

**THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY**

**A CLOUD-BASED APPLICATION DESIGN AND
DEVELOPMENT FOR IN-MEMORY DATA GRIDS**

M.S. Thesis

YILDIRIM MURAT ŐİMŐEK

İSTANBUL, 2013

**THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY**

**THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES
COMPUTER ENGINEERING**

**A CLOUD-BASED APPLICATION DESIGN AND
DEVELOPMENT FOR IN-MEMORY DATA GRIDS**

M.S. Thesis

YILDIRIM MURAT ŐİMŐEK

Supervisor: ASSOC. PROF. DR. V. AĐRI GÜNGÖR

İSTANBUL, 2013

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

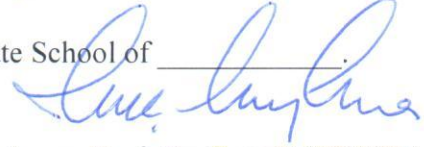
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
COMPUTER ENGINEERING

Name of the thesis: A Cloud-Based Application Design and Development For
In-Memory Data Grids

Name/Last Name of the Student: Yıldırım Murat Şimşek

Date of the Defense of Thesis: 13.06.2013

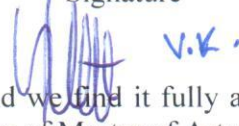
The thesis has been approved by the Graduate School of



Assoc. Prof. Dr. Tunç BOZBURA
Graduate School Director
Signature

I certify that this thesis meets all the requirements as a thesis for the degree of
Master of Arts.

Assist. Prof. Dr. Tarkan Aydın
Program Coordinator
Signature



This is to certify that we have read this thesis and we find it fully adequate in
scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Committee Members

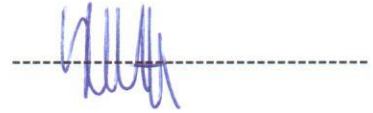
Signature

Thesis Supervisor
Assoc. Prof. Dr. V. Çağrı GÜNGÖR



Thesis Co-supervisor

Member
Assist. Prof. Dr. Selçuk BAKTIR



Member
Assist. Prof. Dr. Mehmet Alper TUNGA



ACKNOWLEDGMENTS

I would like to present my deepest thanks to my thesis supervisor Assoc. Prof. Dr. V. Çağrı G ng r for his valuable guidance, motivation and support throughout this thesis study.

I wish to express my love and gratitude to all my family. I would particularly like to thank my parents for their unlimited support in every stage of my life.

ABSTRACT

A CLOUD-BASED APPLICATION DESIGN AND DEVELOPMENT FOR IN-MEMORY DATA GRIDS

Yıldırım Murat Şimşek

Computer Engineering

Thesis Supervisor: Assoc. Prof. Dr. V. Çağrı Güngör

June, 2013, 116 pages

Cloud computing is defined as the use of computing resources (hardware, software, storage) over a network. It is a service that combines hardware, storage and interface (software) and is supplied for an end-user over the internet. With the help of this infrastructure that consists of strong components, end-users can easily realize the relevant operations on the internet. In this respect, cloud computing can be viewed as the next step for the internet development.

Internet and web technologies change and develop rapidly with each passing day. Also, the demands of end-users increase at the same velocity as these advances. End-users want to do their operations in a fast, reliable and consistent web environment. The establishment of such a web environment is impossible without technological infrastructure. For this reason, many big companies invest on large service infrastructures in order to meet the demands of end-users. In addition, it is required to make changes in the infrastructure because of the constantly increasing number of end-users. The establishment of an infrastructure that develops in accordance with the number of its users and meets all demands of end-users creates difficulties in terms of both cost and management. However, cloud computing with Oracle Coherence provides low-cost solutions because it is flexible, scalable and high performance.

In this study, three cloud-based applications that combine SaaS and PaaS models were developed by using Oracle Weblogic Server, Oracle Coherence and Hazelcast. With the help of the cloud-based architecture and applications, it was demonstrated that websites can work faster, more reliably and consistently. Performance results indicated that Oracle Coherence applications, are superior over the traditional database and the other in-memory data grid, Hazelcast, in terms of the response time, cpu use and memory usage.

Keywords: Cloud Computing, Oracle Coherence, In-Memory Data Grid

ÖZET

BELLEK-İÇİ VERİ KILAVUZLARI İÇİN BULUT-TEMELLİ UYGULAMA TASARIMI VE GELİŞTİRİLMESİ

Yıldırım Murat Şimşek

Bilgisayar Mühendisliği

Tez Danışmanı: Doç. Dr. V. Çağrı GÜNGÖR

Haziran,2013, 116 sayfa

Bulut bilişim, bilişim kaynaklarının (donanım, yazılım, depolama) bir ağ üzerinden kullanılması olarak tanımlanır. Bulut bilişim, donanımı, depolamayı ve arayüzü (yazılım) birleştiren ve internet üzerinden son kullanıcıya sağlanan bir servistir. Güçlü bileşenlerden oluşan bu altyapı yardımıyla, son kullanıcılar internet üzerinden servis alarak ilgili işlemlerini gerçekleştirebilirler. Bu yönüyle, bulut bilişim internet gelişiminin bir sonraki adımı olarak görülebilir.

İnternet ve web teknolojileri her geçen gün hızla değişip ilerlemektedir. Son kullanıcıların istekleri de bu ilerleme ile aynı hızda artmaktadır. Son kullanıcılar, ilgili işlemlerini hızlı, güvenilir, tutarlı bir web ortamında yapmak isterler. Böyle bir ortamın kurulması, teknolojik altyapı olmaksızın mümkün değildir. Bu nedenle, çoğu büyük şirketler son kullanıcıların isteklerini karşılayabilmek için büyük servis altyapılarına yatırım yapmaktadır. Ayrıca sürekli artan son kullanıcı sayısı sebebiyle altyapıda değişikliklerin yapılması gerekir. Kullanıcı sayısına göre büyüyen, son kullanıcıların tüm isteklerini karşılayan bir altyapının oluşturulması, gerek maliyet açısından gerek yönetim açısından zorluklar çıkarmaktadır. Ancak, Oracle Coherence ile bulut bilişim, esnek, ölçeklenebilir ve yüksek performanslı olduğu için düşük maliyetli çözümler sunmaktadır.

Bu çalışmada, SaaS ve PaaS modellerini birleştiren üç bulut-temelli uygulama, Oracle Weblogic sunucusu, Oracle Coherence ve Hazelcast kullanılarak geliştirilmiştir. Bu bulut-temelli mimari ve uygulamalar sayesinde, web sayfalarının daha hızlı, güvenilir ve tutarlı çalıştığı ortaya konmuştur. Performans sonuçları da göstermektedir ki Oracle Coherence uygulamaları, geleneksel veritabanına ve diğer bellek-İçi veri kılavuzu olan Hazelcast'e göre cevap süresi, cpu kullanımı ve bellek kullanımı açısından daha üstündür.

Anahtar Kelimeler: Bulut Bilişim, Oracle Coherence, Bellek-İçi Veri Kılavuzu

TABLE OF CONTENTS

| | |
|--|------|
| LIST OF TABLES..... | vii |
| LIST OF FIGURES..... | viii |
| LIST OF ABBREVIATIONS..... | x |
| 1. INTRODUCTION..... | 1 |
| 1.1 BACKGROUND | 1 |
| 1.2 TRADITIONAL CLIENT/SERVER VS. CLOUD COMPUTING..... | 2 |
| 1.3 CLOUD COMPUTING SERVICE MODELS..... | 5 |
| 1.4 CLOUD COMPUTING DEPLOYMENT MODELS..... | 6 |
| 1.5 THESIS OUTLINE..... | 8 |
| 2. LITERATURE REVIEW..... | 9 |
| 2.1 RELATED WORK ON CLOUD COMPUTING..... | 9 |
| 2.2 THE ORACLE CLOUD..... | 14 |
| 2.3 ORACLE COHERENCE..... | 17 |
| 2.3.1 Key Features of Oracle Coherence..... | 17 |
| 2.3.2 Caching Schemes..... | 18 |
| 2.3.2.1 The replicated scheme..... | 19 |
| 2.3.2.2 The distributed scheme..... | 21 |
| 2.3.2.3 The near scheme..... | 23 |
| 2.3.2.4 The local scheme..... | 25 |
| 3. ARCHITECTURAL DESIGN..... | 27 |
| 3.1 ARCHITECTURE OVERVIEW..... | 27 |
| 3.2 INSTALLATION PREREQUISITES..... | 29 |
| 3.3 INSTALLATION AND CONFIGURATION..... | 31 |
| 3.3.1 Installation of Weblogic and Coherence Servers..... | 31 |
| 3.3.2 Configuration of Weblogic and Coherence Servers..... | 35 |
| 3.3.3 Configuration of Cache and Cluster..... | 44 |
| 3.3.4 Installation and Configuration of Apache Web Server..... | 56 |
| 3.3.5 Installation and Configuration of Oracle Database..... | 57 |
| 3.4 DEVELOPMENT..... | 60 |
| 3.4.1 Java Development..... | 62 |

| | |
|--|------------|
| 3.4.2 Database Development..... | 85 |
| 4. PERFORMANCE ANALYSIS..... | 88 |
| 4.1 COMPARISON OF IN-MEMORY DATA GRIDS WITH THE | |
| DATABASE..... | 88 |
| 4.1.1 The Response Time Difference between Database and In-Memory | |
| Data Grids..... | 88 |
| 4.1.2 The Differences of CPU and Memory Usage between Database | |
| and In-Memory Data Grids..... | 94 |
| 4.2 COMPARISON OF DISTRIBUTED-REPLICATED-NEAR | |
| CACHING SCHEMES IN ORACLE COHERENCE..... | 100 |
| 4.2.1 The Response Time Difference between Distributed-Replicated- | |
| Near Caches..... | 102 |
| 4.2.2 The Differences of CPU and Memory Usage between | |
| Distributed-Replicated-Near Caches..... | 106 |
| 5. CONCLUSIONS AND RECOMMENDATIONS..... | 109 |
| REFERENCES..... | 112 |
| CURRICULUM VITAE..... | 116 |

LIST OF TABLES

| | |
|--|-----|
| Table 2.1: Comparison of Oracle with other cloud platform vendors..... | 16 |
| Table 2.2: Near cache invalidation strategies..... | 25 |
| Table 3.1: Oracle’s system requirements..... | 29 |
| Table 4.1: Comparison Of In-memory Data Grids With The Database..... | 100 |
| Table 4.2: Comparison Of Distributed-Replicated-Near Caching Schemes in Oracle Coherence..... | 108 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1.1: Service delivery models..... | 7 |
| Figure 2.1: Get operations in a replicated cache environment..... | 19 |
| Figure 2.2: Put operations in a replicated cache environment..... | 20 |
| Figure 2.3: Get operations in a distributed cache environment..... | 21 |
| Figure 2.4: Put operations in a distributed cache environment..... | 22 |
| Figure 2.5: Put operations in a near cache environment..... | 23 |
| Figure 2.6: Get operations in a near cache environment..... | 24 |
| Figure 3.1: The architecture of the cloud-based applications..... | 28 |
| Figure 3.2: Weblogic home directory..... | 32 |
| Figure 3.3: Installation types..... | 32 |
| Figure 3.4: JDK selection..... | 33 |
| Figure 3.5: Product installation selection..... | 34 |
| Figure 3.6: Installation of weblogic and coherence servers..... | 34 |
| Figure 3.7: Getting started with the configuration..... | 35 |
| Figure 3.8: Creating a new domain..... | 36 |
| Figure 3.9: Selecting the domain source..... | 36 |
| Figure 3.10: Specifying domain name and location..... | 37 |
| Figure 3.11: Configuring admin server..... | 38 |
| Figure 3.12: Configuring managed servers..... | 38 |
| Figure 3.13: The configuration summary..... | 39 |
| Figure 3.14: Completing the admin and managed server configurations..... | 40 |
| Figure 3.15: Admin console homepage..... | 41 |
| Figure 3.16: List of managed servers..... | 43 |
| Figure 3.17: Selecting the installation options of Oracle 11g database..... | 58 |
| Figure 3.18: Typical install configuration..... | 58 |
| Figure 3.19: Installing products..... | 59 |
| Figure 3.20: Configuration in progress..... | 60 |
| Figure 3.21: The overloaded database of the weather service..... | 61 |
| Figure 3.22: Comparing ExternalizableLite and Serializable in duration..... | 64 |
| Figure 3.23: Adding coherence.jar to the library..... | 64 |

| | |
|---|-----|
| Figure 3.24: Adding hazelcast.jar to the library..... | 65 |
| Figure 3.25: Deployment of active-cache and coherence-web-spi..... | 66 |
| Figure 3.26: Development with Eclipse..... | 68 |
| Figure 3.27: Deployments and status of the applications..... | 70 |
| Figure 3.28: Logging in the homepages of the bahce and bahcesehirhazelcast applications..... | 72 |
| Figure 3.29: Selecting data in the combo boxes..... | 74 |
| Figure 3.30: Database query home page..... | 76 |
| Figure 3.31: Coherence query home page..... | 78 |
| Figure 3.32: Hazelcast query home page..... | 80 |
| Figure 3.33: User operations in THESIS-BAU-M1-MAN1-2..... | 82 |
| Figure 3.34: Session management..... | 84 |
| Figure 3.35: Entity relationship diagram..... | 85 |
| Figure 3.36: TMETEOROLOGY table and total count of rows..... | 86 |
| Figure 4.1: The response time results for the database..... | 89 |
| Figure 4.2: The response time results for Oracle Coherence..... | 91 |
| Figure 4.3: The response time results for Hazelcast..... | 93 |
| Figure 4.4: The results of CPU usage in the database..... | 95 |
| Figure 4.5: The results of memory usage in the database..... | 95 |
| Figure 4.6: Physical I/O on the disk..... | 96 |
| Figure 4.7: The results of CPU and memory usage in Oracle Coherence..... | 97 |
| Figure 4.8: The results of CPU and memory usage in Hazelcast..... | 99 |
| Figure 4.9: The distributed cache server of Oracle Coherence..... | 101 |
| Figure 4.10: The replicated cache server of Oracle Coherence..... | 102 |
| Figure 4.11: The near cache server of Oracle Coherence..... | 102 |
| Figure 4.12: The response time results of the replicated scheme..... | 103 |
| Figure 4.13: The response time results of the near scheme..... | 105 |
| Figure 4.14: The results of CPU and memory usage in the replicated scheme..... | 106 |
| Figure 4.15: The results of CPU and memory usage in the near scheme..... | 107 |

ABBREVIATIONS

| | | |
|------|---|--|
| IT | : | Information technology |
| PC | : | Personal computer |
| NIST | : | National Institute of Standards and Technology |
| SaaS | : | Software as a service |
| PaaS | : | Platform as a service |
| IaaS | : | Infrastructure as a service |
| GUI | : | Graphical user interface |
| XML | : | Extensible markup language |
| JDK | : | Java development kit |
| JVM | : | Java virtual machine |

1. INTRODUCTION

1.1 BACKGROUND

It is evident that the technological changes in the field of computer industry have taken place at a faster pace than *Intel* co-founder, *Gordon E. Moore* estimated. These technological improvements are characterized by a rapid move from the slower to the faster, from the immovable (desktops) to the portable (laptops), from the more expensive to the cheaper.

In his keynote at the 46 th. annual *IDC Directions Conference*, Senior Vice President and Chief Analyst Frank Gens indicated “a significant fork in the IT road”, which resembles to the one that occurred 25 years ago (Preimesberger 2011, p. 1). The first of these turning points in computing technology concerns the launch of desktop computers (PC). In the late 1980s, new types of networks and computing platforms increased the users and the uses of IT because some data processing activities from central servers were transferred to client desktops and caused IT departments to manage and update the applications in thousands of desktop computers (Laszewski and Nauduri 2012, p. 1).

In the late 1990s, with the advent of internet browsers, users could access data and information on personal computers, smartphones and other digital devices without using any other software (Laszewski and Nauduri 2012, p. 1). As client/server and internet computing architectures were adopted by more users, the number of servers also increased in data centers and caused IT organizations to pay huge costs in order to maintain IT infrastructure (Laszewski and Nauduri 2012, pp. 1-2). This, in turn, led to the invention of grid computing, where cheap networked servers are grouped and function as a single large server. Instead of adding different systems, IT departments combined all servers in data centers and reduced the cost of operating data centers. This revolutionary move in the early 2000s later came to be known as cloud computing.

1.2 TRADITIONAL CLIENT/SERVER VERSUS CLOUD COMPUTING

It is true that cloud computing has been a fashionable word in the IT industry, but authorities cannot agree on a specific definition for the term. The term, “cloud”, probably comes from the widespread use of cloud images in the old telephone network diagrams and became popular with the announcement of *IBM* and *Google*’s collaborative project called “*Blue Cloud*” in 2007 (Harshbarger 2011, p. 231, Lamb 2011, p. 66).

Cloud computing can be defined as a type of computing where users can readily access IT resources via the Internet. In its simplest form, it is “the delivery of services over the Internet” (Lamb 2011, p. 66). From the *National Institute of Standards and Technology (NIST)*, Mell and Grance (2011, p. 2) gives the following definition that includes all of the various approaches to cloud computing:

“Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”

According to Gartner’s definition, cloud computing is “a style of computing where massively scalable IT-related capabilities are provided as a service using Internet technologies to multiple external customers” (Josyula et al. 2012, p. 9). In all of these definitions, cloud computing is easily accessible on demand over the Internet.

That’s why, Josyula et al. (2012, p. 10) consider cloud computing as a fourth utility after water, electricity and telephony. For instance, you may not know how electricity is produced or how it flows through your home, but you plug in to get electricity, and only have to pay your bill at the end of the month. Similarly, in the case of cloud computing, you needn’t buy your own IT infrastructure or worry about how your information is stored; instead, you plug in a monitor to get computing services, and only need to pay for your metered usage.

In this sense, cloud computing refers to all the systems “where software is deployed remotely from the user, usually in a vast data center with a lot of computing power, rather than locally on the desktop”, while users can access these systems through the Internet in order to input data, ask for tasks and collect results for local analysis (Lucas 2012, p. 51).

Apparently, there are varied definitions for cloud computing, but in each case, the main difference between traditional client/server computing and cloud computing is the same: “the user transitions from operating on their own mainframe to operating on an Internet-based architecture in the cloud” (Harshbarger 2011, p. 232). Davies (2012, p. 10) explains the change from traditional to new-generation internet-based computing as in the following: an IT manager used to buy a server, install it and provision power, network and cooling for it, which can take ten days; however, today, he can “simply go online and order a new web server or e-mail server and have it instantly deployed on [their] favorite cloud platform” within two minutes.

Therefore, the *NIST* cloud computing model promotes availability and consists of these five essential characteristics: *on-demand self-service* (the ability of a consumer to provision computing capabilities automatically without interacting with each service’s provider), *broad network access* (capabilities available over the network and accessible through standard mechanisms to heterogeneous platforms like mobile phones, laptops etc.), *resource pooling* (serving multiple consumers with different computing resources dynamically assigned and reassigned according to consumer demand), *rapid elasticity* (capabilities rapidly and elastically or even automatically provisioned for the consumer), and *measured service* (automatically controlling and optimizing resource usage by leveraging a metering capability) (Mell and Grance 2009, p. 1).

In short, these five unique characteristics of the *NIST* cloud computing model allows for “easier availability of metered, scalable, elastic computing resources that can be provisioned via self-service and can be accessed from any thin or thick client over the network” (Laszweski and Nauduri 2012, p. 2).

There are several advantages to using cloud computing for companies. The most immediate benefit of cloud computing is related to cost savings. Companies no longer need to invest huge amounts of money on hardware and software infrastructures because the cloud provider already provides them for their IT services. Also, they can save the labor expense that would otherwise be spent to train, implement and maintain these infrastructures because the cloud provider's staff provide support. Rosso (2010, p. 27) also stated that companies can increase or add capabilities without investing in new infrastructure or training new personnel thanks to the use of cloud applications.

Secondly, cloud computing makes it possible for employees to focus on their company's business because they do not have to install hardware and software infrastructures. In addition to time and cost savings, scalability is another determining factor in choosing to work with a cloud provider. As "the cloud provider is responsible for the computing resources necessary to support all of its customers", companies needn't worry about reliable access to scalable IT services (Harshbarger 2011, p. 234).

It is seen that small and medium enterprises choose to migrate to the cloud, as they lack the huge capital for such an IT setup, and cloud computing helps them to stop worrying about IT maintenance and to focus more on their core activities (Narayanan 2010, p. 36). For example, *Salesforce.com* is one of the cloud providers that gives small and medium-size companies the IT functionality that they would never afford to install and run on their own (Mansfield-Devine 2008, p. 9). On the other hand, Dean (2011, p. 67) pointed out that if one takes into consideration scalability, connectivity, high-density power and cooling advantages of cloud computing, the size of companies does not matter because cloud computing seems to be the optimal environment of hosting for both large and small enterprises.

In summary, cloud computing is an inexpensive, flexible, scalable and reliable technology that allows companies to access computing resources at any time and place, as long as they are connected to the Internet.

1.3 CLOUD COMPUTING SERVICE MODELS

Cloud computing service models refer to “the type of service that is being offered (i.e., hardware/software infrastructure, or application development, testing and deployment platform or enterprise software ready for use by subscription” (Laszewski and Nauduri 2012, p. 3). According to the NIST definition of cloud computing by Mell and Grance (2011, p. 1), the cloud model has three categories of services – three service models: *Software as a Service (SaaS)*, *Platform as a Service (PaaS)*, and *Infrastructure as a Service (IaaS)*.

The first of these three service models, *SaaS*, is known to be the earliest model of cloud computing where software companies sell their solutions to companies on the basis of the number of users with a specific set of service-level requirements (Laszewski and Nauduri 2012, p. 3). In this model, applications are easily accessible from various client devices through a thin-client interface like a web browser (e.g. web-based e-mail) (Josyula et al. 2012, p. 13). Here, the consumer doesn’t manage or control the underlying cloud infrastructure (network, servers, operating systems, storage etc.) excepting certain limited user-specific application configuration settings (Mell and Grance 2011, p. 2). In other words, vendors in the *SaaS* model provide hosted services for clients, including all of the necessary hardware, software and data storage, while an Internet browser is the only thing that is needed to operate the software (Lamb 2011, p. 67). The major players in the *SaaS* model include *Oracle*, *Cisco (Webex)*, *Microsoft*, *Salesforce.com* and *Google*.

In the *PaaS* model, the consumer doesn’t manage or control the underlying cloud infrastructure (network, servers, operating systems or storage), either; but has control over the deployed applications and possibly configuration settings for application-hosting environment (Josyula et al. 2012, p. 14). That is to say, the *PaaS* type of cloud computing provides users with platform stacks (*Linux*, *Apache*, *MySQL*) and facilitates the deployment of applications without buying and managing the underlying hardware and software (Josyula et al. 2012, pp. 13-14). With Amazon Elastic Compute Cloud, Savvis and Windows Azure are the major companies that provide the *PaaS* model.

The third service model, *IaaS*, enables the consumer to provision processing, storage, networks, and other fundamental computing resources, and also to deploy and run arbitrary software, including operating systems and applications (Mell and Grance 2011, p. 3). Again, “the consumer does not manage or control the underlying cloud infrastructure but has control over operating systems and deployed applications” (Josyula et al. 2012, p. 14). *Amazon EC2, Telstra, AT&T, Savvis, Amazon Web Services, IBM, HP and Sun* are among the major providers of the *IaaS* model.

Lamb (2011, p. 67) argued that although each one of the service models has a high level of complexity, the distinction between these three categories has become blurred – “cloudy if you will”.

1.4 CLOUD COMPUTING DEPLOYMENT MODELS

Cloud deployment models is concerned with “how the cloud services are made available to users”, and there are four deployment models associated with cloud computing: *public cloud, private cloud, hybrid cloud and community cloud* (Laszewski and Nauduri 2012, p. 5). As the name implies, *public cloud* is open to the general public, while it may be owned, managed and operated by a cloud service provider (Josyula et. al 2012, p. 11, Mell and Grance 2011, p. 3). In this type of cloud deployment model, all users are supported on a subscription basis, and public clouds are mostly used for application development and testing, non-mission-critical tasks like file-sharing as well as e-mail service (Laszewski and Nauduri 2012, p. 5).

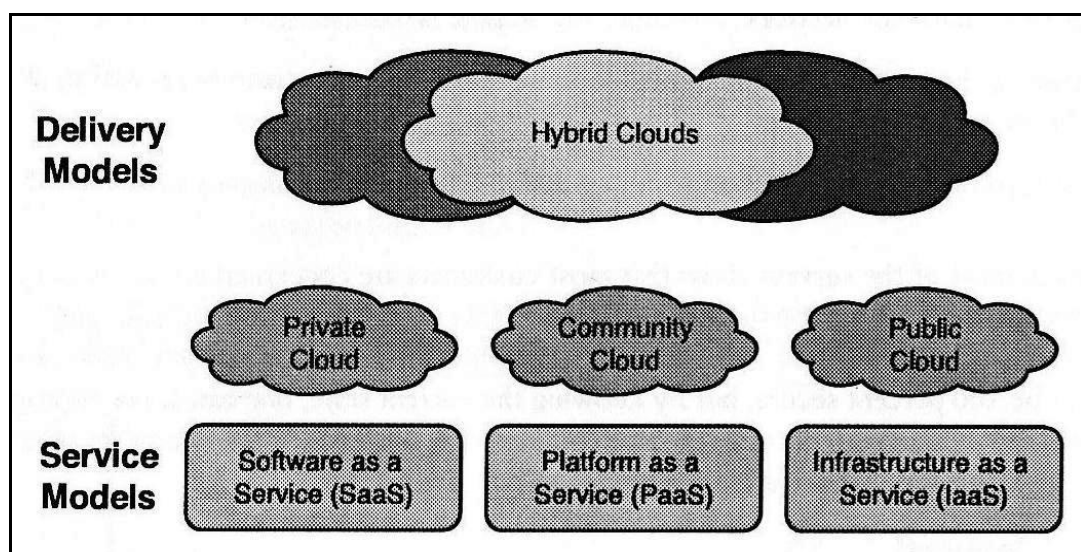
Unlike the external *public cloud*, *private cloud* is internal infrastructure exclusively used by a single organization in order to support various user groups and may exist on-site or off-site (Laszewski and Nauduri 2012, p. 5, Mell and Grance 2011, p. 3). Although *private clouds* are more expensive than *public clouds* because of the cost of acquiring and maintaining, they are better at addressing the security and privacy issues of organizations (Laszewski and Nauduri 2012, p. 5).

A hybrid cloud, on the other hand, combines two or more different cloud infrastructures (private and public) by technology and enables data and application portability, so that they can quickly scale up their IT infrastructure when they need: e.g. if they run out of capacity in a private cloud, they can rapidly reach out to the public cloud for additional resources and continue to operate their business (Josyula et al. 2012, p. 11, Laszewski and Nauduri 2012, p. 5, Mell and Grance 2011, p. 3).

Finally, in the case of a community cloud, the cloud infrastructure is exclusively used by a specific community of consumers from multiple organizations with shared concerns (mission, security, policy and compliance etc.): e.g. universities cooperating in specific areas of research or police departments within a state sharing computing services (Laszewski and Nauduri 2012, p. 5, Mell and Grance 2011, p. 3).

In brief, a public cloud provides service over the Internet, a private cloud over an intranet, and a hybrid cloud shares data and applications between public and private clouds; therefore, “clouds come in a number of forms, ranging from the completely private to the completely public” (Henderson et al. 2010, p. 2). Figure 1.1 illustrates service models and delivery models, and concludes that “all the services can be delivered on any of the cloud delivery models” (Josyula et al. 2012, p. 12).

Figure 1.1: Service delivery models



Source: Josyula et al. (2012), *Cloud Computing: Automating the Virtualized Data Center*

1.5 THESIS OUTLINE

In the present chapter, the following questions have been answered: 1. “what is cloud computing?”, 2. “how did it begin?”, 3. “in what way is cloud computing different from traditional client/server computing?”, 4. “what are the key features of cloud computing?”, 5. “what are the advantages of cloud computing?”, 6. “what kind of services are made available to consumers in cloud computing?” and 7. “how are the cloud computing services delivered to consumers?”.

So far, it has been emphasized that cloud computing is a short cut to providing and managing IT resources and applications as well as reducing the costs of hardware resources for many enterprises. The three fundamental forms of efficiency in the cloud computing model are also summarized by Jason Stowe of *Cycle Computing* as follows: *managing the peaks* (no necessity of purchasing and powering infrastructure to handle peak needs for an application), *economies-of-scale efficiencies* (provisioning large-scale server capacity only when used by multiple applications, and charging only for the resources being used by an application), *increased reliability and lower costs* (reducing the cost of disaster-recovery by enabling access to multiple computing and storage resources with large economies of scale for increased reliability) (Myatt 2010, p. 18).

For the aforementioned reasons, this study aims to design and develop a cloud-based application which combines *SaaS* and *PaaS* models by using *Oracle Weblogic Server*, *Oracle Coherence* and *Oracle Database*, so that websites developed with *Java* can work faster, more reliably and consistently. Moreover, the significance of cloud computing for IT organizations has been mentioned recently in several reports, the results of which will be presented in the second chapter. In addition to the literature review, the second chapter also introduces *Oracle Coherence* (in-memory data grid), and its relationship to cloud computing. In the third chapter, the design of a cloud-based application with *Oracle Coherence* is described. In the fourth chapter, the results of the performance analysis are discussed. Finally, the fifth chapter summarizes the findings of the study and concludes with recommendations for future research.

2. LITERATURE REVIEW

2.1 RELATED WORK ON CLOUD COMPUTING

While most of the literature on cloud computing focuses on the definition of cloud computing, kinds of cloud, deployment types and benefits to IT organizations, there are several other studies surveying the future trends of cloud computing as well as varied opportunities of application in different types of businesses. From now on, the relevant research on cloud computing is summarized.

In his article on storage solutions, Seymour (2013, p. 42) indicated that as the quantity of information grows, the problem of where to keep it increases, and as a result, it is estimated by IT Consultants, *Intergence*, that by 2020, 80% of its workers will be using cloud-based services for work, greatly benefiting mobile users, and local spending on cloud delivery will grow at an annual rate of more than 40% up to 2016.

In the survey, “*The Most Emergent Technologies of 2010*”, by *CIO Insight*’s research program, it was found that cloud computing, which was their tenth-most-emergent technology in the previous year (only 6% of the firms surveyed were developing in the cloud), is No. 2 on their hot-list because twice as many companies have started using it since then (Currier 2009, p. 29).

Cloud computing was also identified as the most strategic technology in two consecutive years (2010 and 2011) by market research/consulting provider *Gartner Inc.* *Gartner* analysts (2010, p. 1) also estimate that the next three years of cloud expansion will split along lines of delivery, either through vendors offering packaged private implementations of methodologies, software and hardware, or through remote management services for clients, and therefore, advise large enterprises to have cloud teams in place for decision management by 2012.

