ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.9

A Revenue Management System based on Microservice Architecture

Asma
M.Tech Student
Department of Information Science
Rashtreeya Vidyalaya College of Engineering, Bangalore, Karnataka, India

Abstract:
The main objective of this paper is to evaluate the developments of revenue management systems over the past decades and discuss the new prospects of research. The paper first introduces the current Revenue Management system and highlights its main drawbacks. The paper summarizes on how the adaptation of new technologies like microservices [2] architecture over the traditional monolithic architecture has an impact on the development process. The deployment of the software on different platforms also plays an important role; the paper also discusses the advantages of deployment of the software on cloud.

I. INTRODUCTION.

The cloud markets, communications and media are undergoing vital transformations which are very challenging to the service providers as they need new strategies. Service providers face vital business challenges on numerous fronts, like most importantly the increased competition locally and globally, erosion of client loyalty attributable to commoditization of services and decreasing margins for ancient service offerings. To contend during this new business environment, service providers should rework their businesses to satisfy the new market demands. A powerful Revenue management system can be of great help to the service providers.

Like, optimize price of customers, increase Profit, enable business agility etc. A good revenue management system will offer complete end to end solution for supporting key business that includes generation, managing and assurance of revenue, capture and collect. With its broad business processes, Revenue management impacts the means that producers introduce new merchandise and services, manage client accounts, payments, and dues and manage overall revenue integrity.

There are some issues in the current revenue management system as follows:

• The systems depend on the centralized architecture. Each system cannot be shared between the IT infrastructures, and data conversion among the systems can only be achieved through the data sharing. The systems communicate with each other in different ways: by point-to-point, via the business bus, etc.
• Each application system is a monolithic structure. With the continuous business expansion, the increase in the number of customers and business demands, the advantages of this monolithic structure have been gradually being lost, and it is facing more and more challenges.
• As the applications become more functional and the code gets more complex, the period for the building and deploying them grows accordingly.
• With the growth of the data and the business, the vertical scaling of the system becomes more and more difficult while the cost is getting much higher.

II. OVERVIEW OF CURRENT REVENUE MANAGEMENT SYSTEM

The current system includes three kinds of applications: basic application, professional application and decision-making application, which support the marketing business operations.

A. Main Drawbacks of Current Revenue Management System

There are four main drawbacks described as follows for the current Revenue Management system:

1) Infrastructure with centralized data storage and centralized architecture The application system has the infrastructure of the centralized data storage and the centralized architecture by using mainframe as database server, X86 server as application server, Linux/UNIX as operating system, Oracle database and WebLogic middleware as basic software, to build the 'silo' systems with the SOA concept. Each system cannot be shared among the IT infrastructures, and the data sharing can only be accomplished through the data conversion among the systems. The systems communicate with each other in different ways: by point-to-point, via the business bus, etc., with lower efficiency and reliability. Once it happens in lack of IT infrastructure resources, the only possible way to improve the processing capability is by upgrading the capacity of the server; however, due to the centralized architecture of the data storage, the processing capability has the ceiling where the system is unable to guarantee the stable operation in the event of business processing fluctuations.

2) Application system with monolithic structure Each application system has the monolithic structure. With the continuous expansion of the Revenue Management, the increasing demands and the increasing number of customers, the advantages of this monolithic structure have been gradually lost, and it is facing more and more challenges. With the expansion of Revenue Management and online business handling, there are lot of issues to be solved, all caused by the increase in the number of customers; on the other hand, more and more developers and testers are joining the development teams where the code base is also rapidly expanding, in the case, the maintainability and flexibility of the monolithic structure are reduced while the costs of testing, building, and maintenance are increased significantly.

3) Low efficiency of building and deploying the entire system Meanwhile, building and deploying the entire system will correspondingly take longer time when the functions of the whole business applications are increasing and the code is becoming more complex. In the current stable deployment pipeline, any minor change of the monolithic application or code submission will trigger the deployment pipeline: compiling the code of the entire application, running the unit tests, checking code, building the deployment packages, function validation, etc., which means that the feedback cycle of the pipeline will become much longer and the efficiency of building the system will become lower.

4) Difficult system scaling With the growth of the revenue management and generated data, the vertical scaling of the entire information system will become more and more complex, which cost is higher. If the horizontal scaling is considered, the cost is also very high as all the code runs on the same process on the server. For example, if the functions of the application system are memory-intensive, such as the statistical query and decision analysis, required to cache large amounts of data, while the other functions are CPU-intensive, such as electricity calculation, requiring huge processing power; then in the horizontal scaling each time, the server running the application will have to have both sufficient memory and strong CPU power to meet the demands. As a result, the overall cost of the infrastructure can be very high, given that each server provides the resources required by the application; furthermore, if some nodes need to maintain the status, i.e., keeping the session information of user's logging in, it will be more difficult for the scaling.

