www.ijiarec.com

ISSN:2348-2079

Volume-7 Issue-1

International Journal of Intellectual Advancements and Research in Engineering Computations

Optimization of webservice-based on control system for balance between network traffic and delay

Mrs.D.Kalaiabirami[1], K.Pavithra[2], M.Pooja[3], K.Priya[4], J.Sindhu Priya[5]

Assistant Professor, Department of CSE, Vivekanandha College of Engineering for Women (Autonomous)

UG scholar, Vivekanandha College of Engineering For Women (Autonomous)

ABSTRACT

Microservices are service-oriented architecture (SOA). In which we can decompose an application into different smaller services. Microservices having load balancer to manage heavy load (traffic) over the network. So that application will be light weight streaming over the network. Java spring boot frame work introduced WebFlux concept. WebFlux is reactive program that simplifies continuous streaming and handles traffic effectively. There is no any polling mechanism to handle multiple requests because polling mechanism will make heavy traffic. Instead of polling mechanism WebFlux uses subscriber and producer concepts that means once the initial http request made to the service then service will send updated data to the subscriber whenever the data being changed in database.

Keywords:

a. **Service :** Spring boot
b. **Database :** Mongo DB
c. **Web UI :** Angular 6

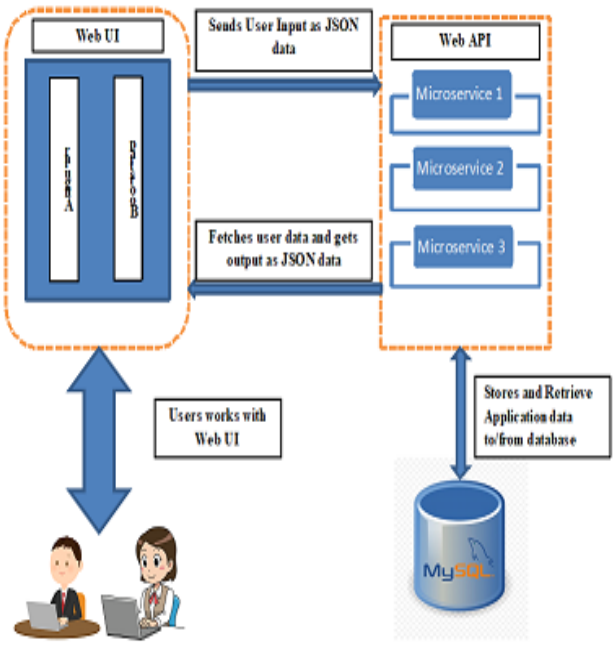
INTRODUCTION

The main endeavor of our project is to Optimizing the network traffic and delay in an application by using Micro services uses load balancer and it applies **Round-robin** (RR) algorithm to handle the http request. In micro services, we can run a module in an application as multiple instances so that each instance gets http request based on algorithm to handle traffic.

Webservice enable so interoperable machine-to-machine communication over networks. Webservices uses a Polling mechanism which is a practical way for a Web service-based control system to enable its actuator for respond to its controller where the actuator does not know exactly when the controller updates its command. A Fast response wants a high polling frequency (polling mechanism handles the event in time-driven mode) which may lead to heavy network traffic. Therefore, how to make the optimal tradeoff between the network traffic and delay in a Web service-based control system becomes an interesting problem. This paper describes the problem of finding the optimal polling frequency control policy of a Web service-based control system as a constrained Markov decision process (CMDP). The policy of that the actuator responds to the controller within a tolerable delay threshold while minimizing network traffic. An algorithm called CMDPA is proposed to solve the problem. Simulation and field experiments show our policy performance.

Author for correspondence:
Department of CSE, Vivekanandha College of Engineering for Women (Autonomous)

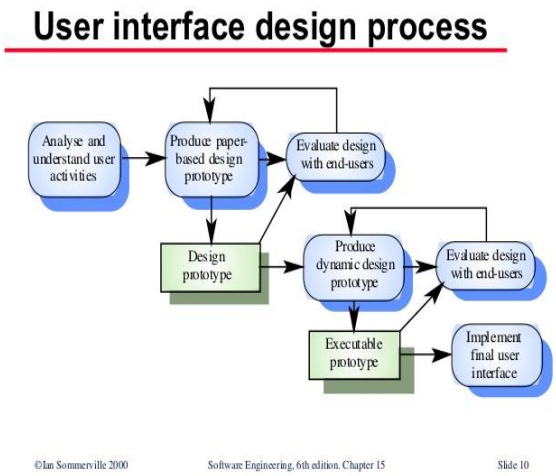
Architecture diagram



WEB UI

User interface design (UI) is the design of user interfaces. It is design of machines and software, such as computers, home appliances, mobile devices, and other electronic devices, with the focus on maximizing usability and the user experience. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals (user-centered design).

The user interface (UI) is everything designed into an information device with which a person may interact. It is also the way through which a user interacts with an application or a website.

