

การประมาณค่าสมรรถนะและค่าใช้จ่ายสำหรับเว็บแอปพลิเคชัน
บนการประมวลผลแบบกลุ่มเมฆที่ยืดหยุ่นได้ของแอมะซอน

นายฐิติณัฐ ตรีนรเศรษฐ์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคอมพิวเตอร์และเทคโนโลยีสารสนเทศ
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

บทคัดย่อและแฟ้มข้อมูลฉบับเต็มของวิทยานิพนธ์ตั้งแต่ปีการศึกษา 2554 ที่ให้บริการในคลังปัญญาจุฬาฯ (CUIR)
เป็นแฟ้มข้อมูลของนิสิตเจ้าของวิทยานิพนธ์ที่ส่งผ่านทางบัณฑิตวิทยาลัย

The abstract and full text of theses from the academic year 2011 in Chulalongkorn University Intellectual Repository(CUIR)
are the thesis authors' files submitted through the Graduate School.

Performance and Cost Estimations for Web Applications on
Amazon Elastic Compute Cloud

Mr. Thitinut Treenorrseth

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in
Computer Science and Information Technology
Department of Mathematics and Computer Science
Faculty of Science
Chulalongkorn University
Academic Year 2011
Copyright of Chulalongkorn University

ฐิติณัฐ ตรีนรเศรษฐ์ : การประมาณค่าสมรรถนะและค่าใช้จ่ายสำหรับเว็บแอปพลิเคชันบนการประมวลผลแบบกลุ่มเมฆที่ยืดหยุ่นได้ของแอมะซอน. (Performance and Cost Estimations for Web Applications on Amazon Elastic Compute Cloud)
 อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร.ชัชวิทย์ อภรณ์เทวีญ, 65 หน้า.

ในปัจจุบันระบบเว็บแอปพลิเคชันได้เข้ามามีความสำคัญอย่างยิ่งในการสนับสนุนการทำธุรกรรมต่างๆผ่านทางระบบออนไลน์ เช่น อีวิลิสเนส, การกระจายข่าวสารรวมถึงเป็นช่องทางการสื่อสาร เนื่องด้วยควมมีประโยชน์ของเว็บแอปพลิเคชัน ทำให้มีจำนวนผู้เข้าใช้บริการบนระบบเว็บแอปพลิเคชันต่างๆเพิ่มมากขึ้นซึ่งในบางครั้งเกินความสามารถที่ระบบจะรับไหวทำให้เกิดปัญหา “ระบบล่ม” ตามมา วิธีแก้ไขมาตรฐานในปัจจุบันคือ การเพิ่มจำนวนของเซิร์ฟเวอร์ หรือ การเพิ่มประสิทธิภาพของเซิร์ฟเวอร์ (เพิ่มหน่วยความจำชั่วคราว, หน่วยประมวลผล หรือ หน่วยความจำถาวร) แต่การแก้ปัญหาดังกล่าวทำให้เกิดความสิ้นเปลืองเนื่องจาก อัตราการเกิดปัญหา “ระบบล่ม” นั้นอยู่ที่ ๒๕ เปอร์เซ็นต์หรือน้อยกว่า ดังปัญหาที่ได้กล่าวมา เราจึงได้นำเสนอวิธีการแก้ปัญหาวิธีใหม่ซึ่งนำบริการของแอมะซอนชื่อว่า แอมะซอนอีลาสติก คลาวด์ คอมพิวต์ หรือ แอมะซอน อีซีทู มาเป็นเครื่องมือการแก้ปัญหา โดยอาศัยลักษณะพิเศษที่เรียกว่า อดได้ สเกลลิง ที่สามารถเพิ่มและลดจำนวนเซิร์ฟเวอร์บนระบบได้โดยอ้างอิงจากข้อกำหนดต่าง ซึ่งในกรณีนี้เราได้กำหนดให้เป็นจำนวนผู้เข้าใช้ระบบ ณ เวลานั้น และหลังจากผลการทดลอง วิธีของเราได้แสดงให้เห็นถึงผลลัพธ์ที่ดีเยี่ยม ยิ่งไปกว่านั้นเรายังได้ศึกษาเพิ่มเติมถึงการปรับเปลี่ยนค่าตัวแปรต่างบนระบบ ซึ่งจากผลลัพธ์แสดงให้เห็นว่าการปรับเปลี่ยนค่า “เวลาครอนแทป” นั้นมีประสิทธิภาพและสามารถทำได้ง่ายที่สุดในการปฏิบัติจริง อย่างไรก็ตามจากผลการทดลองแล้ว วิธีของเรานั้นเหมาะสมกับการลักษณะของผู้เข้าใช้ระบบที่มีการกระจายตัวแบบเอกซ์โพเนนเชียลเท่านั้น

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ลายมือชื่อนิสิต

สาขาวิชาวิทยาการคอมพิวเตอร์ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก
และเทคโนโลยีสารสนเทศ

ปีการศึกษา2554

5273622023 : MAJOR COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
 KEYWORDS : COST ESTIMATION / WEB APPLICATIONS / AMAZON ELASTIC
 COMPUTE CLOUD

THITINUT TREENORRASETH : PERFORMANCE AND COST ESTIMATIONS
 FOR WEB APPLICATIONS ON AMAZON ELASTIC COMPUTE CLOUD.

ADVISOR : ASST. PROF. CHATCHAWIT APORNTewan, Ph.D.,65 pp.

Web base application plays an important role in supporting online activities, such as E-business, contents distribution and a communication portal. There is a big increasing in number of online activity usage that leads to an important problem to a webserver which is a “traffic overload problem.” At present, the conventional solutions are whether to increase the number of webserver or to increase the capability of the server (more RAM, CPU, Cache, HDD). Unfortunately, these solutions bring about waste resulting from overloading problem that accounts for less than 25 percents of the operation time. In addition, hardware capacity has not been used to it maximum over than 70 percent of the time, but left idle instead. This study proposes a new method to solve this problem on Amazon Elastic Cloud Compute (EC2) service. The proposed solution relies on Auto Scaling where the number of system instances (web server) can be increased and decreased based on some pre-defined condition. In this case, the number of current system users is the main focus. An experiment showed that this method provided great performance. It was also found that focusing on tuning of system crontab variable was the most effective method and easiest for real practice. However, the proposed solution is only suitable for the system which predicted user distribution is in exponential form.

Department : Mathematics and Computer Science Student's Signature

Field of Study: Computer Science and Advisor's Signature

Information Technology

Academic Year : 2011

Acknowledgements

First, I would like to thank to my advisor, Assistant Professor Chatchawit Apornthewan, who always guide me the way, give me comments, ideas and advisement. Without him I would never finish this thesis. And prior to his personality and interest, I also consider him as a friend for now and future. Moreover, I would like to thank to Associate Professor Peraphon Sophatsathit who always provide me many academic knowledge and tips since I was entering at Chulalongkorn, in my opinion, he is a role model of good instructor. Also I would like to thank to my friends at Chulalongkorn who always provide me some information and remind me of an academic schedule, especially Guile, Sand, Niel and Nuch without them, my time might be harder than this. And above of all, thank to my family member: my parents, without them I definitely not be here, my brother who always be with me even we both are not a good speaker, my dogs that always guard my home when I not be there and also my fiancée who always with me and encouraged me.

Moreover, during my research and study period at Chulalongkorn, there were many events occurred in my life, some was good some was bad. For all the good things, I thank you all related that make all those good things happen in my life. But for all those bad things, even back in that time they were causing me pain and weary but after all, they just can only make me stronger and more experienced. Therefore graduated my Master Degree here at Chulalongkorn not just only the academic degree but also means a “life” degree to me.

Contents

	Page
Abstract (Thai).....	iv
Abstract (English).....	v
Acknowledgements.....	vi
Contents.....	vii
List of Tables.....	x
List of Figures.....	xi
Chapter	
I Introduction.....	1
1.1 Objectives.....	3
1.2 Scope of the Work.....	3
1.3 Problem Formulation.....	3
1.4 Expected Outcomes.....	4
II Theoretical Background.....	5
2.1 Cloud Computing.....	7
2.2 Amazon Cloud Service.....	11
2.3 Amazon EC2.....	17
2.4 Auto Scaling.....	21
2.5 Elastic Load Balancing.....	22
2.6 Cloudwatch.....	23
2.7 AWS SDK.....	23

Chapter	Page
2.8 Queuing theory.....	24
III Modeling and Tools.....	26
3.1 Launching EC2 instance and perform the configuration.....	28
3.1.1 Launching EC2 instance.....	28
3.1.2 Connect to the instance.....	32
3.1.3 Configure Putty.....	33
3.1.4 Configure the EC2 instance.....	34
3.2 Launch RDS and configuration.....	36
3.2.1 Launch RDS instance.....	36
3.2.2 Create Database.....	39
3.3 Enable and configuration an Auto Scaling.....	39
3.3.1 Enable Auto Scaling.....	40
3.3.2 Configure an Auto Scaling.....	42
a) Create launch configuration.....	42
b) Create Auto Scaling group.....	43
c) Create Auto Scaling policy.....	43
3.4 Create PHP script on an EC2 instance.....	45
3.4.1 Authentication.....	45
3.4.2 RDS connection.....	46
3.4.3 Auto Scaling trigger.....	46
3.5 System working process.....	47

Chapter	Page
IV Experimental Results.....	50
4.1 Functional testing.....	50
4.2 Cost compilation.....	52
V Discussion.....	57
5.1 Solution.....	57
5.2 Quality of service.....	58
5.3 The optimal parameters.....	62
VI Conclusion.....	63
References.....	64
Biography.....	65

List of Tables

Table		Page
1	Pricing table for EC2 instances.....	21
2	Database diagram for application.....	39
3	The cost estimation for conventional method.....	54
4	Conventional method compare with our method.....	55
5	Result on round time tuning for “Best” user distribution.....	59
6	Result on round time tuning for “Bad” user distribution.....	59
7	Result on round time tuning for “Worse” user distribution.....	60
8	Result on round time tuning for “Worst” user distribution.....	60

List of Figures

Figure		Page
1	Workload problem occur at T4 and T5.....	5
2	Solving problem by add more hardware.....	6
3	Solving problem by increase hardware capability.....	6
4	Queuing model components.....	24
5	System model.....	26
6	The EC2 user interface for launch instance.....	28
7	Select the OS for the new instance.....	28
8	Specify instance detail.....	29
9	Select the key pair using to connect to the created instance.....	30
10	Configure instance firewall policy.....	31
11	The detail summary of an instance.....	31
12	New launched instance is ready.....	32
13	Foreign key import successful.....	33
14	Security key using for authentication.....	33
15	Interface of Putty.....	33
16	Connect to instance using Putty successful.....	34
17	Start httpd service on instance.....	35
18	The interface for create new RDS instance.....	36
19	Select the database engine for RDS.....	36
20	Specify RDS detail.....	37

21	Specify some additional configuration.....	38
22	Summary page before launching an instance.....	38
23	Security credential information.....	41
24	Listing all the Auto Scaling command after finish an installation.....	42
25	Command for create Auto Scaling policy.....	43
26	Listing all of the scaling policy of the Auto Scaling group.....	45
27	System working process.....	47
28	One instance to serve 4 users.....	48
29	Two instance to serve 8 users.....	48
30	Four instances to serve 14 users.....	49
31	Two instances to serve 7 users.....	49
32	User distribution over 1 hour.....	51
33	Demand and supply of EC2 instance over time.....	51
34	Four kinds of user distribution.....	54
35	Crontab = 5 mins, Auto Scaling 0 – 10 mins.....	55
36	Crontab = 1- 10 mins, Auto Scaling = 0 mins.....	56
37	Overload time value with respect to crontab round time tuning.....	61

CHAPTER I

Introduction

Technology is one of the most important key for business to overcome the other business competitors. In business, rather than costing, marketing, management policies, technology is the most dominated factor to decide whether who is going to be a winner or a loser. The one that owns better technology or can come up with the most innovative product in that time is likely to be the one who will rise in the market. As mentioned, the better technology brings up many competitive advantages, such as better product quality and functions, lower the product or manufacturing cost. As the importance of technology becomes higher level of the competition, the faster of growth in technology goes. This fact can explain the reason why the growth of technology at present is very fast. This also brings up what is called “technology competition” in business and many other areas. In the field of information technology, the most focusing factor in technology innovation is performance and cost reduction. In order to come up beyond all the competitors, one of the above factors must be achieved.

In this thesis, the new idea on doing an IT-related business is introduced; moreover, I believe that this technology will emerge a big impact on IT business and other related business by this coming year. This mentioned technology is “Cloud computing”, Cloud computing is the technology which base on the concept of centralization and share resourcing. For nowadays, cloud computing is normally understood and is used as service to support many kind of operations and objectives varied by end-users. With high level of comfort, and rapid time of implementation and set up, cloud computing technology is becoming popular in IT area and the expected growth rate for worldwide cloud computing in 2013 is 300 percents. With regarding to the big coming trend of cloud technology, many enterprises are starting to pay their attention on this technology and plan to shift their IT toward the cloud, in order to create cloud-based IT infrastructure environment what is called “Private cloud.” But for end-

users and small companies (non-enterprise), pay for the “public cloud” service is another interested option.

At present there are many reliable cloud service providers likes, Amazon, Google, Microsoft, and Symantec. The service provided is mainly computing resource and storage, which are different for each provider in details. And in this thesis I desire to use the cloud service from Amazon due to the reason that there are many varieties of services and free of charge for some services and the most important is the features provided by Amazon.

Amazon Elastic Compute Cloud (EC2) is one of the products in Amazon Web Service (AWS), which is provided by Amazon. This service based on the concept of “Cloud computing”; in short, I buy the computing resource in form of services not a product. As a service on cloud, the wall of insufficient computing resource is totally broken down because the service provided on cloud can be nearly called as unlimited. Amazon also introduces the service highlight with definition “Pay as you use”, with this notable characteristic, I decide to implement this technology as the solution for the web service project to reduce cost and increase the performance.

The current approach for hosting a web service project on the internet is to have a physical web server which can be either permanently purchased or temporarily rented. However, with any of the proposed solutions, the quantity and the performance of the server have to be large enough to handle the large amount of incoming clients in the peak time. In order to prepare the computing resource to support this kind of scenario, the current conventional approach causes waste of budget and physical resource.

In this thesis, I propose an approach for increasing the performance of web service while also reducing the cost by using Amazon Elastic Compute Cloud (EC2) which is capable of expanding and shrinking the number of computing instances in order to meet an application demand [1,2].

2.1 Objectives

- To investigate Amazon EC2, instance types, system setup and configuration for web applications.
- To estimate the cost of running a web service, given a distribution of user connections over a period of time.
- To reduce the cost of running a web service project.