III SYSTEM MODEL

The life cycle of a revenue management system includes the following four components.

Revenue Generation helps in delivering services to customers, which is priced in a manner to optimize the user, service providers and its shareholders. This helps to gain maximum customer and partner value through agile service delivery and sophisticated account management. Service providers can respond quickly to the market demands with real-time access to customer data and the ability to make innovative offerings, and they can ensure to retain their most profitable customers.
Revenue capture helps in maximizing the market shares using flexible credit card control and competitive pricing models to enable any service for any subscriber. The transactions are captured, pre-authorized, rated and balances are updated as the services are consumed. The risk of revenue leakage can be reduced and the customer satisfaction can be increased with the help of real time interactions.
Revenue Collection makes sure about the appropriate generation of all the bills and the invoices, and also the money collected from the correct debtors. It is then posted to accounts receivables and general ledger account, simultaneously handles payment terms, disputes and settlements. A real-time and exact view of revenue gives insight to respond to market changes.
Revenue analysis makes sure that the transactions are conducted with the fullest possible control, completeness and integrity. It helps in maximizing revenue and minimize loss occurred from fraud and revenue leakage by providing real-time verification, analysis, reporting and control of all events and actions. With the integration of the above four revenue management lifecycle components with the existing business critical systems provides competitive advantage for service providers.

IV. METHODOLOGY

Here we discuss the building of a Revenue Management system based on cloud computing and microservices architecture technologies, suitable for the telecos, and to solve the issues of the current information systems of the centralized infrastructure and monolithic application architecture. The main technical characteristics include as follows:
• Use the distributed infrastructure and make full utilisation of the flexible management and strong horizontal scalability of IT resources enabled by cloud computing technology.
• Utilize the service platform to provide a number of cloud service capabilities for the upper applications, such as flexible scaling, one-click deployment, gray distribution, fault self-healing, and full link monitoring.
• Use the microservices architecture which has the advantages of componentization, decentralization, independent deployment etc.

A. Overview of Technologies

Cloud computing is an increase, use, and delivery model of Internet-based services that typically involves providing dynamically scalable and often virtualized resources over the Internet. Cloud computing is a pay-as-you-use model that provides available, convenient, on-demand network access to a configurable pool of computing resources (such as networks, servers, storage, applications, services), which can be provided quickly, with minimal administrative effort or little interaction with the service provider. Cloud computing can be considered to include the three levels of services: Infrastructure as a Service (laaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [2] [3] [4]. Microservices architecture is an architectural model, which monitors that a single application is divided into a group of small services, services coordinate with each other to provide users with the business value. Each service runs in its own independent process, services communicate with each other by using a lightweight communication mechanism. Each service is built around a particular business, and can be deployed independently in the production environment. The microservices architecture breaks down complex applications into small, modular, specific, low-coupling, and highly autonomous sets of services, each of which is a small application, by analyzing and modeling specific business domains. The main characteristics of microservices architecture include: atomic services, de-centralized service discovery, services as components, technical diversity, business data independence, and infrastructure automation [5] [6] [7] [8]:

1) Atomic services According to the "Single Responsibility Principle" reinforced by Robert C. Martin in one of the object-oriented principles, the more the function is less dependent on other functions, the stronger the cohesion is, to match the standard of "the high cohesion and the loose coupling".
2) Services as components Microservice is small enough and has a single function, and can be independently packaged, upgraded, deployed, roll backed and scaled flexibly, independent of the other microservice to achieve local autonomy. Another advantage of using microservices as a component is to define a clear, language-independent, platform-independent interface among components. However,

the distributed calls are more time-consuming than in-process calls and rely on network stability and reliability.

3) De-centric service discovery all microservices nodes are both control nodes and controlled nodes, receiving and sending messages in real time. In this mode, the heavy-duty enterprise service bus (ESB) is not required. The functionality provided by the message middleware is merged into the various microservices nodes, effectively eliminating the ESB itself as a single failure point of performance, which is one of the main differences between the microservices and SOA architecture.

4) Technology diversity In the microservices architecture, it is allowed to select the appropriate technical solutions to target at solving the specific business issues, more easily selecting new technologies or solutions for the system.

5) Business data independence The service manages the data related to service in the microservices architecture, whose benefits are to provide service data interface integration with the development of the business, rather than integration in the form of database; to choose a more suitable tool to manage or migrate business data.

6) Infrastructure automation The microservices architecture divides the application itself into smaller services, each of which is a separate deployment unit. With the generality of cloud technology, the complexity of deployment and maintenance is significantly reduced. The system resources required can be quickly created and the cost of the application deployment is reduced. The microservices architecture designs a complex application into multiple services, each of which is a standalone, deployable service unit. These features ensure that application software evolves as the business evolves, continually adapting the architecture of the software, discarding unnecessary services, upgrading the services required, and optimizing the architecture with the right technology or tools to keep it up-to-date, keeping the status of continuous evolution.