Dean (2011, p. 66), too, admitted that the cloud is fast-growing and cited *IDC's* predictions that in the next four years public IT cloud services will increase by more than 20 % to become a \$30 billion industry. Dean (2011, p. 66) went on to explain that 91 % of 2,000 IT professionals surveyed believe cloud computing will be the primary IT delivery model by 2015 according to the 2010 *IBM* survey.

New research from the *London School of Economics* and *Accenture* – based on a survey of 1,035 business and IT executives along with in-depth interviews with more than 35 service providers and other stakeholders – has shown that 20 % of respondents are already using cloud capabilities for a variety of services, including corporate email, websites, storage and customer relationship management, while similar numbers plan to make the transition in the next 18 months, which will double the cloud usage within a very short time (Willcocks, Venters and Whitley 2011, p. 1). Cloud computing was also regarded by two-thirds of business and IT executives as “a business service and an IT delivery model that drives innovation in organizations”; at the same time, “half of the executives see it as a new technology platform that can transform organizational forms” (Willcocks, Venters and Whitley 2011, p. 1).

Ernst & Young's *Global Information Security Survey*, “*Borderless Security*”, has indicated that “45 % of its 1,598 respondents from 56 countries have either deployed or are evaluating cloud computing”, which is evaluated as “a surprisingly high number given that the reliability and security level of many cloud services is still unknown” (Hyek 2011, p. 9).

According to a global survey by *KPMG International and Forbes Insight*, which was conducted in 15 countries (including US, Canada, Mexico, France, Germany, Ireland, Italy, Netherlands, Sweden, Switzerland, UK, Australia, China, India and Japan) with 806 senior executives and 123 executives from Cloud service providers in 2011, over half of the businesses and government enterprises have already conducted a full (24 %) or partial (35 %) cloud implementation of some functions (e-mail, sales management and other *SaaS* offerings) (Hill 2011, p. 2). For 19 % of the respondents, spending on cloud represents 10 % or more of total IT expenditures with 65 % citing 10 % or less

(Hill 2011, p. 2). Whereas 46 % of planned implementations are in a *SaaS* environment, significant numbers of end-users are also exploring the use of *IaaS* and *PaaS* models. Among the industries with the strongest adoption of *private clouds* are financial services, healthcare and diversified industrials (averaging 45 % of respondents) (Hill 2011, p. 3).

There are also other studies that relate to different business sectors adopting the cloud model. First of all, it is quite appealing to use the cloud in the construction industry because workers constantly change and new jobsite locations are frequently established (Sage 2012, p. 3). Now that many workers in the field need to access company data timely, traditional client/server software is not helpful because it only provides access to this information from designated locations (Sage 2012, p. 3). Therefore, construction companies should benefit from cloud technology in order to provide greater freedom and easy access to information from satellite offices, job site or customer locations around the world (Sage 2012, p. 3).

Furthermore, Garg (2011, pp. 4-10) explained how cloud computing can be applied to the financial services industry and gave prominent examples of financial services firms that have adopted the cloud model: *NYSE Euronext Capital Markets Community Platform* (a *PaaS* community cloud service for the financial services industry hosting customer applications and services like electronic trading, market data analysis, algorithmic testing and regulatory reporting), *I-Banks using cloud for risk analysis and non-core processes* (*IBM iDataPlex* servers as part of an *IaaS* strategy to build and evaluate risk analysis programs), and *Microsoft Azure DataMarket for the Energy Industry* (*Microsoft DataMarket SaaS* cloud services being used by different players in the energy industry for energy forecasting and analytics applications).

In their white paper, *Appirio* (2009, pp. 1-2) – a cloud solution provider with over 2500 customers and 150 consultants across 23 states – discussed why cloud computing is increasingly important to the services industry, and recommended that professional services organizations engage with their clients, consultants and the broader community by using next generation marketing techniques (instead of traditional on-premise

applications) as in their own example. *Appirio* (2009, p. 2) runs their communication and collaboration on *Google Apps*, their *CRM* on *Salesforce.com*, their marketing on *Marketo*, their financials on *Intacct*, and the core operations of their services business on *PS Enterprise* (a *PSA* application they've built on *Force.com*).

Apart from the construction, financial and professional services industries, the importance of cloud computing is now recognized by life sciences organizations. With the help of a life sciences cloud, biopharmaceutical companies can get direct access to the electronic medical records for the purposes of research and development, and thus, capitalize on its advantages ranging from the analysis of genomic, proteomic and clinical data to coordination of the supply chain, customer relationship management and safety surveillance (Henderson et al. 2010, pp. 4, 11).

Sultan (2010, pp. 112-113) draws attention to the potential of cloud computing for improving efficiency, cost and convenience in the educational sector as in the following examples from American, British, and even African educational establishments: *the University of California* that uses cloud computing in one of their courses for developing and deploying *SaaS* applications, *the Medical College of Wisconsin Biotechnology and Bioengineering Center* that makes protein research more accessible to scientists worldwide thanks to *Google's* powerful cloud-based servers, *Google* and *IBM's* cloud computing university initiative designed to improve computer science students' knowledge, some *UK* higher education institutions like *Leeds Metropolitan University*, *the University of Aberdeen*, *the University of Westminster*, which have adopted *Google Apps* on their students' demand, as well as some *East African* educational establishments like *the National University of Rwanda*, *the Kigali Institute for Education*, *the Kigali Institute for Science and Technology*, *the University of Nairobi*, *the University of Mauritius* providing their students with *Google* cloud services such as *Gmail*, *Google Calendar*, *Google Talk*, *Google Docs*, and *Spreadsheets*.

Another giant cloud provider, *Microsoft*, is also helping *Ethiopia* with 250,000 laptops, all running on *Microsoft's Azure* cloud platform, so that school teachers can download curriculum, follow academic records and securely transfer student data in the education system, without paying for a support system of hardware and software to connect them (Chan 2009, p. 1).

Among all the countries in the world, it is in fact the *U.S. Federal Government* that is leading the movement to the cloud; for example, *Microsoft* has opened a secure private cloud dedicated to the federal government; and the *Navy* is testing public clouds for use from its ships, along with the *Air Force* that is working with *IBM* to design a secure “military-grade” cloud, while President Obama has revealed a long-term cloud initiative (Lillard et al. 2010, p. 334).

In contrast to the North American example, Kuyucu (2011, pp. 462-463) concluded that Turkey is “a newcomer” in the field of cloud computing but is “bearing a great potential for cloud market”, and expected that the recently published *EU* cloud computing policy paper would directly cause Turkey to constitute a basis for policy-making because of its candidate status for EU harmonization.

Similarly, in the comprehensive portfolio on cloud computing technologies by Yapıcı (2010, p. 68) (“*Bulut Bilişim Dosyası*”), the giants of computer technologies like *CISCO*, *HP*, *IBM*, *Google* and *Microsoft* give detailed information about their cloud computing platforms. *HP* and *IBM* also argue that Turkey is one of the countries that needs cloud computing especially for SMEs (Small and Medium Size Enterprises) in the long run, and is already ready for cloud computing with its great economy, the present size and quality of its IT industry (Yapıcı 2010, pp. 79-81).

In conclusion, for some people, cloud computing could be an overhyped technology, whereas others see it as the next revolutionary move in information technology. All in all, cloud computing is increasing its popularity with individuals, companies, health – financial – legal – educational institutions and governments thanks to its cost-effective, reliable, practical and innovative technologies. Although the role of cloud computing is

newly understood in Turkey, the authorities are now taking action to capitalize on its resources. Consequently, cloud computing is not a passing fad but is here to stay.

2.2 THE ORACLE CLOUD

Oracle is known as the chief company which invests largely in the architectures of grid and cloud computing. As their slogan suggests: “Hardware and Software, Engineered to Work Together”, Oracle continually creates new inventions in the fields of software and hardware. This is achieved by “increasing data-center efficiency”; more specifically, “by increasing resource utilization, automation in provisioning, management and monitoring, scalability, and the leveraging of off-the-shelf components and technologies” (Laszewski and Nauduri 2012, p. 7).

Oracle’s strategy can be summed as supporting various cloud computing service and deployment models with “open, complete, and intergrated sets of products from application to disk” (Laszewski and Nauduri 2012, p. 8). Being the recognized leader of cloud services, Oracle provides more than 25 million users on a daily basis with a wide range of services like SaaS and PaaS.

Laszewski and Nauduri (2012, p. 8) lists the following products and technologies by *Oracle*: *SaaS capability* with its *Siebel CRM* solution on demand and on a subscription basis (like *Salesforce.com*), *IaaS products* such as *Oracle Database 11g* and the *Oracle Fusion Middleware 11g* family available on most public cloud service providers (like *Amazon EC2* and *Savvis*), and *PaaS products* for both cloud providers and enterprises that are planning to build their own private clouds with a *DIY* (do-it-yourself) option. As for the *Oracle* products that enable cloud computing, *Oracle Exadata Database Machine*, *Oracle Exalogic Elastic Cloud*, *Oracle VM*, *Oracle Assembly Builder*, and *Oracle Enterprise Manager* can be counted.

Obviously, *Oracle* is not the only cloud provider on earth. Today, there are many different companies that sell similar capabilities such as databases, application servers, development tools, and platforms. According to Laszewski and Nauduri (2012, pp. 13-14), there are at least seven reasons why *the Oracle Cloud* should be preferred over other vendors:

1. “*Oracle* is the only company that offers a full range of products comprising business applications, enterprise software (e.g., databases), application servers, and hardware systems including servers, storage, and networking tools...”
2. “*Oracle* is the first company to offer a highly optimized database platform and compute platforms by combining the latest in hardware and software technologies (*Oracle Exadata, Exalogic*)...”
3. “*Oracle* has a history of bringing important technologies to the realm of mainstream workloads, such as the use of encryption, compression by regular *OLTP*, and data warehousing applications...”
4. “*Oracle* has had a history of providing enterprise software that enables consolidation, scalability, and capacity on demand (e.g., *Oracle Database Real Application Clusters* or *RAC*) since the release of *Oracle Database 10g* and *Oracle Application Server 10g*...”
5. “*Oracle* offers out-of-the-box integration of various features and functionalities in its software offerings [e.g., *Oracle Business Intelligence Enterprise*]...”
6. “*Oracle* software products offer unique features, such as *Oracle Database* row-level locking, which avoids locking rows of data in tables when reading them” as well as “*Oracle Database 10g Real Application* clusters, which allow active-active clustering of multiple instances accessing a single *Oracle* database ...”

7. “Organizations use an array of tools and technologies to manage and monitor different layers of IT system”, but “*Oracle’s consolidated management software, Oracle Enterprise Manager (OEM) Grid Control, can monitor and manage all components in a data center...*”.

Table 2.1 also highlights the major differences between Oracle and other software and hardware sellers in the cloud infrastructure field (Laszewski and Nauduri 2012, p. 14):

Table 2.1: Comparison of Oracle with other cloud platform vendors

| Oracle | Other Vendors |
|--|--|
| Engineered systems (hardware and software) | Usually hardware- or software-only solution |
| Out-of-the-box integrated Oracle products and features | Varying degrees of integration required based on vendor products used |
| Superior performance, cost/performance out of the box | Performance dependent on optimization of all layers |
| Supports all types of workloads (OLTP, DW/DSS) | Appliances exist for specific workloads, mostly concentrating on DW/BI analytics |
| Reduced total cost of ownership due to use of standard x86-based architecture, high-performance characteristics, and engineered system requiring less maintenance effort | Total cost of ownership varies depending on configuration used |
| Unified management infrastructure with applications to disk monitoring capabilities | Requires multiple products to manage and monitor the full stack |

Source: Laszewski & Nauduri (2012), *Migrating to the cloud*

2.3 ORACLE COHERENCE

There are three ways of accessing data from the cloud: *exposing the data using web services*, *replicating the data to the cloud*, and *making data available on the cloud*. The first of these may seem the simplest and most common approach to store data on-premise in many cases. However, *exposing the data using web services* is disadvantageous in that if a request comes from user to the cloud server, the cloud server must request data from the on-premise web service – which increases the response latency, because each user request must go through an additional network hop before it can be satisfied (Gigaspace 2012, p. 3). The second approach, *replicating the data to the cloud* has its own drawbacks: it stores sensitive data on a public cloud and adds the complexity of synchronizing the data and coming up with an appropriate replication strategy (Gigaspace 2012, p. 3). The third solution, *making data available on the cloud*, involves the use of an *in-memory data grid*, so that the data is not stored on the cloud but is made available for processing in the cloud layer, and any changes made to the data are reflected back upon the on-premise data store almost in real time (Gigaspace 2012, p. 4). The fact that the data is available only in memory and never stored on a disk in the cloud is the main strength of an *in-memory data grid*. In-memory replication is also more efficient in that it enables replication in milliseconds rather than a typical delay of minutes for disk replication solutions (Gigaspace 2012, p. 4).

Seović et al. (2010, p. 2), too, emphasized that *in-memory data grids* have become increasingly popular over the last few years because it can solve many of the problems related to performance and scalability, and also improve availability of the system. For this reason, in this study, *Oracle Coherence* is used as the in-memory data grid solution due to its advantages of performance, reliability and scalability.

2.3.1 Key Features of Oracle Coherence

Seović et al. (2010, p. 2) gives the following definition for *Oracle Coherence*: “...an *In-Memory Data Grid* that allows you to eliminate *single points of failure* and *single points of bottleneck* in your application by distributing your application’s objects and related

processing across multiple physical servers”. According to Seović et al.’s (2010, p. 2) definition, *Oracle Coherence* has several important features: managing application objects that are ready for use within the application, distributing application objects across many physical servers, preventing any loss of data or in-flight operations, storing data in memory for high performance and low latency in data access, and distributing both application objects and the processing on these objects.

In the same way, *Oracle* (2012a, p. 2) groups the key features of *Coherence* into four: *caching*, *analytics*, *transactions*, and *events*. *Caching* refers to storing data in the data grid so that a single and consistent view of cached data can be obtained. Also, “reading from the cache is faster than querying back-end data sources...” (*Oracle* 2012a, p. 2). *Analytics* is concerned with querying and analyzing data in memory. With the help of custom analytical functions, *Oracle Coherence* supplies ready-made support for searching, aggregating, and sorting data. In this way, “it parallelizes operations across the entire data grid, ensuring that server failures or slowdowns do not affect calculation results” (*Oracle* 2012a, p. 2). *Transactions* relates to managing transactional data in memory inside the data grid. *Oracle Coherence* guarantees the best in-memory replication and data consistency because “it is suitable for managing transactions in memory until they are persisted to an external data source for archiving and reporting” (*Oracle* 2012a, p. 2). *Events* involves responding in real time to data changes within the data grid. Thanks to its server-side stream processing and interactive technologies (e.g. continuous query) for real-time desktop applications, *Oracle Coherence* provides event-handling technologies capable of handling intense event rates (*Oracle* 2012a, p. 2).

2.3.2 Caching Schemes

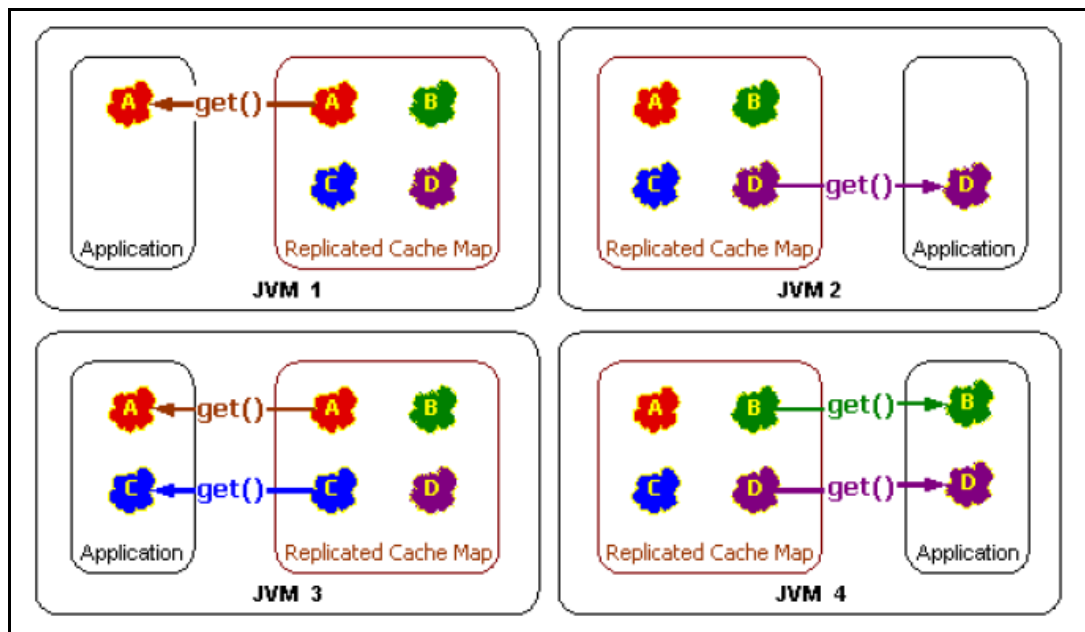
The mechanism that is used for organizing data is called a *scheme*, and *Oracle Coherence* configures four main types of *caching schemes*: *the replicated cache*, *the distributed cache*, *the near cache*, and *the local cache*.

2.3.2.1 The replicated scheme

A *replicated cache* replicates its data to all cluster nodes; that means, “every node [JVM] in the grid that is running the Replicated Cache service has the full dataset within a backing map for that cache” (Seović et al. 2010, p. 81).

As for the *Read* performance in the *replicated scheme*, “the best part of a replicated cache is its access speed” (Ruzzi 2011, p. 9-1). Figure 2.1 shows that “all the data is local to each node” in the replicated scheme; that is, “an application running on that node can get data from the cache at in-memory speed” (Ruzzi 2011, p. 9-1, Seović et al. 2010, p. 82). This is known as “zero latency access”, and “is perfect for situations in which an application requires the highest possible speed in its data access” (Ruzzi 2011, p. 9-1).

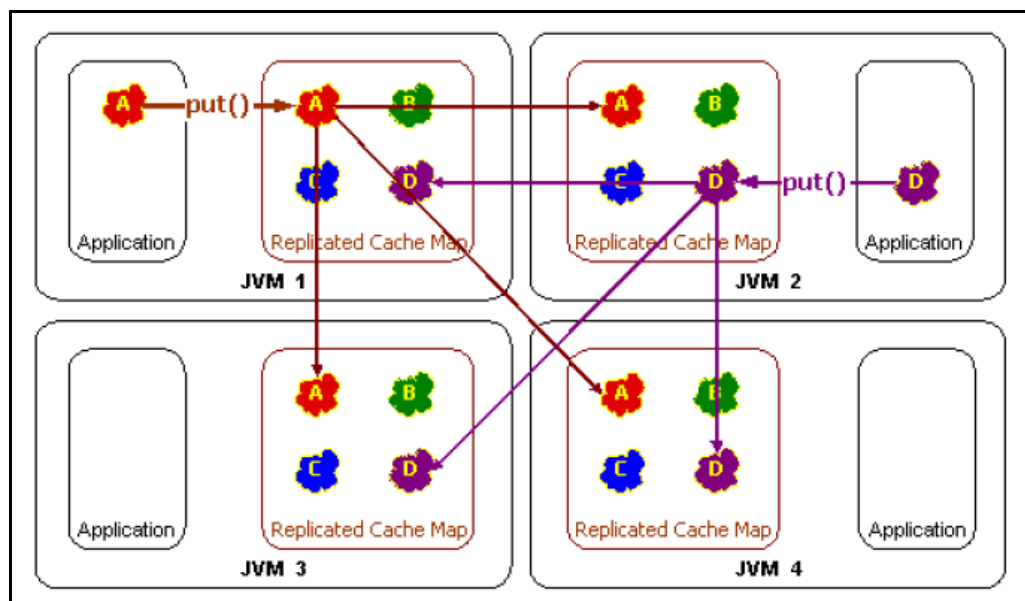
Figure 2.1: Get operations in a replicated cache environment



Source: Ruzzi (2011), *Oracle coherence getting started guide, release 3.7*

As can be seen in Figure 2.2, the *Write* performance of a replicated cache is poor because it has to distribute the operation to all the nodes in the cluster and receive their confirmation that the operation was done successfully (Ruzzi 2011, p. 9-1, Seović et al. 2010, p. 82). This communication with all the other nodes increases “the amount of network traffic and the latency of write operations against the replicated cache” (Seović et al. 2010, p. 82).

Figure 2.2: Put operations in a replicated cache environment



Source: Ruzzi (2011), *Oracle coherence getting started guide, release 3.7*

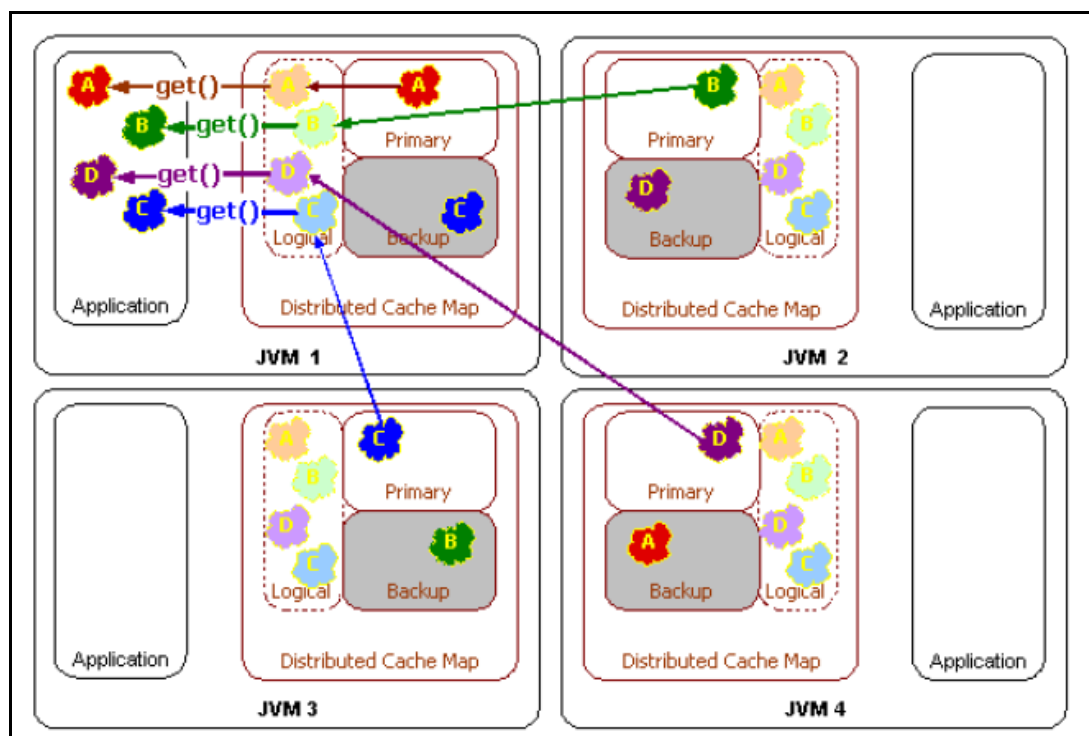
Another limitation of the replicated scheme relates to *data set size*. Because each node holds all the data in the replicated scheme, the total size of all replicated caches is determined by the amount of memory available in a single node (Seović et al. 2010, p. 83). However, when it comes to *fault tolerance*, replicated caches are very elastic in cases of failure, because “there are essentially as many copies of the data as there are nodes in the cluster” (Seović et al. 2010, p. 85). If a node fails, a replicated cache only needs to redirect *read* operations to other nodes and to ignore *write* operations sent to the failed node (Seović et al. 2010, p. 85). The recovery from the failure is really simple because the new node can easily copy all the data from another node. In short, the replicated scheme presents a good choice for “small-to-medium size, read-only or read-mostly data sets” (Seović et al. 2010, p. 85).

2.3.2.2 The distributed scheme

In order to solve the scalability limits of the *replicated scheme*, a *distributed scheme* is configured by *Coherence*. As the name implies, a *distributed cache* is “a collection of data that is distributed (or partitioned) across any number of cluster nodes such that exactly one node in the cluster is responsible for each piece of data in the cache” (Ruzzi 2011, p. 8-1).

The *Read* performance of a *distributed cache* is not as high as a *replicated scheme* because “the reads coming from any single node will require an additional network call”; that is, “if there are n cluster nodes, (n-1)/n operations go over the network” as in Figure 2.3 (Ruzzi 2011, p. 8-1, Seović et al. 2010, p. 86):

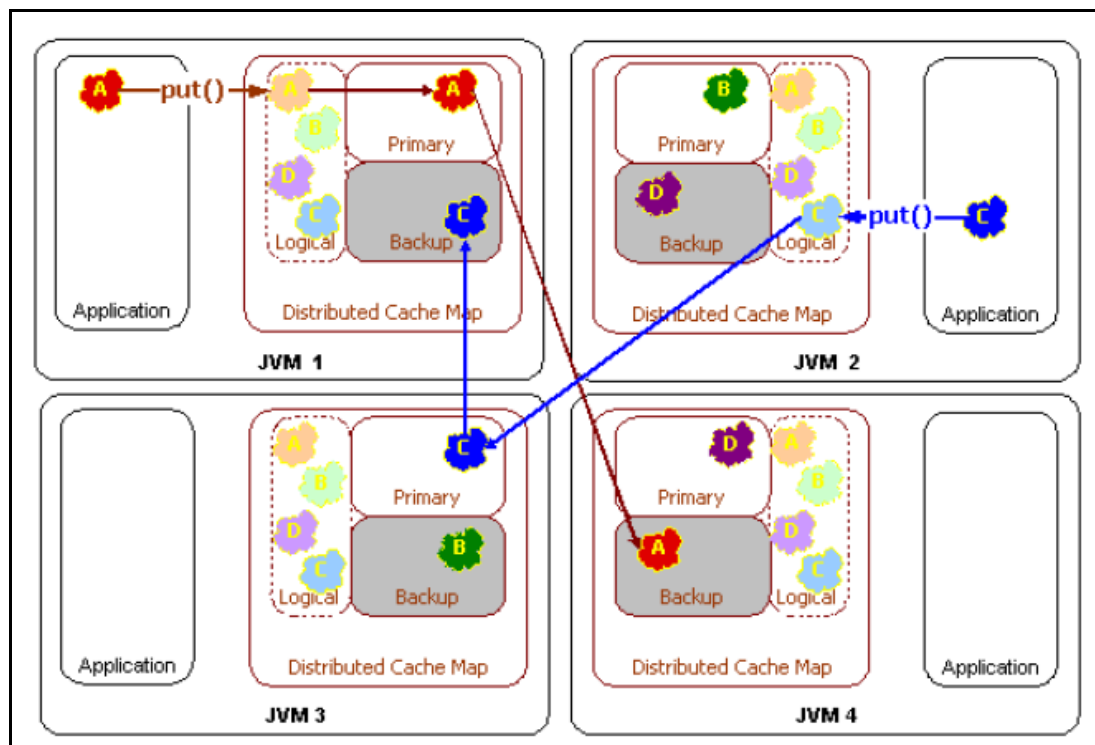
Figure 2.3: Get operations in a distributed cache environment



Source: Ruzzi (2011), *Oracle coherence getting started guide, release 3.7*

The *Write* performance of a *distributed cache* is similar to its *Read* performance in that write operations will also require network access most of the time and use point-to-point communication to achieve the goal in a single network call as in Figure 2.4 (Seović et al. 2010, p. 87):

Figure 2.4: Put operations in a distributed cache environment



Source: Ruzzi (2011), *Oracle coherence getting started guide, release 3.7*

In terms of *data set size*, the distributed scheme is more convenient, because the size of the data set is determined by the total amount of space available to all the nodes in the cluster (Seović et al. 2010, p. 89). As a result, it is possible to manage very large data sets in memory and to scale the cluster in such a way that it can handle growing data sets by only adding more nodes (Seović et al. 2010, p. 89). This support for very large in-memory data sets is the reason for preferring a *distributed scheme* over a *replicated one*.

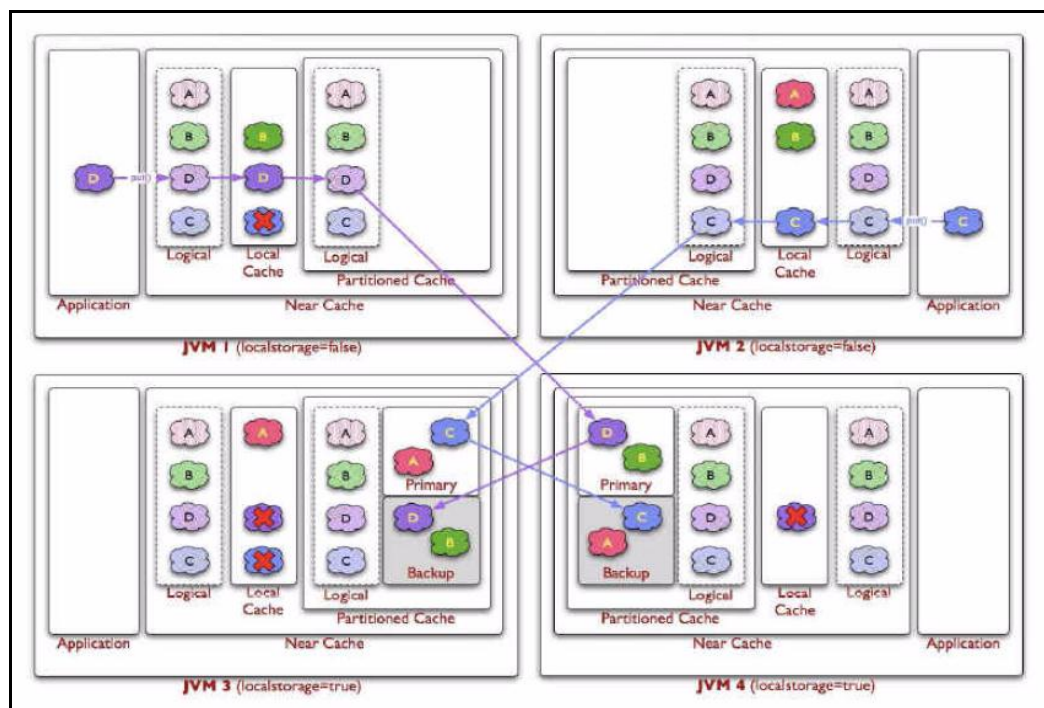
When it comes to *fault tolerance*, the distributed scheme enables users to keep more than one backups of cache data to prevent data loss. In case of a node failure, a distributed cache will notify all the other nodes to have backup copies of the data on the

failed node and also to create new backup copies on different nodes (Seović et al. 2010, p. 90). In addition, when the failed node recovers or a new node joins the cluster, the distributed cache will return some of the data to it by repartitioning the cluster and asking all the other members to move some of their data to the new node (Seović et al. 2010, p. 90). Briefly, the distributed scheme is preferable for large, growing data sets and write-intensive applications.