2.2 Scope of the Work

- There are many cloud-computing providers in market, but on this research I focus only on Amazon Web Services.
- I only interest in a web application with a known distribution of user connections.
- Regarding to the expense, I conduct the experiments only with Micro instance (Linux OS, PHP and MySQL).
- Most of the results come from simulation (not the actual run) due to the budget limitation. For instance, if I perform a real testing for 500,000 user connections over two weeks, this will cost thousands of dollars.

2.3 Problem formulation

The research problem is formulated as “parameter optimization.” There are a number of parameters for using Amazon Compute Cloud. I develop a simulator that takes these parameters and then calculates the cost. The simulation approach is fast and not expensive for trial and error. After a thorough study in the simulator, I will actually test a few cases on Amazon Compute Cloud.

2.4 Expected Outcomes

- A practical method for performance evaluation of an EC2 instance.
- A cost estimation model.

CHAPTER II

Theoretical Background

In this research, we propose a new solution by implementation of Amazon Cloud Service for solving the web service problem called “Traffic Workload problem”, in this thesis; we will call this “Workload problem”. For more description, Workload problem describe the situation when there are too many user connecting to the webserver, so that it cause the webserver to be slow down and unresponsive. For current, a conventional approach is to make sure that the allocated hardware is sufficient for provide service at the peak of users which can be either increase the number of the hardware (Figure 2) or increase the hardware capability (CPU, RAM, Cache, HDD) (Figure 3). But in a real practice, the mentioned solutions bring a lot of waste. As mentioned, our objective is not only to solve the problem bit also consider with cost reduction.











T1	T2	T3	T4	T5
				
 OK	 OK	 OK	 OK	 <u>Down</u>

Figure 1. Workload problem occur at T4 and T5.











T1	T2	T3	T4	T5
				
 OK	 OK	 OK	 OK	 OK

Figure 2. Solving problem by add more hardware.











T1	T2	T3	T4	T5
				
 OK	 OK	 OK	 OK	 OK

Figure 3. Solving problem by increase hardware capability.

First is to improve the performance, refer to the web service problem when the web server itself crashing down due to a large number of incoming client coming to the server in the same time [3]. Second, reduce the cost, in order to avoid the mentioned problem, with current solution, more hardware is needed. But this will cause a lot of waste if we consider on the actual usage and depreciation cost. Moreover, we also represent the cost estimation model for dynamic instance usage, since there is still few of the research which focusing on this aspect [5,6,7,8,9,10]. Therefore, in order to be able to understand the concept of system architecture that we are going to introduce

for solving the problem in the next chapter, knowing of the basic information about some of Amazon Cloud Service is needed.

2.1 Cloud Computing

The word “Cloud” originally comes from the diagram that represents the telephone network in telecommunication field, and later was used in computer network diagram to represent an abstraction of the infrastructure. Originally, Cloud computing is term which describe the service that is delivered to the user over the internet connection. The service in this case can be computation resource, software, data access and storage service which the user does not have to know physical location of the service or understanding of any component device or an infrastructure that is required to provide the service. For example, when we go to Google and do the searching, we do not know the source code or an algorithm behind Google, but we just input the text and receive the result. As from the given example, web service is also one of the Cloud computing services.

Normally, Cloud architecture consists of a large amount of server (may up to ten or hundred thousand of physical server), sometime is called ‘Server farm ‘. These cloud servers are connected together as grid architecture. And communicate with customer through ‘User interaction interface’ which relies on a web protocol. Customer can search for desired service on the web interface called ‘Service catalog’ which contains the list and briefly description of the available service. And when customer ask for the desired service, the requisition will be send to ‘system management’ which play role as a service manager to manage the queuing and resource management to select the appropriate resource among all available at that time. Then the service request will be forward to ‘provisioning services’ which will make a resource reservation on cloud and activate the usage to the customer. After that all the activity and service monitoring will be handled by ‘Monitoring and metering system’

which will monitor the resource usage, condition/health and also collect all the statistic and billing procedure.

For present, the term Cloud computing has been added in more details, for better understanding, Cloud computing can be separate by models which are Private cloud and Public cloud. Private cloud is infrastructure which provides the service for a single organization, in this case propose of the cloud is to consolidate all the IT service into one place in order to retain centralize IT resource management and comfortable such as software maintenance, debug or patch update, usage monitoring and system backup. Another proposes is to save the cost and gain maximum profit, for example, instead of purchase 10 cabling design software license for each department workstation, company can purchase only 5 licenses or less and put them on the cloud and the user just use their thin-client computer to connect to the company cloud when they want to use the software. But with private cloud, company still need to have IT operation team to take care and manage the company cloud.

Another model is Public cloud, which describe the cloud service that is provided by the cloud service provider company such as Amazon, Yahoo Google etc. The Cloud service in this model is available to almost everyone in the world that can connect to the internet. The service will be used by the connected user and the fees are charged according to the actual usage time; mostly the fees will be calculated per hour. As the cloud concept, the user is able to use the computing service without having to know the physical location of the cloud server, or even the components inside.

The cloud can be also divided into following layer.

- 1) Application or software as a service (Saas) – the user pay for the software service that is deliver to user over the internet as request without needing of software installation computation resource on user computer. With this model, the software and its associated data are

hosted in the cloud mostly through the web browser, for current, this kind of model can be realize as thin-client.

2) Platform as a service (PaaS) – similar idea to SaaS, the complete ready to use platform for user. This service is usually for software development team, such that, the development is ready to use, so there is no need for further setting up of underlying hardware or software installation in order to meet the development environment pre-condition. This provide speedy and comfortable to the developer team since after pay for the service, they can start on writing their application on the platform instantly.

3) Infrastructure as a service (IaaS) – all the needed things which is required in order to establish the working infrastructure is provided; for example, the data storage and networking. But typically, the infrastructure is based on the virtualization instead of purchase those server, software, rack space and network equipment.

Moreover, some new model of cloud computing service were introduced likes, Community cloud and Hybrid cloud which refer to the composition of two or more clouds (private or public), which will allows programs and data to be moved easily from one system to another.

Regarding to benefit of the cloud, it also provide some characteristic with many other technical terms likes; Client-server – the architecture form a model which separate between service provider(server) and the service requester(clients) and Grid computing – the physical hardware components are setup to operate in distributed and in parallel form, where a virtual computer is composed of a cluster of network.

By the way, within this thesis, we will focus only on the implementation of public cloud to solve the problem. Moreover the Public cloud service also provides many benefits such as

Agility – an ability to change or re-provision the infrastructure resource as desired.

Cost efficiency – lower start up cost and also break down the entry barrier for the entrepreneur, as the infrastructure does not require a one-time purchase, in contrast it is defined by usage-based as pay as the service you had use.

Location independence – the same concept of web service which provide you the ability to access system over the internet with regardless of the location and device.

Reliability – with using the cloud, the technique which multiple redundant sites are used, this bring a well-designed system and suitable for business process continuity and disaster recovery.

Scalability – with using cloud service, the resource is based on virtualization; therefore, the sizing of the system is nearly unlimited and the responsive is near real-time. So that system can be scale up or down in order to serve the change in number of clients.

Maintenance – with using of cloud service, the maintenance is the part that you do not have to worry, since it is the responsibility of the service provider.

In additional details, the public cloud service is the massively shared computing resource which is provided by the third party company to the customer as service rather than a product and can support up to multiple concurrent end-customers at the time and the cost will be charged by amount of consumption (usually by hour). Therefore the customer will not know (is no need to know) what hardware and software are used as the combination to provide the service for them. The analogy is “If you need milk, would you buy a cow?”, this word is completely explain everything about the cloud

technology, in fact the customer is required to know only on the service which they are going to use on top, but they do not need to know what components are performing underneath. For example, when you want to access your hotmail and send an E-mail, what you need to do is only go to the Internet café, open the web browser, access your hotmail account, compose and send the E-mail then pay your bill. You do not need to know what is the web browser, the operating system, specification of the hardware or even the chair you are sitting while using the computer, you just pay attention to whatever you would like to do by leave the rest to be the business of service provider (Internet café). With this characteristic, you can fully focus on the core business activity rather than the IT environment setup, installation, configuration and maintenance.

As the expectation for the big growing rate of cloud computing technology, during the past ten years there are many cloud service providers step into this market; for current such as Microsoft with "Azure" cloud, Google with "Google App Engine", IBM with "Blue cloud", Verizon and Amazon with "AWS". This is because of the fact that they believe cloud technology will become the core of IT business and a perfect tool for business support and enterprise back-end system in the near future. By the way, for current, there is already on action commercial system that already implemented with cloud computing already like: Google mail, Yahoo mail, Facebook etc.

2.2 Amazon Cloud Service

Among all the cloud services provided on the market, in this thesis we desire to choose "Amazon AWS" to be the tool in our research. The reason we choose Amazon Web Services over other products is because in our personal idea, we believe that Amazon Web Services is more reliable than the others since it is ranked as the number 1 on the "top ten cloud computing service providers" in year 2010 and 2011. The rankings are based on customer traction, solid technical innovation and management track record. This means that this product should be able to survive on the market since they are the best in the area for now and as current status and our

prediction, Amazon seem to be a permanent leader in the market for at least 5 years, so our research can be brought to implement in the real business. And another reason is about the provided service and feature which we feel that Amazon Web Services features is the right tool for the problem.

Amazon Web Services (AWS) is one of the products that are provided by Amazon.com which is originally started their business as an E-commerce core-company likes an online bookstore, DVDs, CDs, MP3, computer software, game etc. Amazon Web Services is a collection of web services which can be together make up a cloud computing platform provided over the internet.

Amazon Web Service was first launched in 2002. In that time, Amazon Elastic Cloud (Amazon EC) was introduced and only compatible with Microsoft Window Server and Microsoft SQL server, but in later, it supports many various kind of platform and such as Linux and OpenSolaris. List of available important web services are shown as following.

- Amazon AWS Authentication - the authentication embedded service which provides authenticate access to the AWS services.
- Amazon CloudFront, a content delivery network (CDN) for distributing objects stored in S3 to so-called "edge locations" near the requester.
- Amazon CloudWatch - provides monitoring capability on all AWS cloud resources and applications including Amazon EC2.
- Amazon DevPay - is a billing and account management system for applications that developer has builded on AWS.
- Amazon Elastic Beanstalk – is a tool for application deploy and manangement
- Amazon Elastic Block Store(EBS) – is a persistent memory block storage for EC2 service.
- Amazon Elastic Compute Cloud(EC2) – is a private server service which allow fully management power to user,and also come with scalable feature.

- Amazon Elastic MapReduce – is a tool which mainly target on researcher data analyst and developer to perform big amount of data processing for cheaper price. With Hadoop framework, it also support Parallel processing application.
- Amazon Elastic Cache – internal caching feature for web server instance.
- Amazon Product Advertising API – In former, it is known as Amazon E-Commerce Service which used for supporting of the electronic commerce.
- Amazon Relational Database Service(RDS) – Database server component.
- Amazon Route53 – Web service domain name management
- Amazon Simple Email Service – E-mail transaction management.
- Amazon Simple Storage(S3) – provides application storage service
- Amazon Simple Queue Service(SQS) – web application message queue management.
- Amazon Simple Notification Service(SNS) – provides web application multiprotocol messaging.
- Amazon SimpleDB – allow database queries function on EC2 and S3.
- Amazon Virtual Private Clour(VPS) – allow secure communication of Amazon Cloud Service with already exist infrastructure.
- AWS Import / Export – provides portable storage device to manage the data transfer into/out of AWS.
- AWS Management Console – Graphic User interface for manage and monitor all the service of AWS.
- AWS Simple Monthly Calculator – support monthly cost estimation.

Naturally, Amazon Web Services is the online service for other web sites or client-side applications; therefore, most of these services are not exposed directly to the end user, but instead, it is tend to offer functionality which is more useful to the developer site. Likewise, in our solution, the services that we gather to making up our system are Amazon CloudWatch, Amazon Elastic Compute Cloud (EC2), Amazon Relational Database Service (RDS) which we can manage and monitor the infrastructure

via AWS Management Console. In the following, we will describe more in depth details about all those services.

Amazon Elastic Compute Cloud (Amazon EC2) is a cloud based web service which provides developer the scalability of computing resource in the cloud. Moreover Amazon EC2 also come with the web interface which allow developer to configure, monitor and complete control over all the resources. Amazon EC2 also reduce the server instances boot time and terminate time to minutes, likewise, this allow the system to quickly scale-up or scale-down with respect to the change. And as a cloud service, Amazon EC2 fee is charged for the actual capacity usage.

Service Highlights

Elastic – Amazon EC2 enables the developer to whether increase or decrease the system capacity of the system not by hours or days but within minutes. Also with using the Web interface control, the developer can control and monitor even hundreds or thousands of server instances simultaneously.

Complete Controlled – Within Amazon EC2 service, the developer have complete control over instances as it located right next to them. With the root access permission on each one, the developer can stop/start or even restart instance while retaining the data on the boot partition. And all those activity can be done through the web control interface.

Flexible – Amazon EC2 come up with many type of server instances, operating systems and also various software packages. Moreover, the developer can select the allocation of instance memory, CPU, instance storage. This option allows the developer to work on any type of desired environments, and also support the development on any platform.

Compatibility – All of Amazon EC2 instance can work in conjunction with other cloud service of Amazon likes Amazon Relational Database Service (RDS), Amazon Simple Storage Service (S3) Amazon SimpleDB etc.

Reliable – With Amazon EC2, replacement of instance can be rapidly performed. Moreover Amazon EC2 Service License Agreement commitment is 99.95% availability for each region.

Features

Amazon EC2 instance also come up with useful add-up features

Amazon Elastic Block Store (EBS) – offer persistent storage for Amazon EC2 instance. Amazon EBS volumes provide off-instance storage that persists independently from the life of an instance. This mean that when developer stop the instance when unused the state of the instance still maintain that state and when the instance is started again, it will be boot with the recently state. This allow developer to stop the instance when unused in order to minimize the cost. Moreover, Amazon EBS provides option to create point-in-time (snapshot) of the volumes and store in Amazon S3. So that the snapshot can be replicate across regions, and also can be shared with other developer and co-worker for using as the starting point for new Amazon EBS volumes.

Multiple Location – Amazon provide many available location for the user, therefore; user can place their system instances in multiple regions and locations around the world as needed in order to perform load balancing, failure protection or even location based service. For current, Amazon EC2 is available in North America (3 locations), South America (1 location), Asia(2 locations) and Europe (1 location).

