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:
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.

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

article xcolor

Abstract

This project aims to evaluate the performance

Write English abstract

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

Einsatz eines Flux-Kompensators für Zeitreisen mit einer maximalen Höchstgeschwindigkeit von WARP 7

Write German abstract

Contents

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.

Write more about project introduction

Chapter 2

Related Work

Write Related Work

Chapter 3

Technological Background

Performance and load testing has been conducted ever since applications came into been. There is a sea of different tools available in internet. In the case of this paper, the server's capacity is going to be driven to its limit, after which instead of staying overloaded, the server will try to scale and consume more clients. This requires the testing tool to be capable of generating a large enough amount of end users.

One of the most popular testing tools is JMeter which is based on Java and cross-platform. It supports multiple protocol and has a very friendly UI to configure test plans. It even produces aggregated test results in different type of graphs. It seems it has everything one asks for except scalibility. When the test requires a bigger scale, JMeter client machine quickly runs into issue that it is unable, performance-wise, to simulate enough users to stress the server or is limited at network level. An option to work around this performance limitation is to utilize remote/distributed testing [1]to control multiple remote JMeter engines from a single JMeter client. In this way, one can replicate a test across many low-end computers and thus simulate a larger load on the server. However, remote mode does use more resources than running the same number of non-GUI tests independently. If many server instances are used, the client JMeter can become overloaded, as can the client network connection. Another inconvenience of using JMeter is to find a ideal environment to host it. While one can execute the JMeterEngine on the application server, one can not ignore the fact that this will be adding processing overhead on the application server the testing results will be somewhat tainted. All put together, it is decided not to use JMeter as load generating and testing tool in the project.

[1]JMeter Remote Testing Apache [2016]

Write about CPU Memory measuring tool

Write about other tools

Chapter 4

Experimental Environments

The experimenting environment is of most importance for technological evaluation. It is also impossible to find a configuration which can do both technology justice. One can easily end up trapped by some technological limitation of a framework that one used or in the case of this project, confined by existing infrastructure provider. However, as the project is conducted under the company context, the limitations have to be accepted and taken into account.

4.0.1 Server in local machine

Locally a virtual box running in Ubuntu 14.04 with 4 CPUs and 12G memory is installed on host machine which is a Macbook Pro with 4 CPUs and 16G Memory. However, the application has to share the resource with database and load generator. Host machine also inevitably runs other applications like outlook, skype office, which adds processing overhead and affect any performance potential of the application.

4.1 Server running environment - Cloud Foundry

Cloud Foundry is an open source software bundle that allows you to run a polyglot Cloud Computing Platform as a Service (PaaS). Initially it was developed as a Java PaaS for Amazon EC2 by Chris Richardson, in 2009 acquired by SpringSource, which was then acquired by VMWare, then handed over to Pivotal. The Cloud Foundry Foundation (http://www.cloudfoundry.org/) is now the maintainer of Cloud

Foundry. More than 50 companies are members of this foundation, such as Pivotal, EMC, HP, IBM, Rackspace, SAP, or VMWare. The Cloud Foundry PaaS is a multi-component automation platform for application deployment. It provides a scalable runtime environment that can support most frameworks and languages that run on Linux. It also contains many components that simplify deployment and release of microservices applications (for instance, Router, Loggregator, Elastic Runtime (Droplet Execution Agents (DEA)), a message bus (NATS), Health Manager, Cloud Controller, etc.)

4.1.1 Cloud Foundry at SAP

Compared with other PaaS offerings, Cloud Foundry has some unique features: it has no limitation on language and framework support and is does not restrict deployment to a single cloud. It is an open source platform that one can deploy to run his apps on his own computing infrastructure, or deploy on an IaaS like Amazon Web Service (AWS), vSphere, or OpenStack. In SAP, it is first integrated with SAP Monsoon, then later shifted to Openstack and now hosted in AWS.

There are two different landscape of Cloud Foundry in SAP. One is called "AWS live". In this landscape, applications run inside containers managed by Warden containerization (a virtualization technique providing isolation on operating system level, which is more efficient than virtual machines). The Warden containers including the applications running inside are managed by DEA that also monitor the application health and provide the management interface to the Cloud Foundry platform.