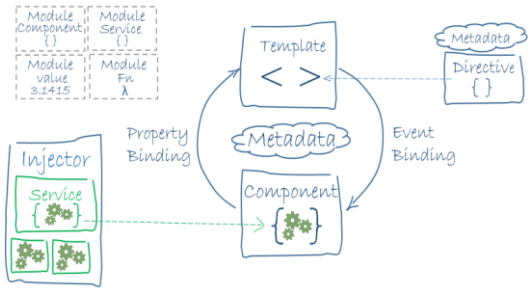


Angular

Angular is a TypeScript-based open-source web application framework. It is introduced by the angular team at Google and by a community of individuals and corporations**.**

Angular recommends the use of Microsoft's **TypeScript** language, which introduces the following features:

- Class-based Object Oriented Programming
- Static Typing
- Generics



Architecture of an Angular application. The main building blocks of Angular JS are modules, components, templates, metadata, data binding, directives, services and dependency injection.

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. AngularJS's data binding and dependency injection eliminate much of the code you would otherwise have to write. It all happens within the browser and making it an ideal partner with any server technology. AngularJS is what like an HTML code had it been designed for applications. HTML is a great declarative language for static documents.

AngularJS is a client-side web application and it is not a single piece in the overall puzzle of building the client-side of a web application. It also handles all of the DOM and AJAX glue code we can once wrote by hand and puts it in a well-defined structure. AJAX makes AngularJS as opinionated about how a CRUD (Create, Read, Update, Delete) application should be built. But while it is opinionated, it also tries to make sure that its opinion is just a starting point we can easily change.

Bootstrap

Booting is the process of starting a computer, specifically it is used to starting its software. The process have chain of stages, in which at each stage a smaller, simpler program loads and then executes the larger, more complicated program of the next stage. It is a process that the computer "pulls itself up by its bootstraps", *i.e.* it improves itself by its own efforts. Booting is also called as chain of events which is start with execution of hardware-based procedures and may it hand-off to firmware and software that is loaded into main memory. Booting also involves a some processes such as performing self-tests, loading configuration settings, loading a BIOS, resident monitors, a hypervisor, an operating system, or utility software.

The **bootstrapping** usually refers to a self-starting process which is supposed to proceed without external input. In computer technology the booting (usually shortened to **booting**) usually refers to the process of loading the basic software into the memory of a computer after power-on or it is a process of reset, especially the operating system which will then take care of loading other software as needed. It contains HYPERTEXT MARKUP LANGUAGE and CASCADING STYLE SHEET-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Though many earlier web frameworks, it concerns itself with front-end development only.

Web API

ASP.NET Web API is a framework which is used to build HTTP services easily that reach a broad range of clients, including browsers and mobile devices. ASP.NET Web API is an ideal platform for building REST full applications on the .NET Framework.

Java Script object Notation (JSON)

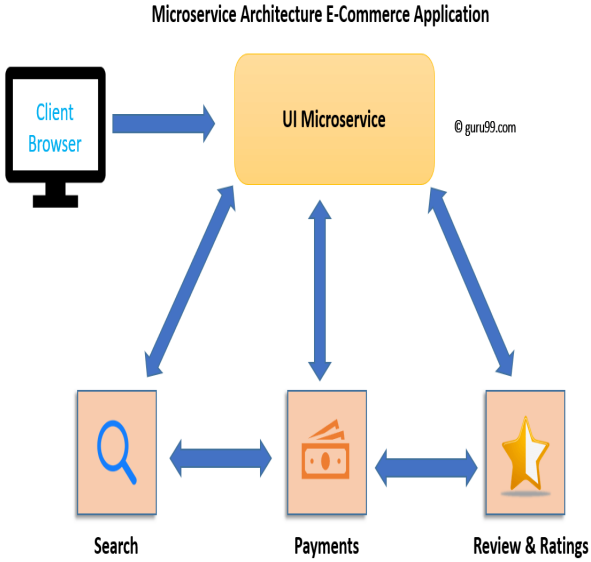
Json is syntax for storing and extracting the data. When exchanging data between browser and server, the data can only be in the form of text. JSON is a text form, and we can convert any JavaScript object into JSON, and send that JSON to the server. Similarly we can also convert any JSON received from the server into JavaScript objects. This the way of data work as JavaScript objects, with no complicated parsing and translations.It is a lightweight data-interchange format.

MicroService

Microservices is a service-oriented architecture pattern where the applications can divide into collection of various smallest independent service units. It is a software engineering approach. The goal of micro service is decomposing an application into single-function modules with well-defined interfaces. These modules can be independently deployed and operated by small teams who own the entire lifecycle of the service within a short time period.

Microservice Architecture

Microservice Architecture is an architectural development style that means it is allow to building an application as a collection of small autonomous services developed for a business domain. Let's take an example of e-commerce application developed with microservice architecture. In this example, each microservice is focused on single business capability. Search, Rating & Review and Payment each have their instance (server) and communicate with each other.