Amazon Virtual Private Cloud – is a service which is provided for company, with this service, company existing IT infrastructure can be expanded and connect with AWS cloud instance securely through a Virtual Private Network(VPN).

Amazon Cloudwatch – is a service which allow user to monitor their AWS resource via the web interface. Many parameters are provided such as CPU utilization, Memory usage, HDD I/O traffic, network usage etc. Amazon Cloudwatch also provide statistic view and graphical view of those parameters, so the user can view historical data of those parameter. Moreover alarm trigger can be perform based on changing of parameter value, for example, alarm when CPU usage is more than 80 percents.

Auto Scaling – allow us to design the system with an ability to scale the system capacity up or down according to the pre-defined conditions. With Auto Scaling, the number of EC2 instances can be increase during highly demand to maintain the performance and decrease when the demand is dropped to minimize the cost.

Elastic Load Balancing – is a component which manages all the incoming traffic across EC2 instances in the system equally. Not only manage the traffic, load balancing also able to detect for unhealthy device in the system and reroute those instance traffic to other healthy instance. Moreover, Load Balancing can also manage the traffic across multiple instances that are located in multiple location.

VM Import – this service enable developer to import their virtual machine images from the existing environment to Amazon EC2 instances. So, developer can immediately migrate their server into Amazon cloud quickly and easily without having to start from the ground up.

High Performance Computing Clusters – is suitable for the job that requires more computation power and complexity than normal, for example, parallel programming job which require both computing power and network performance also, the need of cluster computing and cluster GPU. With this service, the same performance as custom-built infrastructure is served to the user with less cost but more flexibility.

2.3 Amazon EC2

Amazon EC2 provides several type of instance which may suitable to various kind of implementation. The followings are the available instance type.

Micro instance – suitable for low data computation and throughput application, and also suitable for web application that is not require much computation and data transfer.

1) Micro Instance

No. of CPU	2 ECU
Memory	613 MB
Local storage	(EBS storage only)
Platform	32,64 bits

Standard instance – suitable for standard application and general purpose, there are 3 type of standard instance.

1) Small Instance

No. of CPU	1 ECU
Memory	1.7 GB
Local storage	160 GB
Platform	32-bits

2) Large Instance

No. of CPU	2 ECU (2 virtual cores)
Memory	7.5 GB
Local storage	850 GB
Platform	64-bits

3) Extra Large Instance

No. of CPU	2 ECU (4 virtual cores)
------------	-------------------------

Memory	15 GB
Local storage	1690 GB
Platform	64-bits

High-Memory instance – suitable for application that require high throughput rate, for example database related application and memory caching application, there are 3 type of High-Memory instance.

1) Extra large Instance

No. of CPU	3.25 ECU x 2
Memory	17 GB
Local storage	420 GB
Platform	64-bits

2) Double Extra Large Instance

No. of CPU	3.25 ECU x 4
Memory	34 GB
Local storage	850 GB
Platform	64-bits

3) High-Memory Extra Large Instance

No. of CPU	2 ECU (4 virtual cores)
Memory	15 GB
Local storage	1690 GB
Platform	64-bits

High-CPU Instances – suitable for application that requires high computation power (intensive computing).

1) Cluster computer Quadruple Extra Large

No. of CPU	33.5 ECU
Memory	23 GB
Local storage	1690 GB
Platform	64-bits + 10 Gigabit Ethernet

2) Cluster Computer Eight Extra Large

No. of CPU	88 ECU
Memory	60.5 GB
Local storage	3370 GB
Platform	64-bits + 10 Gigabit Ethernet

Cluster GPU Instance – suitable for graphic processing task that require high parallelized processing which have to be on both computation power and network performance. The example is HPC rendering and media processing application.

1) Cluster GPU Quadruple Extra Large

No. of CPU	33.5 ECU + 2x NVIDIA Tesla M2050 GPU
Memory	22 GB
Local storage	1690 GB
Platform	64-bits + 10 Gigabit Ethernet

*** One EC2 Compute Unit (ECU) computing power is equivalent to CPU capacity of a 1.0-1.2 GHz 2007 Xeon processor.

Operating Systems and Software

Amazon EC2 is compatible with many kinds of operating system and software, so the developer can create the most suitable environment for their application. The list of available software is listed as below.

Operating Systems

- Linux Red Hat Enterprise
- Oracle Enterprise Linux
- Amazon Linux AMI
- Fedora
- Window Server
- SUSE Linux Enterprise
- Ubuntu Linux
- Debian Linux

Database Software

- IBM DB2
- Microsoft SQL Server Standard
- Oracle Database 11g
- IBM Informix Dynamic Server
- MySQL Enterprise

Resource Management Software

- StackIQ Rocks+
- Condor
- Hadoop

Web Hosting

- Apache HTTP
- IBM Lotus Web Content Management
- IIS/Asp.Net
- IBM WebSphere Portal Server

Application Development Environments

- IBM Smash
- Ruby on Rails
- JBoss Enterprise Application Platform

Application Servers

- IBM Websphere Application Server
- Oracle WebLogic Server
- Java Application Server

Video Encoding and Streaming

- Window Media Server
- Wowza Media Server Pro

Pricing

As a cloud service, the service charge for Amazon EC2 is calculated by actual usage hour and the minimum usage is not required. Moreover, for recently, Amazon has allow the instance reservation from 1 year up to 3 years, with the service reservation the developer can ensure that the instance will be always available when is needed (suitable for real business usage implementation). And the service prices on each Region are different , as mentioned, the price on the list will show for the Asia Pacific Regions only (Singapore).

		Linux/Unix	Windows
Micro Instances		\$0.02 per hour	\$0.03 per hour
Standard Instances	small	\$0.085 per hour	\$0.12 per hour
	large	\$0.34 per hour	\$0.48 per hour
	extra	\$0.68 per hour	\$0.96 per hour
High-Memory Instances	extra large	\$0.05 per hour	\$0.62 per hour
	double extra	\$1.00 per hour	\$1.24 per hour
	large	\$2.00 per hour	\$2.48 per hour
	quadruple		
extra large			
High-CPU Instances	medium	\$0.17 per hour	\$0.29 per hour
	extra large	\$0.68 per hour	\$1.16 per hour

Table1. Pricing table for EC2 instances.

2.4 Auto Scaling

Auto Scaling is the feature that allows the system to scale-up and scale-down its capacity according to the pre defined condition. With this feature the

developer can ensure that their system will not crash or corrupt by the increasing amount of incoming traffic with the cost minimization.

Features

- Automatically increase the number of instance in the system pool when demand is increase.
- Automatically terminate instance when the demand is decreased
- Scaling policy and trigger action are based on Amazon CloudWatch metrics and alarm.
- Auto Scaling feature can be used with all type of Amazon Elastic Compute Cloud service(EC2) instances.
- No additional charge for enabling of Auto Scaling feature

2.5 Elastic Load Balancing

Elastic Load Balancing is the service that will automatically distributes incoming traffic across EC2 instances. With this service, Application fault tolerance is greater. Moreover, Elastic Load Balancing also able to detects unhealthy instances and redirect the traffic to other instance instead. For the world-wide application that may located servers in many regions, Elastic Load Balancing also able to distribute the traffic across those servers.

Features

- Elastic Load Balancing manages all the incoming traffic of system instances which can be cover in one location or multiple location.
- Additional security and network option can be create when implement with Amazon Virtual Private Cloud (VPC)
- Elastic Load Balancing can detect the health of Amazon EC2 instances. When it detects unhealthy load-balanced Amazon EC2 instances, it no longer routes

traffic to those Amazon EC2 instances and spreads the load across the remaining healthy Amazon EC2 instances.

- When the unhealthy instance is detected in the system, Elastic Load Balancing will reroute the traffic to others healthy instances.
- Elastic Load Balancing also play role in managing of user session on each EC2 instances.
- Elastic Load Balancing supports both IPV4 and IPV6.
- Can also be monitoring by Cloudwatch metrics.

2.6 Cloudwatch

Amazon Cloudwatch provides customer ability to easily monitoring the entire running AWS instance for both real time and statistic monitoring. This feature allow system administrator can keep track of their system instance by monitor each instance standard metrics variable such as instance memory usage, system overall memory usage, CPU utilization, data transaction volume. Moreover the user can create custom metrics for specific purpose monitoring such as number of client connected to the system etc. With Cloudwatch the all the activity can be track and monitor for abnormal event and keep the system in healthy. The example for the

As mentioned, result retrieve from Cloudwatch is not only in the form of raw metric number but also can be represent in form of statistical and graph in order to track back or forecasting. Moreover system administrator can create an alarm to help in monitoring, such as an alarm is on when the CPU usage is more than 80 percents.

2.7 AWS SDK

AWS SDK for provide the language library (.NET,PHP) which allow developer to build up PHP-based application that can work on Amazon Web service platform, such as, show information of the system instance, create or terminate instance, create/update/delete the database and many more. With the current provided library,

the developer can create the application which can interact with almost every type of Amazon Web Service instance.

2.8 Queuing Theory

Queuing theory or Waiting line is mainly concentrate in simulating the customer queuing in order to clarify the total cost of queuing in term of business management. The total cost can be divided into 2 categories.

- Service cost – is the cost which related in establishing of service, support such as staff salary, equipment cost.
- Cost of waiting – is the cost which related to the loss the cause by customer waiting such as dissatisfaction.

To be able to calculate the cost, some parameters must be known, such as amount of people in queue, time spend on waiting.

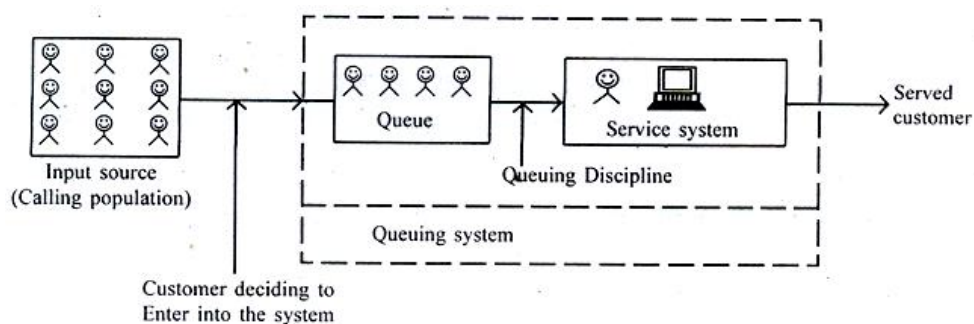


Figure 4. Queuing model components.

As describe in Figure x, there are five components,

- 1) Calling population – the total possible customer which can be divided into 2 cases
 - Finite – in this case, the size of the population is not too big, so that the number of people currently in queue is affect to the size of the queue in the future.
 - Infinite – the size of the population is big enough, so that the number of

people currently in queue is not affect to the size of the queue in the future.

2) Customer arrival – incoming customer can be divided into 2 categories

- Arrival rate – number of incoming customer in one specific time period.
- Interarrival time – time spend on each customer (waiting time for next customer).

3) Waiting line – number of people currently in queue.

4) Processing order – the queue management technique, usually will be First-in-First-out (FIFO), but sometimes can be FIFO with priority queue.

5) Service – property of the service,

- Number of channel -> single channel, multi channels
- Number of step -> single step, multiple steps

CHAPTER III

Modeling and Tools

Modeling and tools

In this chapter we will describe in more details about the solution model and all of the procedure required in setup and configuration the system. As shown below is the system model which we introduce for solving the problem.

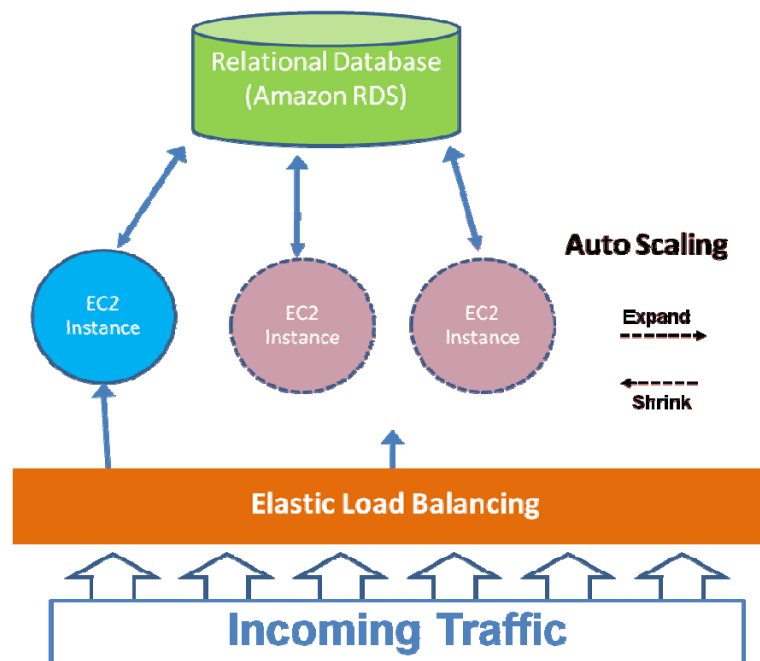


Figure 5. System model.

Mainly, the system combines of Amazon RDS which acts as system database, an EC2 instance as a main web server of the system plus with the Auto Scaling function which will control the number of instances in the group, which designs whether to add more web server instance (a copy of the predefined instance image) into the system to share the incoming traffic or to remove some instances from the group for saving the cost. The Load balancing will distribute and manage the incoming traffic from the client side equally among the instances on the system at that time.