The other landscape is called "AWS Canary". Instead of DEA, each application VM has a Diego Cell that executes application start and stop actions locally, manages the VM's containers, and reports app status and other data to the BBS and Loggregator. Instead of Warden, Garden is used as the containers that Cloud Foundry uses to create and manage isolated environments. Diego architecture improves the overall operation and performance, for example supporting Docker contaniers.

4.1.2 PaaS - Service plans, Backing services and more

Services such as the UAA form the core of a PaaS offering; the choice depends on the domain of the applications. Cloud Foundry distinguishes Managed Services that obey the Cloud Foundry management APIs, and User-provided Services that serve as adapters to external services. In both cases, applications can be wired to service instances; this is called Service Binding.

Write about Cloud Foundry SAP services

4.1.3 Limitations

Write about Cloud Foundry Limitation: DB, Proxy, Test Landscape

4.2 Client running environments

4.2.1 Influential factors for measuring

Write about limitation: network

4.2.2 Client in local machine

Write about Local Virtual Box, all in one machine, process limitation

4.2.3 Client in Cloud Foundry

Write about Cloud Foundry DB limitation

4.2.4 Client in Monsoon

Write about Monsoon IAAS, cost, dc location(minimize network influence)

Chapter 5

Testing Tool

5.1 Load generator - a self implemented scalable client

Write about load generator as simple node application

5.1.1 Design and function of load generator

Write about collecting and storing data

5.1.2 Limitations

Write about limitation of db for load generating

Chapter 6

Measuring Tool

6.1 Response time

Write why care about response time

6.1.1 Recording with Load Generator and its limitations

refer to previous chapter

6.1.2 Retrieving data from ELK and its limitations

Write ELK node application and log quota

6.2 CPU and Memory consumption

Write why care about cpu and memory

6.2.1 Cloud Foundry CLI and its limitations

Write: cf statistics

6.2.2 A self-implemented Ruby application and its limitations

Do I need it?

Chapter 7

Test Scenario

7.1 IO Intensive

Store the current order of product and its meta information, find the which shelf stores the current requested product, assign an idling logistic unit to pick up the itinerary ... I/O operations make up the majority of SAP business scenarios.

Write statistcs to support sap business scinario

In the I/O test conducted in this paper, applications are built to realize a scenario: advertisements are published in a bulletin board and clients can browse through the items.

The PostgreSQL backing service from Cloud Foundry is used as database. As the goal is to test how the application handles large amount of concurrency instead of the database efficiency, data will not be queried in high quantity or complicated joined actions in the test.

To bring Java and node.js to a comparable level, the applications are implemented with minimum use of frameworks so that the overhead or any other influence on performance can be first taken off the table.

Missing figure

Add figure for java application implementation

Figure X shows the implementation of the Java application. It uses embedded Jetty server to handle plain HTTP requests. No REST or any other kind of web services is implemented. The connection with database is plain JDBC. CRUD operations are implemented with query statements. No ORM is utilized. Since the data structure is intentionally kept simple: only one table and with no complex data types, the application without ORM does not produce significantly more boiler codes.

Missing figure

Add figure for java application implementation

Figure X shows the implementation of the node application. It uses the standard node.js library to build the web platform. One other dependency it has is the database connection.

Another library both applications have used is the logging library. The logging library is configured so it can be filtered in elastic search. Because the convenience provided by the existing cloud foundry ELK platform, it is able to log information relevant to single requests and retrieve them through elastic search. This function is used later when comparing the measuring results and determining the bottleneck.

7.2 Compute Intensive

Do I still need it ?

Chapter 8

Test Results and Analysis

8.1 IO Intensive

Write results for local

Write results for CF

Write results for monsoon

8.2 Compute Intensive

Do I need it ?

Chapter 9

Conclusion

Write conclusion

Abbreviations

DEA	Droplet Execution Agents
PaaS	Platform as a Service
AWS	Amazon Web Service

List of Tables

List of Figures

Bibliography

[Apache 2016] APACHE: *Apache JMeter - User's Manual: Remote (Distributed) Testing*. 2016. – URL http://jmeter.apache.org/usermanual/remote-test.html. – [Online; accessed 27-December-2016]

Index

Todo list