2.3.2.3 The near scheme

A *near cache* is defined as “a hybrid, two-tier caching topology that uses a combination of a local, size-limited cache in the front tier, and a partitioned cache in the back tier to achieve the best of both worlds...” (Seović et al. 2010, p. 90). In other words, *the near scheme* combines two caches – a front cache and a back cache “that automatically and transparently communicate with each other by using a read-through/write-through approach” (Ruzzi 2011, p. 11-1). In this way, the near scheme is enabled to have the same zero-latency read access as a replicated cache and to become extremely scalable like a distributed cache.

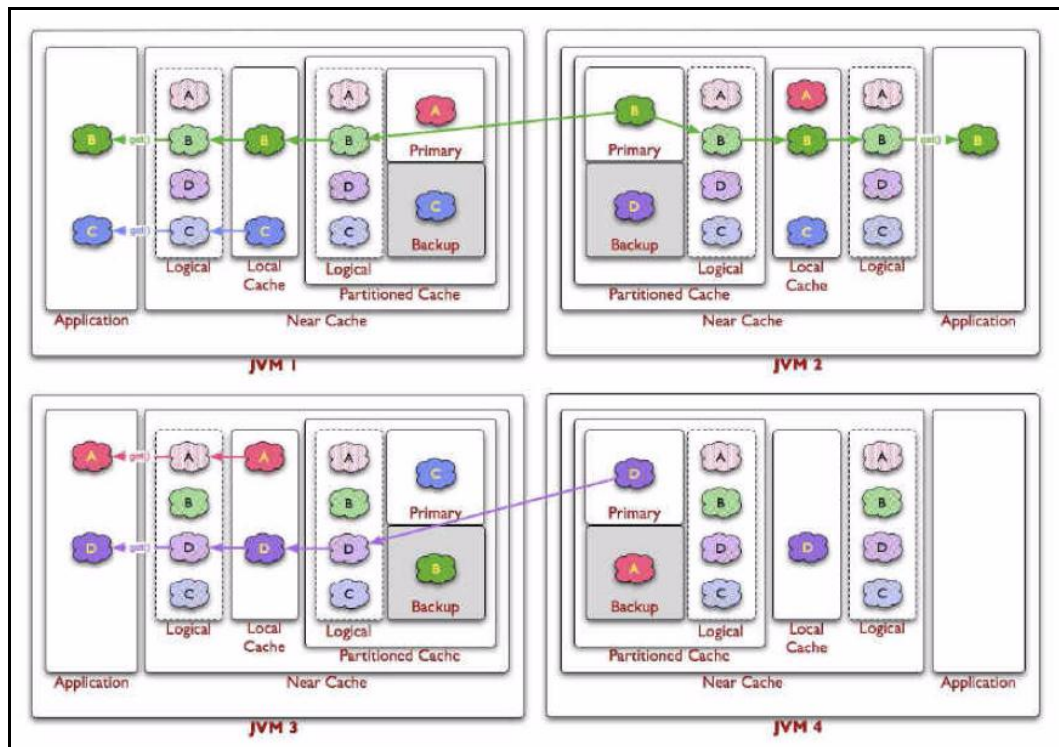
Figure 2.5: Put operations in a near cache environment



Source: Ruzzi (2011), Oracle coherence getting started guide, release 3.7

In Figure 2.5, the data flow in a *near scheme* can be seen: if the client writes an object D into the grid, the object is placed in the local cache inside the local JVM and in the distributed cache backing it (with a backup copy), and if the client requests the object, it can be obtained from the local, or front cache, in object form with no latency (Ruzzi 2011, p. 11-1). Figure 2.6 illustrates the get operations in a near scheme: if the client requests an object that has been expired or invalidated from the front cache, it is automatically retrieved from the distributed cache, and the front cache is updated with the object before it is delivered to the client (Ruzzi 2011, p. 11-2).

Figure 2.6: Get operations in a near cache environment



Source: Ruzzi (2011), *Oracle coherence getting started guide, release 3.7*

In a *near cache* environment, there are also *invalidation strategies* that keep the *front cache* of the *near cache* synchronized with the *back cache*. According to Ruzzi (2011, p. 11-3), *the near scheme* can be configured to listen to certain events in the *back cache* and automatically update or invalidate entries in the *front cache*. Table 2.2 summarizes the four different strategies of invalidating the front cache entries that have changed by other processes in the back cache (Ruzzi 2011, p. 11-3):

Table 2.2: Near cache invalidation strategies

| Strategy Name | Description |
|---------------|--|
| None | This strategy instructs the cache not to listen for invalidation events at all. This is the best choice for raw performance and scalability when business requirements permit the use of data which might not be absolutely current. Freshness of data can be guaranteed by use of a sufficiently brief eviction policy for the front cache. |
| Present | This strategy instructs the Near Cache to listen to the back cache events related only to the items currently present in the front cache. This strategy works best when each instance of a front cache contains distinct subset of data relative to the other front cache instances (for example, sticky data access patterns). |
| All | This strategy instructs the Near Cache to listen to all back cache events. This strategy is optimal for read-heavy tiered access patterns where there is significant overlap between the different instances of front caches. |
| Auto | This strategy instructs the Near Cache to switch automatically between Present and All strategies based on the cache statistics. |

Source: Ruzzi (2011), *Oracle coherence getting started guide, release 3.7*

In brief, the near cache is the best scheme for the read-mostly or balanced read-write caches that must support data sets of any size because it enables users to achieve the read performance of a replicated scheme along with the write performance and scalability of a distributed scheme (Seović et al. 2010, p. 94).

2.3.2.4 The local scheme

A *local cache* is not a clustered service: it is “a cache that is local to (completely contained within) a particular cluster node” (Ruzzi 2011, p. 10-1). However, the local scheme is used in combination with various clustered cache services: commonly used both as the backing map for *replicated* and *distributed caches* and as a front cache for *near caches* (Ruzzi 2011, p. 10-1, Seović et al. 2010, p. 95).

Since it stores all the data on the heap, *the local cache* provides the fastest access speed, both read and write, in contrast to other backing map implementations (Seović et al. 2010, p. 95). *The local cache* can be size-limited: it can restrict the number of entries it caches and automatically evict (remove) entries when the cache becomes full (Ruzzi 2011, p. 10-1).

In addition, the sizing of entries and the eviction policies can be customized: for example, to determine the cache items to be evicted, a combination of LRU (Least Recently Used), LFU (Least Frequently Used) or HYBRID information can be used (Seović et al. 2010, p. 95). According to Ruzzi (2011, p. 10-1), “the local cache supports automatic expiration of cached entries”; in other words, one can also configure *the local cache* to expire cache items on the basis of their age.

3. ARCHITECTURAL DESIGN

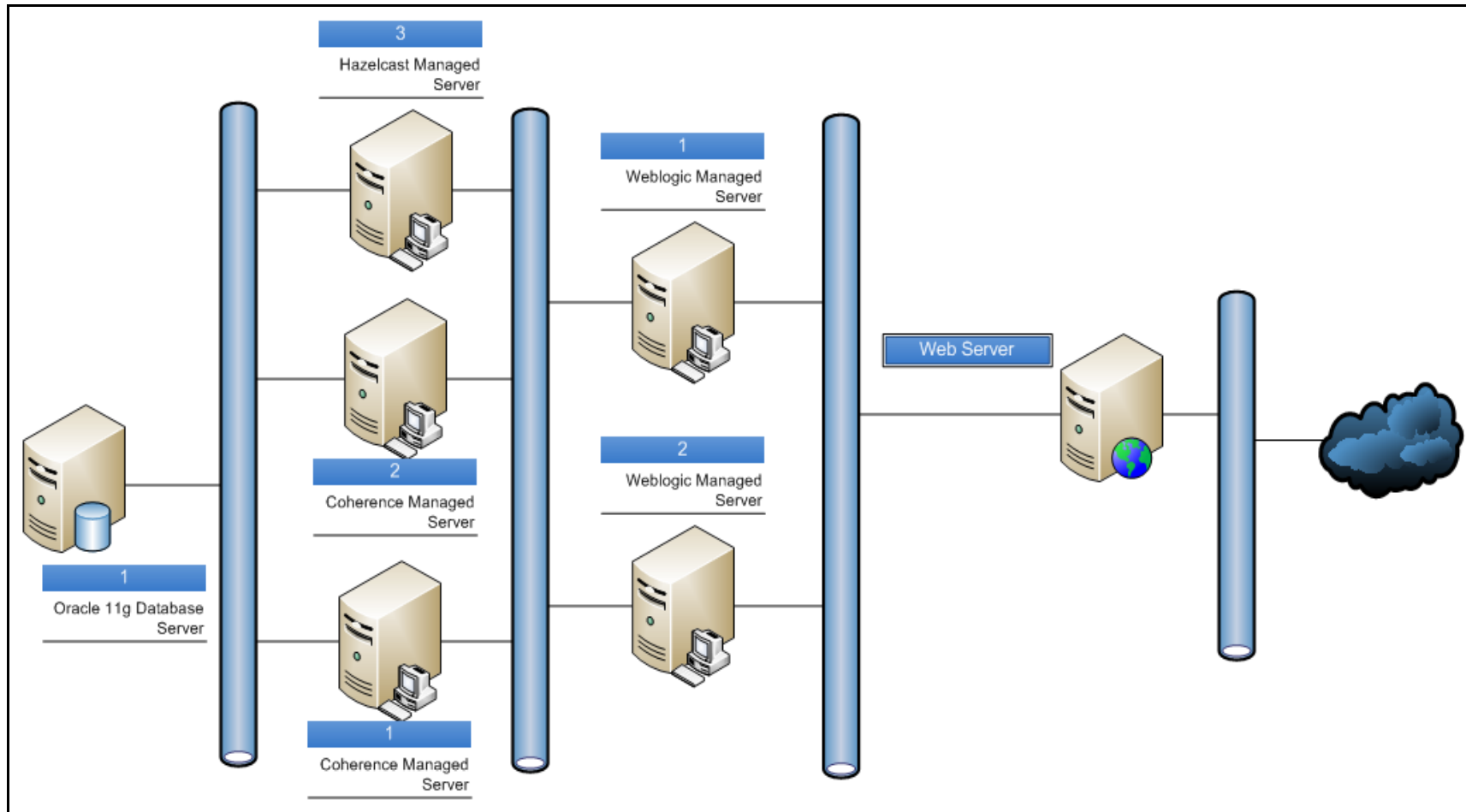
3.1 ARCHITECTURE OVERVIEW

In this study, three cloud-based applications have been designed and developed by using *Java* and two different products of *in-memory data grid*: *Oracle Coherence* and *Hazelcast*. It is considered that the use of *Oracle Coherence* in cloud computing will ensure a rapid, reliable and consistent web environment.

This chapter presents a description of a load-balanced, fault-tolerant and dynamic architecture for optimizing the scalability, reliability, and performance advantages of *Oracle Coherence* in accessing and storing data. For this reason, the following architecture illustrated in Figure 3.1 has been designed by using an *Apache Webserver*, two *Weblogic* managed servers, two *Coherence* managed servers, one *Hazelcast* server and an *Oracle Database*.

The components of this architecture and their specific functions are described as follows: 1. a webserver used for load-balancing and as reverse proxy, 2. two weblogic managed servers used for deploying the web application that will communicate with the Coherence cache, 3. two *Coherence* managed servers used for storing the cache from the *Oracle Database*, 4. a *Hazelcast* server used for both storing the cache from the *Oracle Database* and deploying the Hazelcast web application, 5. an *Oracle Database* used for storing the data in the tables.

Figure 3.1: The architecture of the cloud-based applications



3.2 INSTALLATION PREREQUISITES

Before installing the above servers in the architecture, it is necessary to decide whether to use a physical or a virtual machine. Here, *Oracle's* system requirements should be met. Table 3.1 shows the system requirements determined for *Oracle's* products (Oracle 2013, p. 2-4).

Table 3.1: Oracle's system requirements

| Component | Requirement |
|-------------------------|---|
| Platform configuration | <p>A supported configuration of hardware, operating system, JDK, and database specific to the product you are installing.</p> <p>For the most up-to-date information about other prerequisites and recommendations, such as recommended versions of the JDK, see the Oracle Fusion Middleware Supported System Configurations page at http://www.oracle.com/technology/software/products/ias/files/fusion_certification.html.</p> |
| Processor | 1-GHz CPU |
| Hard disk drive | <p>A complete installation (including SDKs) requires approximately 3.9 GB of disk space. This includes temporary disk space that is needed during installation. Depending on the components you choose to install, and the installer that you are using, less disk space may be needed.</p> <p>For more information, see Section 2.3.4, "Temporary Disk Space Requirements."</p> |
| Memory | A minimum of 1 GB RAM, although Oracle recommends 2 GB of RAM. |
| Color bit depth display | <p>For graphical-mode installation, 8-bit color depth (256 colors) is required.</p> <p>For console-mode and silent-mode installation, there is no color bit depth requirement.</p> |
| JDK | <p>The installation program requires a Java run-time environment (JRE) to run. A JRE is bundled in the Windows 32-bit and Linux x86 installation programs, as well as in some UNIX installation programs (those with file names ending in .bin).</p> <p>For other platforms, the installation program does not install a JDK. File names for these installation programs end in .jar. To run the .jar installation programs, you must have the appropriate version of the JDK installed on your system, and include the bin directory of the JDK at the beginning of the PATH variable definition.</p> <p>Note: It is important that you use a JDK because the installation process assigns values to JAVA_HOME and related variables to point to the JDK directory. All scripts installed by the installation program use this JDK by default, including scripts to start sample applications, the Configuration Wizard, and other development tools.</p> |

Source: Oracle Fusion Middleware Installation Guide for Oracle Weblogic Server, 2013

After choosing the virtual machine, any one of these three operating systems, Linux, Windows or MacOS can be preferred. Because of its performance, use of CPU and memory as well as management, Linux is used in this study. The system features of the virtual machine can be obtained by the following commands below:

1. Kernel Version and System Architecture: `uname -a`

```
[wlsadmin@murat ~]$ uname -a
Linux murat 2.6.32-279.el6.x86_64 #1 SMP wed Jun 13 18:24:36
[wlsadmin@murat ~]$
```

```
Wed Jun 13 18:24:36 EDT 2012 x86_64 x86_64 x86_64 GNU/Linux
```

2. OS Distribution Information: `cat /etc/*-release`

```
[wlsadmin@murat ~]$ cat /etc/*-release
Red Hat Enterprise Linux Server release 6.3 (Santiago)
Red Hat Enterprise Linux Server release 6.3 (Santiago)
[wlsadmin@murat ~]$
```

3. Total Ram of The System: `grep MemTotal /proc/meminfo`

```
[wlsadmin@murat ~]$ grep MemTotal /proc/meminfo
MemTotal:      16334504 kB
[wlsadmin@murat ~]$
```

4. CPU(s) Info of The System: `grep "model name" /proc/cpuinfo`

```
[wlsadmin@murat ~]$ grep "model name" /proc/cpuinfo
model name      : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
model name      : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
model name      : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
model name      : Intel(R) Xeon(R) CPU           X5570  @ 2.93GHz
[wlsadmin@murat ~]$
```

5. Disk Info of The System: `df -h`

```
[wlsadmin@murat ~]$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/mapper/vgroot-lvroot
                20G   6.1G   13G   33% /
tmpfs            7.8G   0     7.8G   0% /dev/shm
/dev/sda1        485M   53M   407M  12% /boot
/dev/mapper/data-data1v
                49G   2.8G   44G    6% /data
[wlsadmin@murat ~]$
```

6. Java Info: `java -version`

```
[wlsadmin@murat ~]$ /data/jdk1.6.0_37/bin/java -version
java version "1.6.0_37"
Java(TM) SE Runtime Environment (build 1.6.0_37-b06)
Java HotSpot(TM) 64-Bit Server VM (build 20.12-b01, mixed mode)
[wlsadmin@murat ~]$
```

It is now possible to start the aforementioned installations within the architecture after meeting these installation prerequisites and system controls.

3.3. INSTALLATION AND CONFIGURATION

First of all, weblogic application servers and coherence servers are installed. Then, Hazelcast, apache webserver and finally Oracle Database are installed. The most important point of installation stage is keeping a record of all the data requested by installation products.

3.3.1 Installation of Weblogic and Coherence Servers

Weblogic and Coherence servers can be installed in three ways: 1. the graphical mode is “an interactive, GUI-based method” for installing the software, and “can be run on both Windows and UNIX systems”, 2. the console mode is “an interactive, text-based method” for installing the software “from the command line, on either a UNIX system or a Windows system”, and 3. the silent mode is “a non-interactive method” where one can specify the installation options by using an XML properties file, and can be run “from either a script or from the command line” (Oracle 2013, p. 2-11).

In this study, the graphic mode is preferred in order to illustrate how the installations are performed, and it is necessary to convert the Linux responses into the graphical mode through Exceed application. The binary files of weblogic and coherence servers are downloaded from the *Oracle*'s website and copied to the machine. After that, the file copied to the directory is extracted with the command, “java -jar wls1036_generic.jar”, and so the binary files of weblogic are now installed. After the installation of the binary files, servers are installed by running the command “./config.sh” in the path, “cd/common/bin”.

When “config.sh” script is run, the screenshot of weblogic home directory appears as in Figure 3.2. While naming the home directory, one must indicate in the folder name

which user and which version of weblogic is being used. This will help to quickly learn the version of weblogic and its user.

Figure 3.2: Weblogic home directory

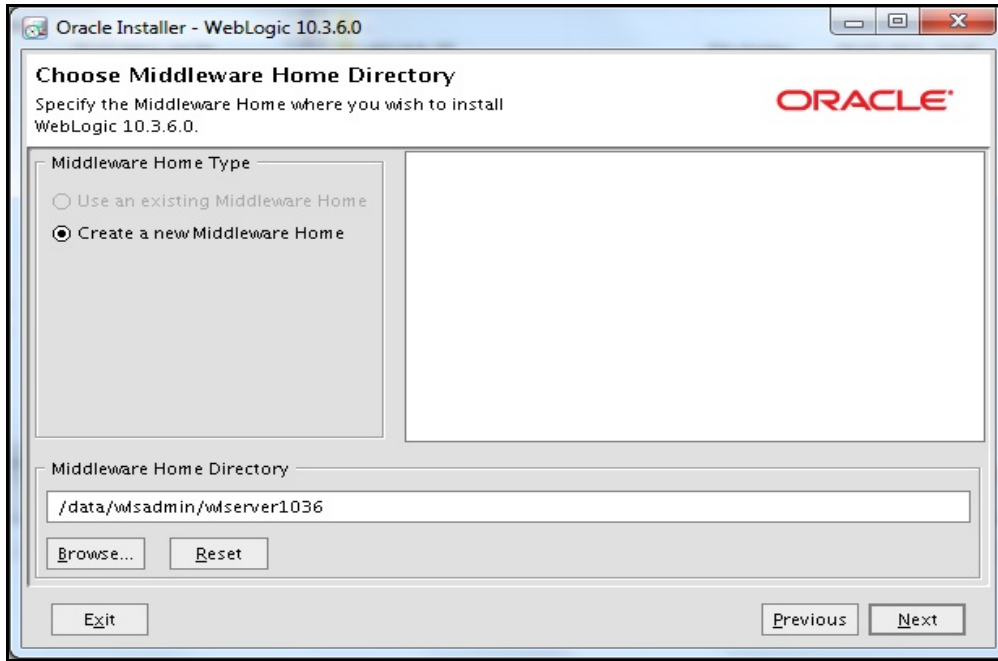
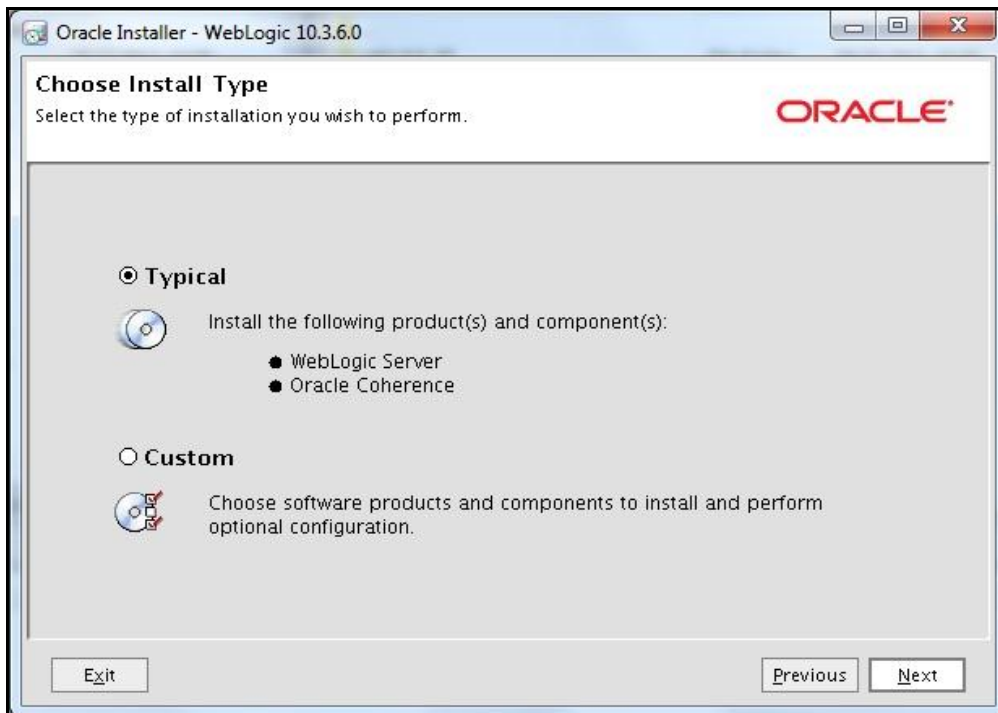
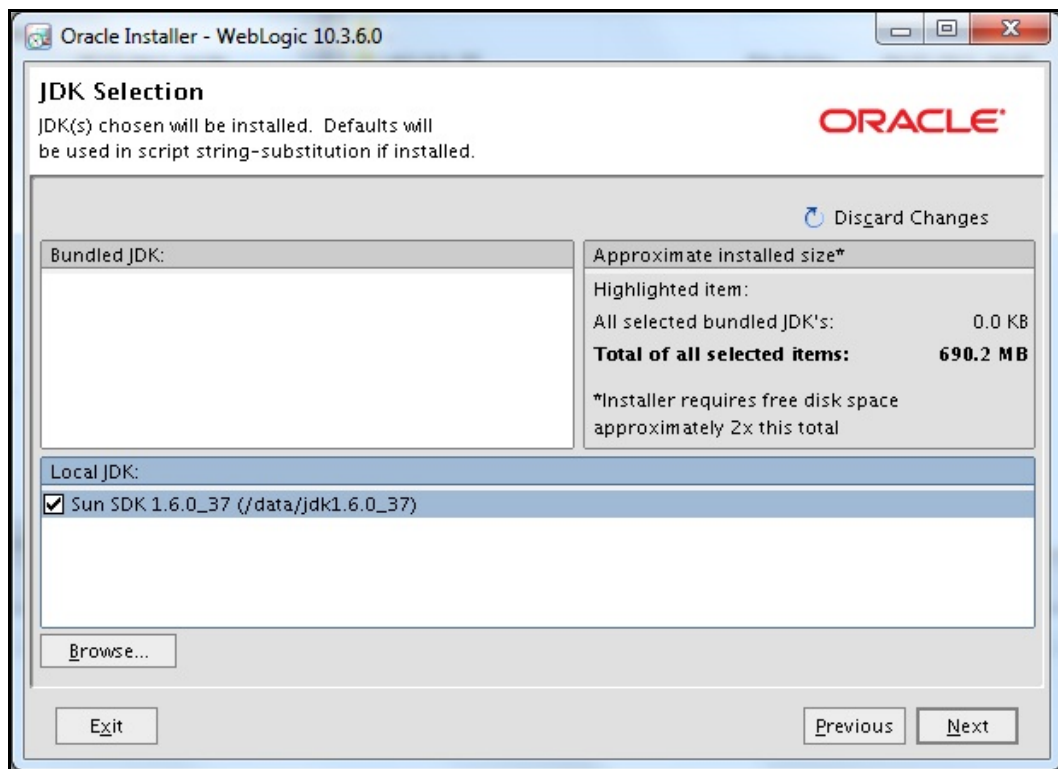


Figure 3.3: Installation types



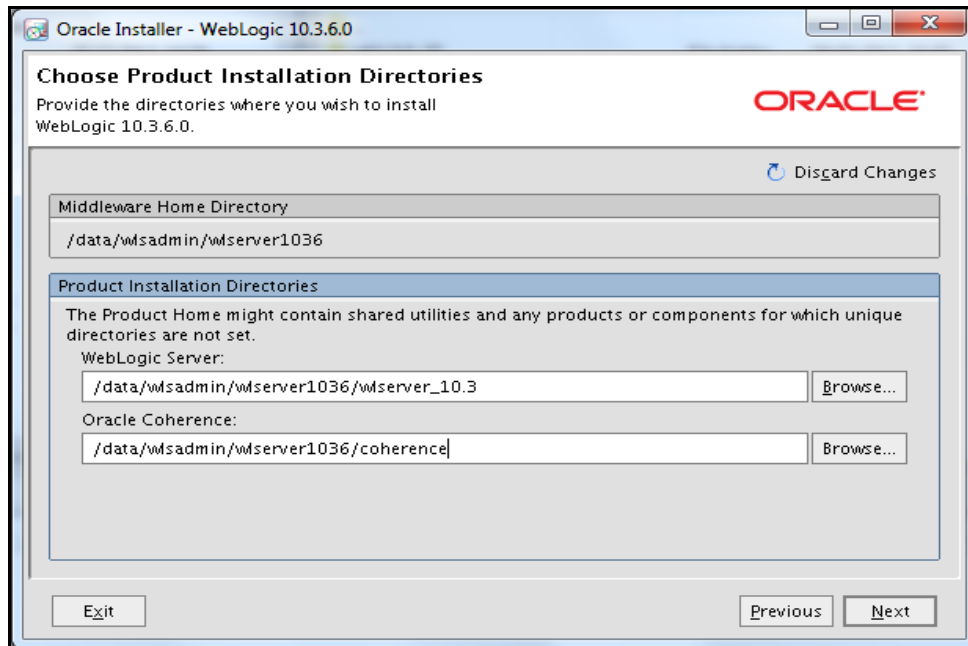
After specifying the path of weblogic home directory, the screenshot of “installation types” appear. Here, the option “typical” is chosen as in Figure 3.3 because all the components of weblogic are needed in this study. After choosing the installation type, JDK is chosen in the next screen. There are two different options of JDK according to the weblogic jar in use: bundled and local JDK. The bundled JDK is already in the installation package, but if the operating system is 64bit, then it is better to choose JDK and carry out the installation manually. In Figure 3.4, JDK is manually installed in the 64bit machine, so the weblogic jar downloaded from the Oracle website is a generic jar and doesn’t include bundled JDK. Next, the path of the local JDK is chosen below:

Figure 3.4: JDK selection



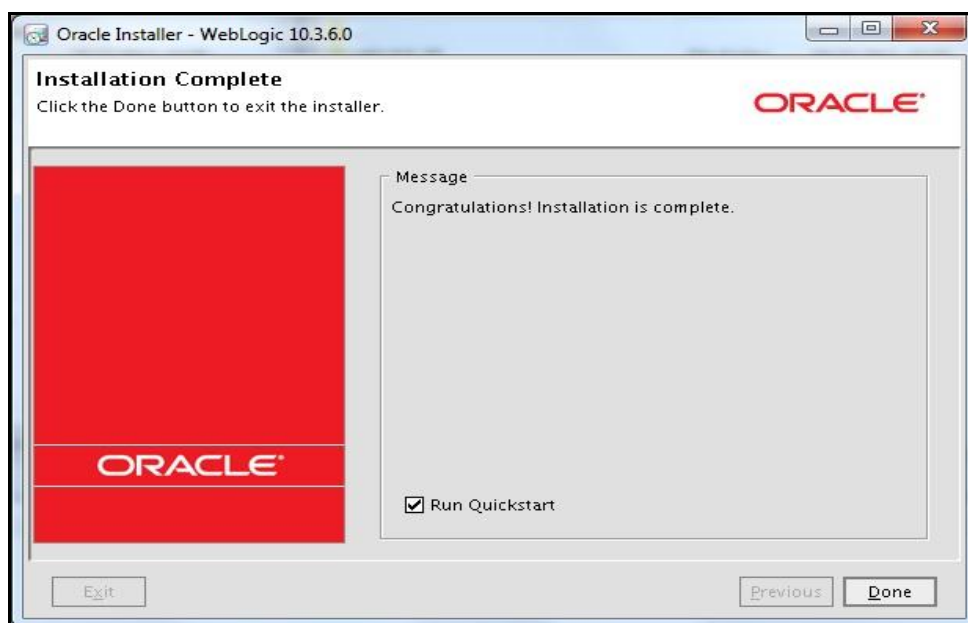
After the JDK choice, the next screenshot is called “product installation directories”. Here, the directories where weblogic application and coherence servers are to be installed are specified under the weblogic home directory as in Figure 3.5. While choosing the directory paths, it is best practice to choose “wlserver_10.3” for weblogic servers and “coherence” for coherence binaries. This is because the content of a folder can easily be understood just by looking at the name of directory.