In able to use the cloud service from Amazon, first we have to sign up for an EC2 user account, and then the system setup procedures are preceded as following steps

- 1) Launch EC2 instance and perform the configuration
- 2) Launch RDS and create the database
- 3) Launch an Auto scaling and perform the configuration.
- 4) Embedded PHP script in EC2 instance

3.1 Launching EC2 instance and perform the configuration

3.1.1 Launching EC2 instance

a) Log in to the account > EC2 > and click launch Instance

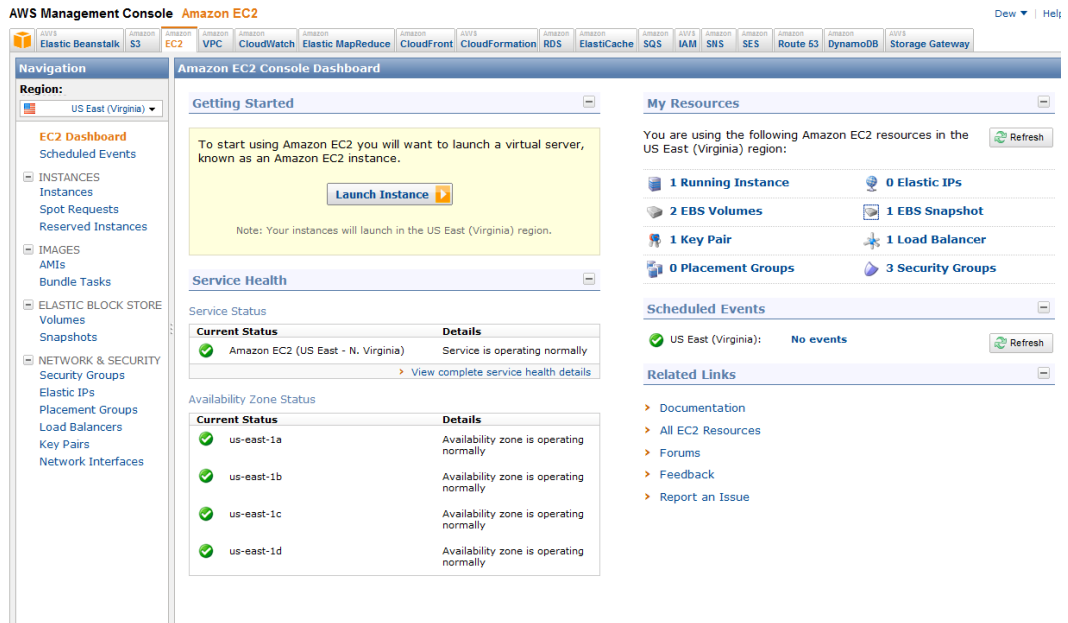


Figure 6. The EC2 user interface for launch instance.

b) Once the launch instance wizard is appear, the first page will let us to select the desired Operating system. In this system we choose “Basic 32-bits Amazon Linux”

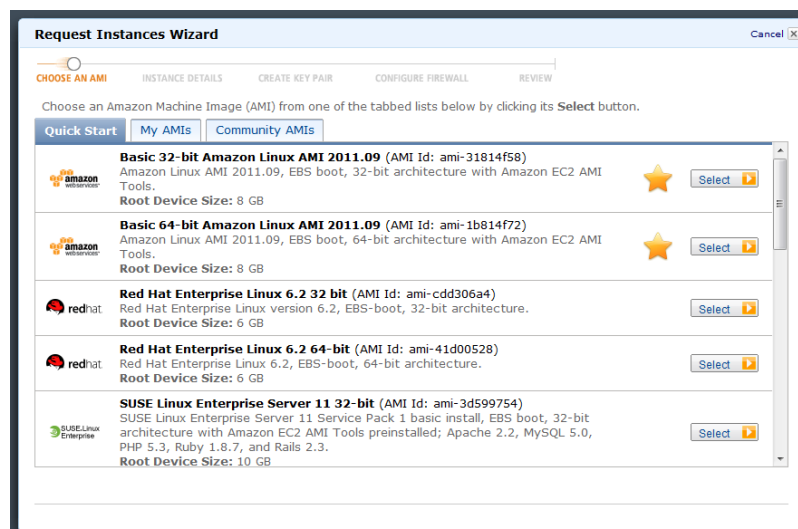


Figure 7. Select the OS for the new instance.

- c) This page will let us to specify the instance details
- number of instance to create (use “1”)
 - Instance type (use “Micro (t1.micro, 613 MB)
 - Availability Zone (us-east-1a)

Figure 8. Specify instance detail.

- d) Then the wizard will let us to identify the *key pair* to use with the instance (the key pair is the tool which allow us to perform the securely connect to the instance after it launches). If we are already have an existing key-pair, just select “choose from your existing Key pairs, but in this case we are just to create our first instance,
- select “ Create a new Key pair” and type in the name of the key pair
 - click “Create and Download your new Key Pair” (You're prompted to save the private key from the key pair to your system.)
 - Save the private key in a safe location in your computer, because it will be used later to connect to the instance.
- e) After the new Key pair is created, the page will refresh again and in this time select “Choose from your existing Key pairs” and select your created key pair in the drop down box.

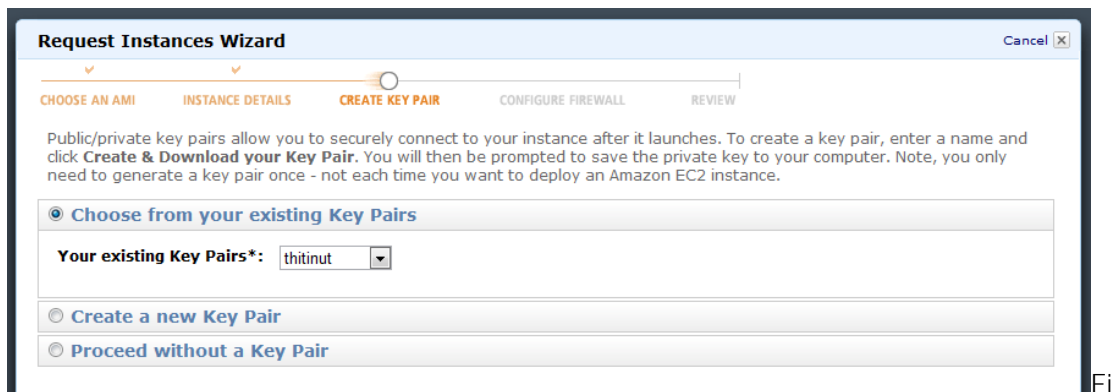


Figure 9. Select the key pair using to connect to the created instance.

- f) Next the wizard will display the page for configure the firewall, where we will be asked to create the *security group*. A security group defines firewall rules for the instances. These rules specify which incoming network traffic should be delivered to our instance (e.g., accept web traffic on port 80). All other traffic is ignored. Therefore we can modify rules for a group at any time. The new rules are automatically enforced for all running instances.

In this case, we will select “Create a new Security Group” and specify the name and the short description for the Security group.

Group name : quick-start-1

Group Description : quick-start-1

With the Security group, we can allow any traffic to come to our instance by open the port and also able to select the source of the traffic that we allow. Likewise, for our instance, we allow 4 kind of traffic

- HTTP (port 80) for web service traffic
- HTTPS (port 443) for web service traffic
- MYSQL (port 3306) for MySQL database traffic
- SSH (port 22) for SSH connection (In real production, the source IP address should be specified in order to retain security).

Request Instances Wizard Cancel

CHOOSE AN AMI INSTANCE DETAILS CREATE KEY PAIR **CONFIGURE FIREWALL** REVIEW

Security groups determine whether a network port is open or blocked on your instances. You may use an existing security group, or we can help you create a new security group to allow access to your instances using the suggested ports below. Add additional ports now or update your security group anytime using the Security Groups page.

Choose one or more of your existing Security Groups

Create a new Security Group

Group Name

Group Description

Inbound Rules

Create a new rule:

Port range:
(e.g., 80 or 49152-65535)

Source:
(e.g., 192.168.2.0/24, sg-47ad482e, or 1234567890/default)

TCP	Port (Service)	Source	Action
	80 (HTTP)	0.0.0.0/0	Delete
	443 (HTTPS)	0.0.0.0/0	Delete
	3306 (MYSQL)	0.0.0.0/0	Delete
	22 (SSH)	0.0.0.0/0	Delete

Figure 10. Configure instance firewall policy.

- g) After finish with entire launching configuration, the last page of the wizard will display the Review of you our configuration, if there is nothing to change, just click “Launch” and within minutes, the instance is ready to use.

Request Instances Wizard Cancel

CHOOSE AN AMI INSTANCE DETAILS CREATE KEY PAIR CONFIGURE FIREWALL **REVIEW**

Please review the information below, then click **Launch**.

AMI: Amazon Linux AMI ID ami-31814f58 (i386)

Name: Basic 32-bit Amazon Linux AMI 2011.09

Description: Amazon Linux AMI 2011.09, EBS boot, 32-bit architecture with Amazon EC2 AMI Tools. [Edit AMI](#)

Number of Instances: 1

Availability Zone: us-east-1a

Instance Type: Micro (t1.micro)

Instance Class: On Demand [Edit Instance Details](#)

Monitoring: Disabled **Termination Protection:** Disabled

Tenancy: Default

Kernel ID: Use Default **Shutdown Behavior:** Stop

RAM Disk ID: Use Default

User Data: [Edit Advanced Details](#)

Key Pair Name: thitinut [Edit Key Pair](#)

Security Group(s): sg-60e87009 [Edit Firewall](#)

Figure 11. The detail summary of an instance.

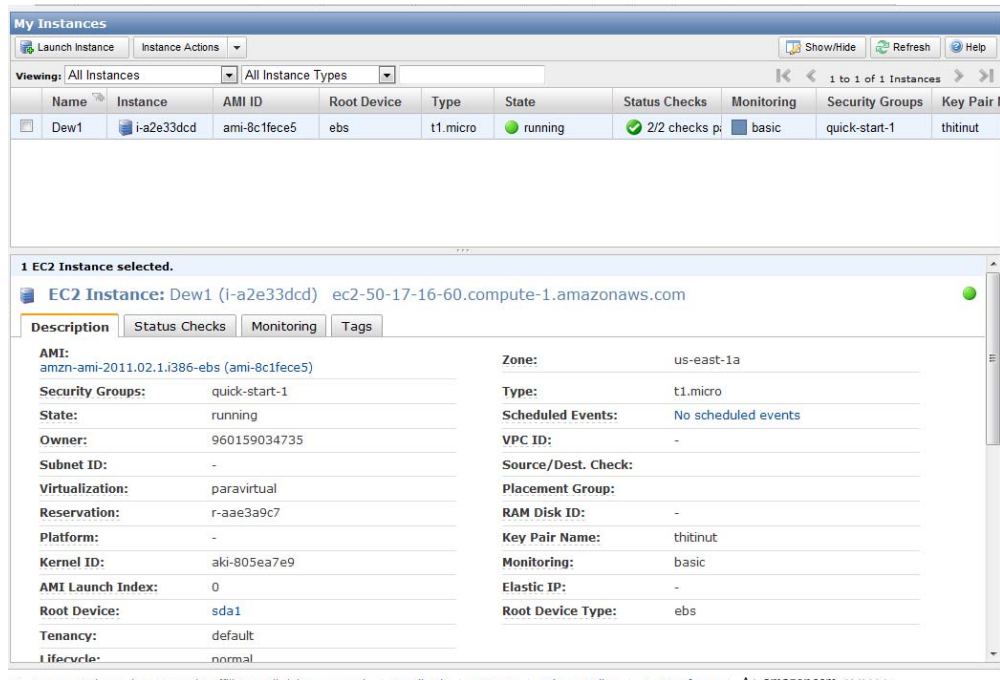


Figure 12. New launched instance is ready.

3.1.2 Connect to the instance

Now we have our instance, in order to connect to the instance by remote access, using the free software name “Putty” is the easiest solution, and also provide comfortable. The following section will provide briefly description on how to remote connect to the instance via the SSH tunnel (refer that Putty is already installed on the PC).

With using Putty, the key pair that we got from creating an instance in previous section is incompatible. Anyway, with the Putty software package there is a tool named “PuTTYgen” which can be used to convert the Amazon giving key pair to the require PuTTY key format. The steps are described as following.

- a) Start PuTTYgen (Program > PuTTY > PuTTYgen)
- b) Click *Load* and browse to the Key location (for example *keyname.pem*)
- c) Select the *.pem* key file and click *Open*, then PuTTYgen will display the following message, just click *OK*



Figure 13. Foreign key import successful.

- d) Another dialog will appear to confirm the new Key generating, click YES, the new key format will be generate (*keyname.ppk*) which is ready to use with PuTTY.

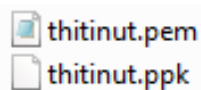


Figure 14. Security key using for authentication.

3.1.3 Configure PuTTY

- a) Start PuTTY, the PuTTY interface can be seen as Figure 15, on the left side is a Category menu and the detail for each menu will be displayed on the right side.

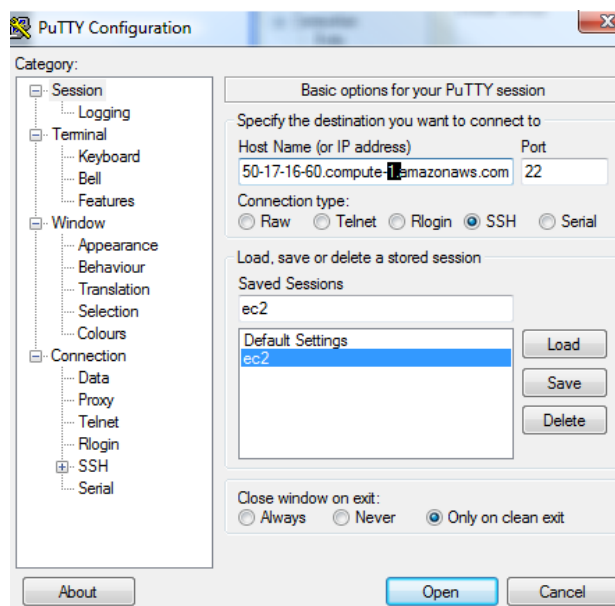


Figure 15. Interface of Putty.

- b) Select *Session* and fill in the “Host Name” using the EC2 instance ID which can be found in the EC2 instance information (Figure 15), For the “port”, we use port 22 which is a standard port for SSH connection.
- c) In the Menu Category select *Connection > SSH > Auth* then in the “Private key file for authentication” input area, browse to navigate to the authentication key location which we just generate using PuTTYgen in the previous section (*keyname.ppk*)
- d) Click ‘Open’, then PuTTY will connect to the instance and prompt for log in name which is “ec2-user” (ec2-user is the default user that is granted when the instance is launched along with root user).

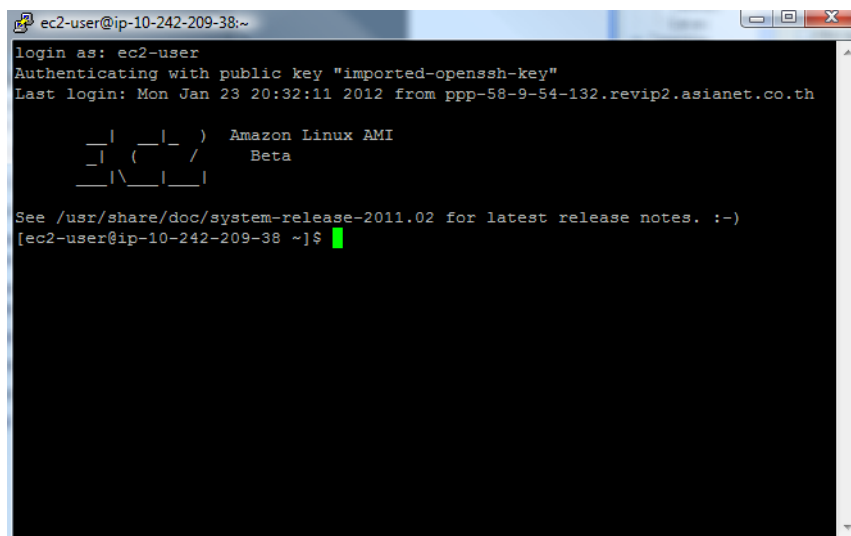


Figure 16. Connect to instance using Putty successful.