B. Overall Framework

As shown in Figure 1, we will take new framework to build the Revenue Management application system, in which the IaaS part includes the cloud operating system and its managed resources; the PaaS part includes the platform services of the data processing, information integration, application building, and cloud service center; and the SaaS part includes the applications of the electric power marketing business refactored with the microservices architecture technology [9] [10] [11].

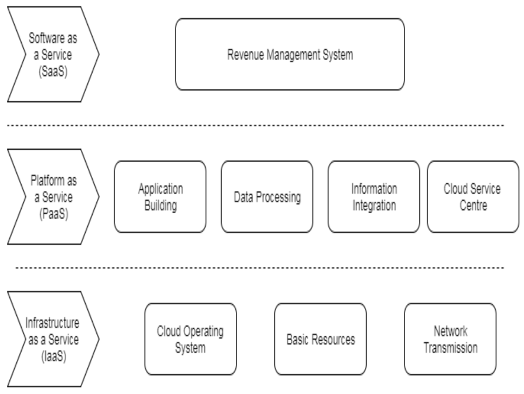


Figure.1. New Framework

C. IaaS Part of Application System with Distributed Infrastructure

The IaaS part of the application system includes the basic resources and network transmission. In the basic resources, the cloud operation system manages the hardware and software resources.

1) Cloud operating system The cloud computing system is mainly responsible for the unanimous allocation of basic resources such as computing, storage, and network, which adopts the PC server as virtual machine, container or physical machine to participate in computing resource allocation and provides the services such as distributed storage and network virtualization.

2) Hardware and software resources The hardware and software resources comprise of server resources, storage resources, network resources and platform software resources. The computing resources mainly comprise the PC servers. By the virtualization of these computing resources, they can be flexibly scaled and efficiently.

3) Network transmission The network transmission refers to the interconnected networks, divided into information intranet, information extranet and Internet access. The information intranet and extranet include the backbone network and the access network, and have two forms of the wired and the wireless.

D. PaaS Part of Application System Providing Cloud Services

The PaaS part of the application system consists of four platform service components: data processing,, application development, information integration and cloud service center.

1) Data processing Data processing comprises distributed relational database, big data platform, unified data access, unified data analysis and other components, providing data storage, computing, analyzing and data replication services.

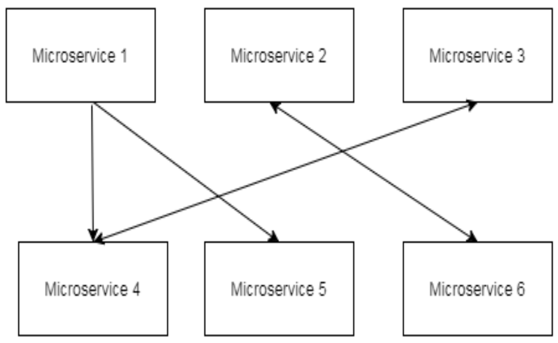
2) Information integration It includes components like distributed service bus, distributed message queue, unified authority, and unified business process and so on, which can provide high performance and reliable service bus to meet the future large-scale applications of electric power marketing business scenarios.

3) Application development It comprises integrated development environment, test environment, application assembly and automated deployment, to provide the development, testing and deployment environment to build the applications for the electric power marketing business.

4) Cloud service center It comprises deployment configuration, monitoring and cloud console to present the services of the common components in the platform to the user in the similar layouts. It can also orchestrate and collect the different services of the platform to create more advanced cloud service capabilities. It can collect operational statistics of the running services for complete analysis to ensure the quality of the cloud services. Through the above platform service components, combined with the cloud operating system in the cloud infrastructure, the service platform provides for the applications with several cloud service capabilities, such as one-click deployment, flexible scalability, fault self-healing, gray release, full-link monitoring etc.

E. SaaS Part of Application System with Microservices Architecture

Based on the above-mentioned IaaS and PaaS parts, the system can provide basic resource management and common platform services to build an integrated electric power marketing business application, realizing the SaaS part of the system. The core idea of microservices design is to form a system with a series of independent services which is deployed separately and has no impact on the other.



The main features of the microservices architecture are as follows in electric power marketing business applications:

• **Simpler development:** one microservice is developed focusing only for one specific function of the business and the complexity is reduced for the development

• **Flexible choice of technical framework:** the choice of technical framework can be varied and unrestricted

• **Independent services:** the deployment of the microservices deployment does not depend on the other ones and its updating does not affect the other microservices

• **Independent on-demand scaling:** the decoupled micro services can be flexibly scaled out based on their own business requirements and characteristics

• **High availability:** the availability of microservice increases as a result of its specific business functions distributed on multiple nodes and the switching capability increases substantially after the failure happens.

