AUTOSCALING WEBSERVICES ON AMAZON EC2

M. A. AZEEZ

This dissertation was submitted to the Department of Computer Science and Engineering of the University of Moratuwa in partial fulfillment of the requirements for the Degree of M Sc in Computer Science specializing in Software Architecture

Department of Computer Science & Engineering
University of Moratuwa,
Sri Lanka
February 2010

96421

ABSTRACT

Fault tolerance, high availability, & scalability are essential prerequisites for any Enterprise application deployment. One of the major concerns of enterprise Application architects is avoiding single points of failure. There is a high cost associated with achieving high availability & scalability. We will look at an economical approach towards automatically scaling Web service applications whilemaintainingtheavailability&scalabilityguaranteesatanoptimumeconomicalcostThisapproach,inv olving the Amazon EC2 cloud computing infrastructure, makes it unnecessary to invest in safety- net capacity & unnecessary redundancy. The Web service application developer should only need to write the application once, and simply deploy it on the cloud. The scalability & availability guarantees should be provided automatically by the underlying infrastructure. Auto scaling refers to the behavior where the system scales up when the load increases & scales down when the load decreases. Auto-healing refers to an approach where a specified minimum deployment configuration is maintained even in the event of failures. Such an approach is essential for cloud deployments such as Amazon EC2 where the charge is based on the actual computing power consumed. Ideally, from the clients' point of view, in an auto scaling system, the response time should be constant and the overall throughput of the system should increase. We will describe in detail an economical approach towards building auto-scaling Apache Axis2 Web services on Amazon EC2. In the course of this article, we will introduce well-known address (WKA) based membership discovery for clustering deployments where multicast-based membership discovery is an impossibility. We will also introduce an approach towards dynamic load balancing, where the load balancer itself uses group communication & group membership mechanisms to discover the domains across which the load is distributed. In a traditional setup, a single load balancer fronts a group of application nodes. In such a scenario, the load balancer can be a single point of failure. Traditionally, techniques such as Linux HA have been used to overcome this. However, such traditional schemes have quite a bit of overhead and also require the backup system to be in close proximity to the primary system. In case of catastrophic situations, this approach can result in complete failure of the system. We will introduce an auto healing scheme in case of load balancer failure using Amazon Elastic fP addresses & a load balancer group, which can overcome these shortcomings.

-

Declaration

The work included in this report was done by me, and only by me, and the work has not been submitted for any other academic qualification at any institution.

[illegible] 25th February 2010

Afkham Azeez Date

I certify that the declaration above by the candidate is true to the best of my knowledge and that this report is acceptable for evaluation for the CS6999 M.Sc. Research Project.

Feb 25, 2010

Sanjiva Weerawarana (PhD) Date

ACKNOWLEDGMENTS

I would like to express profound gratitude to my advisor, Dr. Sanjiva Weerawarana, for his invaluable support, encouragement, supervision and useful suggestions throughout this research work. His continuous guidance enabled me to complete my work successfully.

I am grateful for the support & assistance provided by the management of WSO2, who provided me the research facilities. This work may not have been possible without the Amazon Web services R&D account that was provided by WSO2. This account was extensively used during this research. This work would not have been possible without the support & assistance I received from Filip Hanik, author of the Apache Tribes group communication framework, which is extensively used in my work. I would also like to thank Asankha Perera, software architect at WSO2 & a lead architect of Apache Synapse, who provided advice on architecture & design. I would also like to thank Ruwan Linton, a lead developer of Apache Synapse, and Paul Fremantle, project lead of Apache Synapse, who provided me with design ideas related to dynamic load balancing & load analysis. I would also like to thank my colleague & good friend Amila Suriarachchi for reviewing my work and providing valuable feedback & suggestions for improvements.

I would like to thank Chinthana Wilamuna, the scripting languages expert at WSO2, for providing his expert advice on various aspects, as well as Deependra Ariyadewa & Chamith Kumarage, Linux experts, for providing their expertise.

I am grateful to the open source communities backing the Apache Axis2 & Apache Synapse projects. Their contributions were the foundation upon which this work was built.

I am as ever, especially indebted to my parents for their love and support throughout my life. I also wish to thank my wife, who supported me throughout my work. Finally, I wish to express my gratitude to all my colleagues at WSO2. Many ideas related to this project have come to my mind when having technical & even non-technical discussions with this group of intellectuals.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF TABLES

LIST OF ABBREVIATIONS

AMI	Amazon Machine Image
AWS	Amazon Web Services
CPU	Central Processing Unit
CORBA	Common Object Request Broker Architecture
EC2	(Amazon) Elastic Compute Cloud
ESB	Enterprise Service Bus
JVM	Java Virtual Machine
GCF	Group Communication Framework
GMP	Group Membership Protocol
GMS	Group Membership Service
HA	High Availability
HTTP	Hypertext Transfer Protocol
SLA	Service Level Agreement
SOAP	Historically, Simple Object Access Protocol. Now simply SOAP
S3	(Amazon) Simple Storage Service
WKA	Well-known Address, Well-known Addressing
WSDL	Web Services Description Language