3.1.4 Configure the EC2 instance

Once we connect to the instance, as see in the Figure 16, that now we are in the instance console. With the starting condition of an instance, we need to setting up a web server on an instance. The step are describe as following

- a) Update all the current installed packages on the instance

```
$ sudo yum -y update
```

- b) Install all the software we need in order to initiate the linux web server, for current instance condition we need to install Apache, PHP and the PHP extension a

```
$ sudo yum -y install httpd php php-cli php-gd php-intl php-mbstring php-mysql php-pdo php-soap php-xml php-xmlrpc php-openssl
```

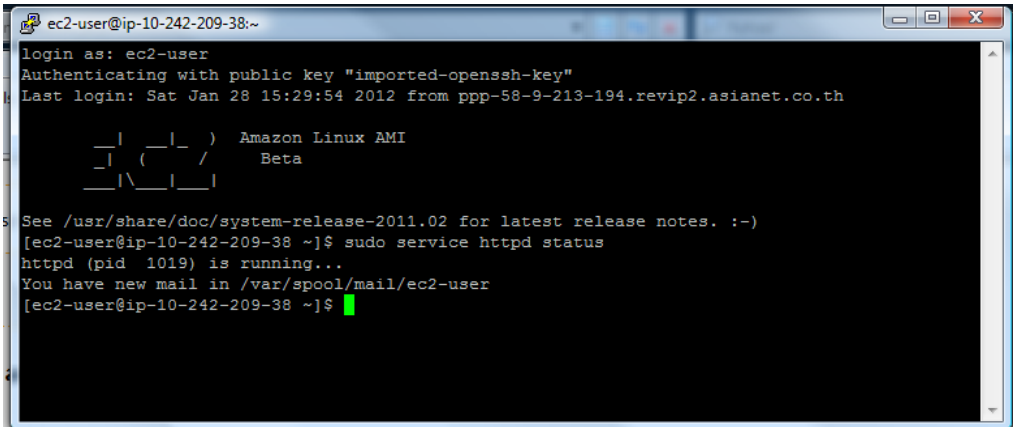
- c) After finish the installation, we need to start the service and configure the service, so that they will automatically start up on instance boot time.

```
$ sudo /sbin/chkconfig httpd on  
$ sudo /sbin/service httpd start
```

- d) After the service is started, we can check for the service condition with command

```
$ sudo service httpd status
```

If the service is running it will show the result like in Figure 17, now we are ready to create the web service on this web server instance already.



```
ec2-user@ip-10-242-209-38:~  
login as: ec2-user  
Authenticating with public key "imported-openssh-key"  
Last login: Sat Jan 28 15:29:54 2012 from ppp-58-9-213-194.revip2.asianet.co.th  
  
  _I_  _I_ ) Amazon Linux AMI  
  _I_ (  _I_ /   Beta  
  _I\  _I\  |  
  
See /usr/share/doc/system-release-2011.02 for latest release notes. :-)  
[ec2-user@ip-10-242-209-38 ~]$ sudo service httpd status  
httpd (pid 1019) is running...  
You have new mail in /var/spool/mail/ec2-user  
[ec2-user@ip-10-242-209-38 ~]$
```

Figure 17. Start httpd service on instance.

3.2 Launch RDS and configuration

3.2.1 Launch RDS instance

After finish with setting up the web server instance, this chapter will show the step in establishing of the Relational Database Service(RDS)

- a) First at the service management console, go to RDS > launch DB Instance to start the launch RDS DB instance wizard.

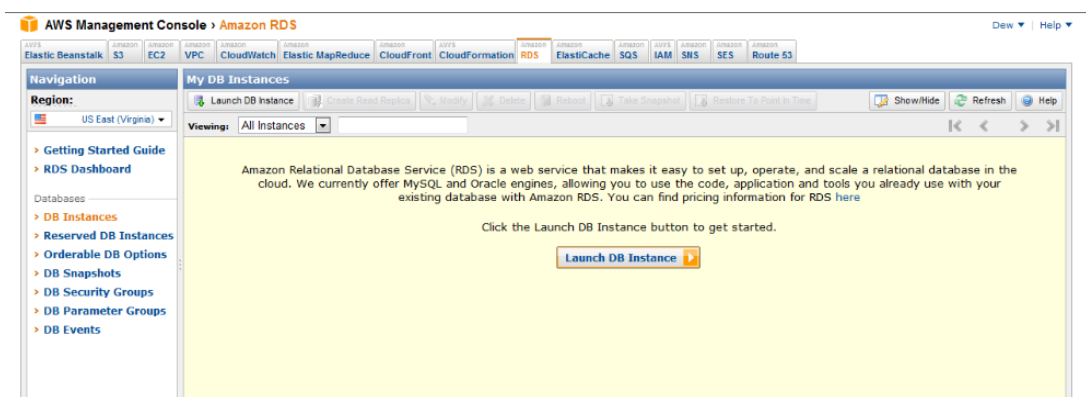


Figure 18. The interface for create new RDS instance.

- b) Select MySQL as a database engine, then input the configuration details as in Figure 19.

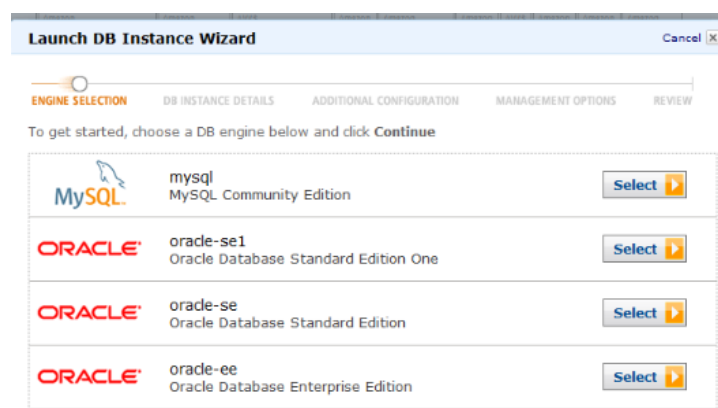


Figure 19. Select the database engine for RDS.

Launch DB Instance Wizard Cancel

ENGINE SELECTION **DB INSTANCE DETAILS** ADDITIONAL CONFIGURATION MANAGEMENT OPTIONS REVIEW

To get started, choose a DB Instance engine and class below

Engine: mysql

License Model: general-public-license

DB Engine Version: 5.1.57 (default)

DB Instance Class: db.m1.small

Multi-AZ Deployment: No

Auto Minor Version Upgrade: Yes No

Provide the details for your RDS Database Instance.

Allocated Storage:* 5 GB (Minimum: 5 GB, Maximum 1024 GB)

DB Instance Identifier:* db1 (e.g. mydbinstance)

Master User Name:* dew (e.g. awsuser)

Master User Password:* •••••• (e.g. mypassword)

[< Back](#) [Continue >](#)

Figure 20. Specify RDS detail.

Important parameter

Engine : mysql – Use MySQL database

DB Instance Class : db.m1.small – Depend on the size of the web application

Multi-AZ Deployment : No – This feature allow database that locate in multiple zone to automatically replicating database update.

Auto Minor Version Upgrade : Yes – Enable the Database instance to automatically receive minor engine upgrades when available.

Launch DB Instance Wizard Cancel

ENGINE SELECTION DB INSTANCE DETAILS **ADDITIONAL CONFIGURATION** MANAGEMENT OPTIONS REVIEW

Provide the optional additional configuration details below.

Database Name: (e.g. mydb)

Note: if no database name is specified then no initial mysql database will be created on the DB Instance.

Database Port: (e.g. 3306)

Availability Zone:

If you have custom DB Parameter Groups or DB Security Groups you would like to associate with this DB Instance, select them below, otherwise proceed with default settings.

DB Parameter Group: default.mysql5.1

DB Security Groups: default

[< Back](#) [Continue](#)

Figure 21. Specify some additional configuration.

Launch DB Instance Wizard Cancel

ENGINE SELECTION DB INSTANCE DETAILS ADDITIONAL CONFIGURATION **MANAGEMENT OPTIONS** REVIEW

Please review the information below, then click **Launch**

Engine: mysql
Engine Version: 5.1.57
License Model: general-public-license
Auto Minor Ver. Upgrade: Yes
DB Instance Class: db.m1.small
Multi-AZ Deployment: No
Allocated Storage: 5
DB Instance Identifier: db1
Master User Name: dew
Master User Password: *****

Database Name: test2
Database Port: 3306
Availability Zone: us-east-1a
DB Parameter Group: default.mysql5.1
DB Security Group(s): default

Backup Retention Period: 1
Backup Window: No Preference
Maintenance Window: No Preference

[< Back](#) [Launch DB Instance](#)

Figure 22. Summary page before launching an instance.

3.2.2 Create Database

After the database instance was created, next we will create the data table. First we need to connect to the EC2 instance via SSH then use the LINUX command

```
$ mysql -u username -ppassword -host=RDS_AMI_ID
```

Then we will be at the RDS commandline console, perform the step as following

```
mysql> show databases;
mysql> use database_name;
mysql> show tables;
```

Now we will create two tables in the database which will use to keep track of the number of current online user on the web application and another one is for a number of web server instances at current time.

Table : User	Table : Server
- Count (int)	- Handle (int)

Table 2. Database diagram for application.

3.3 Enable and configure an Auto Scaling

Configuration of an Auto Scaling is need to be performed through the command line interface (window command prompt) in the local machine. Therefore, we need to set up the environment in our local computer first. The steps for setting up the command line are described as following.

3.3.1 Enable Auto Scaling

- a) Download the AWS command-line tools at <http://aws.amazon.com/developertools/2535>
- b) Save and unpack the archive file in local PC
- c) In order to install the package, at least Java v1.5 or newer is required, to check the current version of Java, open the command prompt and enter

```
C:\> java -version
```

If the java version is not v1.5 or newer, go to <http://www.java.com/en/download/index.jsp> to get the newest version and install.

- d) Set the JAVA_HOME environment variable to point to Java folder.
- e) Set the AWS_AUTO_SCALING_HONE environment variable to point to AWS tool folder.

```
C:\> set JAVA_HOME = <path>
C:\> set PATH=%PATH%;%AWS_AUTO_SCALING_HOME%\bin
C:\> set AWS_AUTO_SCALING_HOME=<path>
C:\> set PATH=&PATH%;%AWS_AUTO_SCALING_HOME%\bin
```

- f) Before we can use Auto Scaling, we must provide the AWS credentials to the command-line tools, therefore we need to use the AWS access keys.
 - Go to <http://aws.amazon.com/security-credentials>
 - Scroll down to 'Access Credentials' to get the 'Access Key ID' and Secret Access Key.

Access Credentials

There are three types of access credentials used to authenticate your requests to AWS services: (a) access keys, (b) X.509 certificates, and (c) key pairs. Each access credential type is explained below.

Access Keys | X.509 Certificates | Key Pairs

Use access keys to make secure REST or Query protocol requests to any AWS service API. We create one for you when your account is created — see your access key below.

Your Access Keys

Created	Access Key ID	Secret Access Key	Status
August 1, 2010	ABGDR2QZ4PWRSWQ6894Q	Show	Active (Make Inactive)

[Create a new Access Key](#)

For your protection, you should never share your secret access keys with anyone. In addition, industry best practice recommends frequent key rotation.

[Learn more about Access Keys](#)

Figure 23. Security credential information.

- Add the 'Access key ID' and 'Secret access key' to the file named *credential-file-path.template* in the AWS tool folder at our local PC.
- g) Set the `AWS_CREDENTIAL_FILE` environment variable to the fully qualified path of the credential file

```
C:\> set AWS_CREDENTIAL_FILE=<path>
```

- h) After perform all the steps, in order to check the completeness, enter the command

```
C:\> as-cmd
```

The command prompt screen will display all the commands of the Auto Scaling tool, the real environment set up is show as below.

```

C:\Windows\system32\cmd.exe
operable program or batch file.
C:\Users\Dew>set JAVA_HOME=C:\Program Files\Java\jre6
C:\Users\Dew>PATH=%PATH%;%JAVA_HOME%\bin
C:\Users\Dew>set PATH=%PATH%;%JAVA_HOME%\bin
C:\Users\Dew>set AWS_AUTO_SCALING_HOME=C:\AS
C:\Users\Dew>set PATH=%PATH%;%AWS_AUTO_SCALING_HOME%\bin
C:\Users\Dew>set AWS_CREDENTIAL_FILE=C:\AS\credential-file-path.template
C:\Users\Dew>as-cmd
-----
Command Name          Description
-----
as-create-auto-scaling-group  Create a new auto scaling group
as-create-launch-config      Create a new launch config
as-create-or-update-trigger   Creates a new trigger or updates an existing trigger.
as-delete-auto-scaling-group  Delete the specified auto scaling group
as-delete-launch-config       Delete the specified launch configuration
as-delete-policy              Delete the specified policy
as-delete-scheduled-action   Delete the specified scheduled action
as-delete-trigger             Delete a trigger.
as-describe-adjustment-types  Describes all policy adjustment types.
as-describe-auto-scaling-groups  Describes the specified auto scaling group(s)
as-describe-auto-scaling-instances  Describes the specified auto scaling instance(s)
as-describe-launch-configs     Describe the specified launch configurations
as-describe-metric-collection-types  Describes all metric colle... metric granularity types.
as-describe-policies           Describes the specified policy/policies
as-describe-process-types      Describes all scaling process types.
as-describe-scaling-activities  Describe a set of activiti...ties belonging to a group.
as-describe-scheduled-actions  Describes the specified scheduled action(s)
as-describe-triggers           Describes a trigger including its internal state.
as-disable-metrics-collection  Disable collection of AutoScaling group metrics
as-enable-metrics-collection   Enable collection of AutoScaling group metrics
as-execute-policy              Executes the specified policy
as-put-scaling-policy          Creates or updates a scaling policy
as-put-scheduled-update-group-action  Creates or updates a scheduled update group action
as-resume-processes            Resumes all suspended scal... given auto scaling group.
as-set-desired-capacity        Set the desired capacity of the auto scaling group
as-set-instance-health         Set the health of the instance
as-suspend-processes           Suspends all scaling proce... given auto scaling group.
as-terminate-instance-in-auto-scaling-group  Terminate a given instance.
as-update-auto-scaling-group   Update specified auto scaling group
help                            Prints the version of the CLI tool and the API.
version

For help on a specific command, type '<commandname> --help'

```

Figure 24. Listing all the Auto Scaling command after finish an installation.

