latest version, the major new features add are the Item object type which can be used to define custom object type and Browser Binding which used to support applications running in a browser.

After 1992, an organization probably will not care about which relational database they were using, because of the power of SQL. Before the standardization of SQL, a developer has to write a particular program for each database using by his application. Switching databases meant the developer has to modify the program for the new database. And in some worse case, the developer has to record the

program which is time consuming and expensive. Just like this situation, the developers working in the world of content management are facing the same challenge. Because ECMS developed by different vendors has its own API, it is difficult for developers to write an application that can work with several CMS. So, it is the reason for CMIS. CMIS is a vendor-neutral, language-independent specification for working with ECM systems. Like SQL, CMIS enables developer to use a single application to work with multiple repositories without coding for different repositories. The creation of the CMIS specification and its broad adoption is almost as significant and game-changing in the content management industry as SQL standardization and the adoption of that standard was to the relational database world. [10]

By choosing repositories that are CMIS compliant, enterprises will gain several benefits. Firstly, the development costs and switching costs are obviously reduced, because enterprise is able to work on multiple repositories with a single application rather than with many particular APIs. Secondly, its easy and flexible for developers to work with CMIS compliant repositories. After learning CMIS, developers will be able to handle most of the fundamental features of CMIS compliant repositories so that they don't need to learn new APIs when they want to use a new repository. Developers are able to select the language or framework which most suits them because CMIS is language neutral. [3]

Fig. 2.1: Benefit of CMIS [10]



2.3.1 CMIS Specification

According to the specification (Version 1.1) [3], CMIS designs an abstraction layer to be layered on top of the existing Content Management System and their existing programmatic interface with the intent which is described as below:
It is not intended to prescribe how specific features should be implemented within those CM systems, nor to exhaustively expose all of the CM system's capabilities through the CMIS interfaces. Rather, it is intended to define a generic/universal set of capabilities provided by a CM system and a set of services for working with those capabilities. [3]

CMIS provides an interface for applications to access a repository. To achieve this, a Data Model is specified by CMIS that defines*the persistent information entities that are managed by the repository.*And a set of services is also specified by CMIS to let the application be able to access and manage those entities. As

said previously, the objective of CMIS is just define a generic set of capabilities provided by CMS, the data model specified by CMIS does not support all the concepts that a full-function ECM repository typically supports. [3]

In the data model, a repository is a container of objects. Objects represent the entities in the repository and each one has a type. There are four base types defined by CMIS, Documents, Folders, Relationships and Policies. Any other specific objects must derive from these four base types. [4, 19]

- A document object represents a standalone information asset. Document objects are the elementary entities managed by a CMIS repository. [4]
- A folder object represents a logical container for a collection of file-able objects, which include documents or sub-folders. Unlike documents, folders can only sit in one folder not several or none. [4]
- A relationship object represents an instance of directional relationship between two objects. [4]
- A policy object represents an administrative policy, such as a retention management policy, which may be applied to one or more controllable policy objects. [4]

Services that CMIS provides to users are focused on four basic features: create, retrieve, update and delete, which is also called GRUD. CMIS also provides other services like navigation and discovery. Following are the descriptions of the basic services that relevant in this project. [4]

- Repository Services are used to allow the client to get the information about the repository which includes the detailed repository information and the

object type definitions that the repository supported. [3]

- Navigation Services are used for the client to traverse the folder tree of the repository and locate the objects. [3]
- Object Services are used to get the information of objects and provide basic GRUD features. [3]
- Discovery Services are used to search the queried object by the client. [3]

In the CMIS specification, it initially support to the following scenarios:

- Collaboration systems
- Portals leveraging Content Management repositories
- Mash-up system, merging information from several different sources
- Federated search, searching one or several content repositories

But the following scenarios are supported which includes WCMS

- Records Management (RM) and Compliance
- Digital Asset Management (DAM)
- Web Content Management (WCM)
- Subscription and Notification Services

2.3.2 Current CMIS Implementation

The implementation of CMIS can be divided into two parts which includes CMIS complaint server and CMIS client. A CMIS complaint server means it can store

content in its repository and provide them through CMIS services. A CMIS client typically means an application that can browse and manage content store in a CMIS compliant server. In this project, both of the two parts are involved.

Up to now, many ECMS have been developed to support CMIS as a part of their service, such as Alfresco, Apache Chemistry InMemory Server, dotCMS and Microsoft SharePoint Server. And there are many existing client applications, such as TYPO3, LibreOffice and Drupal. Additionally, there are three CMIS server libraries that can be used to build a CMIS complaint server which are VB.CMIS, VB.CMIS and OpenCMIS Server Framework (part of Apache Chemistry).

However, there is no official PHP based open source WCMS that has been CMIS compliant. Furthermore, the existing CMIS server library cannot be used to build PHP based CMIS compliant server due to programming language difference.

2.3.3 Use Cases

Said by Laurence Hart (2009) [14], there are three fundamental use cases for the application of CMIS in an enterprise situation, which is Repository-to-Repository (R2R), Application-to-Repository (A2R) and Federated Repositories.

Repository-to-Repository (R2R)

In this situation, the repositories will communicate directly to each other. The two examples for this use case are:

- Federated Records Management which is centrally manages records that are stored in multiple repositories.

- Publish content from one repository to another.

Application-to-Repository (A2R)

In this situation, an application that uses content is connected to a repository that handles the content services. Some examples for this use case are:

Collaboration Systems, like an application to be used as a front end and repository as the back-end.

- Enterprise Software Applications.
- Content Enabled Vertical Applications
- Productivity applications.

Federated Repositories

In this situation, a single interface will be exposed to users while the application works with multiple repositories. Two examples for this use case are:

- Federated Search, like Google Search Appliance which will search across many different repositories.
- Federation instead of migration. Instead of directly migrate content from one repository to another, an application will be used to interact with the new and old repositories.

2.3.4 Industry Response

The meaning of a standard is that the industry will implement it, otherwise, it is useless. According to AIIMs State of the ECM Industry 2011 comments below,

the awareness of CMIS has increased over two years which shows a nice picture. Although the number of vendors who have implemented a CMIS support is not large now, but they are the major vendors in the industry such as IBM and ECM.

The development of CMIS (Content Management Interoperability Services) promises to improve connectivity between ECM systems, and between ECM and other enterprise applications. In our 2009 survey, 69% of our end user respondents were not sure what it is. This fell to 54% in 2010 and 42% this year. The number making a commitment to adopt it as a standard in their organization has doubled since 2009 to 8%, although this is still a small proportion of the overall potential. [17]

2.4 Related Standards

2.4.1 Web Distributed Authoring and Versioning

Web Distributed Authoring and Versioning (WebDAV) is an extension of the Hypertext Transfer Protocol (HTTP). The objective of WebDAV is to provide an interoperability mechanism for users to edit and manage files across remote web servers. WebDAV exposes four main features: maintenance of properties, namespace management, collections and locking. Maintenance of properties enables users to maintain information on web pages; namespace management enables users to map web pages within a URL; collection enables users to management resource; locking enables the same resource will be edited by only one user at one time. [9]

2.4.2 Content Repository API for Java

Content Repository API for Java (JCR) defines a standard for Java based applications to work with multiple repositories. The specification was developed under

the Java Community Process as JSR-170 (Version 1) and as JSR-283 (version 2).

The JCR specification provides a standard interface for different CMS to implement. Thus, like CMIS, the application developers only need to learn one API that is compliant with any JSR-170/JSR-283 compliant repository. Not only vendor neutral, JCR is also architecture neutral. The back-end data storage could be a file system, a WEBDAV repository, an XML-backed system or an SQL-based database.(Javines, 2010) A JCR compliant repository provides four basic operations, read, write, query and delete just as CMISs GRUD.

2.4.3 PHP Content Repository

The PHP Content Repository (PHPCR) is an adaptation of the JCR standard which defines how to deal with a content repository. The PHPCR specification defines an API that combines the best of NoSQL databases with hierarchical data structures. PHP based repositories can implement this API to provide generic access to their content. Moreover, the PHPCR API provides standard ways to handle search, versioning, access control and other features. There are several implementations available now, such as Jackalope Jackrabbit and Midgard2 PHPCR layer.(Bergius, 2011)

2.4.4 Comparison with CMIS

Obviously, there are plenty of benefits to have a standard for the Content Management System world. Thus, many other standards are defined to achieve the same goal as CMIS. But there are still needs for the CMS industry to have a standard like CMIS.

A major disadvantage of the JCR and PHPCR is that they are restricted to a specific programming language and environment. What is worse, they require the ECM to modify some core capabilities in order to support the specification features. In CMIS these constraints don't exist, because CMIS is language neutral and it only defines the generic features that can be implemented to any ECM. Moreover, the JCR standard is not service oriented which means it requires consistent connection and not suitable for some applications, such as mash-up applications. Last, many major vendors participate in the specification, such as IBM, EMC, HP and Oracle. Compare to CMIS, only a few vendors support JCR. [1]

Unlike CMIS which support a custom query language, WebDAV does not support query features. And WebDAV does not support types, properties and relationships, making it very limited with the content that can handle. [9]

2.5 Summary

It has been identified that the significant benefits of WCMS for individuals and organizations. And it is clear that the WCMS is an indispensable part of the organizations and individuals daily work. Because of the urgent needs of integration of multiple repositories, a standard for the CMS industry is important. The definition of CMIS addresses the interoperability problem across multiple repositories, which provides an efficient way for the whole CMS industry. Even through many

major vendors of ECM has implemented the CMIS standard, there are very few vendors of WCMS tried to do that. Especially, there is no PHP based WCMS been implemented to adapt the CMIS standard. With increasing interactive re-

quirements of users, the interoperability of WCMS is more and more important. Thus, implementing CMIS to PHP based WCMS will bring a lot of benefits for the website owners.

