IMPLEMENTING MICROSERVICES USING CLOUD COMPUTING

[1]A.Sindhu Devi [2] Dr.S.Maruthuperumal

[1] Student ,Department of CSE, BIHER, Chennai.

[2] Assistant Professor , Department of CSE, BIHER, Chennai.

ABSTRACT

The world today has more needs on internet. All our lives basically depend on internet where more information we needed are present at most. The internet used nowadays is used by million of people in their computer around the world. These computers are connected to each other through different platforms and different networks. The services needed for one user is nowadays provided with web services. Nowadays due to the vast usage of cloud computing implementing web services using cloud computing is easiest way for web services.

1. INTRODUCTION

A web service is a unique method for sending messages between client and server applications on the World Wide Web. A web service is nothing but a software module that is designed to accomplish a specific set of tasks. Web services can be easily found and implemented over a network in cloud computing. Cloud computing had made easier way to implement web services.

The web service would be able to provide all the needed functions to the client that is connected to the web service. A web service is a set of protocols and standards that allow data exchange between different applications or systems within the World Wide Web. Web services can be used by software programs written in different programming languages and on different platforms to exchange data through the Internet. In the same way, communication on a computer can be internally processed.

Any software, application, or cloud technology whichever uses a unique or standardized Web protocol (HTTP or HTTPS) to connect, process, and exchange data messages over the Internet-usually uses XML (Extensible Markup Language). It is considered as a Web service.

Web services uses programs developed in different languages to be connected between a client and a server by exchanging data over a web service. A client activates or process a web service by submitting an XML request. Whenever an XML request is send the service is also responded with an XML response.

Web Service Components

XML and HTTP is the most fundamental web service platform. The components of Web services are as follows:

A. SOAP (Simple Object Access Protocol)

SOAP stands for "Simple Object Access Protocol". It is a transport-independent messaging protocol. SOAP is built on sending XML data in the form of

SOAP messages. A document known as an XML document is attached to each message.

Only the structure of an XML document, not the content, follows a pattern. The great thing about web services and SOAP is that everything is sent through HTTP, the standard web protocol.

Every SOAP document requires a root element known as an element. In an XML document, the root element is the first element.

The "envelope" is divided into two halves. The header comes first, followed by the body. Routing data, or information that directs the XML document to which client it should be sent, is contained in the header. The real message will be in the body.

B. UDDI (Universal Description, Search, and Integration)

UDDI is a standard for specifying, publishing and searching online service providers. It provides a specification that helps in hosting the data through web services. UDDI provides a repository where WSDL files can be hosted so that a client application can search the WSDL file to learn about the various actions provided by the web service. As a result, the client application will have full access to UDDI, which acts as the database for all WSDL files.

The UDDI Registry will keep the information needed for online services, such as a telephone directory containing the name, address, and phone number of a certain person so that client applications can find where it is.

C. WSDL (Web Services Description Language)

The client implementing the web service must be aware of the location of the web service. If a web service cannot be found, it cannot be used. Second, the client application must understand what the web service does to implement the correct web service. WSDL, or Web Service Description Language, is used to accomplish this. A WSDL file is another XML-based file that describes what a web service does with a client application. The client application will understand where the web service is located and how to access it using the WSDL document.

1.2 Scope of the Project:

This project is mainly used for secure transaction and reliable. Since we are implementing in microservices on cloud it is the most cost effective. The backup and restoration of data is made easier. The storage capacity of cloud computing is vast since the storage capacity of cloud computing is higher.

2. LITERATURE SURVEY

TITLE 1: Application of microservice architecture in cloud environment project development

AUTHOR: Ling Zheng*, and Bo Wei

With the development of the information age, business systems are becoming more and more complex. System development and maintenance are facing huge challenges. In response to this problem, a unified application development platform based on the microservice architecture is proposed. Compared with the traditional single-architecture architecture, the microservices

architecture can split a large and complex application system into a series of service modules that can be independently developed, tested, deployed, operated, and upgraded. This enables the application expansion and application reduction for a large number of Internet companies. Developing complexity and implementing agile development provide more effective methods. This article through a detailed case analysis - the development of the cloud platform system, describes the specific application of the microservice architecture in the actual project development, and discusses the advantages of the traditional single architecture model for the service architecture to build the system. Through research and analysis, it is concluded that the microservice architecture has certain guiding significance for solving problems that may be encountered in enterprise-level applications.