3.3.2 Configure an Auto Scaling

After finish with the command-line tool setup, next we will configure the Auto scaling. The step can be divided mainly into four parts.

- 1) Create launch configuration
 - 2) Create Auto Scaling group
 - 3) Create Auto Scaling policy
- a) Create launch configuration

The launch configuration specifies the type of Amazon EC2 instance that Auto Scaling creates for the system. To create a launch configuration with *as-create-launch-config*, we must specify an Amazon Machine Image (AMI) ID and an Amazon EC2 instance type. As following we create launch config "MyLC" using the following instance.

```
Instance AMI ID : ami-8c1fece5
Instance Type : t1.micro
```

```
C:\> as-create-launch-config MyLC --image-id ami-8c1fece5 --instance-type
```

b) Create Auto Scaling group

After we have defined the launch configuration, we are ready to create an Auto Scaling group. To create an Auto Scaling group with *as-create-auto-scaling-group*, we must specify a group name, a launch configuration, one or more Availability Zones, a minimum group size, and a maximum group size.

In our system we name an Auto Scaling group "DewASGroup" and use the launch configuration you created previously. The 'Availability Zones' determines the physical location of our Auto Scaling instances. For this example, specify a single zone: us-east-1a. Set the minimum size to 0 and maximum size to 20.

```
C:\> as-create-auto-scaling-group DewASGroup --launch-configuration MyLC
--availability-zones us-east-1a --min-size 0 --max-size 20
```

To check the completeness enter the following command

```
C:\> as-describe-auto-scaling-groups --headers
```

```
C:\Users\Dev>as-create-launch-config MyLC --image-id ami-8c1fece5 --instance-type t1.micro
OK-Created launch config
C:\Users\Dev>as-create-auto-scaling-group DewASGroup --launch-configuration MyLC --availability-zones us-east-1a --min-size 0 --max-size 20 --load-balancers
OK-Created AutoScalingGroup
C:\Users\Dev>as-describe-auto-scaling-groups --headers
AUTO-SCALING-GROUP GROUP-NAME LAUNCH-CONFIG AVAILABILITY-ZONES LOAD-BALANCERS MIN-SIZE MAX-SIZE DESIRED-CAPACITY
AUTO-SCALING-GROUP DewASGroup MyLC us-east-1a myLB 0 20 0
C:\Users\Dev>as-put-scaling-policy DewASGroup --name double-group-size --adjustment 100 --type PercentChangeInCapacity --cooldown 600_
```

Figure 25. Command for create Auto scaling policy.

c) Create Auto Scaling policy

A scaling policy command Auto Scaling on how to change the size of the application fleet in response to the desired variable (in this case is the number of

concurrent user), enabling us to specify not only whether we want to scale the group up or down, but also how much. We can express the desired change in capacity as an absolute number, an increment, or as a percentage of the current group size. When a policy is executed, Auto Scaling uses both the current group capacity and the desired change specified in the policy to compute a new desired capacity. Auto Scaling then updates the desired capacity. Moreover each Auto Scaling group can have up to 25 policies.

In our Auto Scaling group, we want to have 10 scaling policies inside the group. 5 policies are using for adding more instances to the group and the other 5 are using for terminate instances from the group. And the properties are as following

Scale-up1 = add 1 instance to group	Scale-down1 = terminate 1 instance from group
Scale-up2 = add 2 instances to group	Scale-down2 = terminate 2 instances from group
Scale-up3 = add 3 instances to group	Scale-down3 = terminate 3 instances from group
Scale-up4 = add 4 instances to group	Scale-down4 = terminate 4 instances from group
Scale-up5 = add 5 instances to group	Scale-down5 = terminate 5 instances from group

Here is an example for creating of Scale-up1

```
C:\> as-put-scaling-policy Scalingpolicy1 --g DewASGroup
"--adjustment=1" --type ChangeInCapacity --cooldown 120 --name scale-up2
```

```
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=2 --type ChangeInCapacity --cooldown 120 --name scale-up2
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:497e5076-0a30-4955-b5f9-4e9588fab77f:autoScalingGroupName/DewASGroup:policyName/scale-up2
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=3 --type ChangeInCapacity --cooldown 120 --name scale-up3
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:9e0d7450-753b-4e2f-a567-a0e577936df9:autoScalingGroupName/DewASGroup:policyName/scale-up3
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=4 --type ChangeInCapacity --cooldown 120 --name scale-up4
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:2b90b406-5654-44b0-9a2e-2c28a8e4d46a:autoScalingGroupName/DewASGroup:policyName/scale-up4
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=5 --type ChangeInCapacity --cooldown 120 --name scale-up5
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:1e20e10e-2063-47b2-b2f8-ffeeb0c53112:autoScalingGroupName/DewASGroup:policyName/scale-up5
```

Figure D2: Create Scaling policy for “scale up”

```
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup "--adjustment=-1" --type ChangeInCapacity --cooldown 120 --name scale-down1
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:4e060bb9-6f41-415c-a01e-16096fcea1ea:autoScalingGroupName/DewASGroup:policyName/scale-down1
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=-2" --type ChangeInCapacity --cooldown 120 --name scale-down2
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:9df7d2cb-d9af-461e-9c8d-26f90acce9a2:autoScalingGroupName/DewASGroup:policyName/scale-down2
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=-3" --type ChangeInCapacity --cooldown 120 --name scale-down3
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:de0fc8f0-e3a4-4051-b718-20f9c006dedf:autoScalingGroupName/DewASGroup:policyName/scale-down3
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=-4" --type ChangeInCapacity --cooldown 120 --name scale-down4
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:3347e71f-9bd7-4360-9e77-8df053d0dffa:autoScalingGroupName/DewASGroup:policyName/scale-down4
C:\Users\Dew>as-put-scaling-policy ScalePolicy1 -g DewASGroup --adjustment=-5" --type ChangeInCapacity --cooldown 120 --name scale-down5
arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:9fd060f-254f-440e-03e4-32b48946a344:autoScalingGroupName/DewASGroup:policyName/scale-down5
```

Figure D3 : Create Scaling policy for “scale down”

After finish with creating of all the Scaling policies, we can check the configuration using the *describe-policies* command.

```
C:\> as-describe-policies -g DewASGroup
```

```
C:\Users\Dewas> as-describe-policies -g DewASGroup
SCALING-POLICY  DewASGroup  scale-down1  -1  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:4ed60bb9-6f41-415c-a81e-16896fce
Name/scale-down1
SCALING-POLICY  DewASGroup  scale-down2  -2  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:9df7d2cb-d9a1-46bc-9c8d-26f90acc
Name/scale-down2
SCALING-POLICY  DewASGroup  scale-down3  -3  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:de0fc8f0-e3a4-4031-b718-20f9c006
Name/scale-down3
SCALING-POLICY  DewASGroup  scale-down4  -4  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:3347e71f-9bd7-4368-9e77-8df053d8
Name/scale-down4
SCALING-POLICY  DewASGroup  scale-down5  -5  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:9f8c868f-254f-440e-83e4-32b48946
Name/scale-down5
SCALING-POLICY  DewASGroup  scale-up1    1  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:e71cdfaf8-bae0-4088-8c89-75c13a9d
Name/scale-up1
SCALING-POLICY  DewASGroup  scale-up2    2  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:497e5076-0a30-4955-b5f9-4e9588fd
Name/scale-up2
SCALING-POLICY  DewASGroup  scale-up3    3  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:9e0d7450-753b-4e2f-a567-a8e57793
Name/scale-up3
SCALING-POLICY  DewASGroup  scale-up4    4  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:2b90b406-5654-44b0-9a2e-2c28a8e4
Name/scale-up4
SCALING-POLICY  DewASGroup  scale-up5    5  ChangeInCapacity  120  arn:aws:autoscaling:us-east-1:960159034735:scalingPolicy:1e20e18e-2063-47b2-b2f8-ffeb0cb
```

Figure 26. Listing all of the scaling policy of the AutoScaling group.

3.4 Create PHP script on an EC2instance

PHP script is response for communicate with the database(RDS) periodically to check for the number of current system user. If the number of current user is more than current system capacity, the PHP script will call the scaling policy to add more instances to the application, but if the number of current user is less than the current capability, some instance will be terminated.

The PHP script mainly can be divided into 3 main parts which are

- 1) Authentication
- 2) RDS connection
- 3) Auto scaling trigger

3.4.1 For Authentication , inside the PHP script has to define 'AWS_KEY', 'AWS_SECRET_KEY', 'AWS_ACCOUNT_ID' and 'AWS_CANONICAL_ID' which can be define in the script as following

```
define('AWS_KEY','AKIAIW6ANSYGLAAF6KVA');
define('AWS_SECRET_KEY','LwVm1S2gnFo6sR7Xu/XlFXBmyfUgpg+UBT9zkhjhp');
define('AWS_ACCOUNT_ID','960159034735');
define('AWS_CANONICAL_ID','7ecdb900c35fa0db195b0c3540b2a8b9e4855ba5f1265295d30a82185f5da97d');
```

3.4.2 For RDS connection, we can create the connection to our application database instance using basic SQL connection command

```

$con = mysql_connect("db1.cxzpsqxsoz4f.us-
east1.rds.amazonaws.com","dew","852456");
if (!$con)
{
die('Could not connect: ' . mysql_error());
}

// Get from DB
mysql_select_db("test2", $con);

$GetUserT1 = mysql_query("SELECT * FROM User");
$GetUserT2 = mysql_fetch_assoc($GetUserT1);
$UserCount = $GetUserT2['Count'];

$GetServerT1 = mysql_query("SELECT * FROM Server");
$GetServerT2 = mysql_fetch_assoc($GetServerT1);
$CHandle = $GetServerT2['Handle'];

```

The important parameters are

UserCount – describe the number of current system online user at the time

CHandle – describe the number of instances which are operating on the system at the time

3.4.3 Auto Scaling trigger, by comparing between UserCount and CHandle, the script can desire whether to add more instance or terminate some instance from the group.

```
if($UserCount >= $CHandle) //need to add instance
```

```
elseif($CHandle > $UserCount) //need to terminate instance
```

After comparing the different between two variables will be store to *\$temp* variable, then *\$temp* will be used for choosing to fire the rule. Triggering of rules is performed through the if-else condition as in following example.

```

if ($temp >= 250)
{
    $CreateMore = $as->execute_policy('scale-up5', array(
        'AutoScalingGroupName' => 'DewASGroup',
        'HonorCooldown' => 'false'
    ));
    mysql_query("UPDATE Server SET Handle = Handle + 250");
}

```

In the example code, describe the case when the number of user more than current application capacity, the script will call for execute Auto Scaling policy “scale-up5” which will add 5 more server instance into the group. In the last line, while defines that one instance serves 50 users, after adding 5 server into the group, the script will call for add number of current users for 250 users (for full source code, please see the Appendix).

After creating the PHP script, In order to make it able to be run-able, we need to set the LINUX crontab function to run the script periodically, in this application is every 5 minutes.

```

[ec2-user@ip-10-242-209-38 ~]$ crontab -l
*/5 * * * * /usr/bin/php /var/www/html/2_TestPage/AS60.php

```

Figure 27. Configure the Linux crontab to run the script

3.5 System working process

For better understanding of the architecture and how it work, Figure 28 to Figure 31 illustrate of how the system work in basic. In this example, suppose that an instance can handle up to 4 users for maximum. Fig. 5 describes the starting condition

which there is only one server instance that can handle 4 concurrent users. In Fig. 6 the number of concurrent user is increase to 8, then one more instance is automatically added into the group (by calling the an Auto Scaling policy). In Fig. 7, the concurrent user increase to 14, then two more instances are added. In Fig.8, the concurrent user decrease down to 10, so one instance will be terminated and there will be only 2 instances left.

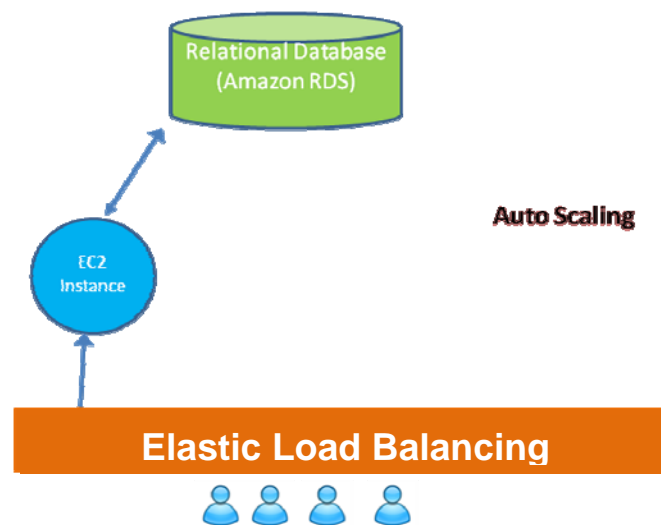


Figure 28. One instance to serve 4 users.

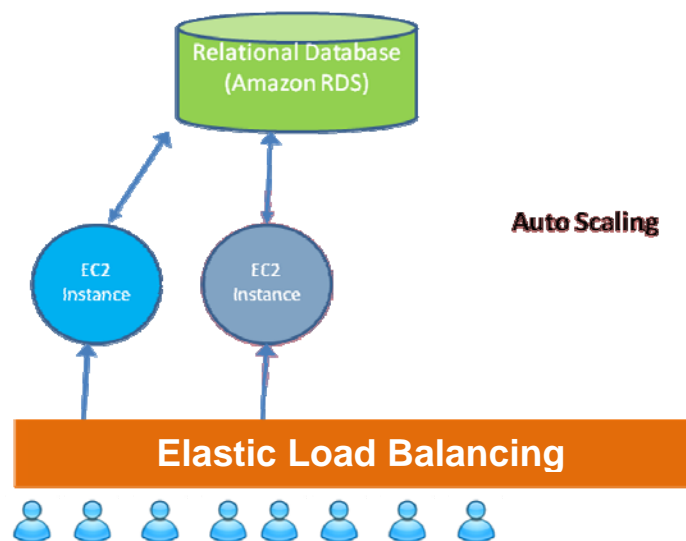


Figure 29. Two instances to serve 8 users.