Figure 3.5: Product installation selection



When the installation is successfully completed, the screenshot in Figure 3.6 appears. One can finish the process by clicking “done”, but because of the necessary configurations after the installations, the option, “Run Quickstart” should be chosen. Then, it is possible to proceed with the configuration screen as in Figure 3.6:

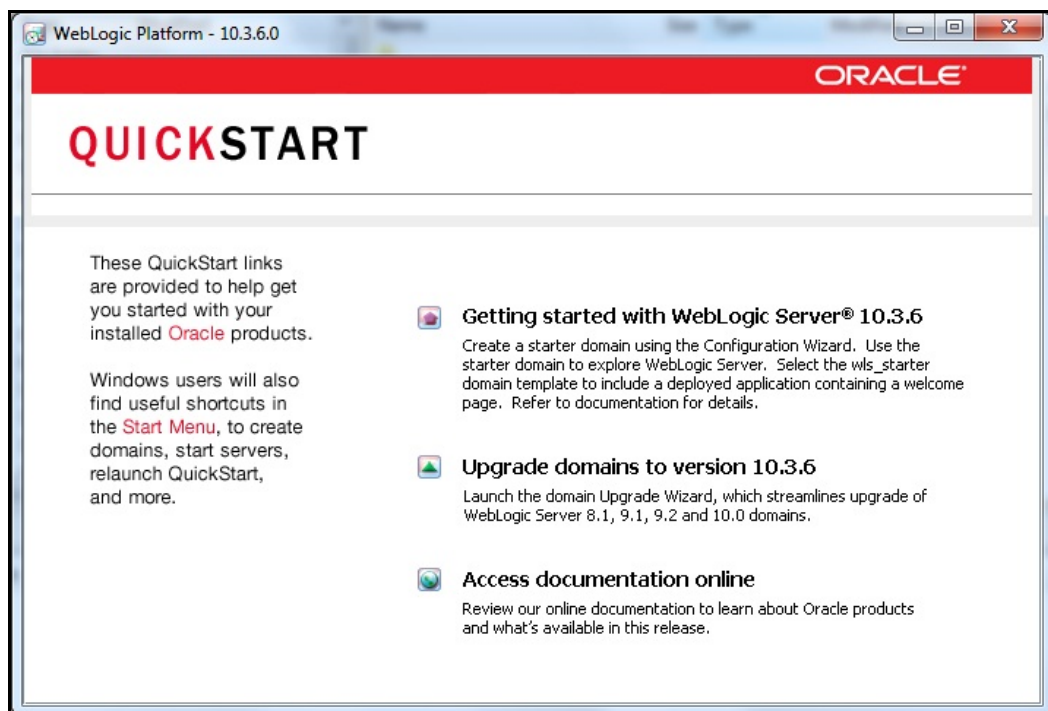
Figure 3.6. Installation of weblogic and coherence servers



3.3.2 Configuration of Weblogic and Coherence Servers

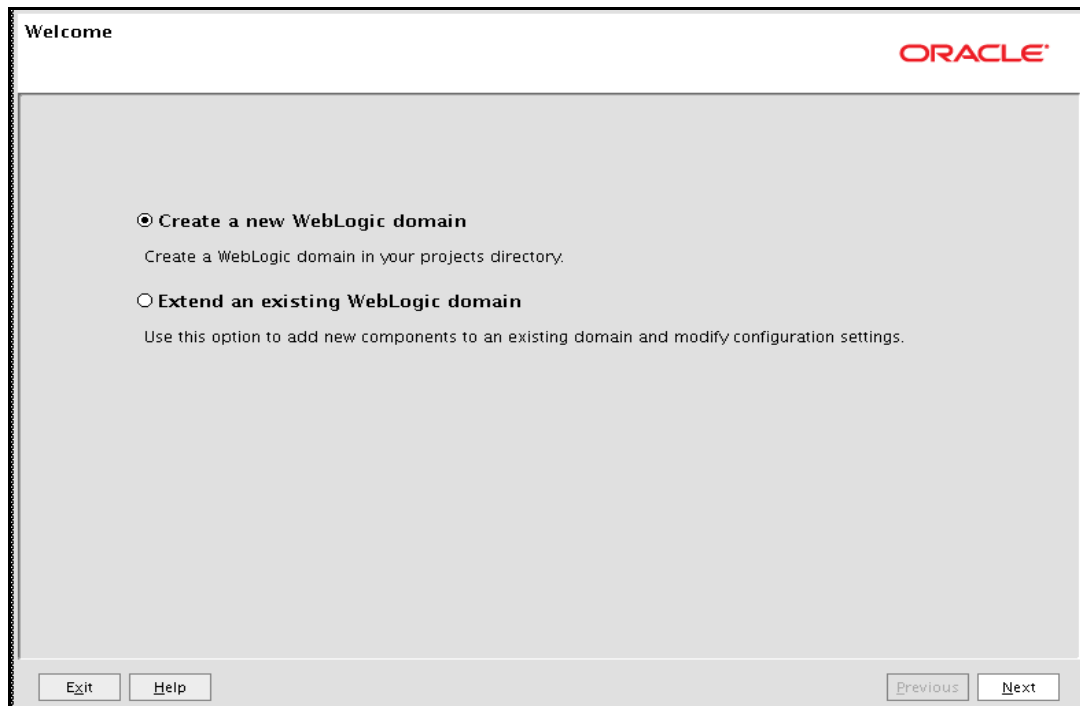
Following the installation stage, there appears the Quickstart screen. In this screen, the domain configuration is made by using the Weblogic Configuration Wizard. Figure 3.7 shows that the configuration process starts with choosing the “Getting started with Weblogic Server 10.3.6” part in the present screen:

Figure 3.7: Getting started with the configuration



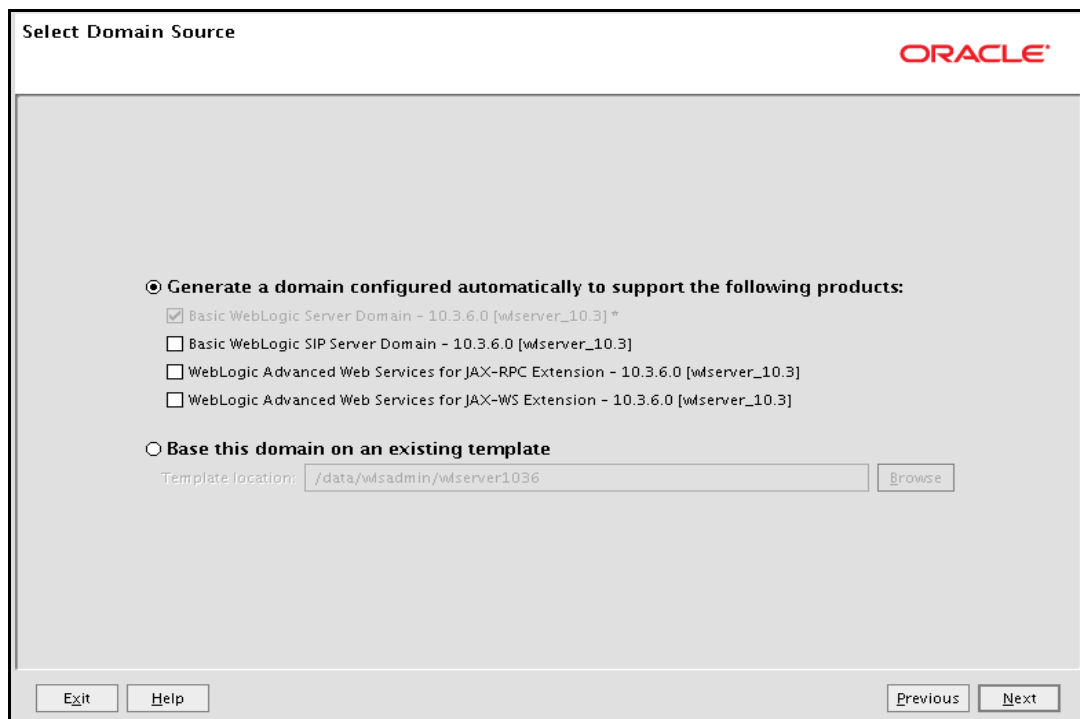
A new domain must now be created for the weblogic server that has been installed. What is meant by the domain is a pool of all the servers here. A server in the weblogic must belong to a domain because all the information of a server can be controlled over the domain. Therefore, “Create a new Weblogic domain” option is chosen as in Figure 3.8.

Figure 3.8: Creating a new domain



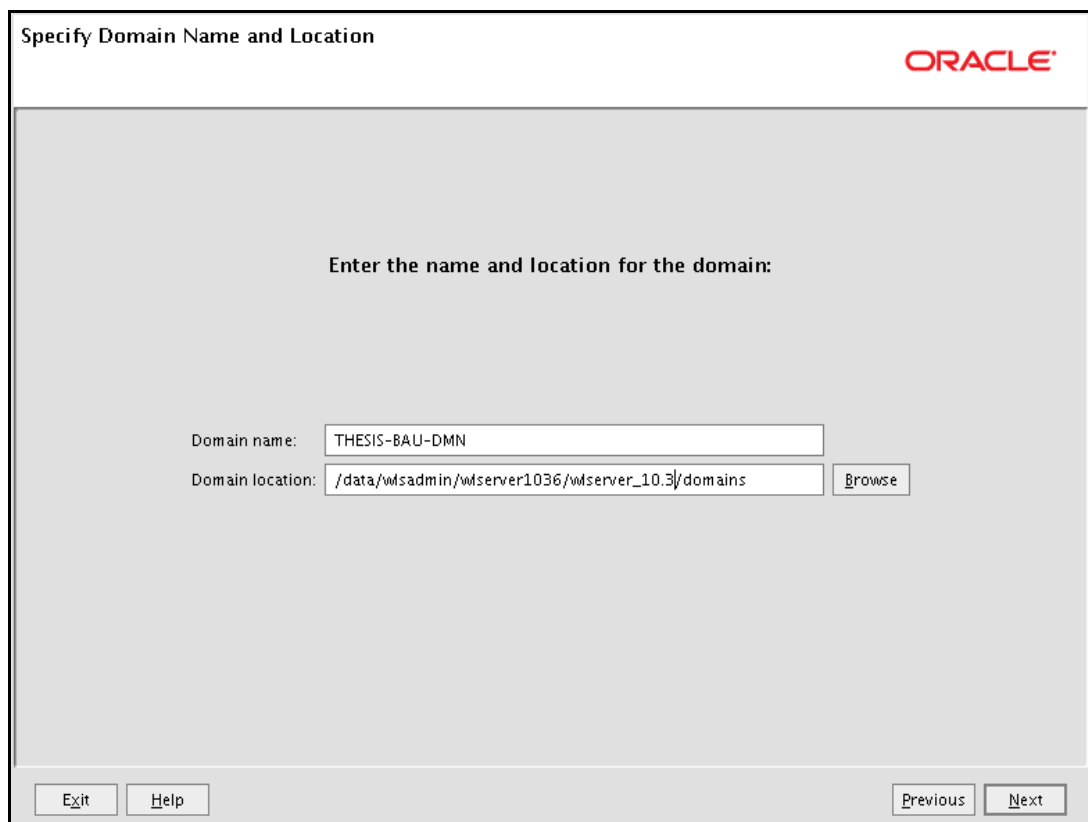
Secondly, which products are to be supported by the domain are determined as in Figure 3.9 below:

Figure 3.9: Selecting the domain source



After making the product choice, the process of domain configuration starts with setting the “domain name” and “domain location” as in Figure 3.10. There are two important points here: firstly, one need to understand from the domain name where and for what job the domain is used; and also while naming a domain folder, it should be ensured that the domain folder includes a domain. Therefore, it is best practice to name the domain folder as “domains” as in the example of Figure 3.10.

Figure 3.10: Specifying domain name and location



Specify Domain Name and Location

ORACLE

Enter the name and location for the domain:

Domain name: THESIS-BAU-DMN

Domain location: /data/wlsadmin/wlserver1036/wlserver_10.3/domains

The next stage in the weblogic and coherence configuration is the admin server configuration that will control the application and coherence servers. Initially, a name should be given to the admin server, starting with the domain name and ending with “admin”. Then, as can be observed from Figure 3.11, the admin server is assigned the IP address of the machine and one of its empty port, preferably one of those ports ending with “00”, because one can easily understand that it belongs to an admin server when he sees a port with “00”.

Figure 3.11: Configuring admin server

Configure the Administration Server ORACLE

[Discard Changes](#)

*Name:

*Listen address:

Listen port:

SSL listen port:

SSL enabled:

Figure 3.12: Configuring managed servers

Configure Managed Servers ORACLE

[Add](#) [Delete](#) [Discard Changes](#) Switch Display

| | Name* | Listen address* | Listen port | SSL listen port | SSL enabled |
|-----|---------------------|---|-----------------------------------|----------------------------------|--------------------------|
| → 1 | THEISIS-BAU-M1-MAN1 | <input type="text" value="10.210.52.90"/> | <input type="text" value="7711"/> | <input type="text" value="N/A"/> | <input type="checkbox"/> |

Figure 3.12 displays the configuration of managed servers: a total of five servers – including two application servers, two coherence servers (where the cache application is to be deployed), and one application server are configured. When the configuration is completed, the other four servers must be added from the admin server console. Managed servers are also assigned a name, listen address and port number. While naming managed servers, it is necessary to start with the domain name again and continue with the machine number and managed server number. In Figure 3.12, the listen address is the same as the address of the admin server and the port number of the listen address is an empty and different port from the port of the admin server.

The configuration of managed servers is followed by the screen of the configuration summary. The configuration summary lists all of the configurations that have been made so far, including the names of the path and servers in the domain. If there is a mistake in the configuration, the configuration summary will help to check and fix it. If the configuration is correct, one can move onto the completion of the configuration as in Figure 3.13:

Figure 3.13: The configuration summary

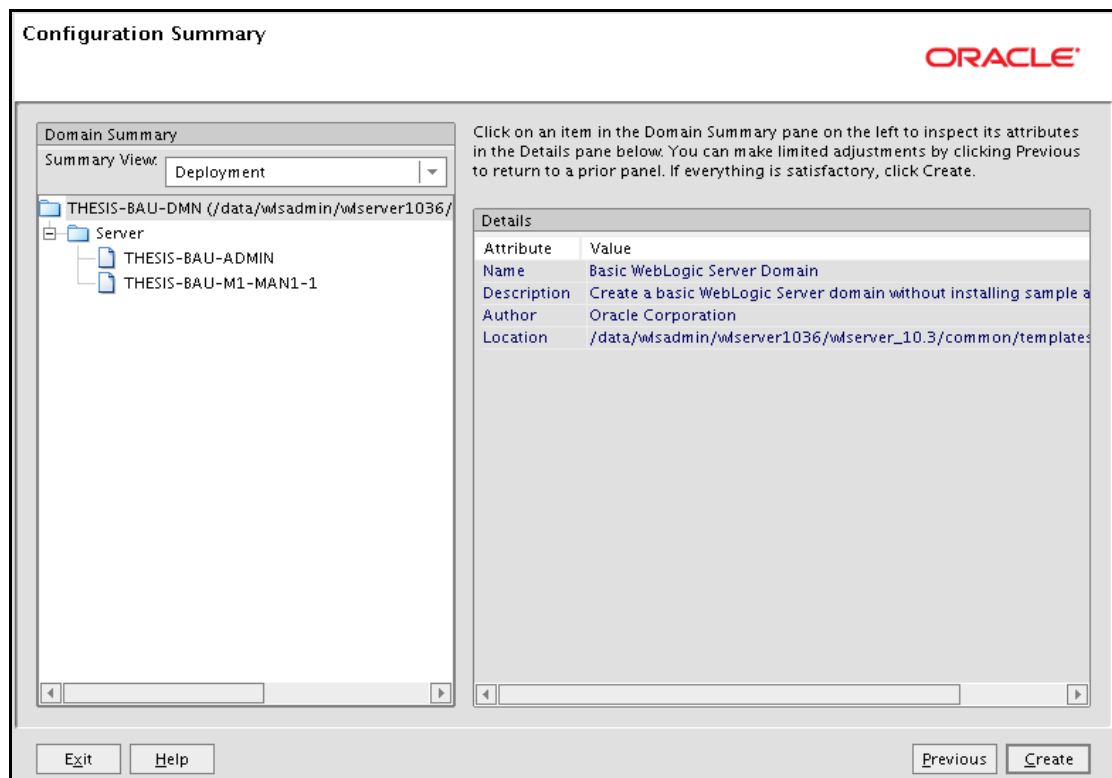
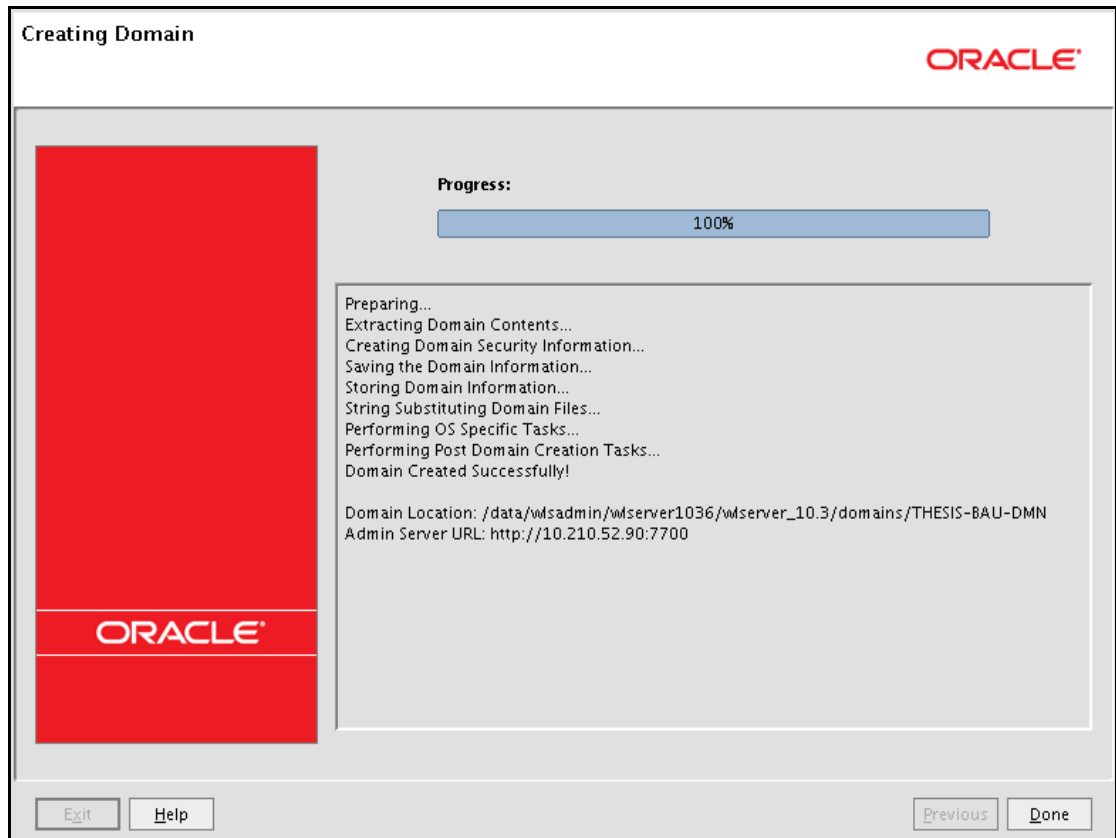


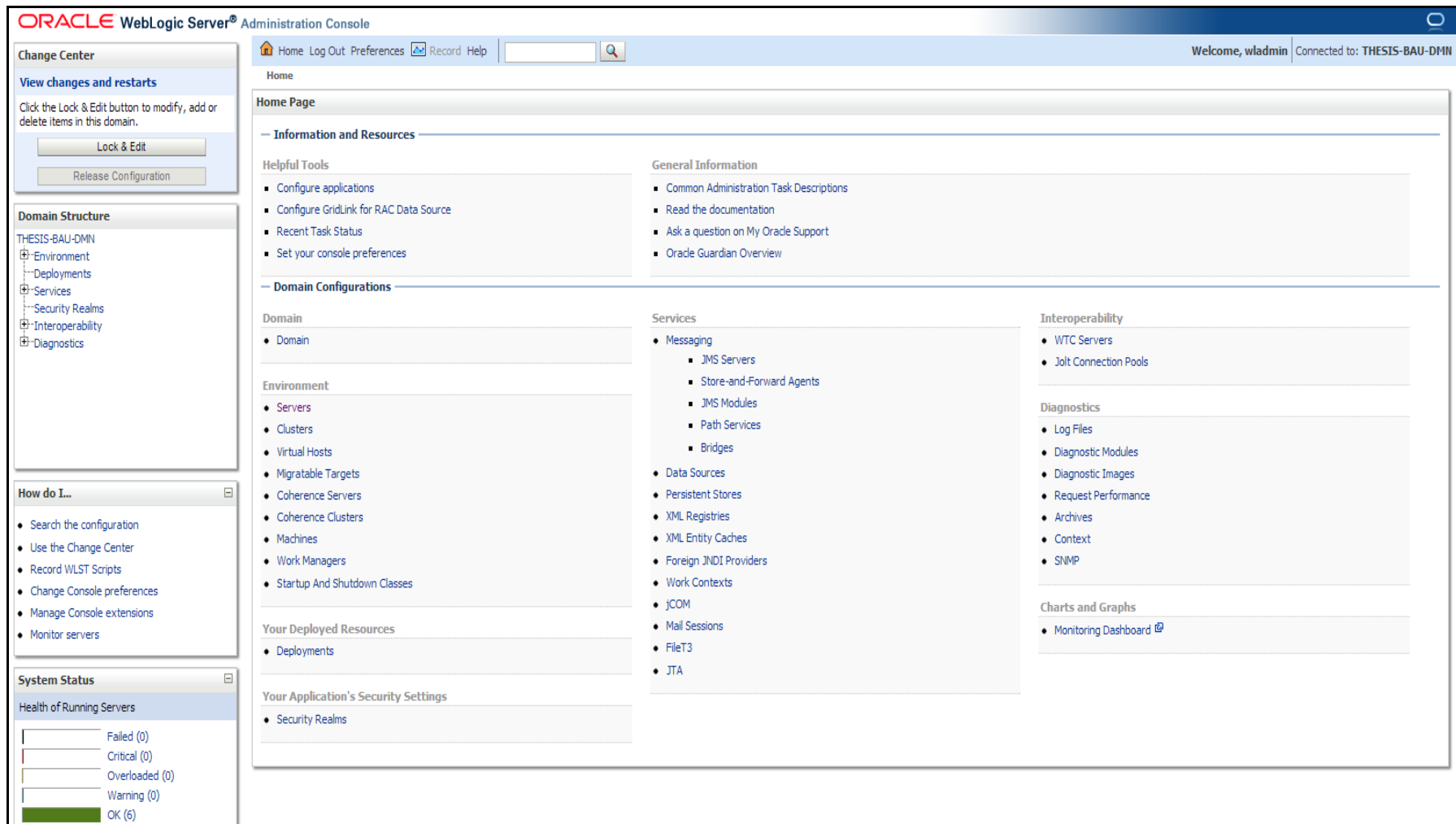
Figure 3.14 shows that all the configurations have been completed successfully and are ready for use in the deployment.

Figure 3.14: Completing the admin and managed server configurations



Finally, the installation and configuration of weblogic application and coherence servers is finished and it is time to connect with the admin console. In order to launch the admin console, it is necessary to use the machine's IP and the port number that has been used in the installation of the admin server. The admin console can be accessed by typing "console" at the end of the IP and port number in the browser: i.e. "<http://10.210.52.90:7700/console>" as in Figure 3.15:

Figure 3.15: Admin console homepage



In conclusion, Figure 3.16 illustrates that the two application servers, two coherence servers and one Hazelcast server in the architecture of this study has been successfully installed and configured; at the same time, all of the servers are now up and running. In the present architecture, the servers labelled as MAN0 represent Coherence servers, the servers labelled as MAN1-1 and MAN1-2 represent the servers in which the front-end application using only Coherence cache is deployed. The server labelled as MAN1-3, on the other hand, is the server that has been installed for Hazelcast and includes the front-end application and the Hazelcast cache.

Figure 3.16: List of managed servers

The screenshot shows the Oracle WebLogic Server Administration Console. The top navigation bar includes 'Home', 'Log Out', 'Preferences', 'Record', and 'Help'. The user is logged in as 'wladim' and is connected to the domain 'THEISIS-BAU-DMN'. The main content area is titled 'Summary of Servers' and shows a list of six managed servers, all in a 'RUNNING' state with 'OK' health. The table columns are Name, State, Health, Listen Port, Heap Free Current, and Heap Size Current.

| Name | State | Health | Listen Port | Heap Free Current | Heap Size Current |
|--------------------------|---------|--------|-------------|-------------------|-------------------|
| THEISIS-BAU-ADMIN(admin) | RUNNING | OK | 7700 | 134793960 | 516751360 |
| THEISIS-BAU-M1-MANO-1 | RUNNING | OK | 7701 | 1888890088 | 2084569088 |
| THEISIS-BAU-M1-MANO-2 | RUNNING | OK | 7702 | 1900199880 | 2084569088 |
| THEISIS-BAU-M1-MAN1-1 | RUNNING | OK | 7711 | 1174466760 | 1547698176 |
| THEISIS-BAU-M1-MAN1-2 | RUNNING | OK | 7712 | 1208973048 | 1547698176 |
| THEISIS-BAU-M1-MAN1-3 | RUNNING | OK | 7713 | 1548269816 | 2084569088 |

The left-hand navigation pane includes sections for 'Change Center' (with 'Lock & Edit' and 'Release Configuration' buttons), 'Domain Structure' (a tree view showing the hierarchy from Environment to Security Realms), 'How do I...' (a list of tasks like 'Create Managed Servers'), and 'System Status' (a bar chart showing the health of running servers: 6 OK, 0 Failed, 0 Critical, 0 Overloaded, 0 Warning).

3.3.3 Configuration of Cache and Cluster

After the coherence servers are installed and configured, there are only a few important configurations left behind. It is essential to add the below parameters to the startup script that starts the coherence servers.

```
CLASSPATH="$CLASSPATH:${COHERENCE_HOME}/lib/coherence.jar"
```

→ It indicates that the managed server is a member of the coherence server.

```
CLASSPATH="$CLASSPATH:${COHERENCE_INITIALIZER}"
```

→ This parameter starts up the managed server as the coherence server.

```
COHER_HOST="`hostname`"
```

→ Managed server is now a coherence server and is named after the value in this parameter.

```
COHER_PORT="33010"
```

→ This is the port that the Coherence server will be using when it is communicating with the other members in the background.

```
CLUSTERNAME="THESIS-COHERENCE-CLUSTER"
```

→ This parameter sets the name of the cluster that the Coherence server is connected to.

```
COHERENCE_OPTIONS="-Dtangosol.coherence.localhost=${COHER_HOST} -
```

```
Dtangosol.coherence.localport=${COHER_PORT}"
```

→ The coherence port and host are added to the Global classpath.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -
```

```
Dtangosol.coherence.clustername=${CLUSTERNAME}"
```

→ The cluster name is added to the Global classpath.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.localport.adjust=true

→ If the port of the coherence server is in use, a new port is automatically generated by setting this parameter.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.override=\${WL_HOME}/domains/conf/CoherenceClusterOverride_thesis.xml

→ This parameter sets in which path the Coherence override xml exists.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.distributed.localstorage=true

→ This parameter indicates that the Coherence server is a cache server and stores caches.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.session.localstorage=true

→ It indicates that the Coherence server is a cache server and stores sessions.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.member=THESIS_\${SERVER_NAME}

→ The member name of the Coherence server is set by this parameter.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.role=THESIS_DISTRIBUTED

→ The role name of the Coherence server is set by this parameter.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.management.remote=true

→ This parameter enables the Coherence cluster to be locally and remotely managed with JMX.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dtangosol.coherence.management=all"
```

→ The Coherence cluster is locally and remotely managed with JMX in this parameter.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dcom.sun.management.jmxremote"
```

→ This parameter enables the JMX library to load on the Global classpath.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dtangosol.coherence.log=${DOMAIN_HOME}/logs/Coherence-  
${SERVER_NAME}.log"
```

→ It tells the Coherence server where to write the coherence logs.

```
echo "COHERENCE_OPTIONS=${COHERENCE_OPTIONS}"
```

→ This parameter prints all the coherence options onto the screen in order to control them at the startup.

```
JAVA_OPTIONS="${JAVA_OPTIONS} ${COHERENCE_OPTIONS}"  
export JAVA_OPTIONS
```

→ It sets all the coherence and java options to the global classpath and the servers starts with this "JAVA_OPTIONS".

The parameters above must be set in all the coherence servers. Since there are two Coherence servers in the current architecture, the Coherence port parameter for the second server needs to be changed into *COHER_PORT="33020"*, and the rest of the parameters dynamically change for the related server. Finally, the detailed configuration of the two coherence servers in the present architecture have been completed.

Besides, there are two other servers (MAN1-1 and MAN1-2), which aren't coherence servers but only include an application. In order for these servers to read the Coherence cache and to share sessions, they need to join in the Coherence cluster. To join them in the coherence cluster, parameters similar to the above should be set to the startup script of the servers.

```
CLASSPATH="$CLASSPATH:${COHERENCE_HOME}/lib/coherence.jar"
```

→ This parameter indicates that the managed server is a member of the coherence server member.

```
COHER_HOST="`hostname`"
```

→ The managed server is now a coherence server and named after the value in this parameter.

```
COHER_PORT="33110"
```

→ This is the port that the Coherence server will be using when it is communicating with the other members in the background.

```
CLUSTERNAME="THESIS-COHERENCE-CLUSTER"
```

→ This parameter sets the name of the cluster that the Coherence server is connected to.

```
COHERENCE_OPTIONS="-Dtangosol.coherence.localhost=${COHER_HOST} -  
Dtangosol.coherence.localport=${COHER_PORT}"
```

→ This is used to set java options to the global classpath.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dtangosol.coherence.clustername=${CLUSTERNAME}"
```

→ It sets the name of the cluster that the Coherence server is connected to.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.localport.adjust=true"

→ If the port of the coherence server is in use, a new port is automatically generated by setting this parameter.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.override=\${WL_HOME}/domains/conf/CoherenceClusterOverride_thesis.xml"

→ This parameter is used to set in which path the Coherence override xml exists.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.distributed.localstorage=false"

→ It indicates that the Coherence server is not a cache server and won't store caches.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.session.localstorage=true"

→ It indicates that the Coherence server is not a cache server and won't store sessions.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.member=THESIS_\${SERVER_NAME}"

→ This parameter sets the member name of the Coherence server.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.role=THESIS_PROCESS"

→ It sets the role name of the Coherence server.

COHERENCE_OPTIONS=\${COHERENCE_OPTIONS} -
Dtangosol.coherence.management.remote=true"

→ This parameter enables the Coherence cluster to be locally and remotely managed with JMX.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dtangosol.coherence.management=all"
```

→ The Coherence cluster is locally and remotely managed with JMX in this parameter.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dcom.sun.management.jmxremote"
```

→ This enables the JMX library to load on the classpath.

```
COHERENCE_OPTIONS="${COHERENCE_OPTIONS} -  
Dtangosol.coherence.log=${DOMAIN_HOME}/logs/Coherence-  
${SERVER_NAME}.log"
```

→ It tells the Coherence server where to write the coherence logs.

```
echo "COHERENCE_OPTIONS=${COHERENCE_OPTIONS}"
```

→ This parameter prints all the coherence options onto the screen in order to control them at the startup.