Chapter 3

Design

In Chapter 2, the current situation of CMIS has been investigated and the important effect of CMIS to web CMS has also been shown. Due to the lack of official CMIS server for web CMS, especially PHP based web CMS, the value of the development of a CMIS server for web CMS is proved. In this chapter, the stakeholders related to this project and the requirements of the production will be discussed. This chapter will also explore the requirements for the evaluation part which can be used to understand and improve the performance of the product. The requirements defined in this chapter will be used to guide the implementation in the next chapter.

3.1 Stakeholders

As described in Chapter 2, a growing number of websites managed by WCMS are created and they will publish plenty of content every day. Since different WCMS has its own advantages, individuals or organizations may use more than one WCMS to build their website. For example, they may use one WCMS to

manage technical documents for internal use and use another WCMS to manage company official website. And some organizations may use different WCMS which most suitable for the situation at that time to manage their content. Furthermore, for the enterprise which spread all over the world, they may need to use the most suitable WCMS to meet the local requirements.

In those examples which need to get content cross different WCMS, they need an easier and efficient way to do that. So the key stakeholders of this project will be the individuals and organizations who need to get content from different WCMS, and in this chapter the requirements will be defined to meet their needs.

3.2 Repository Choice

When choosing the WCMS used to implement the CMIS server, several reasons were taken into account. The WCMS must have a large user base, because the key stakeholders who have the needs to integrate content cross different WCMS usually use the most popular ones. For this reason, Drupal, Joomla and WordPress are the best choices according to 2011 Open Source CMS Market Share which identified them to be the market leader. Secondly, based on time consideration, the WCMS must not be too difficult to learn. Joomla and WordPress have a relatively simple structure for developer to understand that meets this requirement. Thirdly, the WCMS must provide comprehensive APIs for developer to use which most WCMS meets this requirement. Last, the web CMS should own a developer community that can be used to support the development. In the three candidates, WordPress has the shortest learning curve. The APIs of WordPress provide developers an excellent way to build extensions. Apart from those, the community forum of

WordPress has plenty useful blogs and detailed guide which will be a helpful place. As a result, WordPress is selected which will be used to build the CMIS server upon with.

Alfresco is chosen to be the Java based CMS to be compared, as suggested in Aonghus O hAirts (2012) project. [13] Alfresco not only to be a fully CMIS compliant server, but also own an active CMIS section of their community forum. Apart from those, Alfresco has a CMIS Wiki that provides a detailed guide about CMIS.

3.3 Requirements

This section presented the functional and non-functional requirements for the CMIS server and evaluation tool.

3.3.1 Functional Requirements

CMIS Server for WordPress

Because the stakeholders of this project are those who want to get content from different WCMS, the features of the CMIS server must contain:

- Connect to WordPress and use it as a repository
- Get content and information from the repository
- Implement CMIS services to allow the client to browse the repository and get objects
- Expose the service through at least one binding

With the above features, the CMIS severs can achieve the requirements of stakeholders, and advance features can be built upon them.

Because query is also a common feature that stakeholders will use to efficiently get information of content and is a better way to measure the performance of the server, implement query features in the CMIS server is necessary.

Evaluation

The objective of the evaluation is to determine whether or not the CMIS server can be implemented to support WCMS and if the performance is comparable to the ECM CMIS server.

By the evaluation, it should answer the following questions: [15]

- If CMIS can be implemented to support WCMS?
- Is there a better way to implement CMIS into WCMS?
- If the performance of the implementation comparable of ECM CMIS server?
- If the implementation can be easily extended to support other WCMS?

For a CMIS compliant server, the most important evaluation metrics should be response time and resource cost, so that the evaluation will be performed based on simulation test. For example, check the response speed of the server to handle a browse request and how many resource it uses on the server computer. For the simulation purpose, an evaluation tool should be implemented in order to act as a CMIS client. The evaluation tool should be implemented to test all the features of CMIS server for WordPress. For purposes of investigating the performance of the server under different situations, the evaluation tool must have the ability to

simulate any amount of clients and control the number of requests. In addition, the evaluation tool must be able to simulate different client activities. After getting the response from the server, evaluation tool must record the result of each request to a database for the analysis usage.

3.3.2 Non-functional Requirements

The key non-functional requirement for the CMIS server is that the performance should be acceptable for users in a realistic situation. For the stakeholders who want to use the CMIS server, the purpose is to get content from different WCMS. To achieve that purpose, the response time of the implementation should be acceptable and the results must not contain any errors in it under normal situations. In addition, another non-functional requirement is that the server should not have a significant detrimental impact on the server computer.

For evaluation tool, the most important non-functional requirement is that the record of results must be accurate since the analysis based on the data requires accuracy.

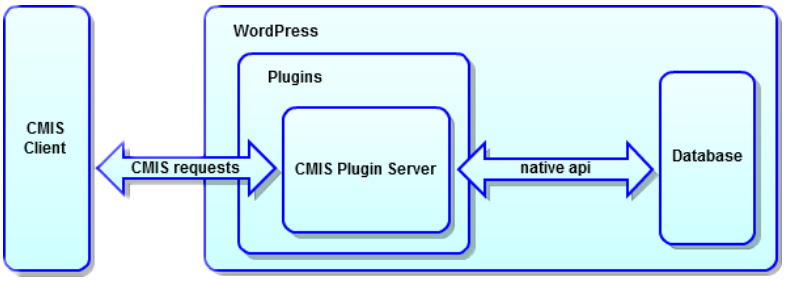
3.4 Design

CMIS Server for WordPress

Based on the above requirements of the CMIS server, the first design is to implement the server as a plug-in for WordPress.

In this design, the CMIS server use plug-in APIs provided by WordPress attach

Fig. 3.1: CMIS Plug-in Server



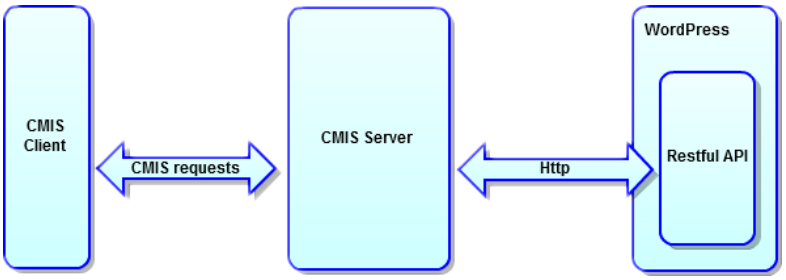
to it and use database APIs to get the information of the content. The client can access the CMIS server through the service binding exposed by the CMIS server, such as AtomPub.

In order to implement the CMIS server for WordPress, the implementation must create the entire server stack in PHP since there is no existing PHP CMIS server library as described in Chapter 2. The implementation must contain the entire domain model defined by CMIS and the services. Additionally, the implementation must contain a binding of the service, such as AtomPub to expose the server to public client. It does require a massive work to be done, but the performance of the implementation should be good, and the implementation can be used by others to build their CMIS server.

The second design is to implement the CMIS server as an external independent component to WordPress.

The Restful API of WordPress is a series of endpoints that provide create, get, update and delete features for developers to manage content stored inside the repository. The response will be returned as JSON objects. WordPress main-

Fig. 3.2: CMIS Server for WordPress as a Independent Part



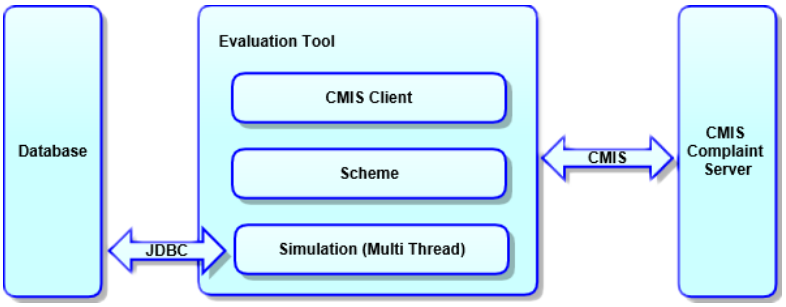
tains an excellence website which introduces the detailed information about each available endpoint, such as required parameters and object type of the response parameters. The Restful API provides a way for this project to implement the CMIS server outside the WordPress. Compared to the first design, this design can be easier implemented. In this design, the CMIS server can be built upon the existing CMIS server library which will save a lot of effort and time, such as OpenCMIS server frame. In addition, because the CMIS server uses Restful APIs to connect to WordPress, it is able to extend to other web CMS which provide Restful API.
[23]

Evaluation Tool

According to the requirements for the evaluation tool, the structure of the evaluation tool is shown as below.

Because the evaluation tool must have the ability of a CMIS client, so its better to build a client inside the evaluation tool. In order to simulate multiple clients, the evaluation tool should use multi threads to simulate the parallel clients. In

Fig. 3.3: Evaluation Design



the real situation, the number of requests that a CMIS has to handler can be different in different time. So a scheme class should be used to control the requests that a thread sends to the server. The scheme should control the interval time between each request which is the thinking time of use. Additionally, it should control the number of thread and number of requests that a single thread sends to the server. After getting a response from the server, evaluation tool should store the result and its information into the database. The data will be used to analyse the performance of the server based on the average result of the oft-repeated stimulation.

3.5 Summary

In this chapter, the stakeholders of this project are identified, which are the organizations and individuals who have large volume of content and need to get them cross different WCMS. The choice of repository is further discussed, which take several considerations into account. The functional and non-functional requirements for CMIS server and evaluation tool are then explored. Based on the requirements, the optional designs are presented. There are two ideas for the CMIS server. The first one is implementing the server as a plug-in for WordPress and

the other is implementing the server as an independent part. The plug-in design is much more difficult since it has to implement the entire CMIS server stack, while second design is easier but rely on the Restful API which may affect the performance of the server. In the last, the design of evaluation tool is proposed. The designs presented in this chapter will be used as a guide for the implementation.