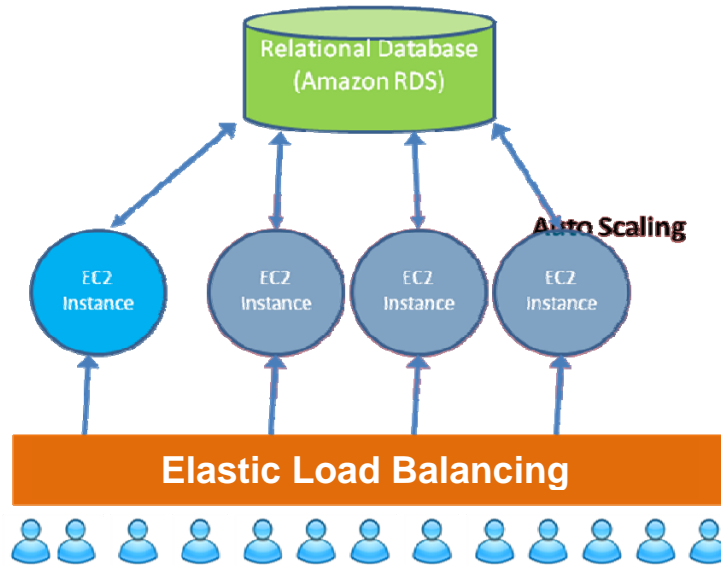


Figure 30. Four instances to serve 14 users.

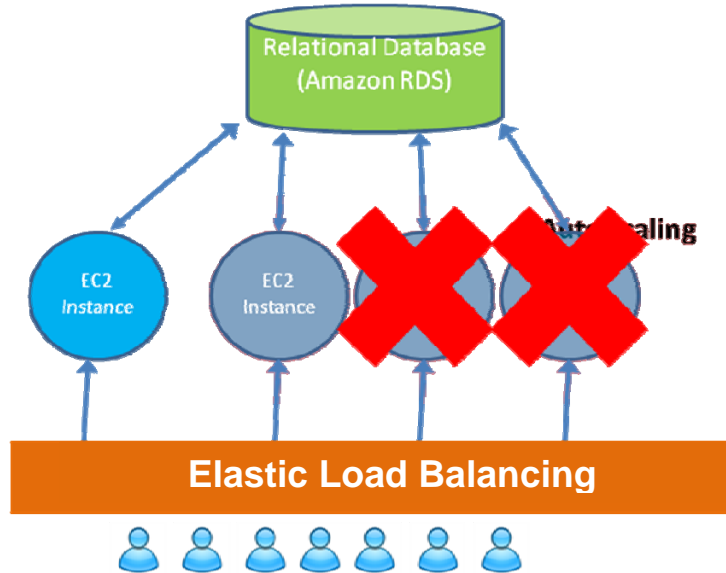


Figure 31. Two instances to serve 7 users.

CHAPTER IV

Experimental Results

In this chapter we will perform the experiment which will be divided into two parts. First, the functional testing (the proof of concept test) will demonstrate how the system really works by modifying the number of parameters in the database (RDS) in order to track the sizing behavior of the system to verify the correctness and completeness of the system configuration and to prove the idea of auto scaling using group policy with the PHP script. Second, the experimental result will be presented in plots to show the cost estimation and total project cost compared to the old-fashion web application practice.

4.1.Functional testing

- a) In this experimental, the system parameter is set up as following.

Auto Scaling group

Max size	20 instances
Min size	0 instance
Cool down time	0 second

Parameters description

- Max size – the maximum number of instance that Auto Scaling can handle.
- Min size – the minimum number of instance that Auto Scaling must handle.
- Cool down time – the period that Auto scaling have to wait before able to take another action

b) Crontab frequency = 5 minutes

Remark: One instance can support 50 users, Experimental period = 1 hour.

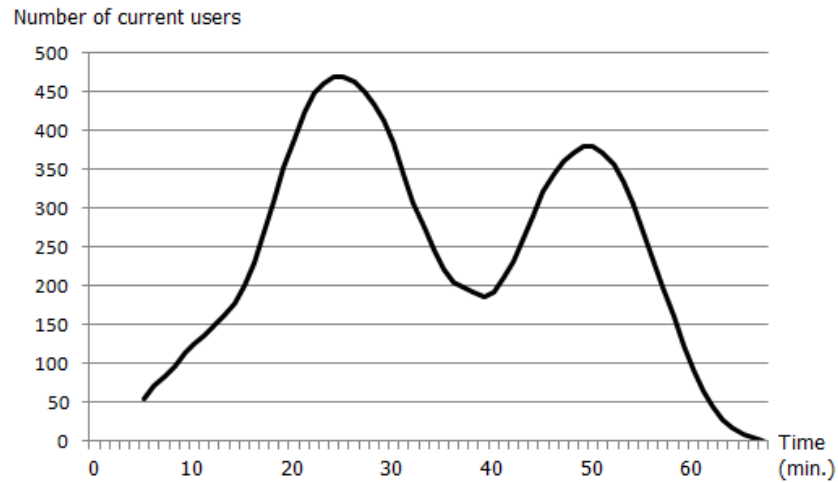


Figure 32. User distribution over 1 hour

Figure 32 represents the number of users on each second of an experimental period. During the experimental, the parameter value that is used to represent the number of current incoming users in the database will be update at every minute.

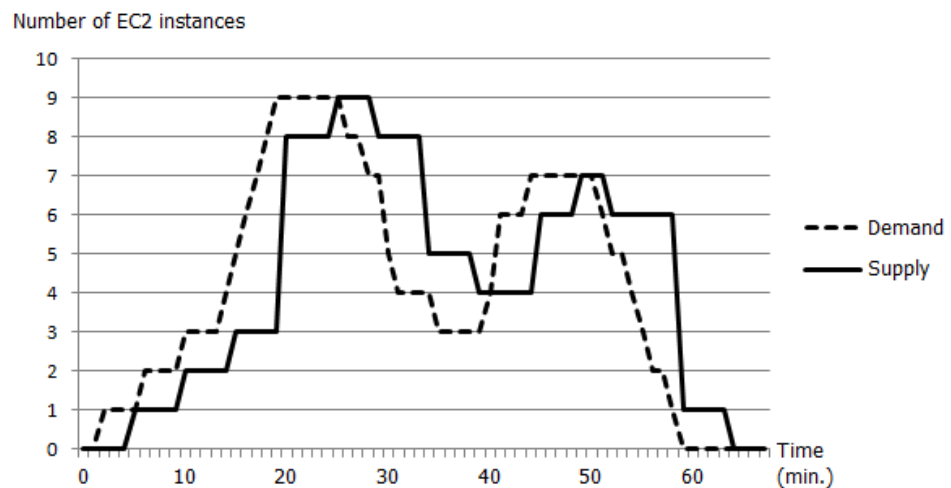


Figure 33. Demand and supply of EC2 instances over time.

Figure 33 describes the result of the experiments. The dash line represents the number of instances that is needed at that period of time which is based on the number of the incoming users at that time. The solid line represents the actual

number of instances operating at that time. As in the graph, we can see that once the demand is increased the system also react to the need of the incoming users by adding more servers into the system with the delay around 5 minutes. Also at the 20th minute, the system show that this technique can support even the rapidly increasing of incoming user by adding four more instances in a row in order to support the number of users at that period. And from 30th minute to 60th minute, the graph shows how the system reacts when the number of users is decreasing. From this experiment, we can conclude that our solution is capable of handling with increasing and decreasing of the incoming users.

4.2. Cost compilation

This part shows the experimental result of total project cost compilation between our solution and current practice method (static number of web server instance). In this experiment we introduce four different kinds of user distribution which are Best, Bad, Worse and Worst as depicted in Figure 34. These user distributions represent the possibility of incoming users of a web project that operates over a predefined period of time. In this experiment, we set the operating period at 14 days and the total number of incoming users is set at 500,000.

This experimental result is simulated by the simulator, therefore the parameters meaning are need to be clarified in order to understand the experimental result.

- User Mean = 10 min, , User SD = 1. A user spends 10 minutes in average to accomplish using web application. The standard deviation is 1 minute.
- User per Instance = 50. In our experiment we define that there must be one instance per 50 active users. In practice, this parameter can be determined easily.

- Auto Scaling = 0 min. This parameter is the “cool-down time” suggested by Amazon. After Auto Scaling takes an action, it will wait for a cool-down time before being able to take another action. This parameter prevents too fast response. If it is zero, Auto Scaling checks the conditions, and takes action every second.
- Crontab = 5 min. Linux Crontab round time that is set up to run a PHP script for every 5minute.
- Boot Mean = 5 min, Boot SD = 0.5. Boot time for duplicating a new instance is 5 min in average. The standard deviation is 0.5 min or 30 sec. The billing starts immediately after booting. The new instance is ready to use in about 5 minutes later.
- Termination Mean = 8 min, Termination SD = 0.8. An average termination time of an instance is 8 min. The standard deviation is 0.8 min or 48 sec.
- Rental Cost = 0.085 US dollar. The cost of renting a “micro” instance per hour is 0.085 US dollar (in US East and Linux usage). It is important to note that *pricing is per instance-hour consumed for each instance, from the time an instance is launched until it is terminated. Each partial instance-hour consumed will be billed as a full hour*
- Daily user distribution. As shown in Fig. 9, there are four kinds of user distributions. Each of them holds 500,000 users

Total system user (per day)

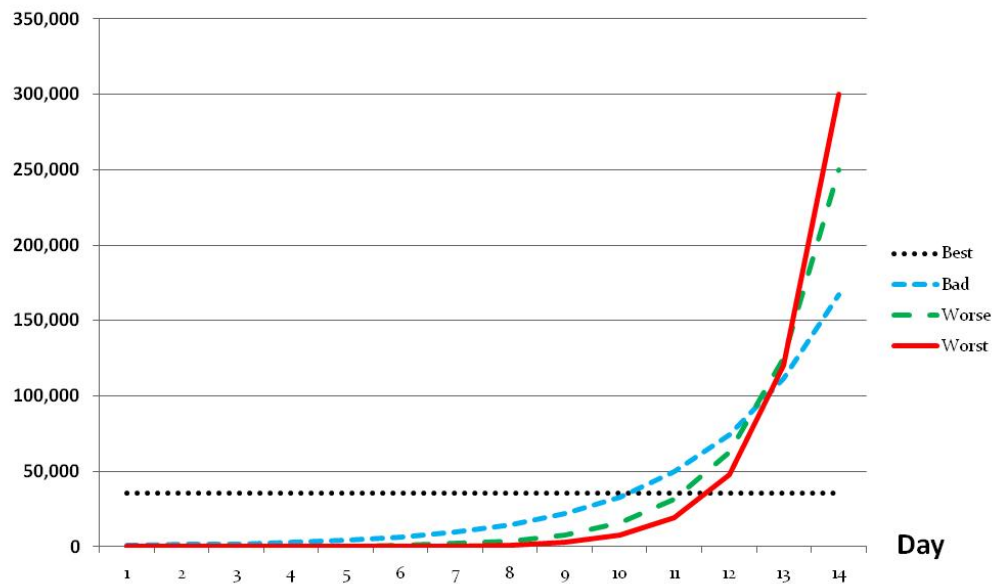


Figure 34. Four kinds of user distribution.

The total estimated project cost for those mentioned user distributions by using the traditional method is shown as following

Distribution	Possible largest user per day	Total estimate cost (\$)
Best	~ 35,718	~ 142.80
Bad	~ 167,242	~ 685.44
Worse	~ 250,015	~ 999.6
Worst	~ 300,000	~ 1199.52

Table 3. The cost estimation for conventional method.

As in the table 3, in order to maintain the system effectiveness, the total cost of the project is affected directly by the user distribution, moreover, the less random level of user distribution (most users likely to use the service at the same time) the more total project cost is.

In contrast to our solution, with the same set of user distributions, the total project cost estimation is extremely reduced except for the “Best case” (ideal user

distribution). And from the result, we can see that the more number of “possible largest user per day” is the more cost saving our solution provides, compare to the traditional approach.

Distribution	Possible largest user per day	Total estimate cost for Our method(\$)	Total estimate cost for Conventional method(\$)	Difference (\$)
Best	~ 35,718	~ 191.45	~ 142.80	+48.65
Bad	~ 167,242	~ 179.96	~ 685.44	- 505.48
Worse	~ 250,015	~ 180.84	~ 999.6	-818.76
Worst	~ 300,000	~ 179.53	~ 1199.52	-1019.99

Table 4. Conventional method compare with our method (Crontab = 5, AutoScaling=0).

Moreover, with our solution more cost saving can be achieved by tuning of the system variable such as the crontab round time and the auto scaling cool-down time. But tuning the crontab round time and leave the auto scaling cool-down time to zero is more preferred since it is easier to be changed which is more comfortable when performing in the real presentation.

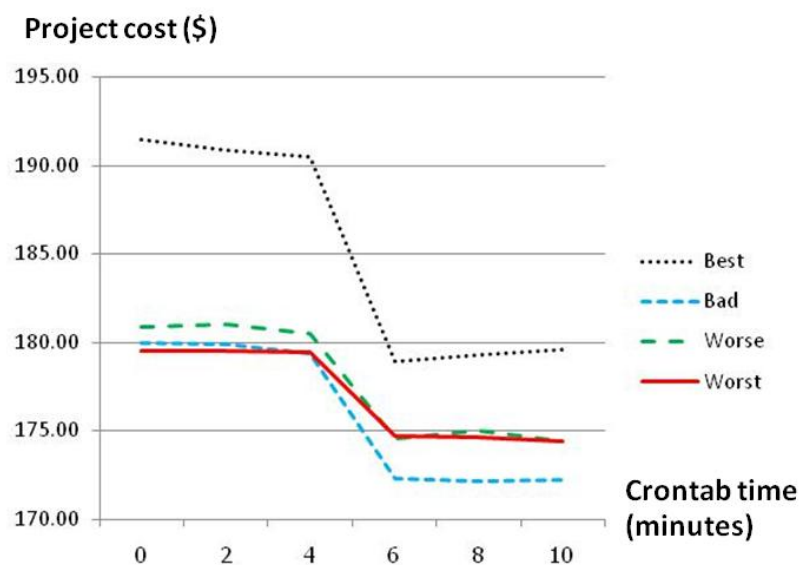


Figure 35. Crontab = 5 mins, AutoScaling 0 – 10 mins

In Figure 35, represent an example on tuning the parameter, the decreasing in project cost estimation is shown with regarding to the tuning of the Auto Scaling cooldown time from 0 to 10, from the graph, the cost drops only when cool-down time equal or more than 5 minutes, this is because any changes during 5 minutes are not observed (crontab round time = 5 minutes). Therefore it is useless to set the cool-down time to be less than the round time. Moreover in Figure 36, the cost estimation for tuning the crontab round time is shown. In this experimental, the Auto scaling cool-down time is set to 0, then the value of the crontab round time can be any number that more than 0. As a result, tuning the crontab value from 1 to 10 shows that the cost is decreased as the crontab round time is increased.