TITLE 2: Microservices: architecture, container, and challenges

AUTHOR: Guozhi Liu , Bi Huang , Zhihong Liang, Minmin Qin , Hua Zhou, Zhang Li

Microservices are emerging as a new computing paradigm which is a suitable complementation of cloud computing. Microservices will decompose traditional monolithic applications into a set of fine-grained services, which can be independently developed, tested, and deployed. However, there are many challenges of microservices. This paper provides a comprehensive overview of microservices. More specifically, firstly, we systematically compare traditional monolithic architecture, service-oriented architecture (SOA), and microservices architecture. Secondly, we give an overview of the container technology. Finally, we outline the technical challenges of microservices, such as performance, debugging and data consistency.

3. EXISTING SYSTEM

A microservices architecture is one type of distributed system, since it decomposes an application into separate components or “services”. For example, a microservice architecture may have services that correspond to business features (payments, users, products, etc.)

3.1 Disadvantage of Existing System:

The legacy Java monolith application is not scalable, and it is one fat JAR which has to be deployed every time.

Due to the size of the application the start time and application initialization time is adversely affected.

Furthermore, additional memory pressure among high other system requirements is forcing the team to think about a strategy to control these issues in the long run.

4. PROPOSED SYSTEM

In order to thrive in today’s volatile, uncertain, complex and ambiguous world, businesses must be nimble, agile and innovate faster. Moreover, since modern businesses are powered by software, IT must deliver that software rapidly, frequently and reliably - as measured by the DORA metrics.

Rapid, frequent, reliable and sustainable delivery requires the success triangle, a combination of three things:

Process - DevOps as defined by the DevOps handbook

Organization - a network of small, loosely coupled, cross-functional teams

Architecture - a loosely coupled, testable and deployable architecture

Teams work independently most of the time to produce a stream of small, frequent changes that are tested by an automated deployment pipeline and deployed into production.

4.1 Advantages of Proposed System:

Microservices architecture allows cross-functional teams to develop, test, problem-solve, deploy, and update services independently, which leads to faster deployment and troubleshooting turnaround times.

Microservices are self-contained, independent deployment module. The cost of scaling is comparatively less than the monolithic architecture.

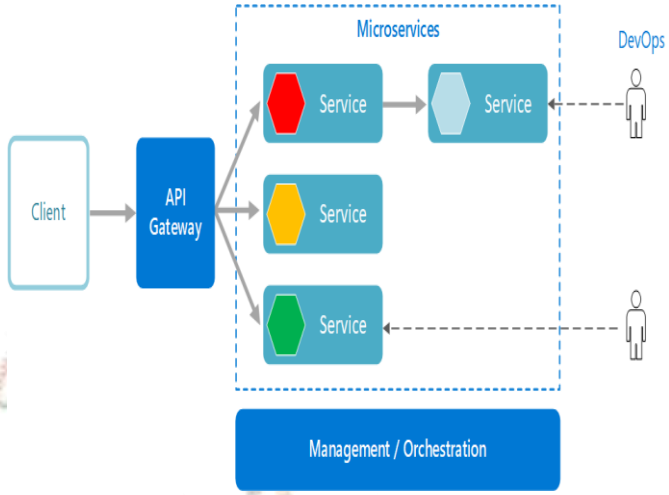
Microservices are independently manageable services. It can enable more and more services as the need arises.

It is possible to change or upgrade each service individually rather than upgrading in the entire application.

Microservices allows us to develop an application which is organic (an application which latterly upgrades by adding more functions or modules) in nature.

It enables event streaming technology to enable easy integration in comparison to heavyweight interposes communication.

5. FLOWCHART DIAGRAM



6. MODULES DESCRIPTION

6.1 List of Modules:

Domain Driven Design – Enabled by implementing Microservices based architecture – Modular routines for each of the functionalities – User, Product, Cart, Order and Reports

Cloud First – Building a highly scalable, resilient, cost effective infrastructure using cloud

Microservices architecture – Aligning each service to have it's own database

Scalable solution – Implementation of containerized application using Dockers and ECS to enable horizontal scaling

Secured Application – Enabled by API Gateway, Lambda Authorizers, Security group & ACL's