Chapter 4

Implementation

In this chapter, the detailed implementations of the prototype CMIS server for WordPress and evaluation tool are presented. The implementations are developed based on the design described in Chapter 3 to meet the requirements. Although the design which describes the CMIS server as a plug-in for WordPress was firstly tried out, this implementation was too complicated. To complete the project and leave time for evaluation, the other option was taken. As a result, implementation of CMIS server for WordPress was developed based on the design which makes the server as an independent component of WordPress. The implementation of evaluation tool can be used to simulate different types of activities in order to evaluate the performance of the server under different situations. The key components of the implementation are presented in detail.

4.1 Plug-in CMIS Server for WordPress

As described in chapter 3, this implementation involves developing an entire server stack for PHP. In order to implement this design, I referred to the Apache Chem-

istry OpenCMIS Server Framework which is implemented in Java. At first, the domain model of CMIS is implemented which includes the entire object and services defines. All of those implementations are developed following the openness principles. But when started developing the service binding after finish the domain model, the implementation is stuck because the lack of knowledge for AtomPub. After discussing with the supervisor, the second design was taken as this design requires more time to learn the relevant technology.

4.2 CMIS Server for WordPress

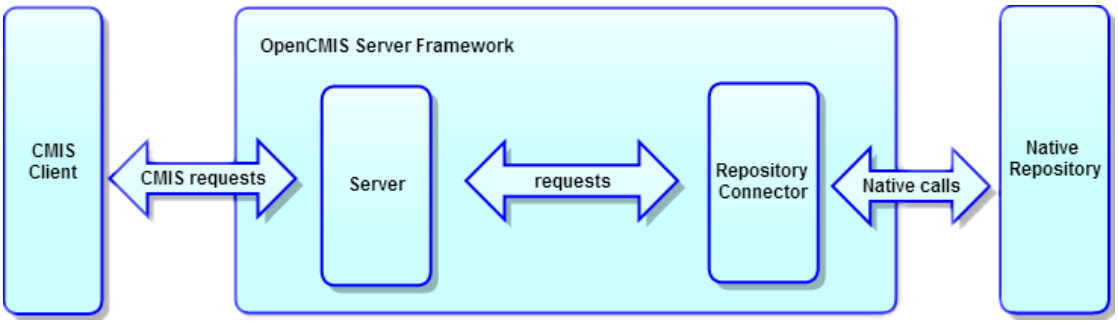
This section presented the detailed implementation of the CMIS server for WordPress.

4.2.1 Architecture

As presented in chapter 2, there are three existing CMIS server library can be used to build the server implementation which is VB.CMIS, NCMIS and OpenCMIS Server Framework. The first two libraries are developed using .NET and OpenCMIS server framework is developed by Java. Because Java is a more familiar language to me, so the implementation the server was developed upon OpenCMIS server framework.

OpenCMIS server framework is a sub-project of Apache Chemistry OpenCMIS project which provide an open source Java implementations of the CMIS specification. The framework provides a skeleton implementation of the server, and a repository connector is required. The framework will collect the requests and data from the CMIS clients and push them to the repository connector which will

Fig. 4.1: OpenCMIS Server Framework

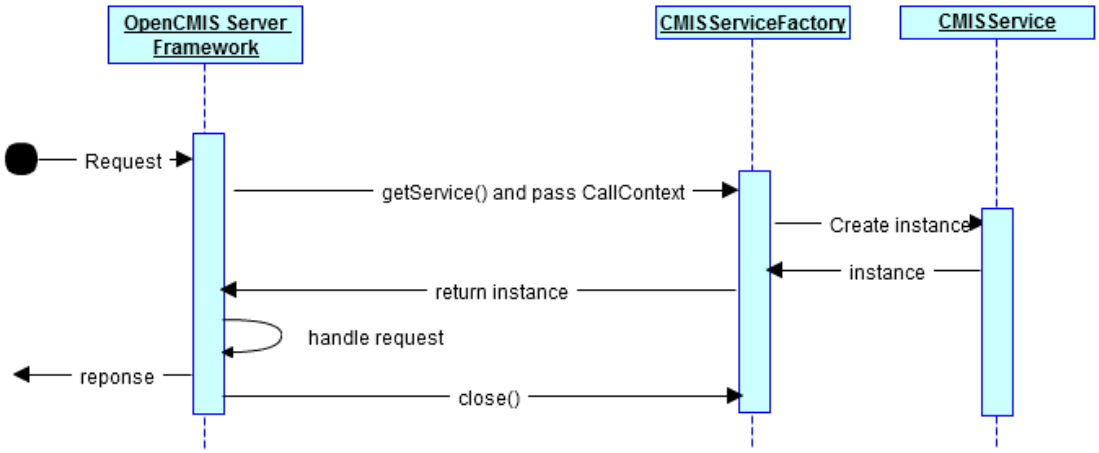


translate the request into native repository calls. [5]

The native calls are different depending on the repository that a server is using, e.g. this project uses Restful APIs to communicate with WordPress. By using theOpenCMIS server framework, the CMIS server implementation only needs to develop the repository connector which responsible for transforming the CMIS calls into WordPress calls and fulfil the necessary services. In the prototype implementation, the repository connector contains below six classes used to achieve the features. While developing the repository connector, I referred to OpenCMIS File Fileshare Repository and InMemory Repository which are the CMIS compliant servers developed by Apache Chemistry. [5]

The CMISServiceFactory class is the entry point of the framework to recognize the repository connector, it manages the object that implements the CMIS service. Once the server receives requests from CMIS client, the getService() method within CMISServiceFactory class is called by the framework and a CallContext object is passed. This CallContext object contains data about the request, such as the used binding, the repository id, username and password. The getService ()

Fig. 4.2: OpenCMIS framework Sequence Graph



method will return an object that implements the CMIS services and this object is used by the framework to handle the request. For each request, the framework will create a new object to handle it and after sending the response, the object will be closed. The figure below shows the sequence graph. [5]

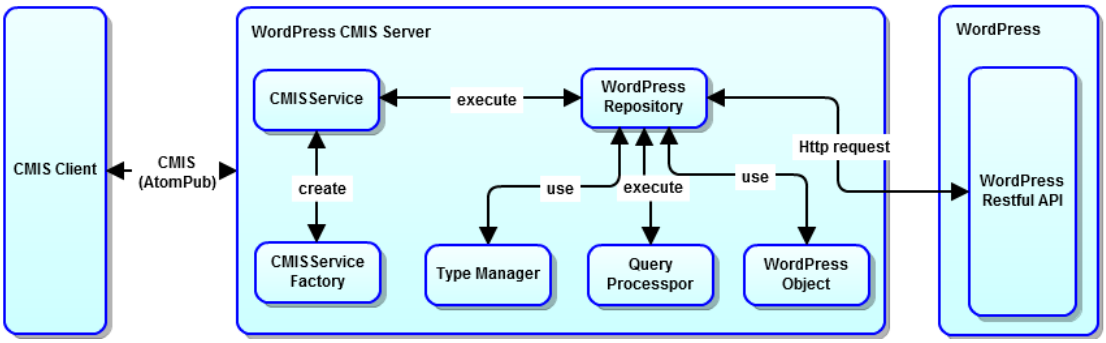
The CMISService class declares the implemented CMIS services in its body and will be created by CMISServiceFactory to return to the framework.

The TypeManagerImpl class defines and stores the data models supported by this server.

The WordPressRepository class is the key class which implements the CMIS services and connects to WordPress to get information about content.

The QueryProcessor class is a part of WordPressRepository which used to process the query requests.

Fig. 4.3: CMIS Server for WordPress Structure



The WordPressObject class is the object definition to store information that gets from WordPress, such as a post.

The figure below shows the relations of the components within the CMIS server for WordPress.

4.2.2 Data Model

As presented in chapter 2, the CMIS has four basic object types which are Folder, Document, Policy and Relationship. After version 1.1, the Item type is added. In the prototype implementation, only Folder and Document type is used to support the model of WordPress. Root directory is essential to a repository, whereas WordPress doesnt have the folder concept. As a result, a virtual root is manually created by the repository. Since post is the most common content that the stakeholders managed in WordPress, so the prototype implementation of this project focus on this content type. Additionally, the WordPress post will be mapped into

the CMIS Document object and metadata of the post will be mapped into Document properties. Although most of the metadata can be mapped into properties of Document type, there are some specific metadata type that a Document does not own. Therefore, in order to fully support the Web CMS object types, some extension types should be defined in CMIS which is not the main target of this project.

In order to follow the principle of openness, a class called WordPressObject is developed to contain the information about post and other objects that will be supported in the future version. Whenever the repository connector needs to get information from WordPress, a method called getInformationFromWordPress() will be executed. Inside this method, it uses HttpClient send request to the WordPress Restful API endpoint and get the response. Because the response object is JSON, a library developed by Json.org is used to handle the response. After reading the response into a Java String object, a JSONObject will be used to parse the response. Finally a WordPressObject is created based on the information get from the JSONObject by key word, and it will be kept holding the information.

Listing 4.1: Key Code Fragment of getInformationFromWordPress()

```
DefaultHttpClient httpClient = new DefaultHttpClient();
HttpGet getRequest = new HttpGet(URL + content);
getRequest.addHeader("accept", "application/json");
JSONObject json = null;
try {
HttpResponse response = httpClient.execute(getRequest);
if (response.getStatusLine().getStatusCode() != 200) {
```

```
        throw new RuntimeException("Failed : HTTP error code : "
        + response.getStatusLine().getStatusCode());
}
BufferedReader br = new BufferedReader(
        new InputStreamReader((
                response.getEntity().getContent())));
        String line;
        String output = "";
        while ((line = br.readLine()) != null) {
                output += line;
        }
                try {
                        json = new JSONObject(output);
                } catch (JSONException e) {
                        e.printStackTrace();
                }
        } catch (ClientProtocolException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
}
```

4.2.3 CMIS Service

In order to meet the requirements of stakeholders for this project, the CMIS services implemented in the prototype include repository service, navigation service,

object service and discovery service. All the services are implemented based on the CMIS specification version 1.1 and all of the parameters are passed as the definition. In the prototype implementation, the main target is providing the ability that let the stakeholders can get content from WordPress repository and execute queries. Therefore, only related services and functions within the services are implemented.