```
JAVA_OPTIONS="${JAVA_OPTIONS} ${COHERENCE_OPTIONS}"  
export JAVA_OPTIONS
```

→ It sets all the coherence and java options to the global classpath and the servers starts with this "JAVA_OPTIONS".

After determining with the startup scripts which of the managed servers will store the cache and session and which of these won't store the cache and session, it is then necessary to mention two important configuration XMLs: "CoherenceClusterOverride.xml", and "Coherence-Cache-Config.xml". "CoherenceClusterOverride.xml" is used in order to identify the coherence cluster members and to determine who will connect to the cluster as a member. Below is presented the details of this xml.

`<?xml version="1.0"?>`

`<coherence>`

`<cluster-config>`

`<authorized-hosts>` → It includes IPs of the hosts that can connect to the cluster.

`<host-address id="murat">10.210.52.90</host-address>`

`</authorized-hosts>`

`<member-identity>`

`<cluster-name system-property="tangosol.coherence.cluster">`

`THESIS-COHERENCE-CLUSTER`

`</cluster-name>` → It determines the unique name that will separate the

coherence cluster from all the other clusters in the network.

`</member-identity>`

`<multicast-listener>` → The Coherence cluster members build multicast communication in order to discover and message each other in the network.

`<address system-property="tangosol.coherence.clusteraddress">`

`225.5.5.225`

`</address>` → This is the IP address for the Coherence cluster to use in multicast communication.

`<port system-property="tangosol.coherence.clusterport">`

`37101`

`</port>` → This is the port for the Coherence cluster to use in multicast communication

`<time-to-live system-property="tangosol.coherence.ttl">`

`2`

`</time-to-live>` → This determines how long the packages used in multicast communication will last. This equals to the maximum number of hops (router, switch and network interface) in the network.

`<join-timeout-milliseconds>`

`10000`

`</join-timeout-milliseconds>` → This is the maximum waiting time for the members to connect to the Coherence cluster member. If the member cannot connect to the cluster within this time limit, the member creates its own cluster.

`</multicast-listener>`

`<port-auto-adjust>`

`true`

`</port-auto-adjust>` → If the port of the Coherence server is in use, a new port is automatically generated by setting this parameter.

`<packet-publisher>`

`<packet-delivery>`

`<timeout-milliseconds>`

`300000`

`</timeout-milliseconds>` → This is the maximum waiting time for the members to send the confirmation for the published package. The member that doesn't send the confirmation within this time limit is marked as "dead".

`</packet-delivery>`

`</packet-publisher>`

`</cluster-config>`

`<logging-config>`

`<severity-level system-property="tangosol.coherence.log.level">`

`9`

`</severity-level>` → This sets the Log severity.

`<character-limit system-property="tangosol.coherence.log.limit">`

`1048576`

`</character-limit>`

`</logging-config>`

`</coherence>`

The following values can be given to this parameter according to Oracle (2011, p. A-31): “0 - only output without a logging severity level specified will be logged; 1 - all the above plus errors; 2 - all the above plus warnings; 3 - all the above plus informational messages; 4-9 - all the above plus internal debugging messages; -1 - no messages”.

The Coherence servers create and configure the cache according to the “Coherence-Cache-Config.xml”. The details of this xml are presented below:

```
<?xml version="1.0"?>
<cache-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://xmlns.oracle.com/coherence/coherence-cache-config"
  xsi:schemaLocation="http://xmlns.oracle.com/coherence/coherence-cache-
  config coherence-cache-config.xsd">
  <defaults>
    <serializer system-property="tangosol.coherence.serializer"/>
    <socket-provider system-property="tangosol.coherence.socketprovider"/>
  </defaults>
```

<caching-scheme-mapping> → This is the place where the Cache names are determined. The naming in the coding should be the same as the naming here, because the name of the cache used in the code should first be controlled here and what kind of caching will be performed is decided here: for example, distributed, near, replicated and local caching.

```
<cache-mapping>
  <cache-name>dist-*/</cache-name>
  <scheme-name>example-distributed</scheme-name>
  <init-params>
    <init-param>
      <param-name>back-size-limit</param-name>
      <param-value>8MB</param-value>
    </init-param>
```



```

    </init-params>
  </cache-mapping>
  <cache-mapping>
    <cache-name>near-*)</cache-name>
    <scheme-name>example-near</scheme-name>
  </init-params>
    <init-param>
      <param-name>back-size-limit</param-name>
      <param-value>8MB</param-value>
    </init-param>
  </init-params>
</cache-mapping>
<cache-mapping>
  <cache-name>repl-*)</cache-name>
  <scheme-name>example-replicated</scheme-name>
</cache-mapping>
<cache-mapping>
  <cache-name>local-*)</cache-name>
  <scheme-name>example-object-backing-map</scheme-name>
</cache-mapping>
<cache-mapping>
  <cache-name>*</cache-name>
  <scheme-name>example-distributed</scheme-name>
</cache-mapping>
</caching-scheme-mapping>

```

<caching-schemes> → It is determined here how caching will be performed, how the backup will be taken, whether the cache will expire, how the read-write backup will be taken and whether it will be delayed.

```

<distributed-scheme>
  <scheme-name>example-distributed</scheme-name>

```

```
<service-name>DistributedCache</service-name>
<backing-map-scheme>
  <local-scheme>
    <scheme-ref>example-binary-backing-map</scheme-ref>
  </local-scheme>
</backing-map-scheme>
<autostart>true</autostart>
</distributed-scheme>
```

```
<near-scheme>
  <scheme-name>example-near</scheme-name>
  <front-scheme>
    <local-scheme>
      <eviction-policy>HYBRID</eviction-policy>
      <high-units>100</high-units>
      <expiry-delay>1m</expiry-delay>
    </local-scheme>
  </front-scheme>
  <back-scheme>
    <distributed-scheme>
      <scheme-ref>example-distributed</scheme-ref>
    </distributed-scheme>
  </back-scheme>
  <invalidation-strategy>present</invalidation-strategy>
  <autostart>true</autostart>
</near-scheme>
```

```
<replicated-scheme>
  <scheme-name>example-replicated</scheme-name>
  <service-name>ReplicatedCache</service-name>
  <backing-map-scheme>
    <local-scheme>
```

```

    <scheme-ref>unlimited-backing-map</scheme-ref>
  </local-scheme>
</backing-map-scheme>
  <autostart>true</autostart>
</replicated-scheme>

<local-scheme>
  <scheme-name>example-object-backing-map</scheme-name>
  <eviction-policy>HYBRID</eviction-policy>
  <high-units>{back-size-limit 0}</high-units>
  <expiry-delay>{back-expiry 1h}</expiry-delay>
  <cachestore-scheme></cachestore-scheme>
</local-scheme>
<local-scheme>
  <scheme-name>example-binary-backing-map</scheme-name>
  <eviction-policy>HYBRID</eviction-policy>
  <high-units>{back-size-limit 0}</high-units>
  <unit-calculator>BINARY</unit-calculator>
  <expiry-delay>0</expiry-delay>
  <cachestore-scheme></cachestore-scheme>
</local-scheme>
<local-scheme>
  <scheme-name>unlimited-backing-map</scheme-name>
</local-scheme>
<read-write-backing-map-scheme>
  <scheme-name>example-read-write</scheme-name>
  <internal-cache-scheme>
    <local-scheme>
      <scheme-ref>example-binary-backing-map</scheme-ref>
    </local-scheme>
  </internal-cache-scheme>

```

```

<cachestore-scheme></cachestore-scheme>
<read-only>true</read-only>
<write-delay>0s</write-delay>
</read-write-backing-map-scheme>
</caching-schemes>
</cache-config>

```

3.3.4 Installation and Configuration of Apache Web Server

The installation of the webserver starts with downloading the latest binary file from Apache's website. The downloaded "*httpd- 2.2.23.tar.gz*" file is uncompressed with the command, "*gzip -d httpd- 2.2.23.tar.gz*" and the package is then extracted with the command, "*tar -xvf http httpd- 2.2.23.tar*" in the related path. The configuration is realized by going to the folder where the package is opened and executing "*./configure*" command. Afterwards, the commands, "*make ve make install*" are executed in the same folder, and the installation is completed according to the desired configuration. The configuration outputs are as follows:

1. Webserver home directory: The three main folders, "*exec, logs, and SubConfs*" directories can be seen here.

```

[root@murat:/usr/local/apachews]# ls
exec  httpd.conf  logs  SubConfs
[root@murat:/usr/local/apachews]# █

```

2. Webserver start, stop, restart scripts and exec directory: The "*start, stop, and gracefulrestart*" scripts of the server can be seen here.

```

[root@murat:/usr/local/apachews]# cd exec/
[root@murat:/usr/local/apachews/exec]# ls
GracefulRestart-apache-WS  start-apachews  stop-apachews
[root@murat:/usr/local/apachews/exec]#

```

3. Webserver configuration files: The two configuration files, "*httpd.conf_WSs*" and "*weblogic.conf_WSs-Redirects_P1of2*" under the main directory "*SubConfs*" can be seen here.

```

[root@murat:/usr/local/apachews/subConfs]# ls
httpd.conf_ws  weblogic.conf_ws-Redirects_P1of2
[root@murat:/usr/local/apachews/subConfs]#

```

In addition, the content of “*webservice redirections*” file is formulated as:

```
##### WS-ReverseProxyRedirections#####  
MatchExpression /bahce  
WebLogicCluster=10.210.52.90:7711,10.210.52.90:7712/CookieName=bahcesession  
MatchExpression /bahcesehirhazelcast  
WebLogicCluster=10.210.52.90:7713/CookieName=hazelcastsession  
##### WS-ReverseProxyRedirections#####
```

The *http/https* request sent to the webservice is first forwarded to the *httpd_conf* file, and the URL is directed to the *redirects* file in *httpd_conf* to be resolved there. In this way, the request reaches the related application server. For example, if this URL, “<http://10.210.52.90/bahce/login.jsf>” is typed in the browser, the webservice will direct a user to one of these two managed servers in the *redirect* file: either the managed server with the IP, “10.210.52.90”, and the port, “7711” or the managed server with the IP, “10.210.52.90”, and the port, “7712”. As for the Hazelcast managed server, the user is directed to the managed server with the IP, “10.210.52.90”, and the port, “7713”, in the *redirect* file, if this URL, “<http://10.210.52.90/bahcesehirhazelcast/login.jsf>” is typed in the browser. As a result, the server binaries of Apache webservice are installed and configured, and how the system works has been demonstrated with two examples.

3.3.5 Installation and Configuration of Oracle Database

Before the installation and configuration of *Oracle Database*, the related binary file for *Linux* or *Windows* is downloaded from the *Oracle* website and copied to the server. Also, the software and hardware requirements of the installation guide (*Oracle* 2012b, p. 2-1) must be controlled. As in Figure 3.17, the downloaded binary files are opened in *Linux* or *Windows* environment, and the process starts with choosing the installation options. It is also clear from Figure 3.17 that both the database installation and configuration will take place at the same time.

Figure 3.17: Selecting the installation options of Oracle 11g database

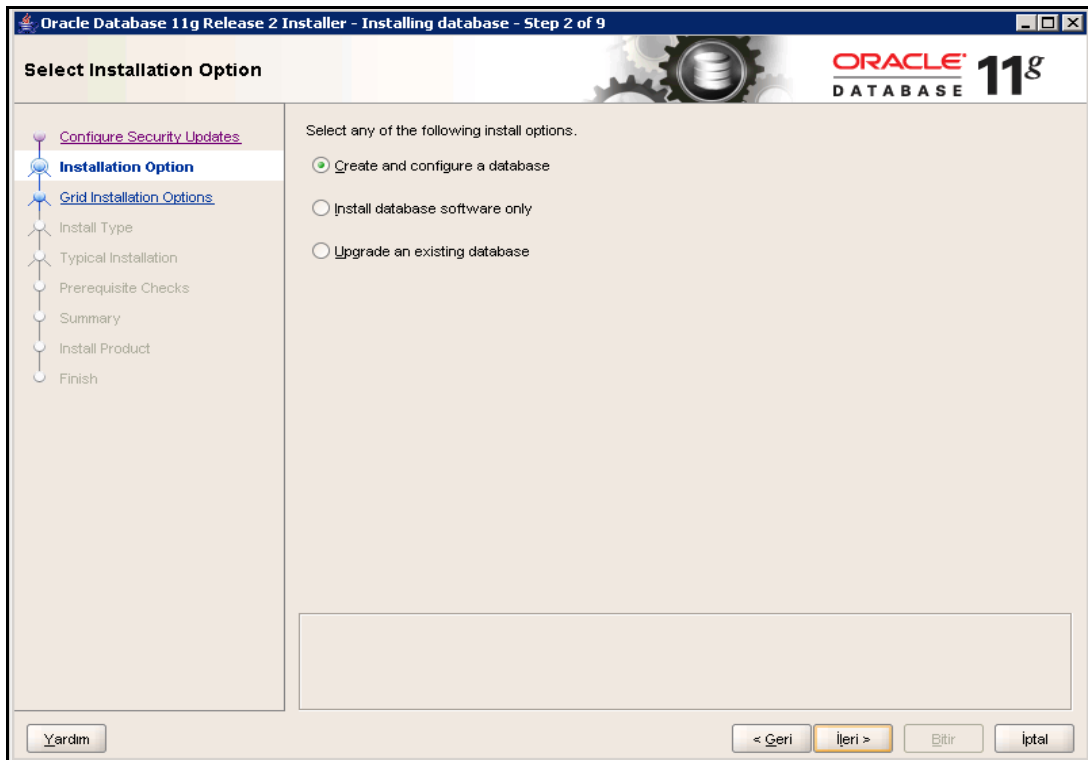


Figure 3.18: Typical install configuration

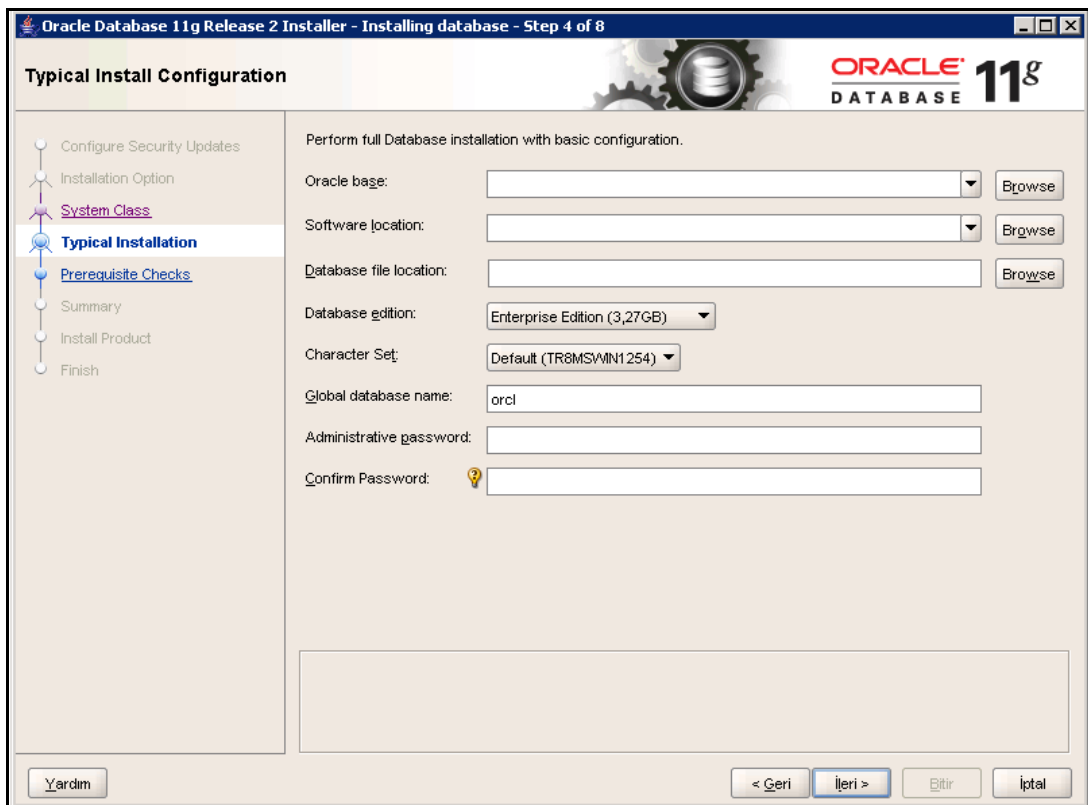
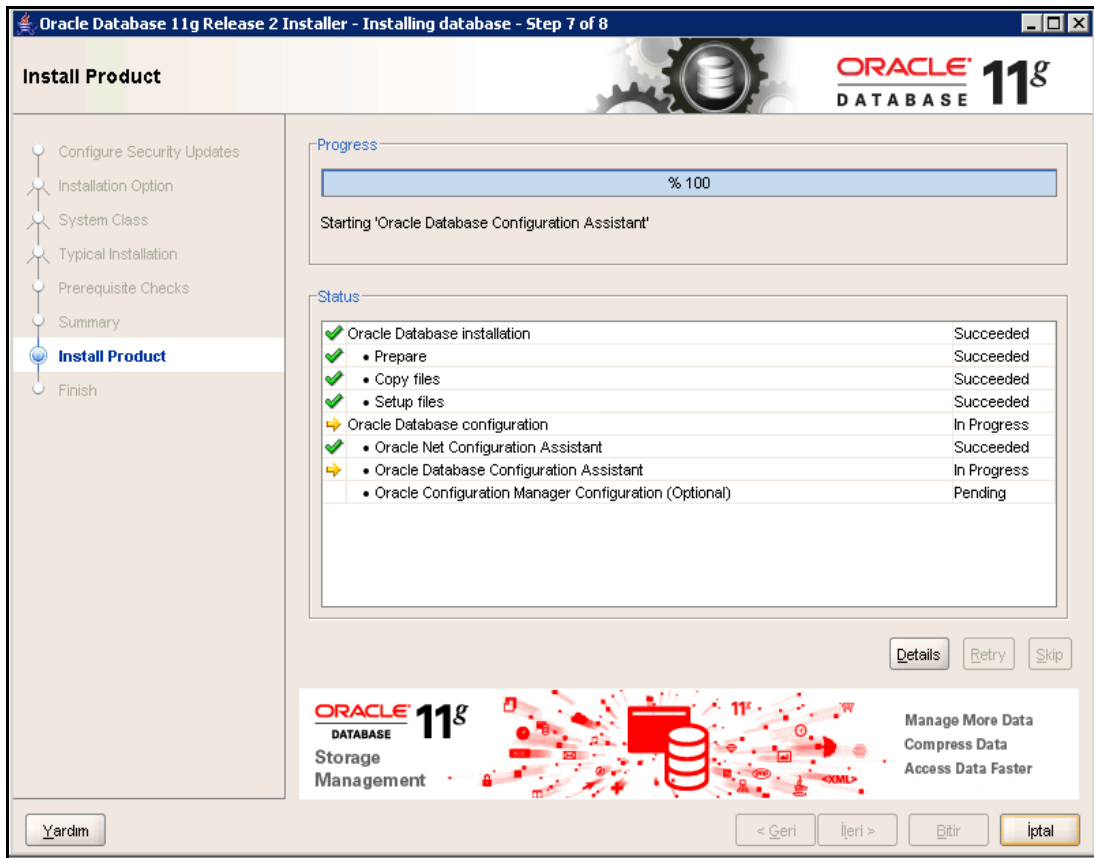


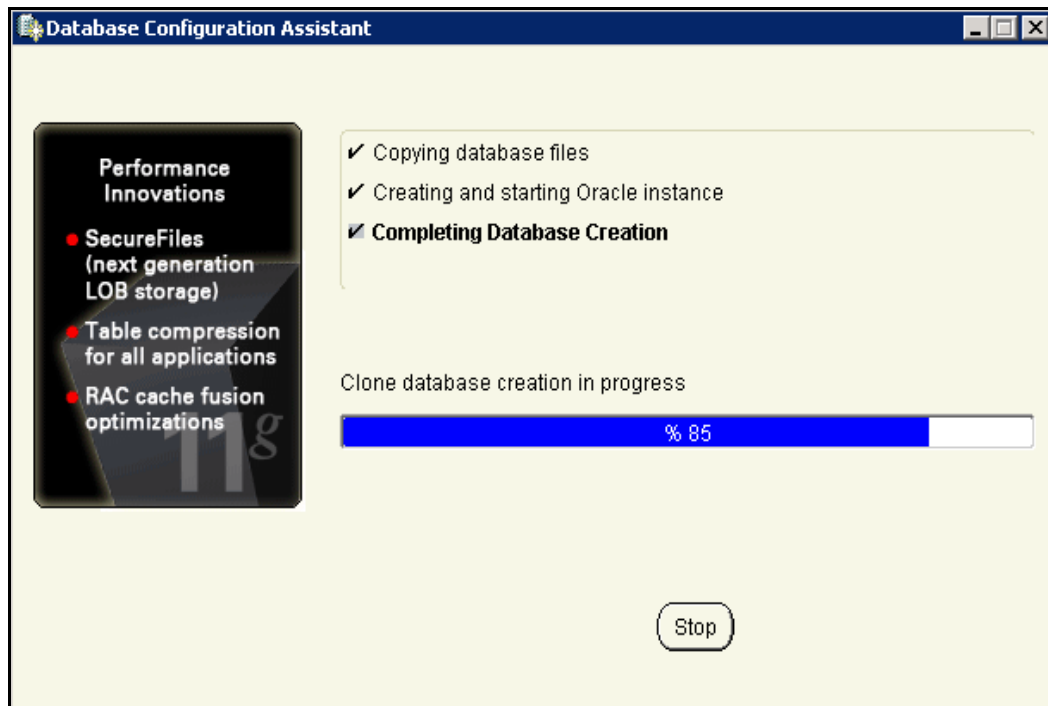
Figure 3.18 shows that the basic configuration of the database necessitates the following information: 1. Oracle base specifies Oracle home directory; 2. Software location is the path where Oracle Db is installed; 3. Database file location is the path where oradata is installed; and 4. Administrative password is set in order to log on the database. After the configuration information is provided, one can start to install and configure the Oracle Database as in Figure 3.19:

Figure 3.19: Installing products



Immediately after the product installation is completed, the configuration process is automatically launched as in Figure 3.20 below:

Figure 3.20: Configuration in progress



Finally, the Oracle 11g Database is ready for use. In order for such applications as “toad, pl/sql developer” to connect to the database, the *tns* address of the database that is installed in the *tnsname.ora* file needs to be added to the applications.

3.4 DEVELOPMENT

So far, all the installations and configurations in the architecture of this study have been successfully completed. From now on, the applications that are using this architecture are described. Just like all the other cloud applications, the applications in this study work with mostly demanded and constantly updated data, whose truth value is of high importance.

In the case of the present architecture, a weather service that provides instantaneously changing measurements of temperature, humidity, pressure etc. and hourly weather forecasts has been postulated to inform institutions (the Turkish Coast Guard Command), websites (CNNTURK), and mobile applications (Yahoo Weather) that should always update their systems. Numerous update requests are being sent to the

systems of such a weather service that need to respond to their clients continuously, consistently and rapidly. Considering the increasing number of requests sent to this weather service, it is a must for them to increase the infrastructure capacity.

Figure 3.21: The overloaded database of the weather service

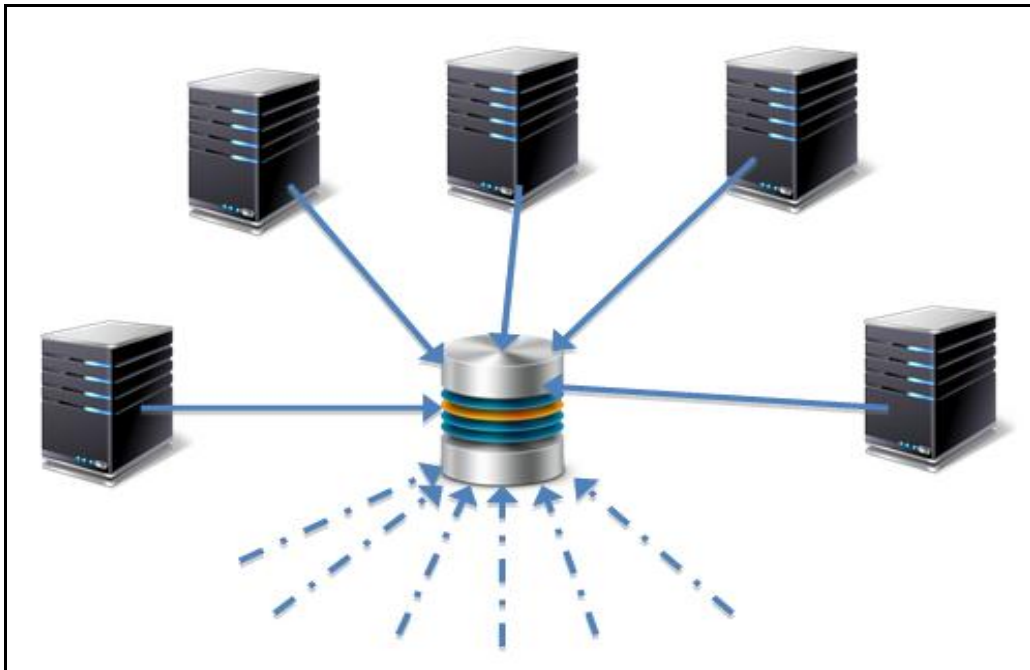


Figure 3.21 illustrates the situation of the overloaded database that receives increasing numbers of requests every day, and it is evident that infrastructure investment becomes a necessity, if these requests are to be sent to a structure whose capacity is constant and predetermined. However, increasing the infrastructure capacity means additional costs of investment, maintenance and repair as well as extra workload. Most importantly, this also complicates the architecture of the system, which will in return cause system problems. The potential cutoffs also lead to financial loss for the weather service.

On the other hand, with the help of a cloud-based application with Oracle Coherence, the existing system can respond to the coming requests (from the external and internal networks) quickly, reliably and consistently without increasing the infrastructure capacity. For this reason, a cloud application has been developed with Oracle Cloud's PaaS and SaaS features in the present study.

Thanks to the use of Oracle Coherence, backend problems have also been eliminated in the production environment. Backend problems occur because: 1. the database can be under too many requests; 2. the database has long-running sqls; 3. the database experiences momentary failures due to workload; or 4. the database can be indirectly influenced by a failure caused by a different application out of the system.

In Oracle Coherence, the most frequently requested data are located over backend in middleware. In this way, database open-close connections are no longer made in middleware, and “object-to-object” communication can start without converting the result of the sql statement into object. In addition, middleware can also be affected by backend problems: thread, heap, CPU problems may appear in middleware. With Oracle Coherence, *http session management* is performed, and as a result, the problems in middleware are resolved without being noticed by the end-users.

3.4.1 Java Development

All of the applications have been developed on the Java platform by using spring, primefaces. For Coherence and Hazelcast developments, Coherence Developer Guide and Hazelcast Documentation are used (available on their websites). The important issues that should be taken into consideration during the development are detailed here. First of all, Oracle Coherence uses the NamedCache interface in order to locate the data over backend to middle tier or to reach the cache in the cache server(s). The developer uses this interface in the code to create a new cache similar to the following:

```
NamedCache sehir = CacheFactory.getCache("sehir");  
sehir.put(country.getNcountry()+ "-" + city.getNcity()+ "-" + date, datas);
```

As can be seen from the code, a cache named “sehir” is created and the data is put in this cache. If the developer wants to read data from the cache, he requests a cache name by using the NamedCache interface. If the requested cache name is present in the cache server, the requested data can be obtained from the cache by searching with a key as in the following example:

```
NamedCache sehir = CacheFactory.getCache("sehir");
sehir.get(searchCountryValueCoh+"-"+searchCityValueCoh+"-
"+searchDateValueCoh);
```

On the other hand, Hazelcast, the counterpart of Oracle Coherence puts the data from the backend into the cache by using the Map interface or reaches the cache in the cache server. The developer uses this interface to create a new cache in the code. A cache named “sehir” is created in the code and the data are put into this cache as follow:

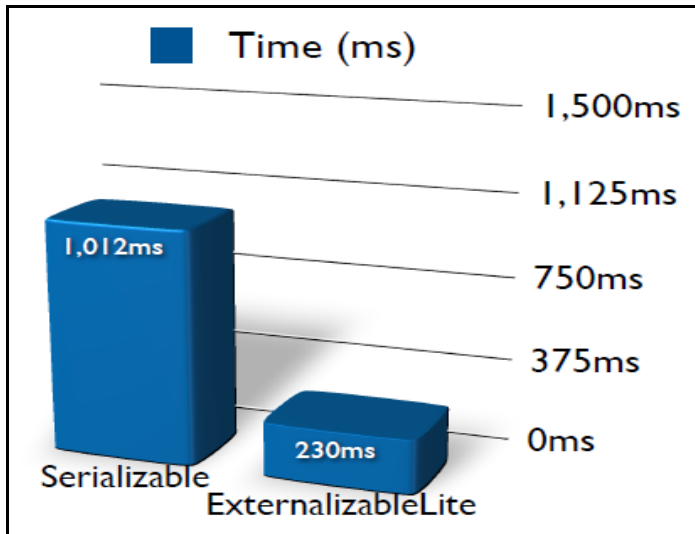
```
Map<String, List<DataColumnCoherence>> mapSehir= Hazelcast.getMap("sehir");
mapCustomers.put(country.getNcountry() + "-" + city.getNcity() + "-" + date, datas);
```

If the developer wants to read the data from the cache, he requests a cache name by using the Map interface. If the requested cache name is present in the cache server, the requested data can be obtained by searching with a key as below:

```
Map<String, List<DataColumnCoherence>> mapSehir = Hazelcast.getMap("sehir");
mapSehir.get(searchCountryValueHaz+"-"+searchCityValueHaz+"-
"+searchDateValueHaz);
```

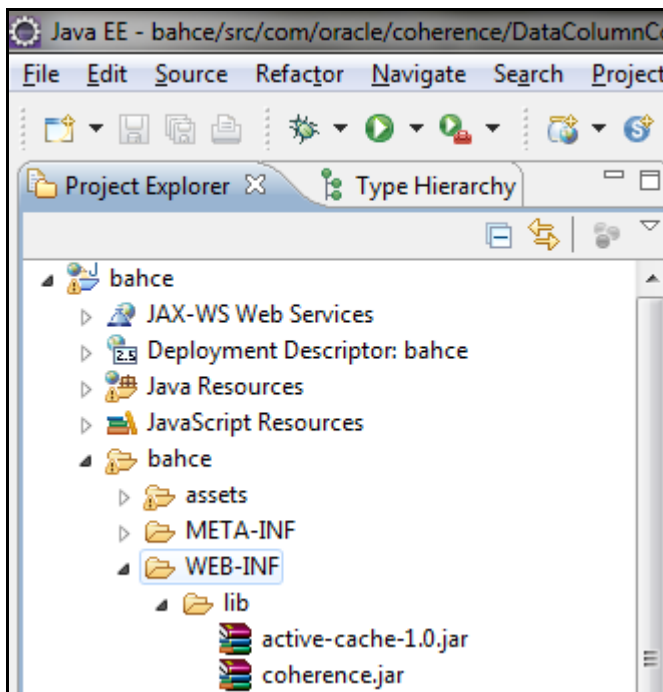
If an object in Java needs to be stored in the disk and memory or sent to another environment over the network, this object is implemented as serializable. In Oracle Coherence, there is an interface called *ExternalizableLite* that extends the serializable interface. Since *ExternalizableLite* has a much better performance than the serializable interface, developers need to put the data in the *ExternalizableLite* type while putting the data into the cache. As for the advantages of the *ExternalizableLite* in Figure 3.22, it is typically 6x faster, and the resulting serialization is much smaller, while it emits less garbage for both serialization and deserialization (Tangosol 2007, p. 72).

Figure 3.22: Comparing ExternalizableLite and Serializable in duration



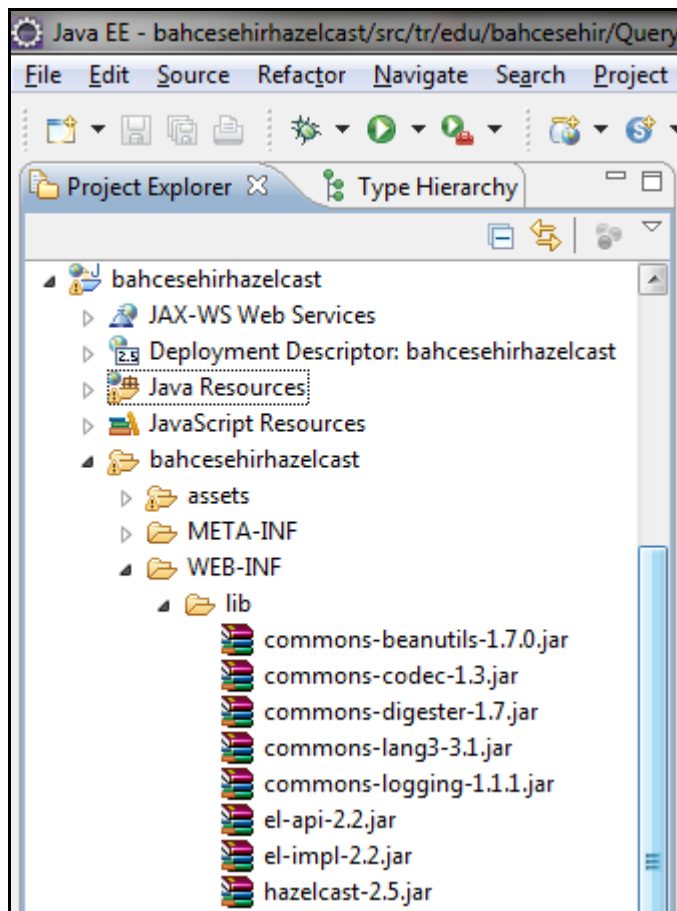
In addition, developers need to add *coherence.jar* to the class paths and the library of the applications in order to be able to use all the features of the Oracle Coherence as evident from Figure 3.23 below:

Figure 3.23: Adding coherence.jar to the library



Similarly, developers need to add *hazelcast.jar* to the class paths and the library of the applications to be able to use all the features of Hazelcast as can be observed in Figure 3.24:

Figure 3.24: Adding hazelcast.jar to the library



With its feature of “*http session management*”, Oracle Coherence provides a consistent and reliable web environment. In order to use this Coherence feature in the developing application, the below xml must be added in the web project. Because Coherence has many parameters for session management, the suitable parameters for the developing application are chosen and added to the web.xml as in the following:

```
<context-param>  
    <param-name>coherence-scopecontroller-class</param-name>  
    <param-value>  
com.tangosol.coherence.servlet.AbstractHttpSessionCollection$GlobalScopeController  
    </param-value>
```

```

</context-param>
<context-param>
  <param-name>coherence-session-member-locking</param-name>
  <param-value>>false</param-value>
</context-param>
<context-param>
  <param-name>coherence-sticky-sessions</param-name>
  <param-value>>false</param-value>
</context-param>

```

If an application needs to have session management and be deployed in the weblogic domain, the following library definition must be added to the weblogic.xml of the application:

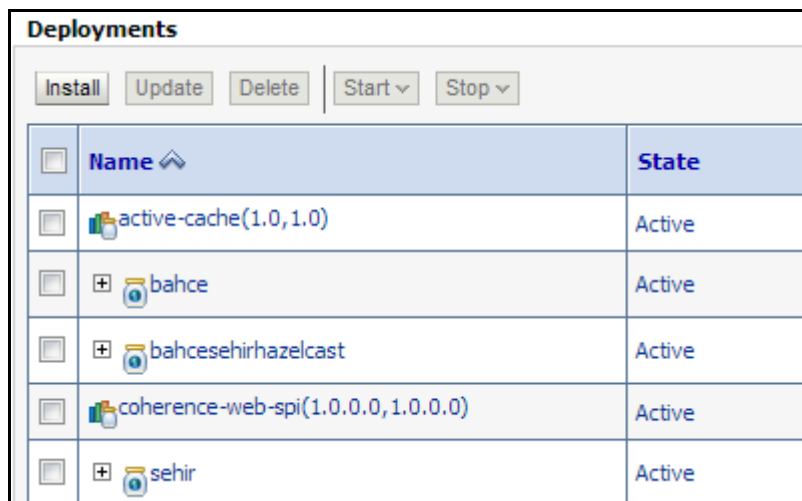
```