V. RELATED WORK

Docker[3] is a platform which that packages applications in containers. Docker is of very great help when it comes to collaborating the code with co-workers. These containers are bit like virtual machines, but rather than creating a complete virtual OS, it allows the same Linux kernel to be used by the applications. The advantages of using containers are many. Like, it allows the developer to package up an application along with all needed parts such as libraries and other dependencies and ships it all as one package. By doing so, the developer can run the application on any other Linux machine without any customized setting. Also the applications running on a container could be twice as fast as one in a virtual machine.

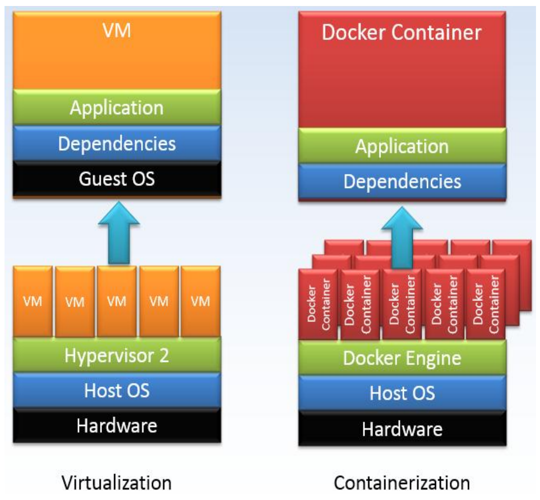


Figure.1. A comparison between virtualization and containerization.

The above figure explains about the virtualization in the VM and a Container.

VI. CONCLUSION

The paper proposes the method of building the electric power marketing business application system based on the cloud computing and microservices architecture technologies to solve the issues of the current system with the concentrated infrastructure and the monolithic architecture which are difficult to support the business processing in very large scale and the evolution of the electric power marketing business. On the technical side, it puts forward the overall framework of the electric power marketing business application system based on the cloud computing and microservices architecture technologies with the following characteristics:

• The IaaS part of the application system has the distributed infrastructures with the IT resource elastic management and powerful horizontal scalability enabled by the cloud computing technology;

• The PaaS part of the application system is the service platform with the components of the data processing, information integration, application building, and cloud service center to provide the several cloud services, such as one-click deployment, flexible scaling, fault self-healing, gray distribution, full-link monitoring and other cloud service capabilities

• The SaaS part of the application system has the microservices architecture with several advantages, such as servitization, componentization, decentralization, independent deployment etc.

VII. REFERENCES

[1]. Xiang Wu, Hong Ouyang, Lijuan Dong, Hongli Van, "Research and application of marketing business application system of State Grid", Electric Power Information Technology, 2011(2), pp. 49-54.

[2]. Thomas ErI, Zaigham Mahmood, Ricardo Puttini, "Cloud computing concepts, technology & architecture", Pearson Education, 2013.

[3]. Kai Hwang, Geoffrey C. Fox, Jack J. Dongarra, "Distributed and cloud computing: from parallel processing to the Internet of Things", Publishing House of Morgan Kaufumann, 2011.

[4]. Jiongjiong Gu, "Technologies and practice of cloud computing architecture", Publishing House of Tsinghua University, 2016.

[5]. Sam Newman, "Building microservices", O'Reilly Media, 2015.

[6]. J. Thones, Microservices. Software IEEE, vol. 24, no. 3, pp. 116-116, 2015.

[7]. J. Stubbs, W. Moreira and R. Dooley, "Distributed Systems of Microservices Using Docker and Serfnode", Science Gateways (IWSG) 2015 7th International Workshop.

[8]. C. Boettiger, "An introduction to Docker for reproducible research [J]", ACM SIGOPS Operating Systems Review, vol. 49, pp. 71-79, 2015.

[9]. Armin Balalaie, Abbas Heydarnoori and Pooyan Jamshidi, "Migrating to Cloud- Native Architectures Using

Microservices: An Experience Report", Advances in Service-Oriented and Cloud Computing, vol. 567, pp. 201-215, 2015.

[10]. W. Stefan, T. Eddy and J. Wouter, "Comparing PaaS offerings in light of SaaS development", Computing, vol. 96, no. 8, pp. 669-724, 2014.

[11]. D. Bernstein. Containers and cloud: From lxc to docker to kubernetes. IEEE Cloud Computing, (3): 81--84, 2014.

[12].J. I. Fernández-Villamor, C. Iglesias, and M. Garijo, "Microservices: Lightweight service descriptions for REST architectural style," in Proc. 2nd Int. Conf. Agents ArtifIntell, 2010, pp. 186-189.