4.2.3.1 Repository Service

Repository service is used to get information about the repository which includes repository information and information about object type definitions. In this prototype implementation the below methods are implemented:

- getRepositoryInfos() returns the repository list available in from the endpoint. In the prototype implementation, the method returns the repository of yanzhangk.wordpress.com.
- getRepositoryInfo() returns the repository information and capabilities of the repository, in this project, returns the information of yanzhangk.wordpress.com.
- getTypeDescendants() returns the set of descendants under a specified type which defined in the TypeManager class according to the type id.
- getTypeDefinition() returns the type definition of a specified type which defined in the TypeManager class according to the type id.
- getTypesChildren() returns the children types under a specified type which defined in the TypeManager class according to the type id

With the implemented methods, the client can get the available repositories and their information from the service endpoint. And the type definitions supported

by this repository are also available through the three type methods.

4.2.3.2 Navigation Service

Navigation service is responsible for travel through the folder hierarchy of a repository, whereas WordPress repository only has a virtual root folder. Consequently, the methods that related to get folder information are not relevant to this project. As a result, only two methods of navigation service are implemented which are getChildren() and getObjectParents().

The getChildren() method returns the objects contained in a specified folder, as there is only a root folder in this repository, so the other objects will be returned. The getObjectParents() method returns the parent folder of a specified object, as all the objects in this repository are in the root folder, so the root folder is always returned. As a result, the client can browse the repository through this service.

4.2.3.3 Object Service

In order to achieve the purpose that gets content from the repository, the prototype implementation implemented four methods presented as below.

- sgetAllowableActions() returns the allowed actions for a specified object according to the object id.
- getObject() returns the information of a specified object according to the object id.
- getObjectByPath() - returns the information of a specified object according to the path.

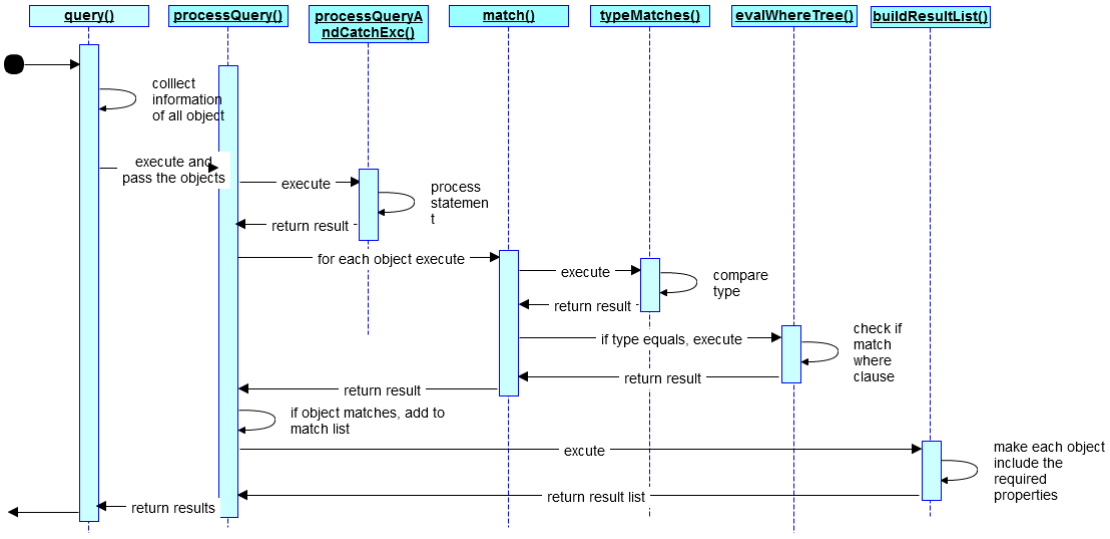
- getProperties() returns the properties of a specified object according to the object id

By implementing the methods, the client can get the information, properties and allowed actions of an object by using its id or path. Therefore, clients can get content from the repository.

4.2.3.4 Discovery Service

Discovery service is responsible for search the query-able object inside the repository. Since query is required for the performance evaluation part, the query() method is implemented in the prototype implementation. The class QueryProcessor is developed to handle the query requests. Within the class, the query statement is divided into several chunks and processed by different methods.

Fig. 4.4: Query Sequence Graph



In the above figure, the fundamental process of query is presented. Firstly, the class collects information of all objects from WordPress and stores them. Secondly, the query statement is processed and divided into chunks. Thirdly, each candidate object is compared to the query object which collected from the query statement. If the object type is equal to the query object, it will be checked if it matches the statement in where clause. If the result is true, put the object into the match list. Finally, before returning the matched objects, each object in the match list will be processed to include the properties which required in the query statement.

4.2.3.5 Build the Result

Before each method which returns CMIS object, the result has to be compiled from the WordPressObject that stored in the program to CMIS object. The method compileObjectType is responsible for this process. It will create and add the corresponding properties as well as allowable actions to the CMIS object according to the WordPressObject. Since the CMIS provides the feature that allows clients to control the number of results in each page, the results list has to be processed before return them to the client.

4.3 Evaluation Tool

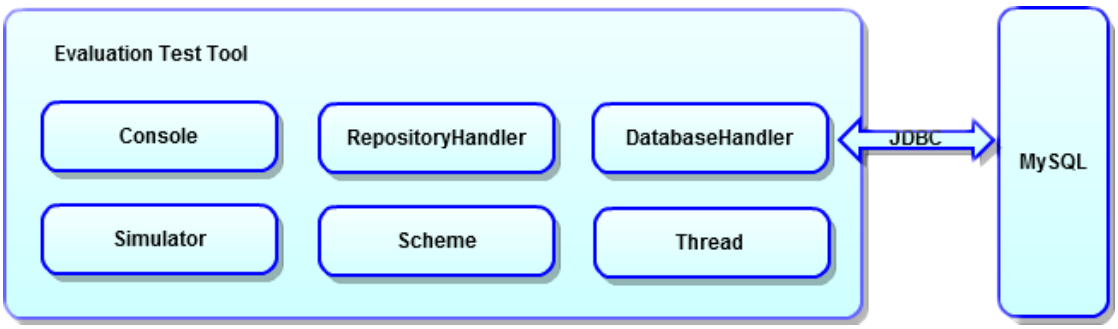
This section presents the implementation of an evaluation tool.

4.3.1 Architecture

According to the design presented in chapter 3, the evaluation tool should include CMIS client features which can be used to simulate common CMIS client activities. Additionally, the tool must use multi-thread to simulate multi-client.

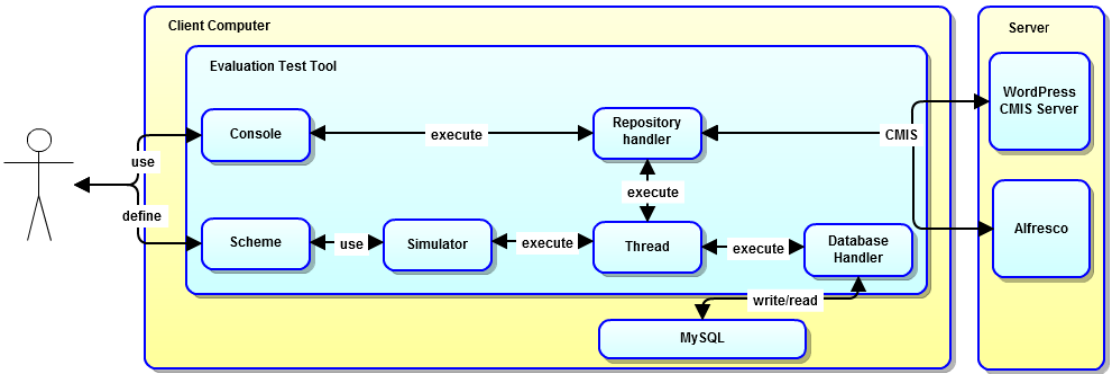
The implemented components of the evaluation tool are shown in the figure below.

Fig. 4.5: Evaluation Tool Structure



- Console provides a client for users to execute operations and display the information of response.
- RepositoryHandler is actually a CMIS client implementation that can connect to the CMIS compliant server and execute CMIS client operations, such as browse folder and query.
- DatabaseHandler is responsible for write or read data from MySQL database. Thread will call invoke this DatabaseHandler to store the information of the response.
- Scheme defines the parameters for one simulation to perform different workload.
- Each simulation is controlled by its corresponding simulator, and in this prototype implementation, there are four different kinds of simulation.
- Thread simulates one single CMIS client and performs the specified job, and it will be invoked by simulator.

Fig. 4.6: Evaluation Tool Structure



4.3.2 CMIS Client Features

As presented in the section 4.2.1, the RepositoryHandler needs to implement the features of CMIS client. This prototype implementation uses the OpenCMIS client API to develop client implementation.