Caching and Performance– Caching enabled by Redis cache for faster search

Faster to Market – DevOps integration with CI/CD and Code Pipeline

Event driven integration – Communication between decoupled services

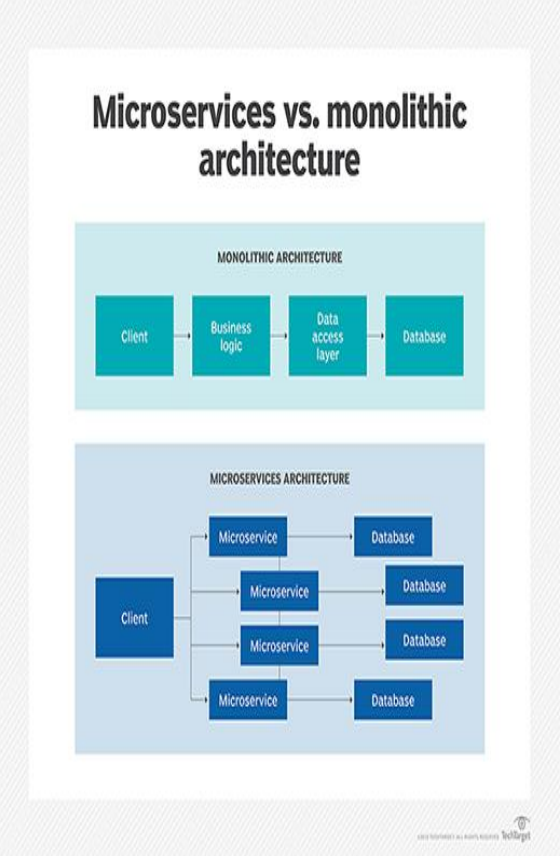
Operational excellence – Elastic search and Kibana integration to view logs and errors

6.2 Modules Description

1. Microservices

Microservices make up the foundation of a microservices architecture. The term illustrates the method of breaking down an application into generally small, self-contained services, written in any language, that communicate over lightweight protocols. With independent microservices, software teams can implement iterative development processes, as well as create and upgrade features flexibly.

Teams need to decide the proper size for microservices, keeping in mind that an overly granular collection of too-segmented services creates high overhead and management needs. Developers should thoroughly decouple services in order to minimize dependencies between them and promote service autonomy. And use lightweight communication mechanisms like REST and HTTP.



The difference between monolithic and microservices design

2. Containers

Containers are units of software that package services and their dependencies, maintaining a consistent unit through development, test and production. Containers are not necessary for microservices deployment, nor are microservices needed to use containers. However, containers can potentially improve deployment time and app efficiency in a microservices architecture more so than other deployment techniques, such as VMs.

The major difference between containers and VMs is that containers can share an OS and middleware components, whereas each VM includes an entire OS for its use. By eliminating the need for each VM to provide an individual OS for each small service, organizations can run a larger collection of microservices on a single server.

The other advantage of containers is their ability to deploy on-demand without negatively impacting application performance. Developers can also replace, move and replicate them with fairly minimal effort. The independence and consistency of containers is a critical part of scaling certain pieces of a microservices architecture -- according to workloads -- rather than the whole application. It also supports the ability to redeploy microservices in a failure.

Docker, which started as an open-source platform for container management, is one of the most recognizable providers in the container space. However, Docker's success caused a large tooling ecosystem to evolve around it, spawning popular container orchestrators like Kubernetes.

3. Service mesh

In a microservices architecture, the service mesh creates a dynamic messaging layer to facilitate communication. It abstracts the communication layer, which means developers don't have to code in inter-process communication when they create the application.

Service mesh tooling typically uses a sidecar pattern, which creates a proxy container that sits beside the containers that have either a single microservice instance or a collection of services. The sidecar routes traffic to and from the container, and directs communication with other sidecar proxies to maintain service connections.

Two of today's most popular service mesh options are Istio, a project that Google launched alongside IBM and Lyft, and Linked, a project under the Cloud Native Computing Foundation. Both Istio and Linked are tied to Kubernetes, though they feature notable differences in areas such as support for non-container environments and traffic control capabilities.

4. Service discovery

Whether it's due to changing workloads, updates or failure mitigation, the number of microservice instances active in a deployment fluctuate. It can be difficult to keep track of large numbers of services that reside in distributed network locations throughout the application architecture.