The communication on microservices is a stateless communication where both pair of request and response is independent. Hence, Microservices can communicate effortlessly. In the Microservice Architecture, the Data is grouped. Each Microservice has its separate data store.

- Separate data store for each Microservice
- Keep code of a similar level of maturity.
- Separate build for each Micro service
- Always treat- sever as stateless.

MySQL

MySQL is a database management system

A database is a structured collection of data which may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, we need a database management system such as MySQL Server. Since computers are very good at handling collection of large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

MySQL databases are relational

A relational database stores a data into separate tables does not putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed of access. MySQL software is Open Source. Open Source means which is available for anyone to use and modify the software. The MySQL Database Server is very fast, reliable, scalable, and easy to use.
MySQL Server was originally developed for handle the collection of large databases much faster than existing solutions and has been successfully used in highly demanding production environments. Even it is a under constant development, MySQL Server today offers a rich and useful set of functions. The main reason of my sql is its connectivity, speed, and security makes its server highly suited for accessing databases on the Internet.

MySQL Server works in client/server or embedded systems

The MySQL Database Software. it is a client/server system which consists of a multithreaded SQL server to support different back ends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

We can also use MySQL Server as an embedded multithreaded library that we can link into our application to get a smaller, faster, easier-to-manage standalone product.

EXISTING SYSTEM:

The webservice system considers a client request as a whole request which may lead to heavy network traffic. So it takes more time to response client side. Thus the micro service used to solve this problem. It aid to control the network traffic and delay.

PROPOSED SYSTEM

Microservices are service-oriented architecture (SOA). In which we can decompose an application into different smaller services. Microservices having load balancer to manage heavy load (traffic) over the network. So that application will be light weight streaming over the network.

CONCLUSION

In microservices, we can run a module in an application as multiple instances so that each instance gets http request based on algorithm to handle traffic. There is no any polling mechanism to handle multiple requests because polling mechanism will make heavy traffic. So the client receive their response from server side as soon as possible. Thus the module you can break into individual parts, the more tooling, forethought, and insight what you need to invest to know. We can solve true immutable delivery to reduce configuration drift, and then the Linux containers helps to enable service isolation, rapid delivery, and portability.

REFERENCES

[1]. L. Bass, I. Weber, and L. Zhu, DevOps: A Software Architect's Perspective. Addison-Wesley Professional, 2015.
[2]. M. Fowler and J. Lewis, "Microservices."http://martinfowler.com/articles/microservices.html,March 2014. [Last accessed 3, 2015].

[3]. A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Migrating to cloud-native architectures using microservices: An experience report," in In Proceedings of the 1st International Workshop on Cloud Adoptionand Migration, 11, 2015.
[4]. A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices migration patterns," Tech. Rep. TR-SUTCE-ASE-2015-01, Automated Software Engineering Group, Sharif University of Technology, Tehran,Iran, 2015
[5]. A. Brunnert, A. van Hoorn, F. Willnecker, A. Danciu, W. Hasselbring, C. Heger, N. R. Herbst,P. Jamshidi, R. Jung, J. von Kistowski, A. Koziolek, J. Kroß, S. Spinner, C. V¨ogele, J. Walter, andA. Wert, "Performance-oriented devops: A research agenda," CoRR, vol. abs/1508.04752, 2015.
[6]. J. Humble and D. Farley, Continuous delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional, 2010.
[7]. A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst, "Continuous monitoring of software services: Design and application of the kieker framework," research report, Kiel University, 2009.
[8]. S. Newman, Building Microservices. O'Reilly Media, 2015.
[9]. M. Stine, Migrating to Cloud-Native Application Architectures. O'Reilly Media, 2015.
[10]. V. Vernon, Implementing Domain-driven Design. Addison-Wesley Professional, 2013.
[11]. M. Nygard, Release It!: Design and Deploy Production-Ready Software. Pragmatic Bookshelf, 2007.
[12]. C. Pahl, "Containerization and the paas cloud," IEEE Cloud Computing, 3, 24–31.
[13]. R. Daigneau, Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Addison-Wesley Professional, 2011.
[14]. G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Professional, 2004.
[15]. P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: A systematic review," IEEE Transactions on Cloud Computing, 1, 2013, 142–157.
[16]. B. Henderson-Sellers, J. Ralyt, P. Agerfalk, and M. Rossi, Situational Method Engineering. SpringerVerlag Berlin Heidelberg, 2014.
[17]. P. Jamshidi, C. Pahl, S. Chinenyeze, and X. Liu, "Cloud migration patterns: A multi-cloud architectural perspective," in Proceedings of the 10th International Workshop on Engineering Service-Oriented Applications, 2014.
[18]. A. Ahmad, P. Jamshidi, and C. Pahl, "Classification and comparison of architecture evolution reuse knowledgea systematic review," Journal of Software: Evolution and Process, 26(7), 2014, 654–691.