In addition to the CMIS specification, the OpenCMIS client API introduces a session concept which enables applications to get control on the client side cache behaviour. The session object is the main object that an application has to work with and it will attach to a repository to get information. The SessionFactory interface is responsible for the application to create a Session object and get available repositories from a CMIS service binding end point. Additionally, OpenCMIS client API provides CMIS object implementation which shared with server framework, such as repository, folder and document. [5]

In this prototype implementation, three features are implemented in RepositoryHandler. Firstly, it can connect to a CMIS compliant server using the Session object provided by the OpenCMIS client API. Before create the Session object,

some parameters must be passed to the SessionFactory object to get the available repositories. Then the Session object can be created to attach to a specified repository. Secondly, the RepositoryHandler can be used to browse the repository which includes the sub-feature that can get the children of a specified folder. Lastly, the RepositoryHandler can execute query to the server and display the response.

Listing 4.2: Key Code Fragment to create a Session

```
SessionFactory sessionFactory =
                        SessionFactoryImpl.newInstance();
Map<String, String> parameter =
                        new HashMap<String, String>();
parameter.put(SessionParameter.USER,
                        this.repositoryUserName);
parameter.put(SessionParameter.PASSWORD,
                        this.repositoryPassword);
parameter.put(SessionParameter.ATOMPUB_URL,
                        this.repositoryURL);
parameter.put(SessionParameter.BINDING_TYPE,
                        BindingType.ATOMPUB.value());
List<Repository> repositories =
                        new ArrayList<Repository>();
repositories = sessionFactory.getRepositories(parameter);
Repository repository = repositories.get(0);
parameter.put(SessionParameter.REPOSITORY_ID,
                        repository.getId());
session = sessionFactory.createSession(parameter);
```

4.3.3 Simulator

The objective of the evaluation tool is to measure the performance of the CMIS server implementation under different situations, so the evaluation tool must simulate different workload for the server. The Scheme is used to control the workload of a simulation. Inside the Scheme class, it defines the information for the simulation, such as thread number, request number and interval time. The interval time is used to simulate the thinking time that a user used after receiving the response and execute the next operation. Since this parameter has the same impact as the thread number on the response time, the interval time in this project will be set to a normal value, e.g.2 seconds.

In order to evaluate the performance of the CMIS server in a comprehensive way, the evaluation tool is implemented to contain three kinds of simulators. Each simulator can be used to simulate one particular operation.

- Browse Simulator. Since browse is the most common operation that users will execute, the performance of the server under this simulation is meaningful to users.
- Query Simulator. Query this is also a common operation for users and it is a relatively complicated operation which can best represent the performance of the server. Additionally, query simulator can use different query statements to perform different workload.
- Network Cost Simulator since this prototype implementation use Restful API to get information from the repository, so the time cost can be divided

into two parts this. This simulator can be used to identify the time cost of getting a response from the Restful API to get a more accurate performance of the server.

Listing 4.3: Key Code Fragment for Browse Simulator

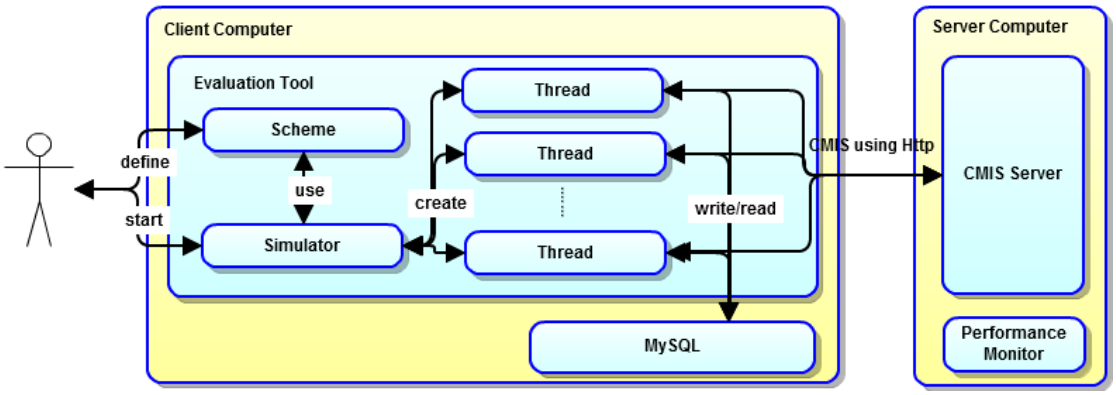
```
public BrowseSimulator() {
scheme = new Scheme();
handler = new RepositoryHandler(alfrescoURL,
alfrescoUserName, alfrescoPassword, alfrescoType);
//handler = new RepositoryHandler(wordpressURL,
wordpressUserName, wordpressPassword, wordpressType);
scheme.setName("B001");
scheme.setCount(20);
scheme.setInterval(2000);
scheme.setThreadNumber(40);
}
public void doBrowseSimulation() {
        try {

        for (int i = 0;
        i < scheme.getThreadNumber(); i++) {

        Thread t = new Thread(new
                        BrowseThread(scheme, handler));
        t.start();
}
```

Within each simulator, it creates multi threads according to the thread number defined in the Scheme class. Each single thread will invoke the RepositoryHandler to execute the specified operation. Before executing the operation, the thread will record the start time. After receiving the response from the server, the thread will calculate the time cost for the mean time. Then the data will be stored into the database by invoking the DatabaseHandler. The stored data includes scheme name, thread number and time cost. After finishing the simulation, the entire data stored in the database will be retrieved to be analysed.

Fig. 4.7: Evaluation Process



4.4 Technical Setup

The CMIS server for WordPress is implemented using Maven in Eclipse, and the OpenCMIS server framework is added to the project as a dependency by using the following fragment [5]:

Listing 4.4: code

```
<dependency>
    <groupId>org.apache.chemistry.opencmis</groupId>
    <artifactId>chemistry-opencmis-server-support</artifactId>
    <version>x.y.z</version>
 </dependency>
```

After finishing the implementation, it was built into a war file and put into the application server to be running. The application server used in this project is Resin which is a Java based web application server.

The evaluation tool was developed as a Java project in Eclipse and can be run in Eclipse.

4.5 Summary

This chapter presented the prototype implementation according to the design described in chapter 3. The implementation of the Plug-in CMIS Server is first presented. Due to time consideration, this implementation cannot be finished with this project. Then the CMIS server for WordPress based on the OpenCMIS server framework is introduced. The architecture and functionality of the implementation are presented in detail. Last, the implementation of evaluation tool is described. With the OpenCMIS client API, the tool implements the features in the design and support three kinds of simulator. After implementing the CMIS server and evaluation tool, the implementation work is finished. The next chapter will present the evaluations of the server implementation.

Chapter 5

Evaluation

This chapter describes the performance evaluation of the CMIS server for WordPress (CSWP) compare to Alfresco. According to the design in chapter 3, the evaluation tool will be put on the client computer and another server computer is used to hold the Alfresco and CSWP. Before the evaluation, the same documents are created for the two servers. The performance valuation will be performed in two parts. The first part measures the response performance of the server on the client side which takes response time and the number of errors into account. The performance data will be collected by the evaluation tool for this part. The other part measures the resource utilization of the CMIS server application on the server side. The performance data will be collected by using the Performance Monitor of Windows on the server computer.

5.1 Features Evaluation

This evaluation is used to check if the CSWP correctly handle the requests as it declared. The console of the evaluation tool is responsible for this evaluation.

Firstly, the console will send request to CSWP to get available repositories. After receiving the repository list, create the connection to the first one. Secondly, the console will execute browse operation to check the navigation and object service of CSWP. Thirdly, the console will execute the query operation to check the discovery service. After receiving response from CSWP, the result will be compared with the Documents stored in WordPress.

By performing this evaluation, the results show that CSWP does return the right response the client and achieve the features as it declared. This result indicates that the CMIS can be used to support WCMS and this server meets the functional requirements defined in the Chapter 3.

5.2 Features Evaluation

This evaluation is used to check if the CSWP correctly handle the requests as it declared. The console of the evaluation tool is responsible for this evaluation. Firstly, the console will send request to CSWP to get available repositories. After receiving the repository list, create the connection to the first one. Secondly, the console will execute browse operation to check the navigation and object service of CSWP. Thirdly, the console will execute the query operation to check the discovery service. After receiving response from CSWP, the result will be compared with the Documents stored in WordPress.

This evaluation is used to check if the CSWP correctly handle the requests as it declared. The console of the evaluation tool is responsible for this evaluation. Firstly, the console will send request to CSWP to get available repositories. After

receiving the repository list, create the connection to the first one. Secondly, the console will execute browse operation to check the navigation and object service of CSWP. Thirdly, the console will execute the query operation to check the discovery service. After receiving response from CSWP, the result will be compared with the Documents stored in WordPress.

5.3 Evaluation Metrics

Since the first part of the evaluation focus on the response performance of CSWP, the following metrics are defined to measure the performance.

- Total cost, which records the response time of the entire simulation. It represents the overall response speed of the server under the specified workload.
- Average cost which records the response time of a single request in the simulation. It is calculated by the total cost and the number of requests. And it represents the response speed of the server. In the real world, it represents the time a client has to wait to get the response after sending a request.
- Server Cost which represents the time for the server to handle a single request. For Alfresco, this is the same to Average cost. Since CSWP uses the Restful API to get information from WordPress, the response time can be divided into two parts which are the time of CSWP to handle the request and the time to get information from WordPress. As a result, the server cost of CSWP means the Average cost subtract the time for CSWP to get information from WordPress. With this metric, it is easier to investigate if the Restful API has a significant impact on the performance.

- Network Cost which stands for the time for the server to get information from WordPress. It is used to calculate the server cost for CSWP in this project.

- Request per Second which can represent the limit performance of the server under the specified workload.

- Errors which record the number of errors that the server returns while the entire simulation.

For the second part of the evaluation, three counts of the Performance Monitor are recorded to represent the resource utilization of the server application. The explanations of the counts refer to Microsoft TechNet.

- Processor Time. This counter shows the percentage of the time that the processor spends on non-idle thread. It is calculated by subtracting the percentage of time that the processor spends on idle process. It indicates the time that the processor is busy.