Service discovery helps service instances adapt in a changing deployment, and distribute load between the microservices accordingly. The service discovery component is made up of three parts:

A service provider that originates service instances over a network;

A service registry, which acts as a database that stores the location of available service instances; and

A service consumer, which retrieves the location of a service instance from the registry, and then communicates with that instance.

Service discovery also consists of two major discovery patterns:

A client-side discovery pattern searches the service registry to locate a service provider, selects an appropriate and available service instance using a load balancing algorithm, and then makes a request.

In a server-side discovery pattern, the router searches the service registry and, once the applicable service instance is found, forwards the request accordingly.

Data residing in the service registry should always be current, so that related services can find their related service instances at runtime. If the service registry is down, it will hinder all the services, so enterprises typically use a distributed database, such as Apache ZooKeeper, to avoid regular failures.

5. API gateway

Another important component of a microservices architecture is an API gateway. API gateways are vital for communication in a distributed architecture, as they can create the main layer of abstraction between microservices and the outside clients. The API gateway will handle a large amount of the communication and administrative roles that typically occur within a monolithic application, allowing the microservices to remain lightweight. They can also authenticate, cache and manage requests, as well as monitor messaging and perform load balancing as necessary.

Additionally, an API gateway can speed up communication between microservices and clients by standardizing messaging protocols translation and freeing both the client and the service from the task of translating requests written in unfamiliar formats. Most API gateways will also provide built-in security features, which means they can manage authorization and authentication for microservices, as well as track incoming and outgoing requests to identify any possible intrusions.

There are a wide array of API gateway options on the market to choose from, both from proprietary cloud platform providers like Amazon and Microsoft and open source providers such as Kong and Tyk.

7. CONCLUSION

This project is used to implement microservices using cloud computing for easy accessible and reliable transaction of messages. The cost of this project is less. The security on transaction is also high when compared to the existing system.

8. REFERENCES

1. Suyel Namasudra, Pinki Roy, Balamurugan Balusamy, Pandi Vijayakumar, "Data accessing based on the popularity value for cloud computing", *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp.1-6, 2017.

2. Tomas Harach, Petr Simonik, Adela Vrtkova, Tomas Mrovec, Tomas Klein, Joy Jason Ligori, Martin Koreny, "Novel Method for Determining Internal Combustion Engine Dysfunctions on Platform as a Service", *Sensors*, vol.23, no.1, pp.477, 2023.

3. C. Wang, Q. Wang, K. Ren, W. Lou, “Privacy-preserving public auditing for data storage security in cloud computing”, INFOCOM 2010, pp. 1-9, 2010.

4. Alwi Maulana, Pradana Ananda Raharja, "Design and Testing on Migration of Remiss-Supply in Banking System to Microservice Architecture", *2022 IEEE International Conference on Communication, Networks and Satellite (COMNETSAT)*, pp.168-173, 2022.

5. V. S. Devi Priya, S. Sibi Chakkaravarthy, "Containerized cloud-based honeypot deception for tracking attackers", *Scientific Reports*, vol.13, no.1, 2023.

6. Xiaoming Yu, Wenjun Wu, Yangzhou Wang, "Dependable Workflow Scheduling for Microservice QoS Based on Deep Q-Network", *2022 IEEE International Conference on Web Services (ICWS)*, pp.240-245, 2022.

7.C.Erway, A.Kupcu, C.Papamanthou, and R.Tamassia, ―Dynamic provable data possession,‖ in Proc. of CCS'09. Chicago, IL, USA: ACM, 2009.

8. Towards Secure and Dependable Storage Services in Cloud Computing Cong Wang, Student Member, IEEE, Qian Wang, Student Member, IEEE, Kui Ren, Member, IEEE, Ning Cao, Student Member, IEEE, and Wenjing Lou, Senior Member, IEEE- 2011.

9. Addressing cloud computing security issues, Future generation computer systems (2011).

10. P. Xu, H. Chen, D. Zou, and H. Jin, "Fine-grained and heterogeneous proxy re-encryption for secure cloud storage," Chin. Sci. Bull.,vol. 59, no. 32, pp. 4201–4209, 2014.

11. E.-J. Yoon, Y. Choi, and C. Kim, "New ID-based proxy signature scheme with message recovery," in Grid and Pervasive Computing(Lecture Notes in Computer Science), vol. 7861. Berlin, Germany: Springer-Verlag, 2013, pp. 945– 951.