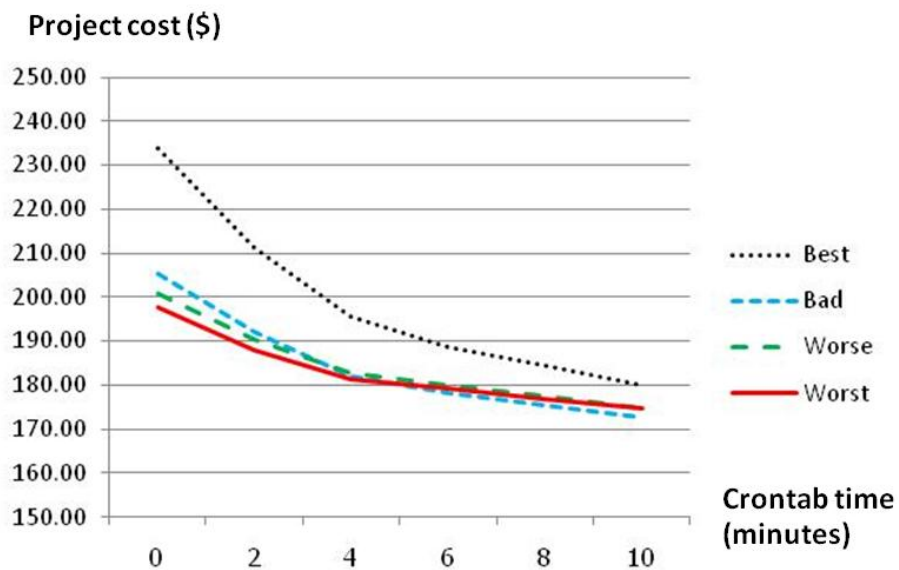


Figure 36. Crontab = 1 – 10 mins, AutoScaling = 0 mins.

CHAPTER V

Discussion

In this chapter we will discuss on two topics. First will discussion on our problem solution including the result and possible further improvement. Second, we discuss on possible way of measuring the quality of service.

5.1.Solution

Originally, our solution is designed to used Amazon Cloudwatch as an main component to play role in measure the amount of an incoming traffic but after some development and research, comparing between using of Amazon Cloudwatch and RDS + PHP script, we see that using the PHP script is a better solution since it is more comfortable, free, agile and also can be further implement with more advanced or complicated techniques, for example, with using a PHP script, we can defined the round time to any number but with Amazon Cloudwatch, the round time is fixed to only 1 and 5, also additional fees is charged when using round time as “:1” (detailed monitoring). Therefore, our final result does not require using of the Cloudwatch as a main system component.

For further development of this solution, the algorithm for trigger the event is still wide-open for development, since this part rely on the coding technique of the PHP script. For current, the event trigger measures the amount of user at a certain point of time (similar to pick up the sample every 5 minutes). Development on this portion can be done with many different ideas such as: finding the average between time period and call for new instance with respect to the average value, or even perform the prediction for an amount of incoming user in the next time period with respect to the

passed user amount. As mentioned, implementation of PHP script brings up wide open for better solution.

5.2. Quality of service

As seen from the experimental, although we can tune some parameters to cut the cost, but in realistic, there is no such thing as a free lunch. The quality of service might be deteriorated to compensate the cost reduction. We define two quantities for measuring the badness of service. First, “overload time” is defined as the sum of time period that EC2 instance has been overloaded (an instance is serving smaller amount of user than its capacity). Second, “overload users” is defined as a set of numbers of excessive users per an instance in every second of the overload time. In our simulator, the time period scale is changed into seconds. Hence, the number of overload users is reported as min, max, and average. The quality of service is depicted in Table 5, 6, 7 and Table 8. It is seen that the cost drops with increasing of crontab round time, but the quality of service obviously deteriorates (as in Figure 37). The increasing of overload users describes the degrading of the service quality. Users have to wait longer for a response. Finally, the system becomes unresponsive.

Once the system has degraded, users will stay on the system longer due to large response time. If an instance dies, all users on the instance have to restart the application. However, our simulator does not take the effect of system degradation into account. Therefore, the overload users may produce an additional cost as mentioned.

Crontab round time	Best				
	Cost	Overload time	Overload user (per instance per second)		
			Hrs	Min	Avg
10	178.93	52.59	0.17	2.23	188.80
20	169.24	67.65	0.17	2.96	210.70
30	165.84	73.66	0.17	3.39	219.20
40	162.78	75.78	0.17	3.84	221.80
50	160.40	76.57	0.17	4.31	228.30
60	161.25	76.95	0.17	4.69	234.80
3 hours	154.39	84.89	0.17	9.16	240.60
6 hours	149.16	84.27	0.17	16.26	241.60
12 hours	137.74	88.78	0.17	29.42	252.10
24 hours	121.32	97.29	0.17	52.70	249.20
48 hours	97.32	120.66	0.17	83.07	255.70

Table 5. Result on round time tuning for “Best” user distribution.

Crontab round time	Bad				
	Cost	Overload time	Overload user (per instance per second)		
			Hrs	Min	Avg
10	172.41	30.53	0.04	2.42	29.29
20	166.94	37.81	0.04	3.29	37.28
30	165.40	41.36	0.04	3.90	34.75
40	163.27	42.70	0.04	4.19	37.25
50	162.32	43.00	0.04	3.63	34.05
60	147.98	44.60	0.04	4.72	36.34
3 hours	135.29	60.53	0.04	8.57	42.30
6 hours	126.71	73.39	0.04	11.17	43.75
12 hours	119.87	107.79	0.04	15.23	49.25
24 hours	106.10	166.27	0.07	17.33	44.16
48 hours	95.10	183.63	0.12	33.46	86.34

Table 6. Result on round time tuning for “Bad” user distribution.

Crontab round time	Worse				
	Cost	Overload time	Overload user (per instance per second)		
			Hrs	Min	Avg
10	174.64	22.96	0.03	2.93	49.91
20	170.62	29.28	0.03	4.43	62.46
30	169.41	32.00	0.03	5.39	62.60
40	167.86	31.27	0.03	6.26	61.54
50	167.10	31.73	0.03	5.37	63.94
60	141.61	32.65	0.03	8.14	70.89
3 hours	129.29	46.28	0.03	14.45	70.13
6 hours	123.07	57.35	0.03	20.66	71.41
12 hours	112.51	84.52	0.03	25.42	66.49
24 hours	94.03	132.06	0.09	36.02	78.46
48 hours	73.54	136.32	0.16	64.91	152.64

Table 7. Result on round time tuning for “Worse” user distribution.

Crontab round time	Worst				
	Cost	Overload time	Overload user (per instance per second)		
			Hrs	Min	Avg
10	174.65	19.39	0.02	2.97	75.64
20	171.36	23.39	0.02	4.75	80.38
30	170.61	25.11	0.02	6.77	90.74
40	169.24	25.47	0.02	7.87	87.57
50	168.91	24.49	0.02	6.10	93.26
60	138.04	27.65	0.02	10.88	94.35
3 hours	123.62	36.16	0.02	21.41	92.78
6 hours	115.14	48.47	0.02	33.67	105.79
12 hours	104.73	67.73	0.02	44.78	101.47
24 hours	83.78	111.04	0.08	50.98	97.11
48 hours	59.89	111.20	0.20	111.69	289.58

Table 8. Result on round time tuning for “Worst” user distribution.

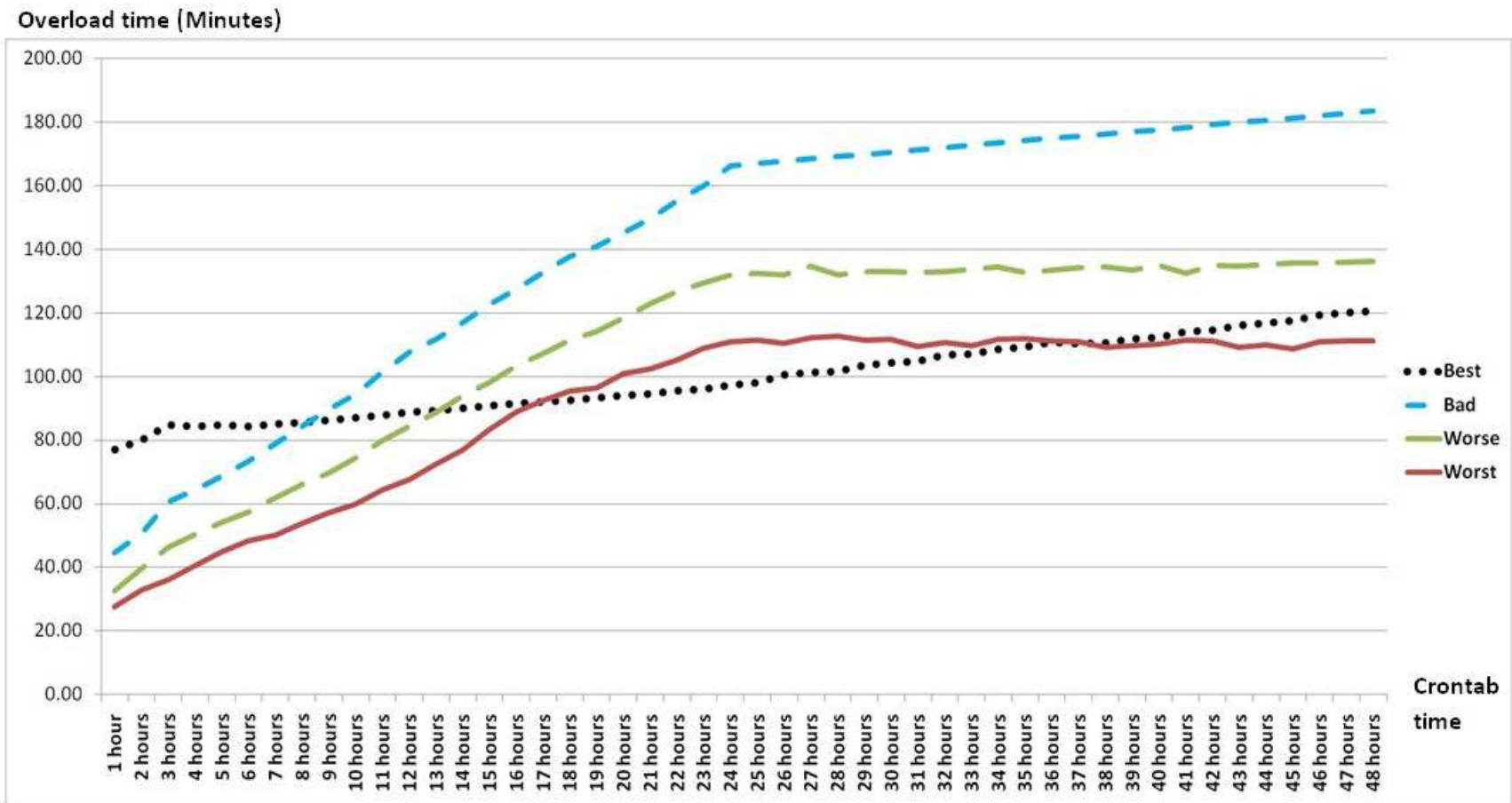


Figure 37. Overload time value with respect to Crontab time round time tuning.

5.3. The Optimal parameters

Lastly, everyone wants to know the optimal parameters, but this is a multi-objective optimization. There are two objectives, 1) minimizing cost and 2) maximizing quality of service. The two objects seem to be contradictory to each other. In our opinion, a wide range of parameters is applicable. For instance, as seen from Table 5, 6, 7 and Table 8, choosing the CronTab round time between 10 and 60 min is fine for all user distribution. But setting the CronTab round time at 10 hours is unacceptable because in the worst case the system is overloaded by $21.97 / 50 = 43.94\%$ in average for 60.71 hours. A tradeoff between the cost and the quality of service is up to your decision.

CHAPTER VI

Conclusion

With this research we have represent the new solution on running the web application project which bring better in both efficiency and effectiveness. For efficiency, compare to the traditional method (using the static number of web server instance), our solution provide the 24/7 of availability to the system user. Mainly discuss in this thesis is the issue about the nature of the system user for the applicant project that most of the user will try to login and submit the application around last 2 -3 days before closing date(deadline), such as an entrance examination or contest. Which the opening period for submit the application is fixed. Finally with implementation of PHP script, further development is wide-opened for any developer and any algorithm.

Moreover, we also show how to estimate the cost of running a web application on Amazon EC2. Our simulation shows that the cost is subjected to user distributions. The exponential distributions better utilize EC2 instances, hence lowering the cost compared to that of uniform distribution. The most important parameter that reduces the cost is tuning of the round time, but the quality of service degrades in the exchange of cutting cost. A key issue of parameter optimization is to find an acceptable tradeoff between the cost and the quality of service which will be different on each project based on the priority level and concerning of the project manager.

References

- [1] M. Armbrust et al., A view of cloud computing, Communication of the ACM 53 (4)(2010) : 50-58.
- [2] Amazon EC2. Available from: <http://aws.amazon.com/ec2>.
- [3] Case Studies. Available from : <http://aws.amazon.com/solutions/case-studies/>.
- [4] Simple Monthly Calculator. Available from :
<http://calculator.s3.amazonaws.com/calc5.html>.
- [5] E. Deelman, G. Singh, M. Livny, B. Berriman, J. Good, The cost of doing science on the cloud: The Montage example. International Conference for High Performance Computing, Networking, Storage and Analysis (SC'08) (2008) : 1-12.
- [6] A. Spellmann, R. Gimarc, M. Preston, Leveraging the cloud for green IT: predicting the energy, cost and performance of cloud computing. Computer Measurement Group Conference (CMG'09) (2009).
- [7] D. Kondo, B. Javadi, P. Malecot, F. Cappello, D.P. Anderson, Cost-benefit analysis of cloud computing versus desktop grids, The 23rd Parallel and Distributed Processing Symposium (IPDPS'09) (2009) : 1-12.

Biography

Mr. Thitinut Treenorraseth was born in 1986. He obtained his degree in Computer Science from the Mahidol University, Bangkok, Thailand, in 2007. And in the same year, he joined Card System and Marketing Co.,Ltd as a system support staff. In 2009, he resigned from Card System and Marketing and join YIC Asia Pacific Corporation Limited as IT operation staff and still working for this company for present.