- Available MBytes. This counter shows how many megabytes of physical memory are currently available for use by processes.

- Pages/Sec. This counter indicates the number of pages that either were retrieved from disk due to hard page faults or written to disk to free space in the working set due to page faults. This counter can be combined with the Available Mbytes counter to indicate the memory usage of the server implementation.

5.4 Simulation

This section presents the test suite of different simulations as well as the performance result collected corresponding to each test suite. Since simulator is controlled by the scheme, each test suite has several schemes. In order to get a more objective result, each test suite is simulated three times. To make a fair situation for the two servers, ten documents are created with the same properties. Additionally, killing the irrelevant processes as many as possible on the server side computer.

5.4.1 Browse Simulation

Fig. 5.1: Browse Simulation Test Suite

Scheme Name	Interval (ms)	Requests/Thread	Thread
B001	2000	10	5
B002	2000	10	20
B003	2000	20	20
B004	2000	20	40

Both Alfresco and CSWP will be evaluated by this test suite. In test B001 and B002, the value of request per thread is set to same in order to investigate the performance of the server under different workload. Set the same value of the thread parameter of test B002 and B003 to investigate the performance of the server when work in a longer time. In test B004, the server will receive 800 requests in a short period which means a relatively large workload for the server, so it can be used to see the performance under that situation. The table below shows the performance data of this test suite.

Fig. 5.2: Browse Simulation Result

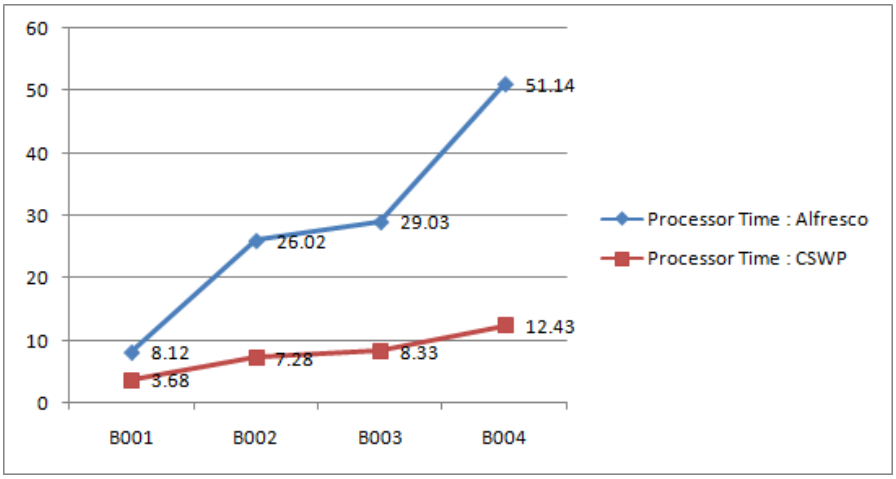
Name	Total Cost (ms)	Average Cost (ms)	Network Cost (ms)	Server Cost （ms）	Request/ Second
B001 (Alfresco)	21637	432.74	/	432.74	2.31085640
B002 (Alfresco)	154466	772.33	/	772.33	1.29478331
B003 (Alfresco)	357316	893.29	/	893.29	1.11945728
B004 (Alfresco)	827943	1034.93	/	1034.93	0.96625009
B001 (WordPress)	256512	5130.24	2215.24	2915	0.19492265
B002 (WordPress)	1291363	6456.82	2457.25	3999.57	0.15487512
B003 (WordPress)	2583205	6458.01	2208.57	4249.44	0.15484640
B004 (WordPress)	5357208	6696.51	2637.12	4059.39	0.14933151

The data in this table indicate several results. Firstly, by comparing B001 and B002, the results indicate the response time will be longer if the workload is added. Secondly, by comparing B002 and B003, the results indicate the response time will be longer if work in a long time. Thirdly, from test B001 to B004, the response times of Alfresco show a rising trend when the workload is increased, while the response times of CSWP didnt show a clear change.

Compare to Alfresco, the response speed of CSWP is much slower. The average response time of the CSWP is about 12 times of the Alfrescos response time. The client has to wait about 6 seconds to get the result from CSWP which is too long in the real situation. Even get rid of the impact of Restful API, the response time of CSWP is still about 4 times bigger and requires 4 seconds for the client.

This also can be reflected by the Request/Second metric which shows the limit performance of CSWP is not good as Alfresco.
Therefore, the implementation of the CMIS service related to browsing operation must be optimized since the response time is too long.

Fig. 5.3: Process or Time of Browse Simulation



This graph shows the average Processor Time value of the two servers while processing the browses requests. As shown in the graph, Alfresco needs nearly 3 times more time for the CPU to handle the processes. Alfresco also shows a faster increasing trend while CSWP shows a smoother trend. As a result, CSWP needs less CPU resource to handle requests. However, consider that Alfresco is a much complicated system that CSWP, it is hard to say that the CPU usage of Alfresco is all used to handle browse requests.

The Available MBytes and Pages/Sec can be combined to indicate that while the browse simulation, the two servers did not face a memory shortage problem. Be-

Fig. 5.4: Available MBytes of Browse Simulation

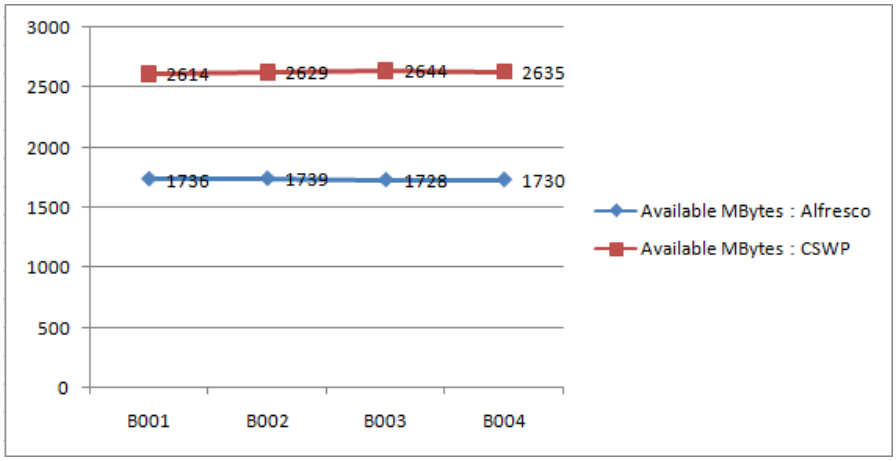
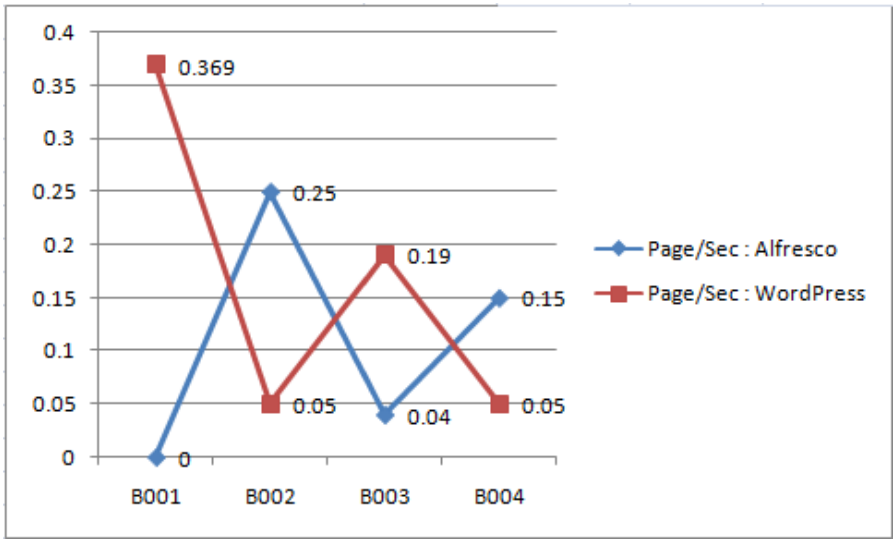


Fig. 5.5: Page Per Sec of Browse Simulation



cause the available memory of the server are the In addition, the CSWP server seems to consume less memory than Alfresco. As a result, the CSWP will not have a significant detrimental impact on the server computer under this simulation.

5.4.2 Query Simulation

Fig. 5.6: Query Simulation Test Suite

Scheme Name	Statement	Interval (ms)	Repetition	Thread
Q001	SELECTE cmis:name FROM cmis:document	2000	10	10
Q002	SELECTE cmis:name FROM cmis:document	2000	10	20
Q003	SELECTE cmis:name FROM cmis:document	2000	20	40
Q004	SELECTE * FROM cmis:document	2000	10	10
Q005	SELECTE * FROM cmis:document	2000	10	20
Q006	SELECTE * FROM cmis:document	2000	20	40

This test suite is also designed to investigate the different performance of the server when changing only one parameter. The comparison of Q001 and Q002 indicates the performance of the server under different workloads. The performance of servers under test Q001 and Q004 shows the impact of statement complexity on the servers. Test Q003 and Q006 are used to investigate the performance of the server under relatively large workload.

