hochschule mannheim

Performance Comparison between JAVA and Node.js in terms of Scaling in Cloud

Zheng Zeng

Bachelor Thesis

for the acquisition of the academic degree Bachelor of Science (B.Sc.)

Course of Studies: Computer Science

Department of Computer Science

University of Applied Sciences Mannheim

11.11.2015

Tutors

Prof. Peter Knauber, Hochschule Mannheim

Jens Keller, SAP SE

Zeng, Zheng:
Performance Comparison between JAVA and Node.js in terms of Scaling in Cloud / Zheng Zeng. –
Bachelor Thesis, Mannheim: University of Applied Sciences Mannheim, 2015. **??** pages.

Zeng, Zheng:
/ Zheng Zeng. –
Bachelor-Thesis, Mannheim: Hochschule Mannheim, 2015. **??** Seiten.

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Ich bin damit einverstanden, dass meine Arbeit veröffentlicht wird, d. h. dass die Arbeit elektronisch gespeichert, in andere Formate konvertiert, auf den Servern der Hochschule Mannheim öffentlich zugänglich gemacht und über das Internet verbreitet werden darf.

Mannheim, 11.11.2015

Zheng Zeng

Abstract

Performance Comparison between JAVA and Node.js in terms of Scaling in Cloud

Contents

1 Introduction **1**

2 Related Work **3**

3 Experimental Environments **4**
3.1 Server running environment - Cloud Foundry 4
3.1.1 Cloud Foundry at SAP 4
3.1.2 PaaS - Service plans, Backing services and more 4
3.1.3 Limitations 4
3.2 Client running environments 4
3.2.1 Influential factors for measuring 4
3.2.2 Client in local machine 4
3.2.3 Client in Cloud Foundry 4
3.2.4 Client in Monsoon 4

4 Testing tool **5**
4.1 Apache JMeter 5
4.1.1 JMeter introduction 5
4.1.2 JMeter Limitations 5
4.2 Load generator - a self implemented scalable client 5
4.2.1 Design and function of load generator 5
4.2.2 Limitations 5

5 Measuring tool **6**
5.1 Response time 6
5.1.1 Apache JMeter and its limitations 6
5.1.2 Recording with Load Generator and its limitations 6
5.1.3 Retrieving data from ELK and its limitations 6
5.2 CPU and Memory consumption 6
5.2.1 Cloud Foundry CLI and its limitations 6
5.2.2 A self-implemented Ruby application and its limitations . . 6

6 Test Scenario **7**
6.1 IO Intensive 7

6.2 Compute Intensive 7

7 Test Results and Analysis **8**
7.1 IO Intensive 8
7.2 Compute Intensive 8

8 Conclusion **9**

List of Tables **v**

List of Figures **vi**

Source references **vii**

Chapter 1

Introduction

The promise of cloud has ushered the software engineering into a new era. Technology is to be used anytime and anywhere to untangle issues, provide solutions, bring people together and change their ways of living. This new era of net-centric web-based applications redefines how software is delivered to a customer and how the customer uses the delivered software. The digitization of economy impacts everyone. New businesses and leaders are emerging from nowhere. Companies appear and disappear much faster than ever before.

SAP , a company which has been renown for its on premise ERP solutions, also faces the pressure from customers to reduce TCO and increase agility. It has since long announced its cloud strategy to help customer master the digital economy. The reach of SAP systems is to be extended through cloud based access, ultimately reaching everybody everywhere in entirely new ways. Cloud Computing allows on demand software provisioning with Zero-Installation and automatic configuration at low cost and immediate access in ultra-scalable data centers, which leads to the next generation networked solutions. In the process of leverage cloud infrastructure to increase the business agility and to lower the TCO of customers, ultimately new types of applications are enabled.

Cloud solutions come with new capabilities and technical challenges for cloud software development. Different cloud software development platforms and frameworks that applied with different programming languages are used inside SAP. JAVA EE and Node.JS are the most prominent ones. There have been on-going discussions about how to take a choice between them for SAP applications considering the SAP environment. This paper aims to discuss this issue in terms of their respective performance and scalibility. The reason is simple. Among all the major

advantages brought by the cloud paradigm, scalability is the one that makes cloud computing different to an "advanced outsourcing" solution. Performance, which heavily affects customer satisfaction is a key metric.

Chapter 2

Related Work

Chapter 3

Experimental Environments

3.1 Server running environment - Cloud Foundry

3.1.1 Cloud Foundry at SAP

3.1.2 PaaS - Service plans, Backing services and more

3.1.3 Limitations

3.2 Client running environments

3.2.1 Influential factors for measuring

3.2.2 Client in local machine

3.2.3 Client in Cloud Foundry

3.2.4 Client in Monsoon

Chapter 4

Testing tool

4.1 Apache JMeter

4.1.1 JMeter introduction

4.1.2 JMeter Limitations

4.2 Load generator - a self implemented scalable client

4.2.1 Design and function of load generator

4.2.2 Limitations

Chapter 5

Measuring tool

5.1 Response time

5.1.1 Apache JMeter and its limitations

5.1.2 Recording with Load Generator and its limitations

5.1.3 Retrieving data from ELK and its limitations

5.2 CPU and Memory consumption

5.2.1 Cloud Foundry CLI and its limitations

5.2.2 A self-implemented Ruby application and its limitations

Chapter 6

Test Scenario

6.1 IO Intensive

6.2 Compute Intensive

Chapter 7

Test Results and Analysis

7.1 IO Intensive

7.2 Compute Intensive

Chapter 8

Conclusion

List of Tables

List of Figures

Listings