<library-ref>
  <library-name>coherence-web-spi</library-name>
</library-ref>

```

There are operations that need to be completed in the server for http session management. The “*active-cache-1.0.jar*” and “*coherence-web-spi.war*” packages must be deployed in both the application servers and the coherence servers as in Figure 3.25:

Figure 3.25: Deployment of active-cache and coherence-web-spi



Two of the three applications developed here include the front-end codes and webservice codes. The names of these two applications are “*bahce*” and

“bahcesehirhazelcast”. The front-end applications have been designed to show the performance differences between the traditional and new-generation cloud systems when there is no load on the servers where the applications are deployed.

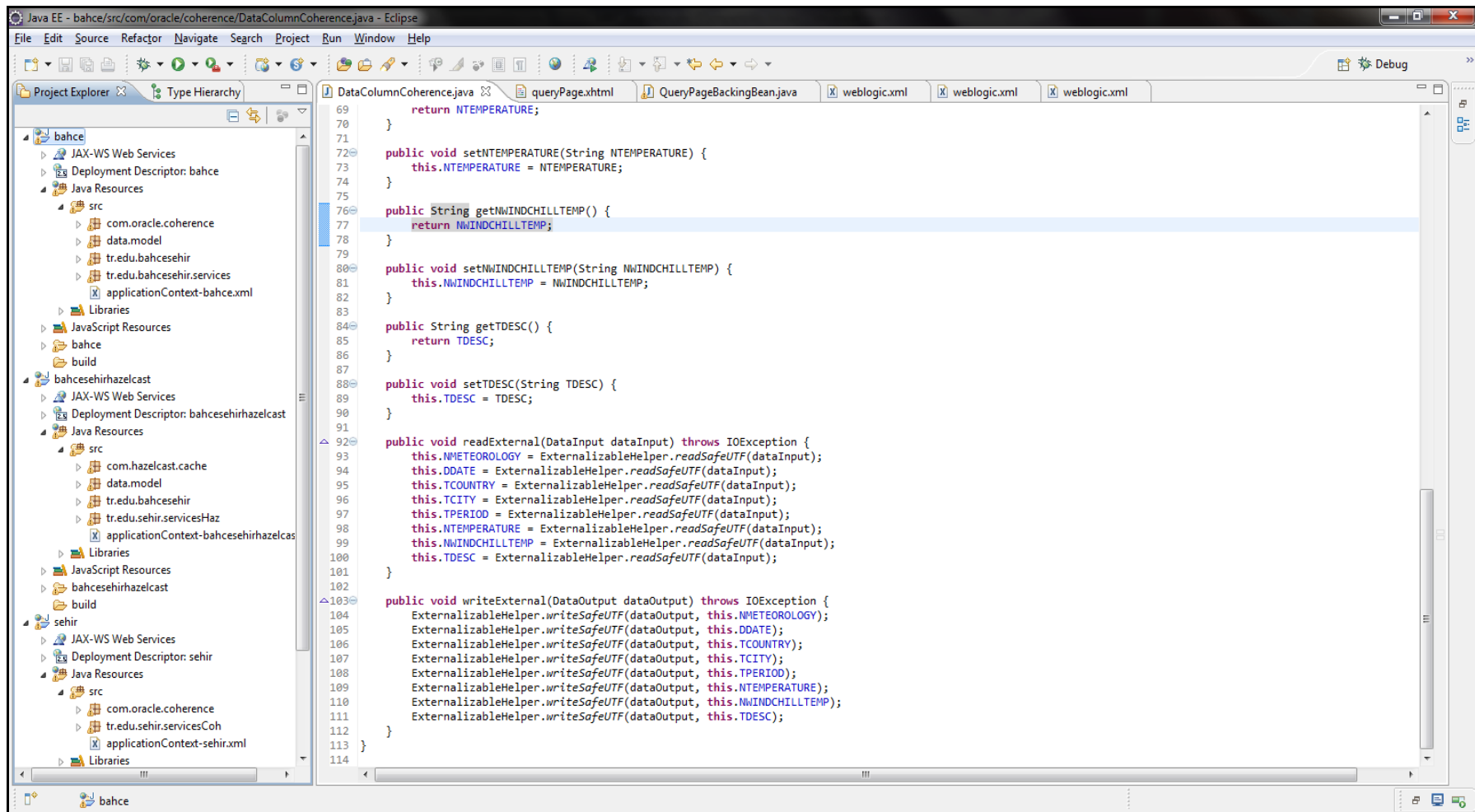
The front-end codes of the *“bahce”* application have been written in such a way that from which source the data can be received faster will be demonstrated: from the Coherence cache or the database. The webservice codes of this application have been designed in such a way that it can access the database or the Coherence cache directly. In this way, the differences between the traditional and the cloud systems can easily be demonstrated by using these webservices in the load test. In addition, the *“bahce”* application has the feature of session management, and therefore, have been targeted to the *“THESIS-BAU-M1-MAN1-1”* and *“THESIS-BAU-M1-MAN1-2”* servers.

The *“Bahcesehirhazelcast”* application has been developed in the same way as the *“bahce”* application, except that the technology in use is Hazelcast instead of Oracle Coherence, and the server it is deployed into: *“THESIS-BAU-M1-MAN1-3”* server.

The two applications, *“sehir”* and *“bahcesehirhazelcast”* include the codes for storing the cache. Both applications have been developed to store nearly 11000 rows of a 240000-row table in cache by connecting to the database. The *“sehir”* application works in the servers of Coherence caching and the *“bahce”* front-end application uses the data stored in the cache of the *“sehir”* application. On the other hand, the *“Bahcesehirhazelcast”* application works in the servers of Hazelcast caching. Since it includes the front-end codes in the same package, the *“Bahcesehirhazelcast”* application reads the data it has stored in the cache. The *“sehir”* application has been targeted to *“THESIS-BAU-M1-MAN0-1”* and *“THESIS-BAU-M1-MAN0-2”* servers, while the *Bahcesehirhazelcast”* application has been targeted to the *“THESIS-BAU-M1-MAN1-3”* server.

Figure 3.26 indicates the developmental phases of the *“bahce”*, *“sehir”*, and *“bahcesehirhazelcast”* applications in the Eclipse environment.

Figure 3.26: Development with Eclipse



These three applications are then exported as war file over the Eclipse and deployed to the weblogic server. For weblogic server deployment, it is necessary to connect to the admin console and to deploy the applications from the “Deployments” screen as in Figure 3.27:

Figure 3.27: Deployments and status of the applications

ORACLE WebLogic Server® Administration Console

Home Log Out Preferences Record Help

Welcome, wladmin Connected to: THESIS-BAU-DMN

Home > Summary of Deployments

Summary of Deployments

Control Monitoring

This page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. Installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Install button.

[Customize this table](#)

Deployments

Showing 1 to 5 of 5 Previous | Next

| <input type="checkbox"/> | Name | State | Health | Type | Deployment Order |
|--------------------------|-------------------------------------|--------|--------|-----------------|------------------|
| <input type="checkbox"/> | active-cache(1.0, 1.0) | Active | | Library | 100 |
| <input type="checkbox"/> | bahce | Active | OK | Web Application | 100 |
| <input type="checkbox"/> | bahcesehirhazelcast | Active | OK | Web Application | 100 |
| <input type="checkbox"/> | coherence-web-spi(1.0.0.0, 1.0.0.0) | Active | | Library | 100 |
| <input type="checkbox"/> | sehir | Active | OK | Web Application | 100 |

Showing 1 to 5 of 5 Previous | Next

Change Center

View changes and restarts

No pending changes exist. Click the Release Configuration button to allow others to edit the domain.

Domain Structure

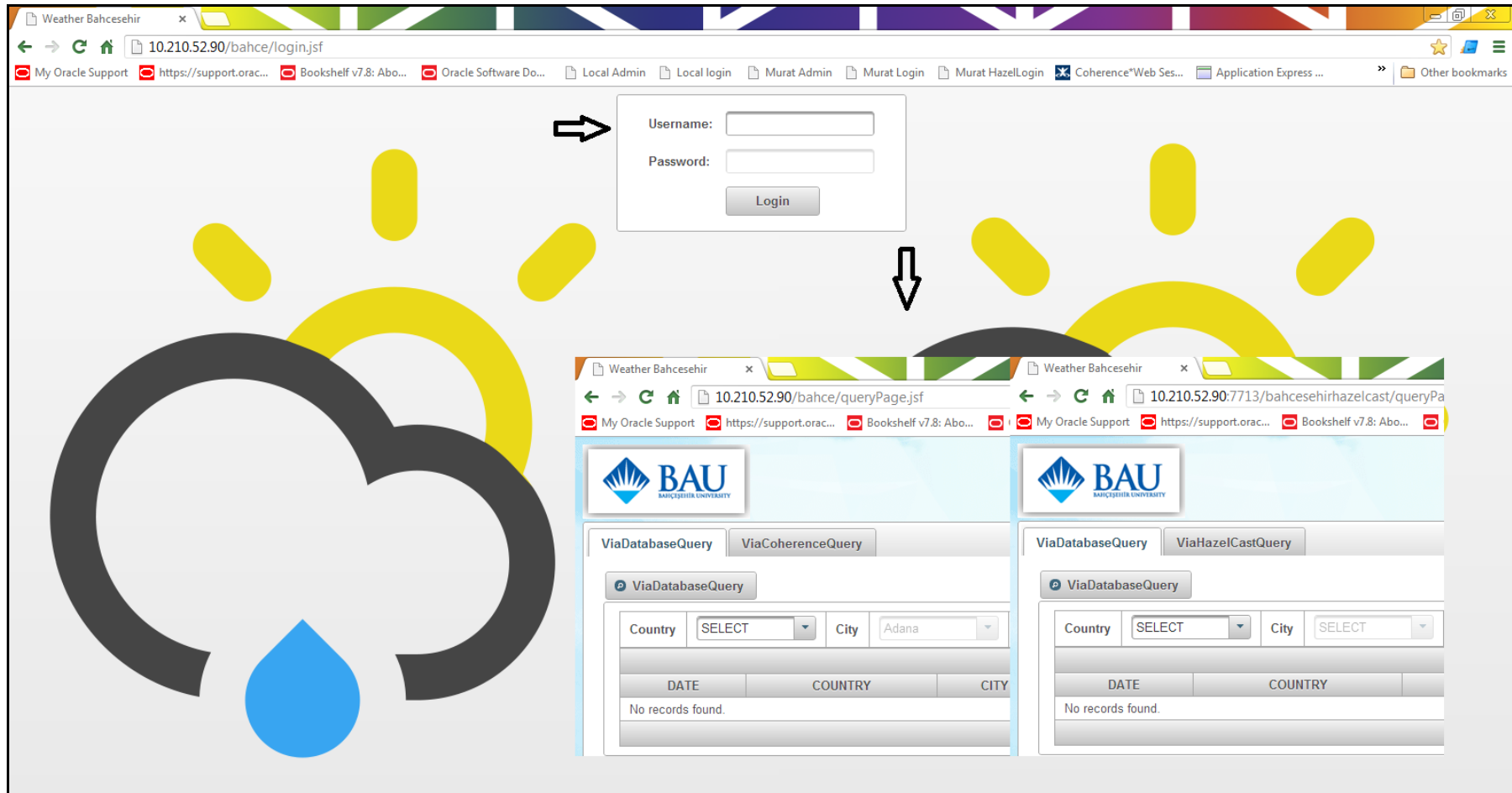
- THESIS-BAU-DMN
 - Environment
 - Deployments**
 - Services
 - Security Realms
 - Interoperability
 - Diagnostics

How do I...

- Install an Enterprise application
- Configure an Enterprise application
- Update (redeploy) an Enterprise application
- Start and stop a deployed Enterprise application
- Monitor the modules of an Enterprise

Only when these three applications are active on all the servers, the front-end applications are now ready for use. The front-end applications can now be accessed from the following URLs: “<http://10.210.52.90/bahce/login.jsf>” and “<http://10.210.52.90:7713/bahcesehirhazelcast/login.jsf>”. When these two URLs are typed in the browser, the login screen in Figure 3.28 will appear: the user can enter the necessary information, create a session in the system, and reaches the home page of the related applications. It is clear from the flowchart in Figure 3.28 that there are two different screens in the home page of each application, and divided into two groups: ViaDatabaseQuery for the traditional system, and for the cloud system, ViaCoherenceQuery or ViaHazelcastQuery screens.

Figure 3.28: Logging in the homepages of the bahce and bahcesehirhazelcast applications



After the user has logged in, he needs to choose between the combo boxes in order to be able to see the data on the screen. Figure 3.29 shows that it is obligatory for the user to choose the combo boxes sequentially: if the user doesn't choose "country", then "city" and "date" choices will appear as read-only. The reason for this obligation is that the correct data can be shown on the screen, the requested data can be classified, and in this way, the response duration of this data can be taken instead of the bulk data. If there wasn't such an obligation, the user might choose "country" or "city" randomly, and the city chosen might not be located in the country chosen. In that case, no data would be shown on the screen and the systems would be engaged by the requests sent in vain.

Figure 3.29: Selecting data in the combo boxes

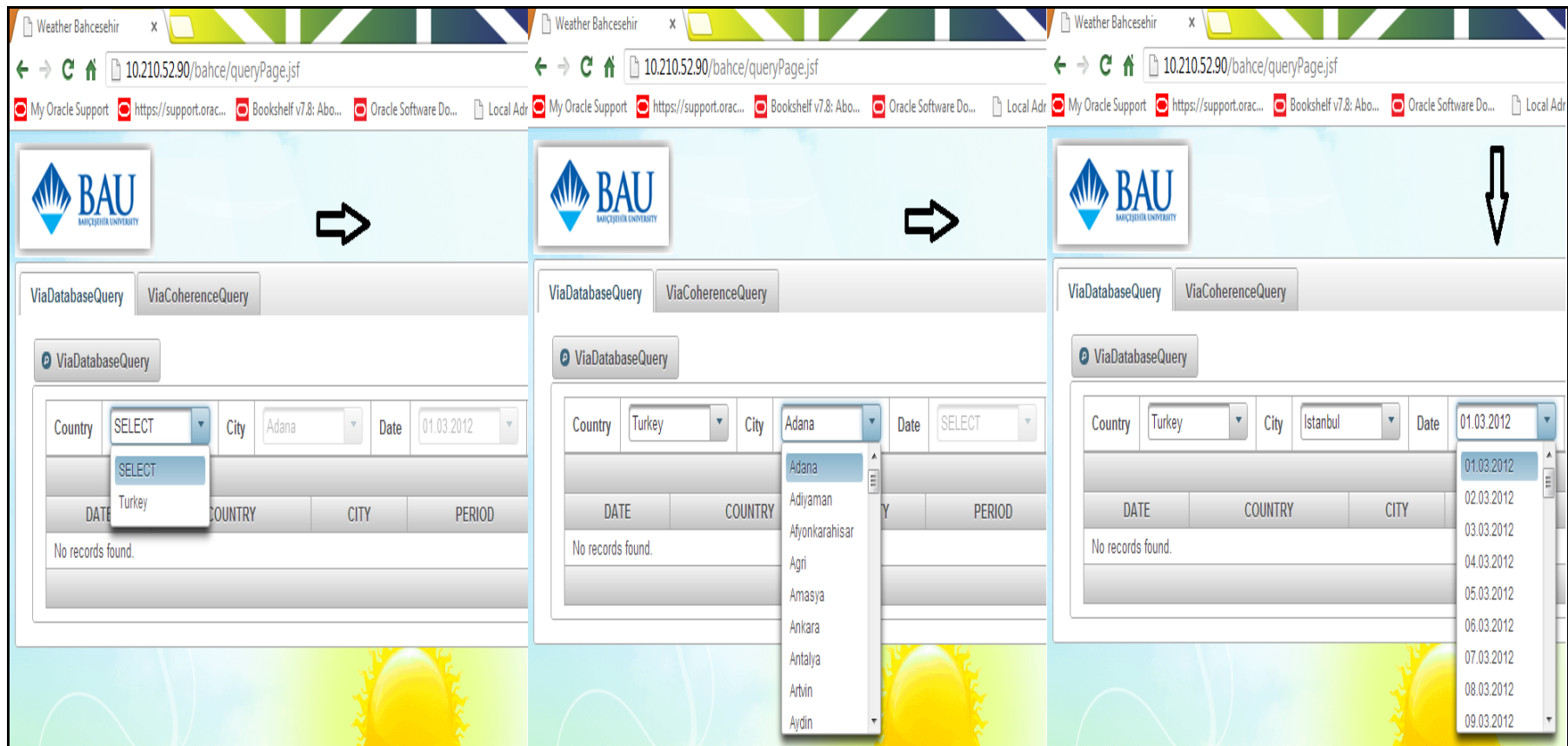


Figure 3.30 illustrates that the user makes a query by choosing the “Country”, “City”, and “Date” in the ViaDatabaseQuery screen. The system goes to the database, makes a query according to the chosen data, and shows the data on the screen in 0.388 seconds as in Figure 3.31:

Figure 3.30: Database query home page

Weather Bahcesehir x

10.210.52.90/bahce/queryPage.jsf

My Oracle Support https://support.orac... Bookshelf v7.8: Abo... Oracle Software Do... Local Admin Local login Murat Admin Murat Login Murat HazelLogin Coherence*Web Ses... Application Express ... Other bookmarks

BAU
BAHÇEŞEHİR UNIVERSITY

ViaDatabaseQuery ViaCoherenceQuery

ViaDatabaseQuery Linux-2.6.32-279.el6.x86_64-1.6.0_37-THESIS-BAU-M1-MAIN-1

Country Turkey City Istanbul Date 01.03.2012 Statement Duration :0.388

Showing 1-2 out of 2

| DATE | COUNTRY | CITY | PERIOD | TEMPERATURE | WINDCHILLTEMP | DESCRIPTION |
|------------|---------|----------|--------|-------------|---------------|-------------|
| 01.03.2012 | Turkey | Istanbul | Day | 22 | 25 | Clear |
| 01.03.2012 | Turkey | Istanbul | Night | 15 | 18 | Clear |

Showing 1-2 out of 2

Similarly, the user makes a query in the ViaCoherenceQuery screen by choosing the same “Country”, “City”, and “Date” data as ViaDatabaseQuery. Then, the system goes to the Coherence cache this time, makes a query according to the chosen data, and shows the data on the screen in 0.0020 seconds as can be seen from Figure 3.31:

Figure 3.31: Coherence query home page

The screenshot shows a web browser window with the URL `10.210.52.90/bahce/queryPage.jsf`. The page features the BAU (Balıkesir University) logo and navigation tabs for `ViaDatabaseQuery` and `ViaCoherenceQuery`. The `ViaCoherenceQuery` tab is active, displaying a search form with the following parameters:

- Country: Turkey
- City: Istanbul
- Date: 01.03.2012
- Statement Duration: :0.0020

The search results are displayed in a table with the following columns: DATE, COUNTRY, CITY, PERIOD, TEMPERATURE, WINDCHILLTEMP, and DESCRIPTION. The table shows two rows of data for the date 01.03.2012.

| DATE | COUNTRY | CITY | PERIOD | TEMPERATURE | WINDCHILLTEMP | DESCRIPTION |
|------------|---------|----------|--------|-------------|---------------|-------------|
| 01.03.2012 | Turkey | Istanbul | Night | 15 | 18 | Clear |
| 01.03.2012 | Turkey | Istanbul | Day | 22 | 25 | Clear |

Likewise, the user makes a query in the ViaHazelCastQuery screen by choosing the same “Country”, “City”, and “Date” data as ViaDatabaseQuery. Afterwards, the system again goes to the Hazelcast cache, makes a query according to the chosen data, and shows the data on the screen in 0.019 seconds as in Figure 3.32:

Figure 3.32: Hazelcast query home page

The screenshot shows a web browser window with the URL `10.210.52.90:7713/bahcesehirhazelcast/queryPage.jsf`. The page features the BAU (Balıkesir University) logo and navigation tabs for `ViaDatabaseQuery` and `ViaHazelCastQuery`. The `ViaHazelCastQuery` tab is active, showing a search form with the following parameters:

- Country: Turkey
- City: Istanbul
- Date: 01.03.2012
- Statement Duration Hazelcast: :0.019

The search results are displayed in a table with the following columns: DATE, COUNTRY, CITY, PERIOD, TEMPERATURE, WINDCHILLTEMP, and DESC. The table shows two rows of data for the date 01.03.2012.

| DATE | COUNTRY | CITY | PERIOD | TEMPERATURE | WINDCHILLTEMP | DESC |
|------------|---------|----------|--------|-------------|---------------|-------|
| 01.03.2012 | Turkey | Istanbul | Night | 15 | 18 | Clear |
| 01.03.2012 | Turkey | Istanbul | Day | 22 | 25 | Clear |

Another risk is that while the user is working on the screen, the server in which the user's session is stored may suddenly shut down due to a system failure. If the system is traditional, all the user's data become inaccessible along with the server because all the processes the user have been doing on the screen are stored in the session of the server that has shut down. Figure 3.33 also demonstrates that the user has logged in the system and made a database query in the server, "THESIS-BAU-M1-MAN1-2" and lasted 0.321 seconds:

Figure 3.33: User operations in THESIS-BAU-M1-MAN1-2

The screenshot shows a web browser window with the URL `10.210.52.90/bahce/queryPage.jsf`. The page features the BAU (Bahçeşehir University) logo and navigation tabs for `ViaDatabaseQuery` and `ViaCoherenceQuery`. The `ViaDatabaseQuery` tab is active, displaying a search form with the following parameters:

- Country: Turkey
- City: Istanbul
- Date: 01.03.2012
- Statement Duration: :0.321

Below the search form, a table displays the results, showing 1-2 out of 2 records. The table has the following columns: DATE, COUNTRY, CITY, PERIOD, TEMPERATURE, WINDCHILLTEMP, and DESCRIPTION.

| DATE | COUNTRY | CITY | PERIOD | TEMPERATURE | WINDCHILLTEMP | DESCRIPTION |
|------------|---------|----------|--------|-------------|---------------|-------------|
| 01.03.2012 | Turkey | Istanbul | Day | 22 | 25 | Clear |
| 01.03.2012 | Turkey | Istanbul | Night | 15 | 18 | Clear |

A red box highlights the system version information: `Linux-2.6.32-279.el6.x86_64-1.6.0_37-THESIS-BAU-M1-MAN1-2`.

In the Oracle Coherence, if a server shuts down due to a system failure, the sessions in this server are copied to the other servers, and the users can continue the processes that they have been doing in this server without repetitions. For instance, if the “THESIS-BAU-M1-MAN1-2” server (in Figure 3.34) shuts down due to a system failure, the user continues the processes on the “THESIS-BAU-M1-MAN1-1” server. Since the session of the server that has shut down is copied to the other server, the statement duration doesn’t change (0.321 seconds again), while it is only the server information that changes in Figure 3.34:

Figure 3.34: Session management

The screenshot shows a web browser window with the URL `10.210.52.90/bahce/queryPage.jsf`. The page features the BAU (Balıkesir University) logo and two tabs: `ViaDatabaseQuery` (selected) and `ViaCoherenceQuery`. Below the tabs is a search form with the following fields:

- Country: Turkey
- City: Istanbul
- Date: 01.03.2012
- Statement Duration: :0.321

The search results are displayed in a table with the following data:

| DATE | COUNTRY | CITY | PERIOD | TEMPERATURE | WINDCHILLTEMP | DESCRIPTION |
|------------|---------|----------|--------|-------------|---------------|-------------|
| 01.03.2012 | Turkey | Istanbul | Day | 22 | 25 | Clear |
| 01.03.2012 | Turkey | Istanbul | Night | 15 | 18 | Clear |

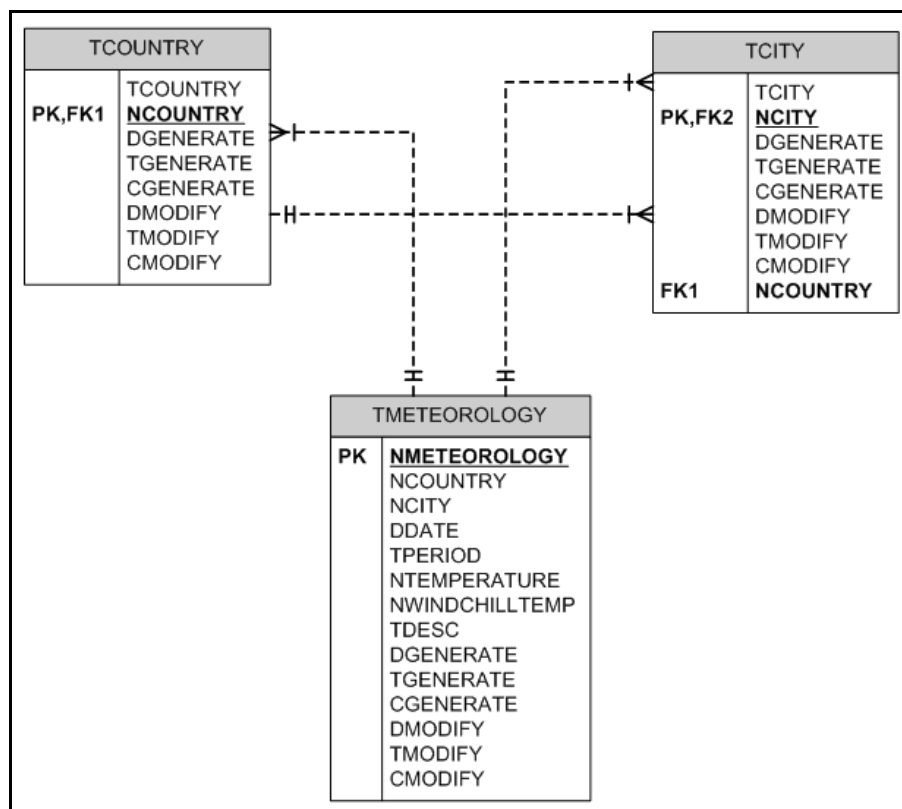
Two orange boxes highlight specific elements: one around the `Statement Duration :0.321` field and another around the session ID `Linux-2.6.32-279.el6.x86_64-1.6.0_37-THESES-BAU-IM1-MAN1-1` located in the top right corner of the page content area.

In summary, even when there is not much load applied to the systems, and the requests are sent to the systems with only a limited amount of data, the traditional system seems to be much slower than the cloud system in terms of the response duration. Just by looking at the statement duration, one can claim that the Oracle Coherence works faster than the database and the Hazelcast, while the Hazelcast works faster than the database. This is because the Oracle Coherence uses ExternalizableLite Serialization instead of the default serialization.

3.4.2. Database Development

In order to compare the traditional and cloud systems, and also to locate the backend data into the middleware, a database must be developed for the new-generation cloud. For this reason, an architecture consisting of three tables is needed in the database for the front-end applications that are coded with java.

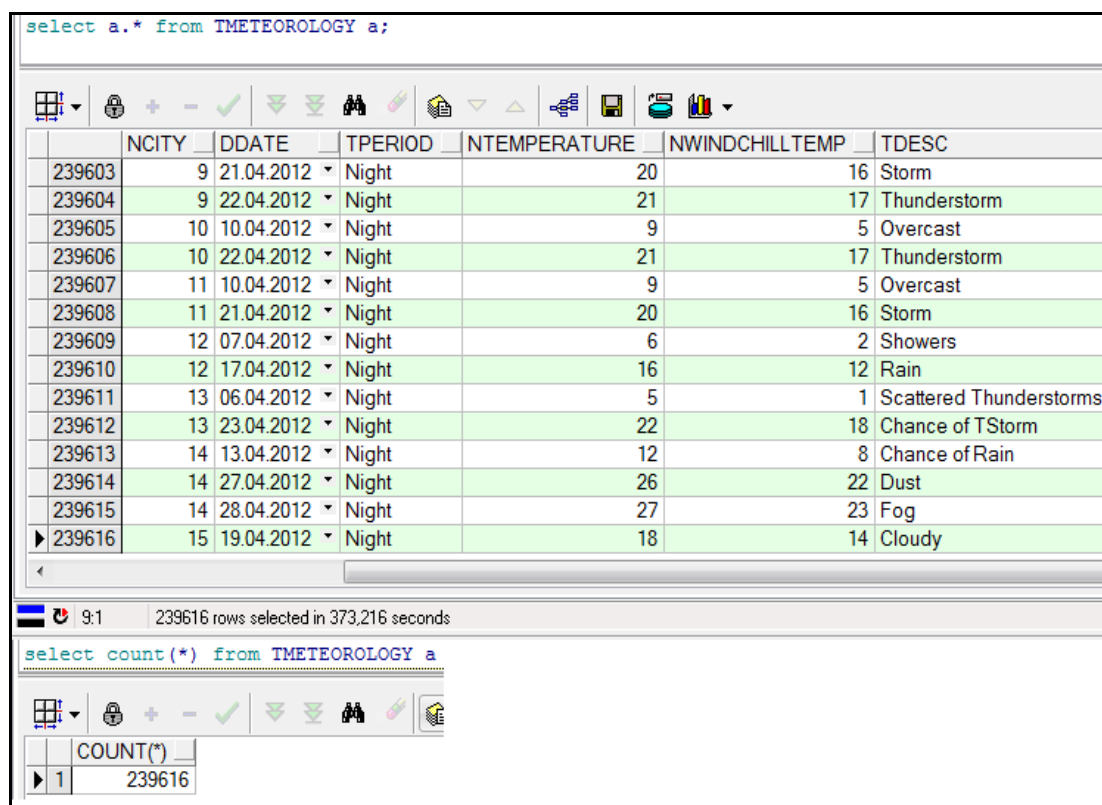
Figure 3.35: Entity relationship diagram



According to Figure 3.35, the names of these three tables are as follows: TCOUNTRY includes country data; TCITY city data, and TMETEOROLOGY has country, city, date, period, temperature etc. data. It can also be observed from Table 3.35 that there is 1 to many relationship between TCOUNTRY and TCITY tables; whereas TMETEOROLOGY table has 1 to many relationship with the TCOUNTRY and TCITY tables.

After these three tables are created, the data need to be inserted into the tables. In this study, considering its relationships to the other tables, almost 240000 rows of data have been inserted into the TMETEOROLOGY table by writing a pl/sql procedure as can be seen from Figure 3.36:

Figure 3.36: TMETEOROLOGY table and total count of rows



Consequently, the processes in production environment cannot be done with a single table; instead, more than one table or view are used. In order to obtain a common result from these tables or views, joint statements are written. Every joint statement means

loss for database execution time. Still, in this study, the data of the front-end applications have been created with joint statements and by using more than one table. Therefore, performance results similar to the production environment can be obtained in the phase of the load test.

4. PERFORMANCE ANALYSIS

4.1 COMPARISON OF IN-MEMORY DATA GRIDS WITH THE DATABASE

This chapter presents the results of the performance analysis used in comparing the new-generation, cloud-based in-memory data grids with the traditional database. Two performance tests have been carried out by using *Apache Jmeter* – “an open source software” that is “designed to load test functional behaviour and measure performance” (Apache 2013). *Apache Jmeter* can load and performance test many different server types: *web – HTTP, HTTPS; SOAP; database via JDBC; LDAP; JMS; mail – SMTP(S), POP3(S) and IMAP(S); and native commands or shell scripts* (Apache 2013).

In this study, the webservice test – SOAP – has been done for the “*bahce*” and “*bahcesehirhazelcast*” applications by simulating the behaviours of 1000 users (threads) on the systems. The users of this study requested 1000 different data at intervals of one second from the *database, Oracle Coherence* and *Hazelcast* systems, and thus provided load in each of these three systems. The performances of the traditional database and the two in-memory data grids were then compared in terms of the response time, CPU usage and memory usage. The Oracle Coherence application (“*bahce*”) used the distributed caching scheme in the first part of the performance analysis here because there is only one caching scheme in Hazelcast – the distributed cache. However, the comparisons of the three caching schemes within the Oracle Coherence system were also made in the second part of the performance analysis.

4.1.1 The Response Time Difference between Database and In-Memory Data Grids

Response time indicates the length of time spent on completing a request, and the longer the response time is, the longer the user waits for the response of the system. Figure 4.1 illustrates the results of the response time for the database. According to Figure 4.1, the average response time for the database is 236 seconds.

Figure 4.1: The response time results for the database



Figure 4.2 shows that the average response time for Oracle Coherence is 24 ms, when the same amount of load is put on the Oracle Coherence.

Figure 4.2: The response time results for Oracle Coherence

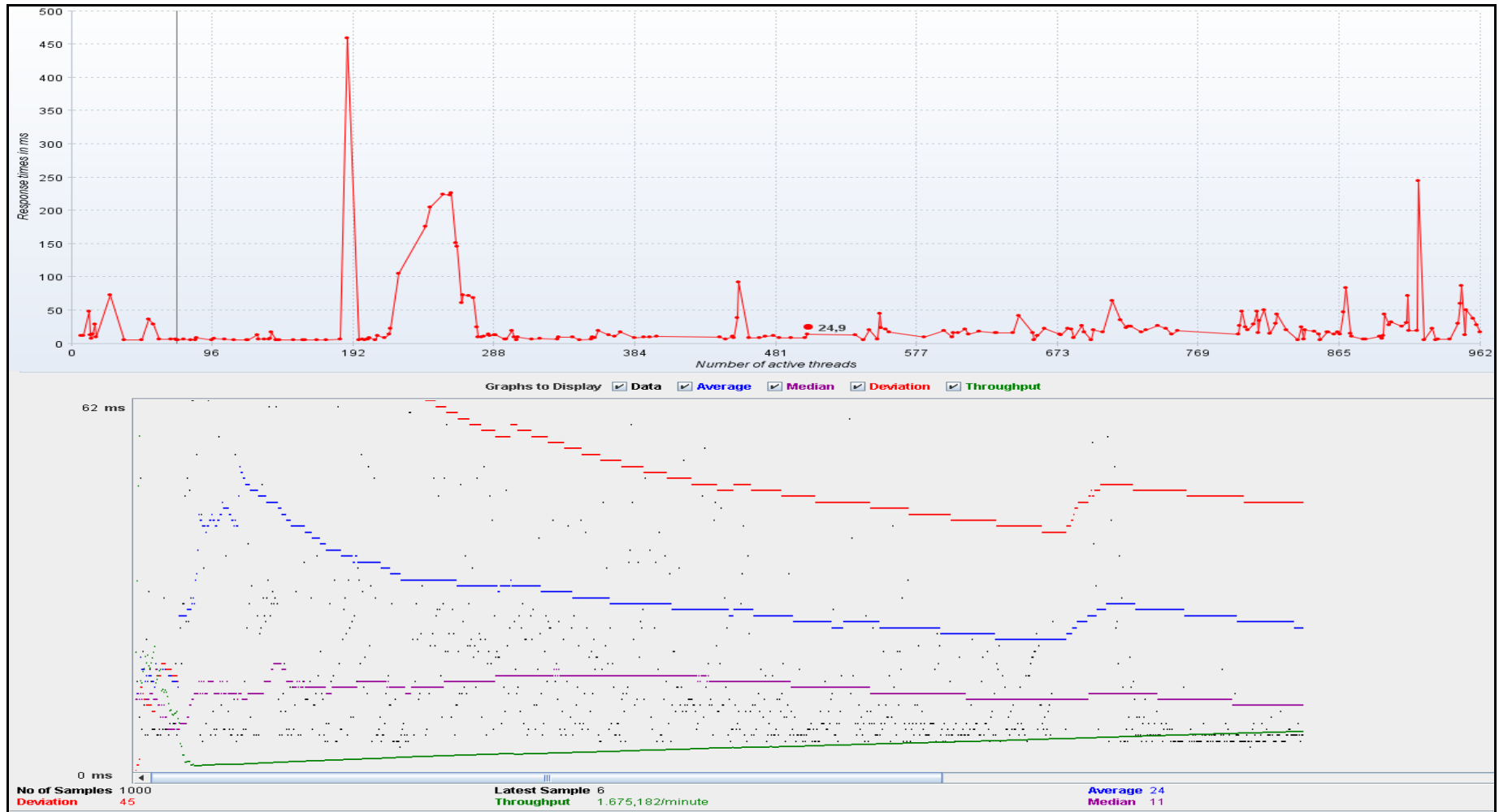
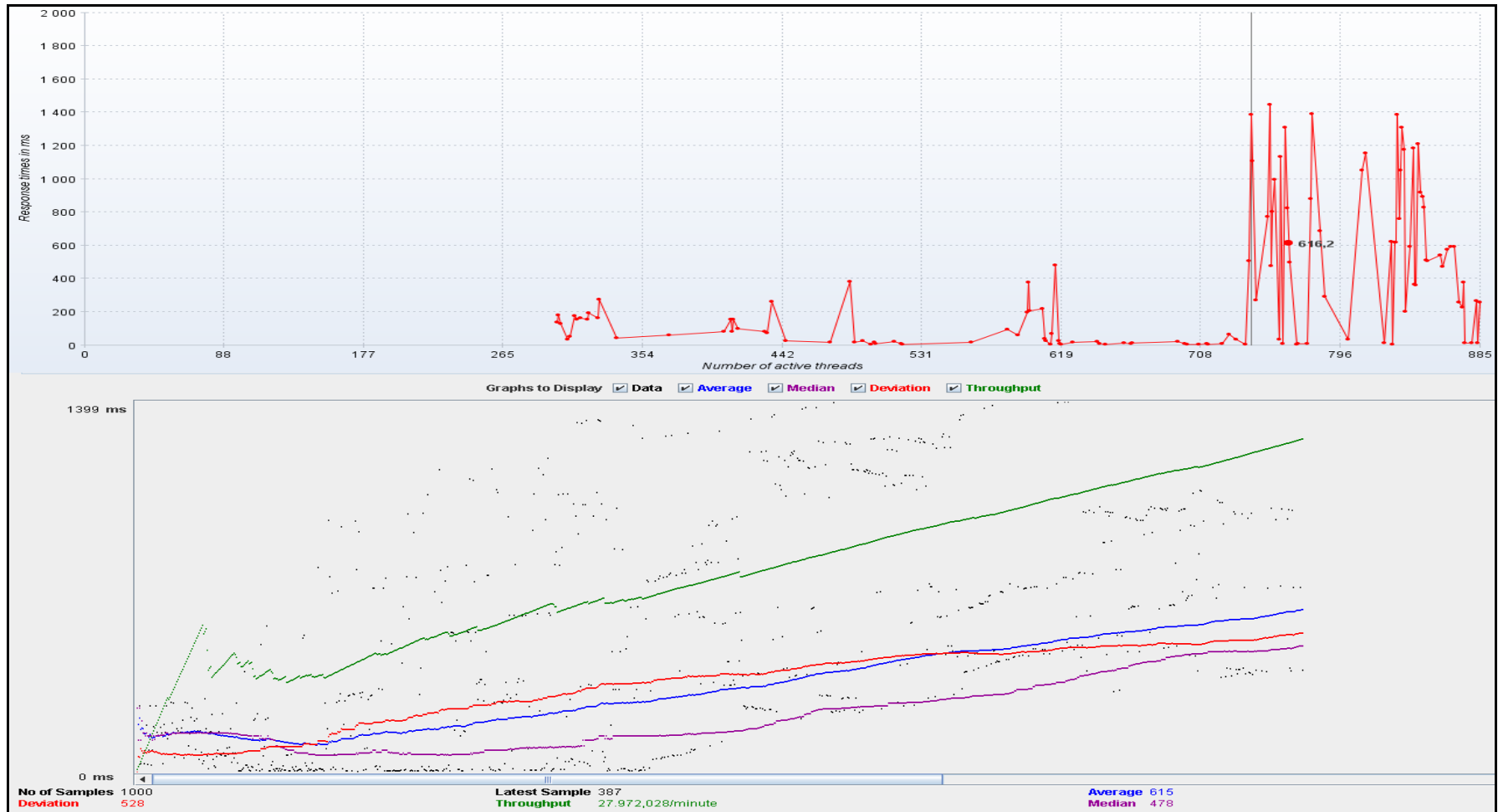


Figure 4.3 indicates that the average response time for Hazelcast is 615 ms in the same situation.

Figure 4.3: The response time results for Hazelcast



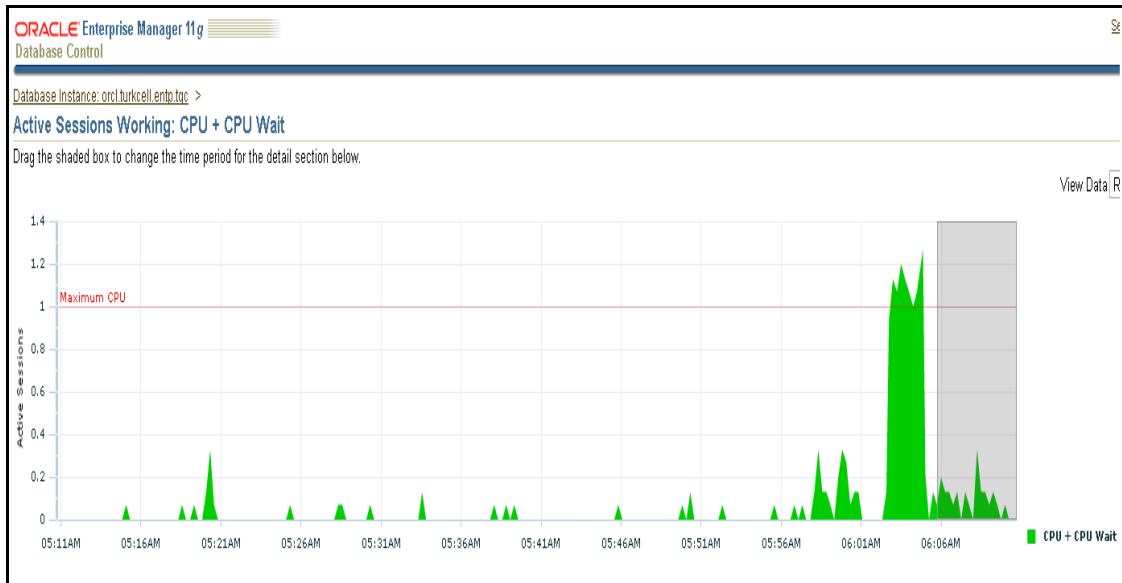
Compared to the database and Hazelcast, it is Oracle Coherence that responds the fastest to the requests sent by 1000 users. Secondly, there comes the Hazelcast, which is slower than the Oracle Coherence, whereas it is the database that is slowest in processing 1000 different data and responding to 1000 users. The long response time of the database may cause the user to continually enter the system, which will in return lead to a further slowdown in the middle tier. However, with the Oracle Coherence, there is no waiting in the backend system, and the thread pools are not filled up; as a result, CPU and memory usage won't increase. In short, the user of the Oracle Coherence wouldn't complain about the slowness of the system.

4.1.2 The Differences of CPU and Memory Usage between Database and In-Memory Data Grids

CPU is related to the amount of work the processor is performing, and the processing power is reduced if an application is running at the maximum level of CPU. That's why, it is important to control how much of the CPU power is consumed by an application. High CPU usage prevents the creation of new processes as well as causing system slowdowns and even system shutdowns if the 100% CPU usage is persistent. For this reason, the CPU usage shouldn't be close to the maximum value of 100%.

Another important factor in the performance analysis is memory usage, which refers to the amount of memory being used by an application when it starts up or is running. The memory allocated in the system is limited both in the database and in Java applications. As a result, the application slows down, if the allocated memory for the application is used at the maximum level. It is important to control the usage of the allocated memory by the applications. With the purpose of measuring the CPU and memory usages by the database, two tools – *Oracle Enterprise Manager* and *TOAD for Oracle* – were used. Figure 4.4 presents the results of the CPU usage in the database: the CPU usage of the database reaches the maximum level of 1 GB within five minutes.

Figure 4.4: The results of CPU usage in the database



In Figure 4.5, the usage of SGA (System Global Area) memory can be observed: SGA memory refers to the shared memory area for Oracle processes in the RAM, and shared pool is a RAM area within the SGA. Shared pool is especially important for the database because it stores the identical sql statements and prevents repeated parsing of sql. It is evident from Figure 4.5 that 500 mg of the SGA memory (700 mg) is set as shared pool, and almost 450 mg of the shared pool is filled up due to the load. Since the shared pool is overloaded, the identical sql statements cannot be stored here, and the database starts to parse every sql statement.

Figure 4.5: The results of memory usage in the database

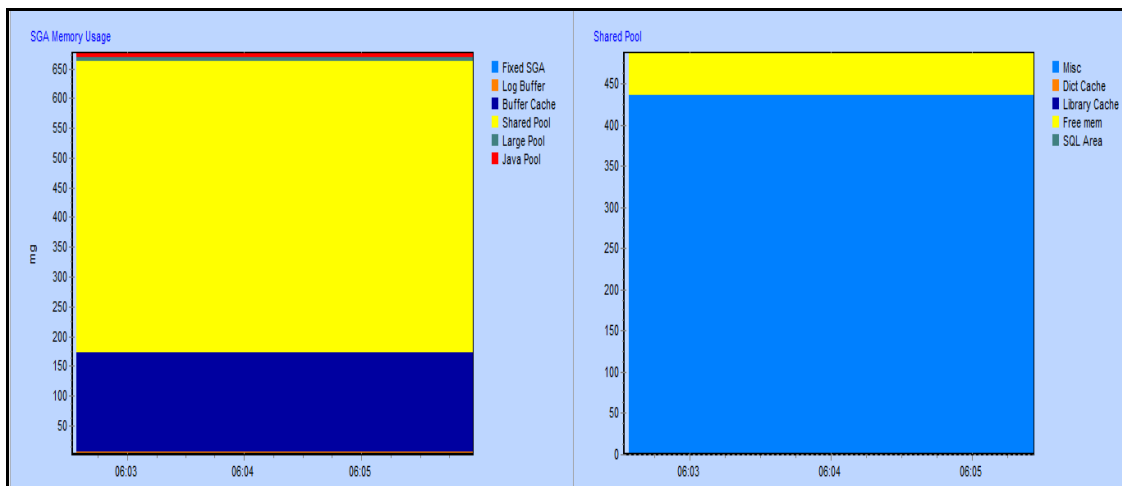


Figure 4.6: Physical I/O on the disk

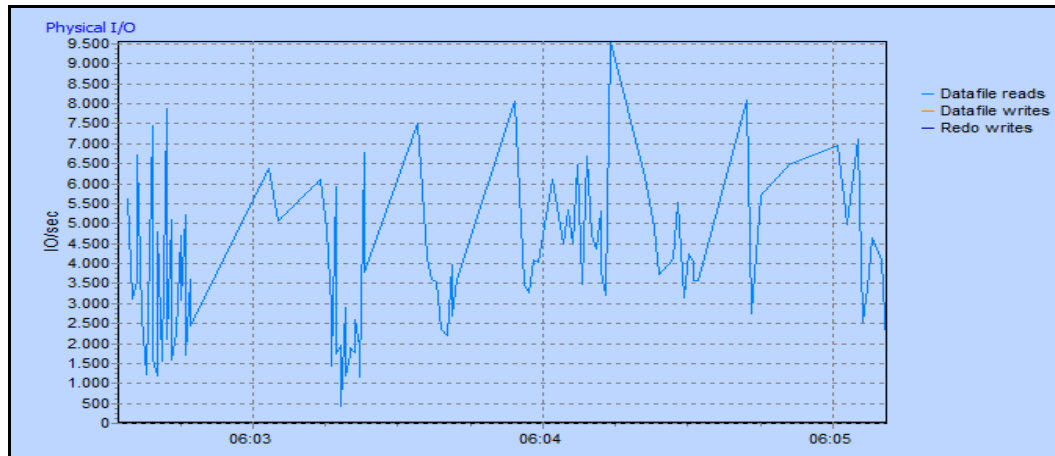
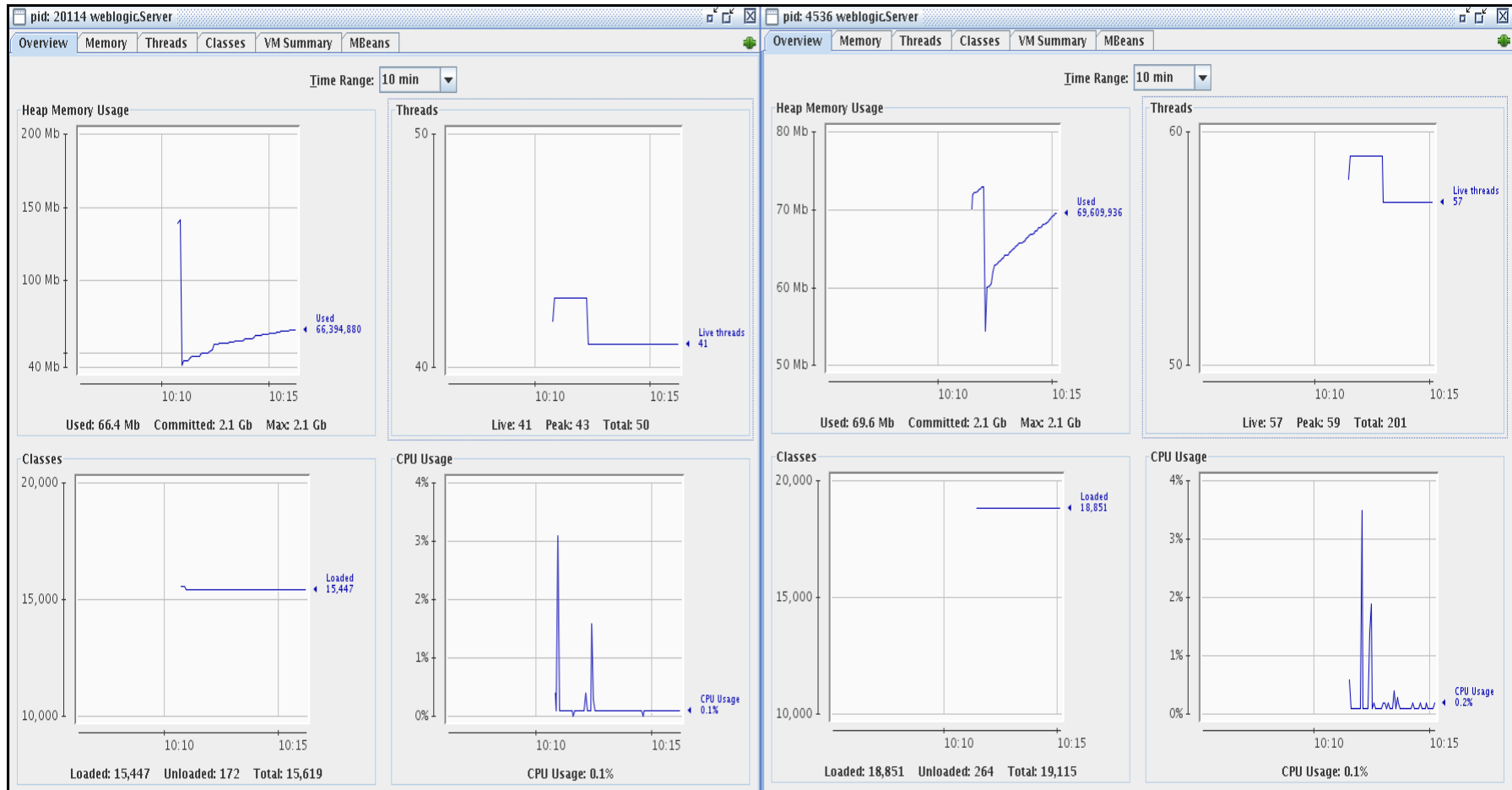


Figure 4.6 demonstrates the frequency of data file reads in the database: within 60 seconds, the frequency of data file reads increased from 500 IO to 9500 IO. Therefore, when the database is under the load, it performs too many physical I/O reads.

In order to measure the CPU and memory usages by the Oracle Coherence, *JConsole* was used. Since the coherence servers were in the distributed caching scheme during the load test, two different CPU diagrams were obtained for the two Coherence servers with respect to the process IDs. It is clear from Figure 4.7 that the average CPU usage equal to 1.7 %, while the average memory usage amounts to 68 mb in the two servers. Besides, it can be seen from Figure 4.7 that there is a sharp decrease in the memory usage – from 140 mb to 40 mb, and a sudden increase in the CPU usage – from 0.1 % to 3 %. This is because of the garbage collection before the load test. The aim of the garbage collection is to supply consistent data for CPU and memory checks.

Figure 4.7: The results of CPU and memory usage in Oracle Coherence



With the help of *JConsole* again, the CPU and memory usages by Hazelcast were calculated. Figure 4.8 shows that the CPU usage increased from 0.6 % to 16 % during the load test, while the memory usage increased from 100 mb to 159 mb.