By comparing the results of Q001 and Q002, the data show that increasing 10 threads did not have clearly impacted on the servers. Comparing the Q001 and Q004, as well as other pairs which have same parameters except query statement, the results show that the complexity of query statement did have an impact on

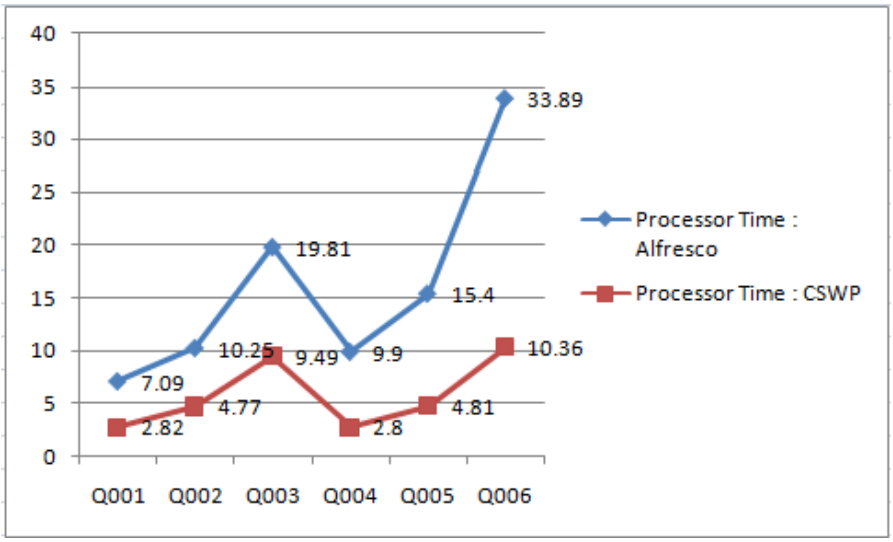
Fig. 5.7: Query Simulation Result

Name	Total Cost (ms)	Average Cost (ms)	Network Cost (ms)	Sever Cost （ms）	Request/Second
Q001 (Alfresco)	15975	159.75	/	159.75	6.259780908
Q002 (Alfresco)	32220	161.1	/	161.1	6.207324643
Q003 (Alfresco)	127922	159.9	/	159.9	6.253810916
Q004 (Alfresco)	26023	260.23	/	260.23	3.842754486
Q005 (Alfresco)	110370	551.85	/	551.85	1.812086618
Q006 (Alfresco)	501894	627.37	/	627.37	1.593962072
Q001 (WordPress)	241686	2416.86	2377.41	39.45	0.413760003
Q002 (WordPress)	480980	2404.9	2354.25	50.65	0.415817706
Q003 (WordPress)	1920118	2400.15	2333.83	66.32	0.416641061
Q004 (WordPress)	222039	2220.39	2186.65	33.74	0.450371331
Q005 (WordPress)	449744	2248.72	2206.7	42.02	0.444697428
Q006 (WordPress)	1800160	2250.2	2178.42	71.78	0.444404942

the servers performance.

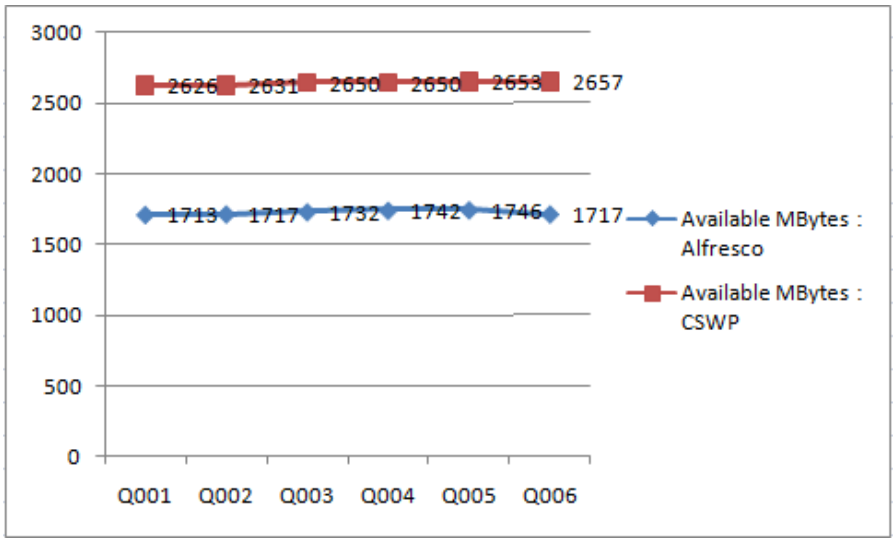
Comparing the average cost of each test for Alfresco and CSWP, the response time of CSWP is more than 10 times longer than Alfresco. But a 2 second delay is acceptable in a realistic situation. And considering that CSWP uses the Restful API to get information, the time that CSWP used to handle the requests is not long. After subtracting the time spent on getting information, the time of CSWP used to handle the query is much less. Even through the server cost indicates the query performance of CSWP is better than Alfresco, the whole response speed of CSWP is still not as good as Alfresco. Additionally, the Alfresco may have many custom object types in its repository which may cost more time to search among them. However, one certain thing that can be indicated from the results is the time spent on getting information is severely affecting the efficiency of CSWP.

Fig. 5.8: Processor Time of Query Simulation



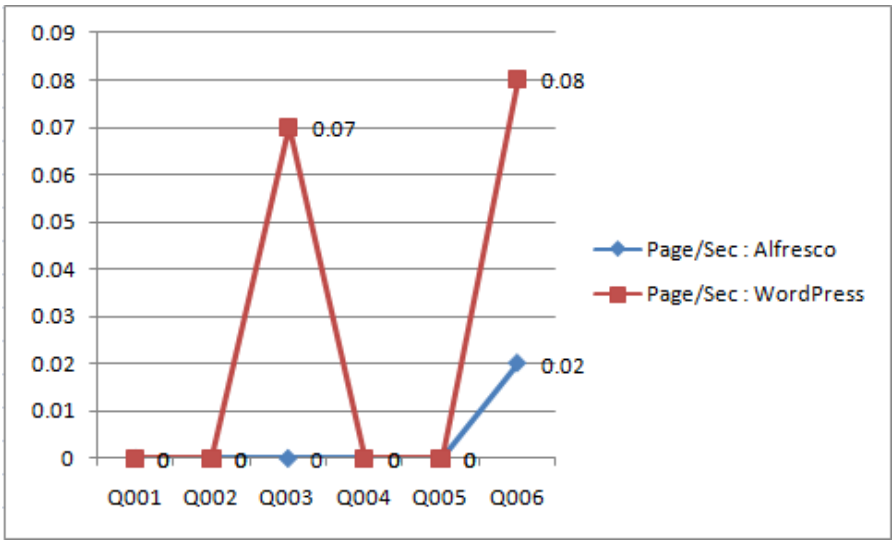
The above graph shows the CPU usage while handling query requests. As revealed by the graph, the CSWP uses less CPU resource to handle query requests than Alfresco. In addition, both servers show an increasing trend while the adding the number of threads. By comparing the value of Q001 and Q004, Q002 and Q005, as well as Q003 and Q006, the values indicates Alfresco uses more time to handle the more complex query statements. Yet CSWP had not shown to be affected a lot by the complexity of the query statement.

Fig. 5.9: Available MBytes of Query Simulation



The memory usage of query simulation shows a similar state to the browse simulation, the two servers did not require more memory to handle the requests. Additionally, the CSWP used less memory compare to Alfresco.

Fig. 5.10: Page Per Sec of Query Simulation



5.4.3 Workload Simulation

The purpose of this simulation is to investigate the performance of the server when under a large workload and check errors exist in the response to see if the server meets the limitation. Since browse is the most common request, the basic operation is set to browse to simulate a large number of parallel clients which request browsing the repository. The following test suite is designed to achieve the purpose.

The number of threads is increased slower in the starting stage to check the performance of the server and avoid receiving many errors. And increasing the number of threads to a much larger amount to see if the server meets the limitation.

The above table reveals that the response time of both servers is very long, and the response speed of Alfresco is faster than CSWP. The average response time of Al-

Fig. 5.11: Workload Simulation Test Suite

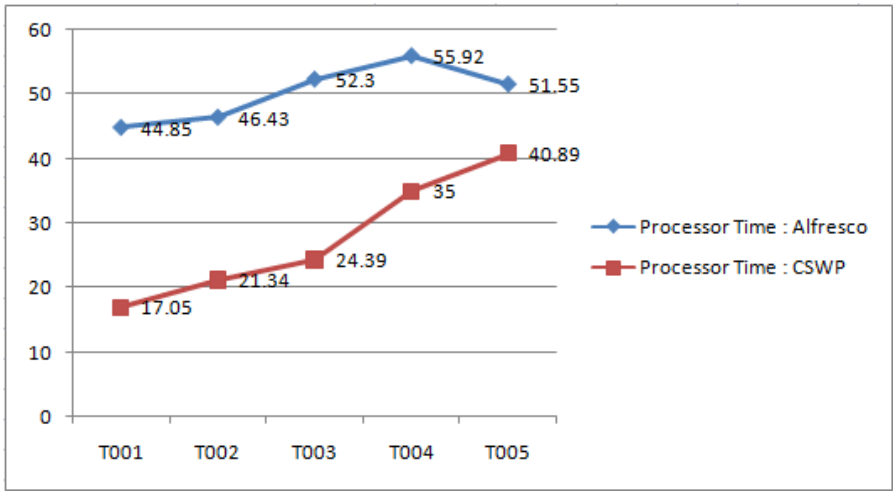
Name	Thread
P001	100
P002	130
P003	150
P004	200
P005	300

Fig. 5.12: Workload Simulation Result

Name	Thread	Average Cost (ms)	Error	Request/Second
L001 (Alfresco)	100	12676.6	0	0.07888551
L002 (Alfresco)	130	13731.3	12	0.07282632
L003 (Alfresco)	150	15050.4	13	0.06644342
L004 (Alfresco)	200	21022.5	47	0.04756808
L005 (Alfresco)	300	15622.43	142	0.06401053
L101 (WordPress)	100	17335.3	0	0.05768576
L102 (WordPress)	130	17244.36	2	0.05798997
L103 (WordPress)	150	17834.63	1	0.05607069
L104 (WordPress)	200	20160.25	1	0.04960256
L105 (WordPress)	300	23288.32	74	0.04293998

fresco is about 80 percent of CSWP. But the errors are contained in the response of Alfresco is more than CSWP. In particular, when the number of threads increases to 300, half of the responses are error from Alfresco. Compare to Alfresco, it seems that CSWP can work better under the 300 parallel requests. However, considering Alfresco is designed to be run on the professional servers, maybe the resource of a normal personal computer is not enough for it.