Figure 4.8: The results of CPU and memory usage in Hazelcast



In conclusion, when compared to the database, the in-memory data grids in this study use much less CPU than the database that tends to use the maximum capacity of CPU. When the in-memory data grids are compared, it is found that the Oracle Coherence (1.7%) uses almost 10 times less CPU than Hazelcast. High CPU usage causes system slowdown and unless the system capacity is increased, the system cannot function properly. For this reason, the Oracle Coherence seems more advantageous in terms of CPU usage.

Compared to the database, the in-memory data grids in this study use much less memory than the database that tends to use the maximum capacity of memory. When the in-memory data grids are compared, it is found that the Oracle Coherence (68 mb) uses 2.5 times less memory than Hazelcast. A temporary solution for high memory usage is increasing memory so that the applications shouldn't slow down. Therefore, the use of the Oracle Coherence is found to be more convenient than both the database and Hazelcast, if memory usage is in question.

Table 4.1: Comparison Of In-memory Data Grids With The Database

| | Response Time | CPU | Memory |
|------------------|----------------------|------------|---------------|
| Database | 236 sn | Overloaded | 450 mg |
| Coherence | 24 ms | 1.7% | 68mb |
| Hazelcast | 615 ms | 16% | 159mb |

4.2 COMPARISON OF DISTRIBUTED-REPLICATED-NEAR CACHING SCHEMES IN ORACLE COHERENCE

In the second part of the performance analysis, the performance results of the three caching schemes – distributed, replicated, and near caches – in the Oracle Coherence were compared in terms of the response time, CPU and memory usage. When the Oracle Coherence was compared to Hazelcast, the results for the distributed caching were already presented in the first part of the performance analysis, and therefore are only reviewed here. Below is presented the information on the three types of Coherence schemes and VM (Virtual Machine) summary for their application servers. Firstly, Figure 4.9 illustrates the distributed cache server named “THESIS-BAU-MAN0-2”.

Figure 4.9: The distributed cache server of Oracle Coherence

The screenshot displays the Oracle Coherence weblogicServer interface for a distributed cache server. The left pane shows a tree view with 'Cache' expanded to 'DistributedCache'. The right pane shows 'Attribute values' for the cache:

| Name | Value |
|-------------------|-------|
| AverageMissMillis | 0.0 |
| AveragePutMillis | 0.0 |
| BatchFactor | 0.0 |
| CacheHits | 6940 |
| CacheHitsMillis | 0 |
| CacheMisses | 0 |
| CacheMissesMillis | 0 |
| CachePrunes | 0 |

Below the attributes, system and VM information is shown:

- Operating System:** Linux 2.6.32-279.el6.x86_64
- Architecture:** amd64
- Number of processors:** 4
- Committed virtual memory:** 4,744,780 kbytes
- Total physical memory:** 16,334,504 kbytes
- Free physical memory:** 5,617,140 kbytes
- Total swap space:** 4,128,760 kbytes
- Free swap space:** 4,076,152 kbytes

VM arguments include: `-Dweblogic.Name=THESIS-BAU-M1-MAN0-2 -Xms2g -Xmx2g -Xmn600m -XX:SurvivorRatio=8 -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:PermSize=256m -XX:MaxPermSize=256m -verbose:gc -XX:+PrintGCTimeStamps -XX:+PrintGCDetails -Xloggc:/data/wlsadmin/wlserver1036/wlserver_10.3/domains/THESIS-BAU-DMN/logs/GC_logs.THESIS-BAU-M1-MAN0-2 -Dweblogic.management.server=http://10.210.52.90:7700 -DHostName_WLSName=murat-M1-MAN0-2 -Dweblogic.Stdout=/data/wlsadmin/wlserver1036/wlserver_10.3/domains/THESIS-BAU-DMN/logs/System_Out.M1-MAN0-2 -DAPPLPLOGDIR=/data/wlsadmin/wlserver1036/wlserver_10.3/domains/THESIS-BAU-DMN/logs/AppLogs -DLOGTRAILER=THESIS-BAU-M1-MAN0-2`

When the Coherence is set to the replicated scheme, it means that all the member servers in the cluster store the local cache. Figure 4.10 illustrates the replicated scheme of the Coherence servers, where the front-end server, “THESIS-BAU-MAN1-2”, stores the cache in its local environment.

Figure 4.10: The replicated cache server of Oracle Coherence

The screenshot displays the Oracle Coherence weblogicServer interface for a replicated cache server. The left pane shows a tree view with 'Cache' expanded to 'ReplicatedCache'. The right pane shows 'Attribute values' for the cache:

| Name | Value |
|-------------------|---|
| AverageGetMillis | 0.0 |
| AverageHitMillis | 0.0 |
| AverageMissMillis | 0.0 |
| AveragePutMillis | 0.0 |
| BatchFactor | 0.0 |
| CacheHits | 15552 |
| CacheHitsMillis | 0 |
| CacheMisses | 5184 |
| CacheMissesMillis | 0 |
| CachePrunes | 0 |
| CachePrunesMillis | 0 |
| Description | Implementation: com.tangosol.net.cache.LocalCache |

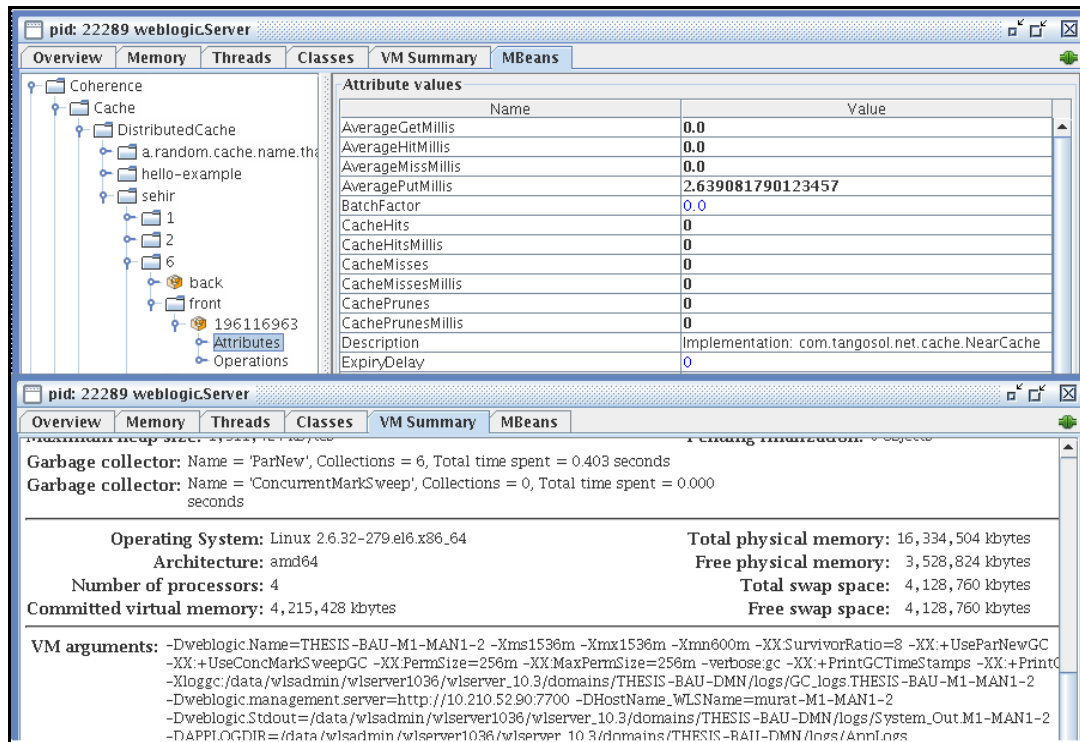
Below the attributes, system and VM information is shown:

- Operating System:** Linux 2.6.32-279.el6.x86_64
- Architecture:** amd64
- Number of processors:** 4
- Committed virtual memory:** 4,215,428 kbytes
- Total physical memory:** 16,334,504 kbytes
- Free physical memory:** 3,528,824 kbytes
- Total swap space:** 4,128,760 kbytes
- Free swap space:** 4,128,760 kbytes

VM arguments include: `-Dweblogic.Name=THESIS-BAU-M1-MAN1-2 -Xms1536m -Xmx1536m -Xmn600m -XX:SurvivorRatio=8 -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:PermSize=256m -XX:MaxPermSize=256m -verbose:gc -XX:+PrintGCTimeStamps -XX:+PrintGCDetails -Xloggc:/data/wlsadmin/wlserver1036/wlserver_10.3/domains/THESIS-BAU-DMN/logs/GC_logs.THESIS-BAU-M1-MAN1-2 -Dweblogic.management.server=http://10.210.52.90:7700 -DHostName_WLSName=murat-M1-MAN1-2`

When the Oracle Coherence is set to the near scheme, all the member servers in the cluster store the local and distributed cache. It can be observed from Figure 4.11 that the front-end server, named “THEESIS-BAU-MAN1-2”, functions as the near cache.

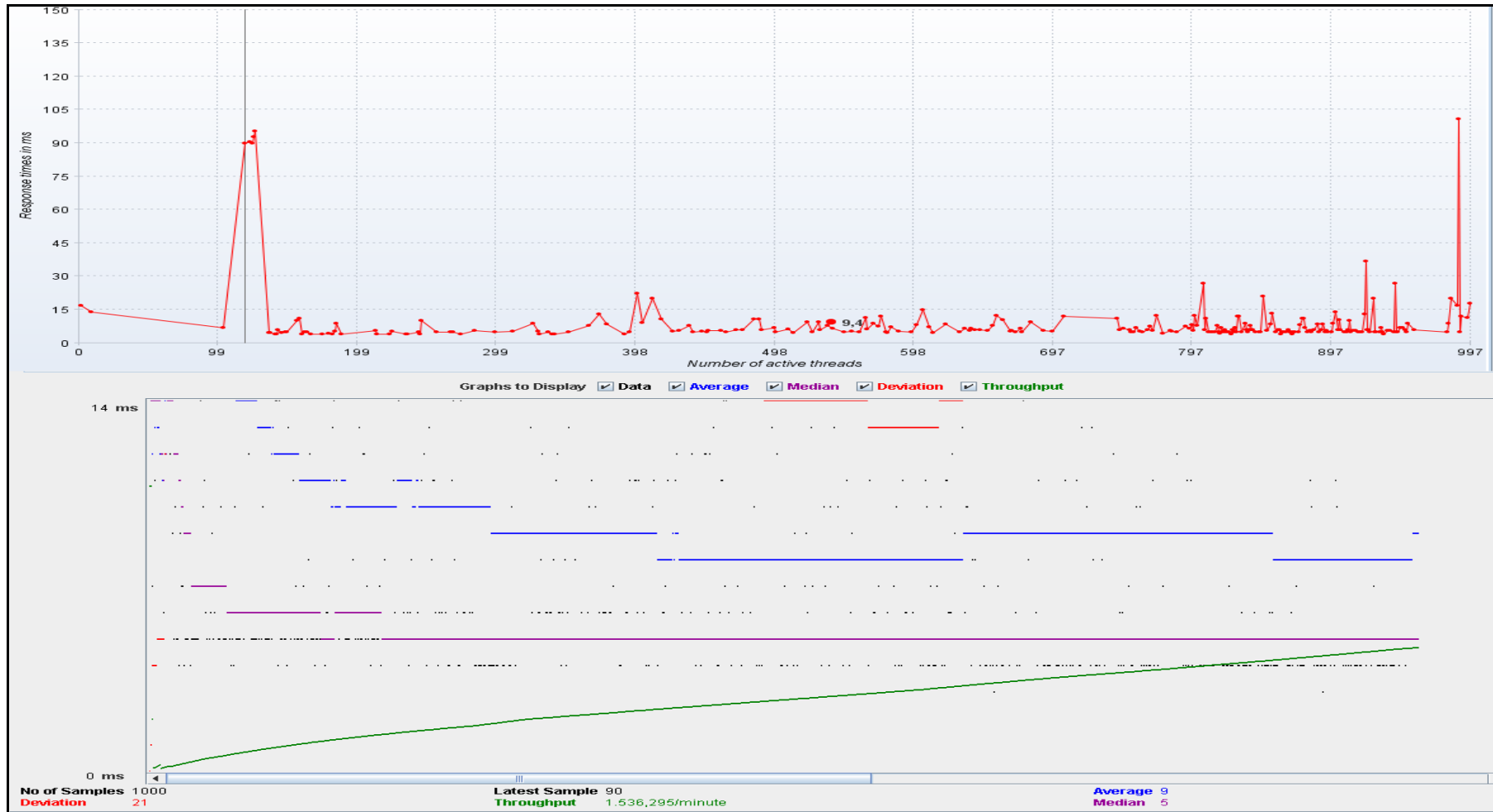
Figure 4.11: The near cache server of Oracle Coherence



4.2.1 The Response Time Difference between Distributed-Replicated-Near Caches

The average response time for the distributed scheme was previously found to be 24 ms in Figure 4.2. When the same load test was applied to the replicated scheme, the average response time was calculated as 9ms as in Figure 4.12. The reason why the replicated scheme was so fast at responding to the requests of 1000 users is that each server now stores the cache in its local environment and obtains the requested data from its own local cache.

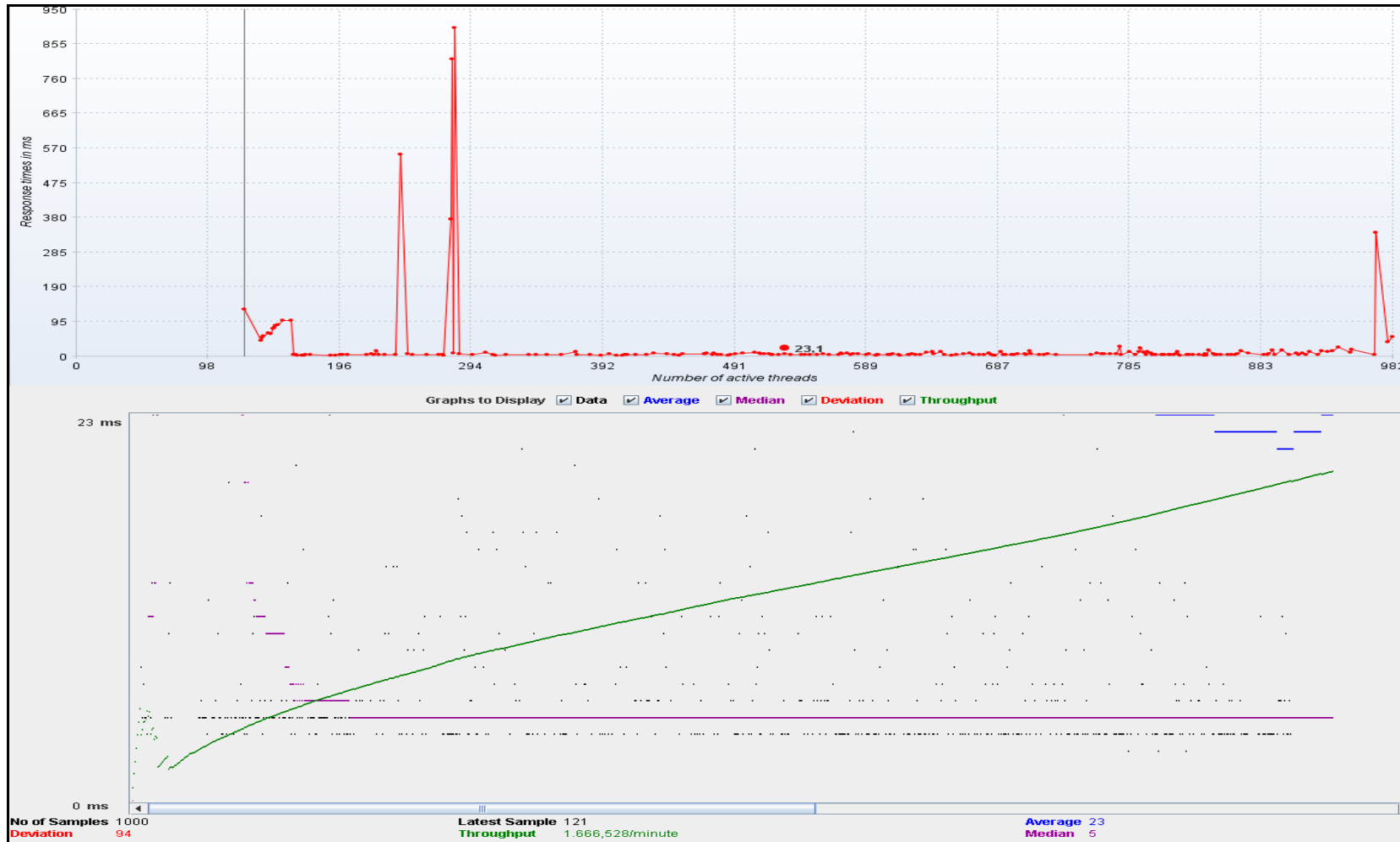
Figure 4.12: The response time results of the replicated scheme



According to Figure 4.13, the near cache provided data in 23ms at the end of the load test. This is because the near scheme uses distributed and local caches. Now that the cache is divided into the front and back tier, the data storage is shared between the front and back tiers, and the data that does not exist in the front tier is obtained from the back tier.

According to the response time results in all the three caching schemes, the near cache (23 ms) responded at a similar speed to the distributed cache (24 ms) because the data used in the performance analysis were not located in the front tier and obtained from the distributed cache.

Figure 4.13: The response time results of the near scheme

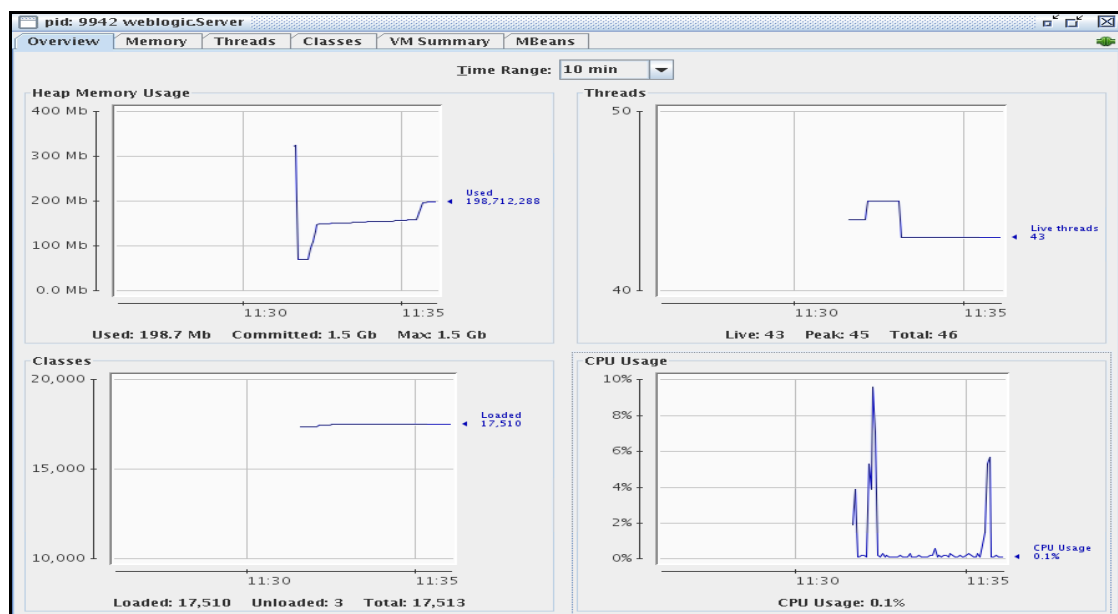


Briefly, while the distributed and near caches are similar in terms of the response time (24 ms, 23 ms), it is the replicated cache that has the shortest response time among all the three schemes (9 ms). The reason is that all the servers in the replicated scheme store the cache in their local environment and can reach the data faster than the other caching types. As a result, it is advisable to prefer the replicated cache with medium-sized cache for a fast system. If the cache size is big, near cache or distributed cache may be preferred, although the system will not work as fast as in the replicated scheme.

4.2.2 The Differences of CPU and Memory Usage between Distributed-Replicated-Near Caches

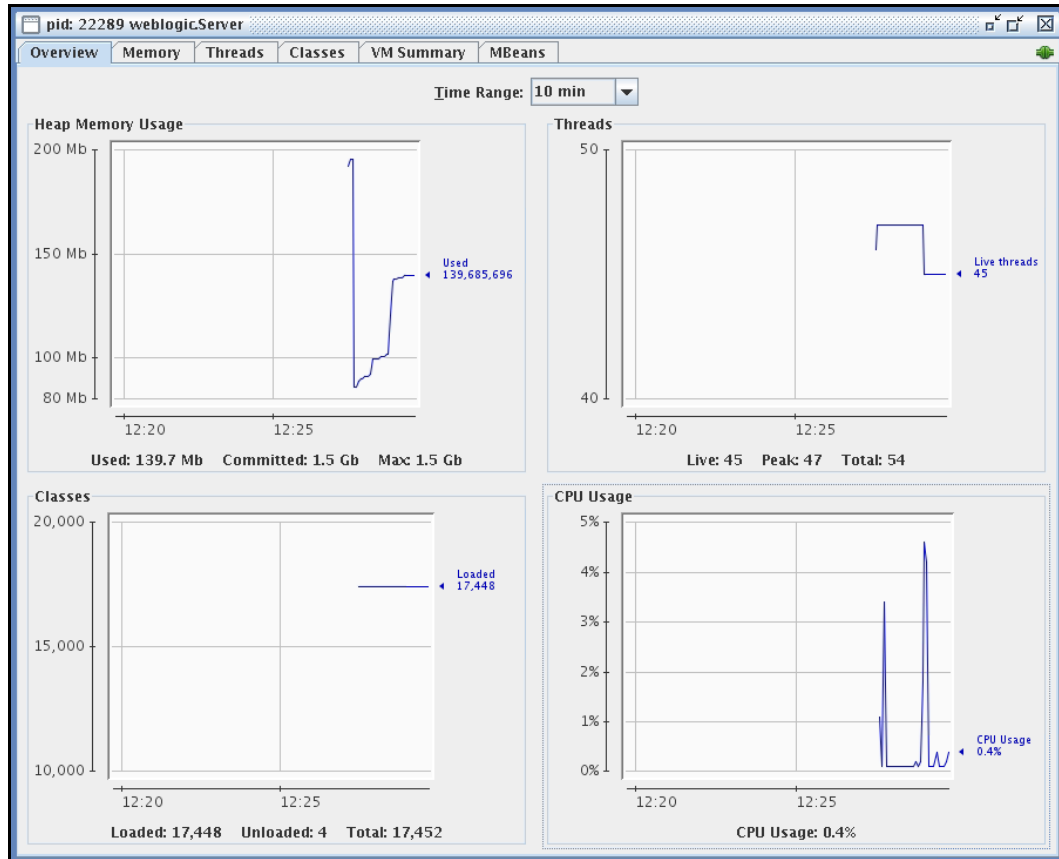
Jconsole is also used here to measure CPU and memory usages of the three caching schemes in the Oracle Coherence. The CPU usage of the distributed cache was previously calculated as 1.7 % under the load (Figure 4.7), while its memory usage equaled to 68 mb. Figure 4.14 shows that the CPU usage of the replicated scheme amounts to almost 6 %, whereas its memory usage is 199 mb. In addition, due to garbage collection, there is a sharp decrease in the memory usage of the replicated scheme – from 310 mb to 90 mb, as well as a sudden increase in its CPU usage – from 0.1 % to 9 %.

Figure 4.14: The results of CPU and memory usage in the replicated scheme



When the same load test is applied to the near cache, the results presented in Figure 4.15 have been obtained:

Figure 4.15: The results of CPU and memory usage in the near scheme



According to Figure 4.15, the near cache used nearly 5 % of the CPU power, whereas it used nearly 140 mb of the total memory (1.5 GB). Because of the garbage collection, the CPU usage suddenly increased from 0.4 % to 4 %, while the memory usage sharply decreased from 200 mb to 90 mb.

In summary, when the CPU usages of the cache schemes are listed in an increasing order, the distributed scheme (1.7%) consumes the least CPU power, with the near scheme (5%) in the second row, and finally, the replicated scheme uses the biggest part of the CPU capacity (6%). The reason why the near and replicated schemes make similar and more use of the CPU power is that the servers that use both of these two schemes store the cache in their local environments.

Also, when these three schemes are compared in terms of their memory usage, it is again the distributed scheme that consumes the least amount of the memory (68 mb). Following the distributed scheme, the near scheme uses 139 mb of the memory, but it is the replicated scheme (198 mb) that makes the highest consumption of the memory capacity. Since the servers of the near and replicated schemes store the cache in their local environments, the distributed scheme is again far ahead of these two caching schemes, if the memory usage is being considered.

In conclusion, the Oracle Coherence is found to be superior over the database and the other in-memory data grid, Hazelcast, in terms of the response time, CPU and memory usages. As for the best option among the three caching schemes, the distributed cache is found to be superior over the near and replicated caches in terms of CPU and memory usages. However, when it comes to the response time factor, the replicated cache has outperformed the distributed and near caches. In this study, the distributed scheme was preferred because it separated the front-end servers from the Coherence servers, and saved the system from extra work load and additional capacity (CPU and memory).

Table 4.2: Comparison Of Distributed-Replicated-Near Caching Schemes in Oracle Coherence

| | Response Time | CPU | Memory |
|--------------------------|----------------------|------------|---------------|
| Distributed Cache | 24 ms | 1.7% | 68 mb |
| Replicated Cache | 9 ms | 6% | 199 mb |
| Near Cache | 23 ms | 5% | 140 mb |

5. CONCLUSIONS AND RECOMMENDATIONS

Cloud computing is an innovative way of computing where clients can access computing resources over the internet without buying and installing their own IT infrastructure. Since clients are only responsible for paying their metered usage and needn't care about the functioning or maintenance of services, cloud computing is resembled to utilities like water, electricity and telephone. Although its popularity is increasing with individuals, different businesses and governments, authorities are still divided on adopting the cloud. While some people regard cloud computing as an overhyped technology, others see it as the next revolutionary move in information technology due to its cost-effective, reliable, practical and innovative solutions.

In this study, three cloud-based applications were designed and developed by using Java and two different in-memory data grids: the “*bahce*” and “*sehir*” applications with Oracle Coherence, and the “*bahcesehirhazelcast*” application with Hazelcast. To optimize scalability, reliability and performance of Oracle Coherence in accessing and storing data, an Apache Webserver, two Weblogic managed servers, two Coherence managed servers, one Hazelcast server and an Oracle Database were used in the design of the architecture. The architecture of this study simulated a weather service that must respond to update requests of their clients continuously, consistently and rapidly.

After the architectural design and development of the applications were described, the performances of the Oracle Database and the two in-memory data grids were tested by using Apache Jmeter and analyzed to reveal the differences between the traditional client/server computing and internet-based cloud computing, and also to compare the performances of the two cloud products, Oracle Coherence and Hazelcast. A further analysis was undertaken to identify the optimal caching scheme of the three: distributed, replicated, and near caches in Oracle Coherence. In the same way as the database and in-memory data grids, the performances of the three caching schemes were compared in terms of the response time, CPU and memory usages. The findings of the performance analyses can be summarized as in the following:

1. When the database and in-memory data grids were compared in terms of the response time, Oracle Coherence was found to be the fastest at responding to the requests. After Oracle Coherence, there comes its counterpart, Hazelcast, while the database was the slowest. Since there was almost no waiting in the backend system of Oracle Coherence, its users are supposed to have no complaints about the slowness of the system.

2. With respect to CPU usage, Oracle Coherence was the most economical of all, using almost 10 times less CPU than Hazelcast. Unlike the in-memory data grids, the database consumed the maximum capacity of CPU. Since high CPU usage leads to system slowdowns, the users of Oracle Coherence are at an advantage due to its low CPU usage.

3. In terms of memory usage, Oracle Coherence was again the most economical of all, using 2.5 times less memory than Hazelcast. Unlike the in-memory data grids, the database consumed the maximum capacity of memory. Although increasing its capacity might temporarily solve the problem of high memory usage, Oracle Coherence was found to be more convenient than the database and Hazelcast.

4. When the three caching schemes in Oracle Coherence were compared in terms of the response time, the distributed and near caches provided similar results. However, the replicated cache outperformed them with the shortest response time because all the servers in the replicated scheme store the cache in their local environment and allow faster access to the data. As a result, the replicated cache can be preferred as long as the cache is medium-sized for a faster system.

5. When the three caching schemes in Oracle Coherence were compared in terms of CPU usage, the distributed scheme was found to consume the least CPU power, while the near and replicated schemes made similar and more use of the CPU power. This is because they store the cache in their local environments.

6. When the three caching schemes in Oracle Coherence were compared in terms of memory usage, the distributed scheme was the most economical of all. The near cache had relatively less memory consumption than the replicated scheme. This is because the servers in the near and replicated schemes store the cache in their local environments.

In the light of these findings, Oracle Coherence is superior over the database and the other in-memory data grid, Hazelcast, in terms of the response time, CPU and memory usages. Considering the response time factor, it is advisable to prefer the replicated cache in Oracle Coherence, but the distributed scheme can be the best option for low CPU and memory usage. However, the distributed scheme was especially used in this study because it saved the system from extra work load and additional capacity by separating the front-end servers from the Coherence servers.

REFERENCES

Books

- Josyula, V., Orr, M. & Page, G., 2012. *Cloud computing: Automating the virtualized data center*. Indianapolis: Cisco Press.
- Laszewski, T., & Nauduri, P., 2012. *Migrating to the cloud: Oracle client/server modernization*. Waltham: Syngress.
- Lillard, T. V., Garrison, C. P., Schiller, C. A., & Steele, J., 2010. The future of cloud computing. *Digital forensics for network, internet, and cloud computing*. Boston: Syngress, pp. 319-339.
- Ruzzi, J., 2011. *Oracle coherence getting started guide, release 3.7*. CA: Oracle USA Inc.
- Seović, A., Falco, M. & Peralta, P., 2010. *Oracle coherence 3.5: Create internet-scale applications using Oracle's high-performance data grid*. Birmingham: Packt Publishing Ltd.

Periodic Publications

- Currier, G., 2009. On the rebound. *CIO Insight*. (110), pp. 28-36.
- Davies, A., 2012. Using the cloud. *Broadcast Engineering*. **54** (5), pp. 10-15.
- Dean, K., 2011. Data centers: where clouds connect. *Mission Critical*. **4** (3), pp. 66-68.
- Harshbarger, J. A., 2011. Cloud computing providers and data security law: building trust with United States companies. *Journal of Technology Law & Policy*. **16** (2), pp. 229-255.
- Kuyucu, A. D. H., 2011. The playground of cloud computing in Turkey. *Procedia Computer Science*. **3**, pp. 459-463.
- Lamb, J., 2011. Technology forecast: plenty of clouds. *Employee Benefit Adviser*. **9** (12), pp. 66-67.
- Lucas, R., 2012. Remote possibilities. *Professional Engineering*. **25** (11), pp. 51-52.
- Mansfield-Devine, S., 2008. Danger in the clouds. *Network Security*. **12**, pp. 9-11.
- Myatt, B., 2010. Cloud computing is here to stay until something better comes along. *Mission Critical*. **3** (2), pp. 18-22.
- Narayanan, C. V., 2010. Testing, the 'cloud' testing, the way. *Siliconindia*. **13** (7), pp. 36-38.
- Preimesberger, C., 2011. Why it's time to embrace cloud now: IDC. *Channel Insider*, 16 Mar. p. 1.
- Rosso, A., 2010. In the clouds. *Collector*. **76** (5), pp. 26-28.
- Seymour, R., 2013. Storage solutions. *The Middle East*. (441), p. 42.
- Sultan, N., 2010. Cloud computing for education: A new dawn?. *International Journal of Information Management*. **30** (2), pp. 109-116.
- Willcocks, L., Venters, W., & Whitley, E. A., 2011. Clear view of the cloud: The business impact of cloud computing. *Accenture Outlook Point of View*. (1), pp. 1-2.

Other Publications

- Apache, User Manual, 2013. Apache JMeter Overview [online], <http://jmeter.apache.org/> [accessed 2 February 2013].
- Appirio, White Paper, 2009. Professional services cloud: What cloud computing means for the services industry [online], CA, Appirio Inc., http://thecloud.appirio.com/rs/appirio/images/PScloudWP_Appirio.pdf [accessed 4 February 2013].
- Chan, S. P., 2009. Microsoft cloud computing gets down to earth [online], Seattle Times, http://seattletimes.com/html/microsoft/2009458942_microsoftazure13.html [accessed 27 September 2012].
- Garg, A., 2011. Cloud computing for the financial services industry [online], US, Sapient Global Markets, http://www.sapient.com/content/dam/sapient/sapientglobalmarkets/pdf/thought-leadership/GM_Cloud_Computing.pdf [accessed 19 November 2012].
- Gartner Inc., Press Release, 2010. Gartner identifies the top 10 Strategic Technologies for 2011 [online], <http://www.gartner.com/newsroom/id/1454221> [accessed 20 January 2013].
- Gigaspaces, White Paper, 2012. Bridging the cloud with in-memory data grid [online], Gigaspaces & Razorfish, <http://www.gigaspace.com/WhitePapers> [accessed 18 November 2012].
- Henderson, S., Lin, S., Solomon, M. & Hines, C., 2010. The wisdom of the cloud: cloud computing in the life sciences industry. *Executive Report*. New York: IBM Global Business Services.
- Hill, S., 2011. Clarity in the cloud: A global study of the business adoption of cloud. *KPMG International and Forbes Insight*. UK: KPMG International Cooperative.
- Hyek, P., 2011. Cloud computing issues and impacts. *Global Technology Industry Discussion Series*. UK: EYGM Limited.
- Mell, P. & Grance, T., 2011. The NIST definition of cloud computing. *Reports on Computer Systems Technology*. Gaithersburg: National Institute of Standards and Technology.

- Oracle, Developer's Guide, 2011. Oracle coherence 3.7 [online], Oracle Corporation, http://docs.oracle.com/cd/E18686_01/coh.37/e18677.pdf [accessed 5 August 2012].
- Oracle, Data Sheet, 2012a. Oracle coherence 3.7.1 [online], Oracle Corporation, <http://www.oracle.com/us/products/middleware/cloud-app-foundation/in-memory-data-grid/026050.pdf> [accessed 4 January 2013].
- Oracle, Installation Guide, 2012b. Oracle database 11g release 2 (11.2) for Linux [online], Oracle Corporation, http://docs.oracle.com/cd/E11882_01/install.112/e24321.pdf [accessed 10 January 2013].
- Oracle, Installation Guide, 2013. Oracle weblogic server [online], Oracle Corporation, http://docs.oracle.com/cd/E28280_01/doc.1111/e14142.pdf [accessed 4 January 2013].
- Sage Software Inc., Media, 2012, <http://na.sage.com/~media/CCA9F66BB3D34720B334565F840AD3E5.pdf> [accessed 15 March 2013].
- Tangosol Inc., Infosheet, 2007, Coherence Training [online], <http://www.art2dec.com/coherence/worksh/Coherence-3.2.x-Training-2.3-A4-2UP.pdf> [accessed 10 November 2012].
- Yapıcı, C., 2010. Bulut bilişim dosyası [online], http://www.tubisad.org.tr/Tr/Library/Analizler/bulut_bilisim_dosyasi.pdf [accessed 12 December 2012].

CURRICULUM VITAE

Full Name : Yıldırım Murat Şimşek

Address : Turkcell Maltepe Plaza, Yeni Mh. Pamukkale Sk. No:3 34880 Soğanlık mevkii /Kartal-İSTANBUL

Birth Place / Year : Mersin, 1985

Language(s) : İngilizce

Primary School : Salim Güven Primary School, 1996

High School : Yusuf Kalkavan Anatolian High School, 2003

University : Kadir Has University, Computer Engineering, 2009

MSc : Bahçeşehir University, 2013

Graduate School : Institute of Science

Name of Program : Computer Engineering

Work Experience : Turkcell 2008 -....