Fig. 5.13: Process or Time of Workload Simulation



From the graph 5.1.3, the CPU usage of Alfresco is sustaining in a high state and may meet the limitation in test T005 since the ratio is suddenly dropped. On the contrary, the CSWP shows a relatively low CPU usage in the beginning and increasing to a high state. Nevertheless, the CSWP seems not meet the limitation under 300 request.

Fig. 5.14: Available MBytes of Workload Simulation

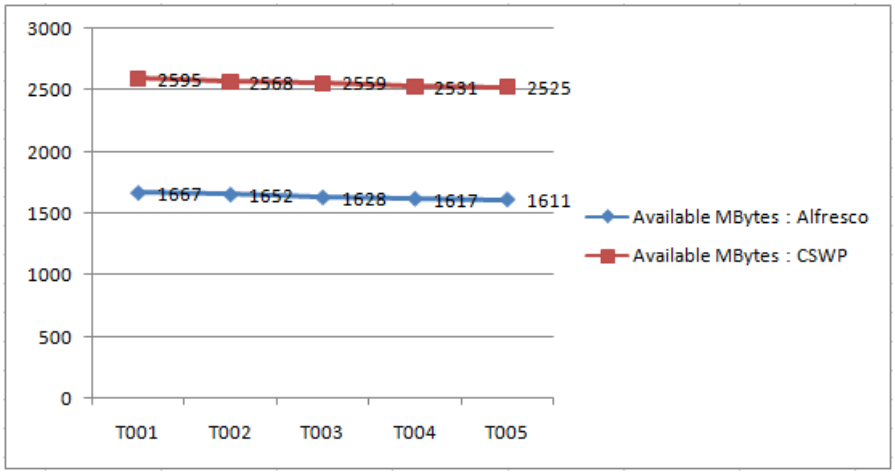
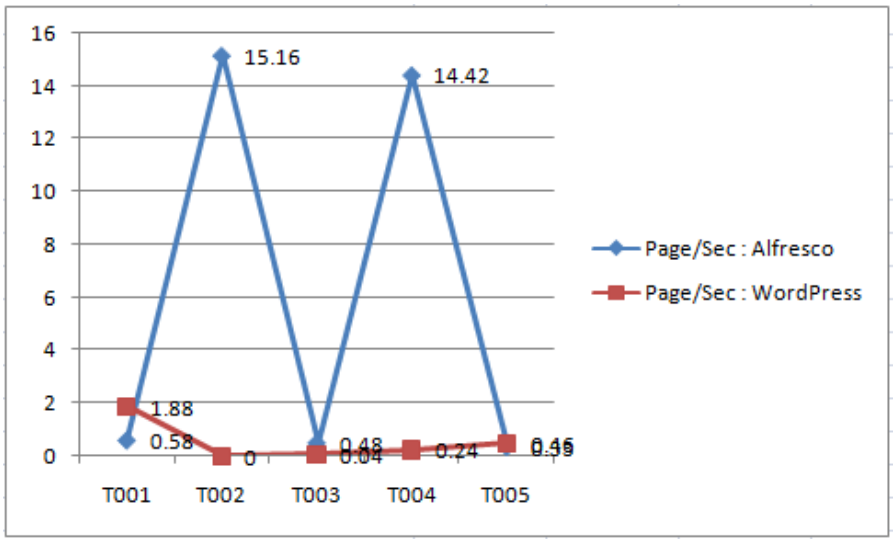


Fig. 5.15: Page Per Sec of Workload Simulation



As indicated by the Available MBytes and Page/Sec counters, CSWP uses less memory than Alfresco and both servers consumed more memory as the number of threads is increased.

5.4.4 Network Simulation

For the sake of investigating the accurate performance of CSWP, this simulation is designed. In order to get the corresponding time cost for each previous test, the operation of this simulation is implemented to be the same as the operation performed to WordPress in the previous simulations. For example, in test B001, each thread will execute browse operation which will require the server to get all the objects in root folder of WordPress. In this simulation, the corresponding test will simulate the operation that gets all the objects in root folder to get the response time of WordPress. As a result, the time spent on getting information from WordPress is investigated. The results of this simulation are added to the previous simulations as the metric Network Cost.

5.5 Improvement

As indicated from the browse simulation, the response time of CSWP is too long for the users in a realistic situation. In order to improve the performance, a caching mechanism is implemented. Within the Repository Connector of CSWP, the WordPressRepository instance will continue pulling the informationof content from the WordPress and store them in the Repository Connector. Once receiving a request from the client, the CSWP will return the information which stored in the WordPressRepository instance instead of getting it from WordPress. In this

way, the performance of CSWP will be better after the first caching. But this design may consume more resource of the server computer.

5.6 Summary

In the chapter, the evaluation of the features is first presented which shows CSWP meet the functional requirements of the design. Then the performance evaluation metrics are described which can be divided into two parts. the first part of the matrices indicates the response speed of the server and the second part shows the impact that the server has on the computer. The different simulation designs are further introduced, as well as the results of each simulation. The results show that performance of CSWP is not as good as Alfresco, especially the browse feature. However, the CSWP does not have a serious impact on the performance of the server computer. Under the large workload, the CSWP shows a normal reaction that can be accepted. Lastly, a caching mechanism is implemented to CSWP to improve the performance of browse feature.

Chapter 6

Conclusion

This project achieves its purpose which is developing a CMIS server for open source PHP based Web CMS and investigate the feasibility and performance issues compare to an existing Java based CMIS server implementation. The CMIS server for WordPress is implemented as the requirement. As described in the evaluation chapter, it can be used to provide CMIS services for clients which meet the requirements of stakeholders. This indicates that CMIS can be used to support WCMS. And a tool is implemented to achieve the performance evaluation purposes. In the performance evaluation process, the CMIS server for WordPress and Alfresco are evaluated under different situations. Even though the performance of the CMIS server for WordPress is not as good as Alfresco, it did not have a significant detrimental impact on the server computer. Additionally, an improvement is done to the CMIS server for WordPress to enhance the performance of browse. Furthermore, since the architecture of this implementation is not locked to WordPress, it can be easily extended to support other WCMS.This ability provides a potential way to improve the interoperability as the purpose of CMIS.

Chapter 7

Future Work

A few more works can be done upon this project:

- As said in the implementation chapter, the basic object types of CMIS cannot contain all the metadata of Web CMS objects, so the future work can investigate the possibility of developing an extension to CMIS to fully support Web CMIS Objects.
- This project investigates the browse and query performance between the CMIS server for WordPress and Alfresco, a future work can be done to investigate other CMIS features.
- The CMIS server implementation of this project is developed for WordPress, a future work can modify the implementation to make it support other Web CMS.
- Although the implementation of first design is not finished, a future work can be continued to implement a PHP based CMIS server library.

Bibliography

[1] *Content Repository for Java Technology API Specification (JSR-283).* Day Software AG, 2009.

[2] *Essential Elements of a Web Content Management Solution.* Ektron, 2009.

[3] *Content Management Interoperability Services (CMIS) Version 1.1.* OASIS, 2013.

[4] Alfresco. Cmis. http://wiki.alfresco.com/wiki/CMIS.

[5] Apache. Apache chemistry opencmis 0.9.0. http://chemistry.apache.org/java/0.9.0/maven/apidocs/.

[6] Associates, W. Effective web content management: Empowering the business user while it maintains control. *Ektron Inc* (2001).

[7] Bernd Simon, Symeon Retalis, S. B. Building interoperability among learning content management systems.

[8] Daniel MICAN, Nicolae TOMAI, R. I. C. Web content management systems, a collaborative environment in the information society. *Informatica 20 Economic? 13* (2009), 20.

[9] E. James Whitehead, J. World wide web distributed authoring and versioning (webdav): An introduction.

[10] Florian Mller, Jay Brown, J. P. *CMIS and Apach Chemistry in Action.* Manning, 2012.

[11] Goodall, G. The evolution of web content management: A tool for both it and marketing professionals. http://blog.infotech.com/news-analysis/the-evolution-of-web-content-management-a-tool-for-both-it-and-marketing-professionals/.

[12] Goodwin, S., and Vidgen, R. Content, content, everywhere... mmm time to stop and think? the process of web content management. *COMPUTING & CONTROL ENGINEERING JOURNAL* (2002).

[13] hAirt, A. O. An open localisation interface to cms using oasis content management interoperability services. Master's thesis, University of Dublin, Trinity College, 2012.

[14] Hart, L. Three fundamental cmis use cases. http://wordofpie.com/2009/08/17/three-fundamental-cmis-use-cases/.

[15] I, D. E. N. N. I. S. N. K. A. N. G. Adapting content management interoperability services for web content management systems. Master's thesis, Royal Institute of Technology, 2010.

[16] McKeever, S. Understanding web content management systems: evolution, lifecycle and market. *MCB UP limited 9* (2003), 103.

[17] Moore, M. Content management interoperability services. *OLC 26* (2012), 115.

[18] Ngoma, S. *Functional Usability and Flexibility of Web Content Management Systems*. PhD thesis, 2011.

[19] Potts, J. *Getting Started with CMIS*. Optaros, Ecmarchitect.com, CMIS, 2009.

[20] Waldt, D. Content management interoperability services (cmis). *Gilbane Beacon* (2009).

[21] water&stone. 2011 open source cms market share report.

[22] Wayne Powel, C. G. Web content management systems in higher education. *Educause Quarterly 2* (2003).

[23] Wordpress. Rest api. http://developer.wordpress.com/docs/api/.