Architecture Design for Rule-based Clinical Decision Support System on Cloud Computing

Ali Osman Cibikdiken¹,*, Mucahit Buyukyilmaz² and Levent Korkmaz³

¹Necmettin Erbakan University, Dep. of Computer Eng., Konya, Turkey
²Necmettin Erbakan University, ACDAL, Konya, Turkey
³Necmettin Erbakan University, Medicine Faculty, Konya, Turkey

Keywords: Clinical decision support systems, Rule-based system, Cloud computing.

Abstract. In recent years, it is getting harder to make a diagnosis and follow up treatments because of the increment at a knowledge of medicine and discovery of new diseases. It is almost impossible to follow new diseases, treatments, and developments at the science of medicine, at the same time, while examining daily polyclinic patients. In this study, a system architecture for patient follow-up has been described which is also supporting doctors’ decisions with rule-based decision-making algorithms. The proposed architecture is designed to work on the cloud computing. In addition to rule-based analysis, the necessary components for statistical and machine learning are also introduced to the architecture.

Introduction

Nowadays, applications of cloud computing concepts have increased in different scientific and business areas. Healthcare domain is one of these fields. There are many healthcare applications have been provided by IT companies. Also, many research articles in healthcare cloud computing solutions have been published [1]. Healthcare organizations have many reasons for using cloud services. One of the most important of these reasons is that they want to work with patient information in a fast, safe and efficient manner.

There are doctors, physicians, specialists, patients, pharmaceutical companies, and IT solutions firms are in healthcare domain. Electronic Health Record (EHR) systems save information when the patient is admitted to a hospital. The diagnostic information that collects by physicians after diagnosing the patient is stored EHR. Doctors can get advice by sharing the health information in EHR with consulting experts. The cloud infrastructure can provide several advantages in the healthcare domain through systems such as health information management system, laboratory information system, radiology information system, pathology information system, etc. Health organizations can prefer to use public or private cloud systems [2].

Clinical Decision Support System (CDSS) included tools and knowledge to aid medical personnel in performing clinical decisions. CDSS presents adhering to clinical guidelines, preventing drug-allergy interactions, providing standard order sets, and linking to clinical knowledge resources. Rule-based services can be a fundamental component in CDSS. This component includes some services: suggestion services for treatments and medications, overlapping prescriptions, drug-drug interactions, age contraindications, drug-disease interactions, alert registration service, drug allergies, order sets, clinical pathway order sets, link to external knowledge resources, medication dose calculator, etc. [3, 4]. Cloud infrastructure provides to collect the health information from different data sources in a different hospital as input data for the rule-based component.

In this study, a system architecture has been introduced a rule-based clinical decision support system on cloud infrastructure.
**Used Technologies**

Proposed architecture in this study, has been entirely built top of open-source technologies which can work on distributed on cloud environments. All components of the architecture can be multiple instances. Linux container offered to manage this concurrency and scalability. As a result of multiple running instances, a Load Balancer offered in front of services. By this way, all components can scale up or down to service small clinics to large hospitals without service interruption. A Container Orchestration service has been offered to manage containers to scale by the load. Kubernetes or Docker Swarm can be used as a container orchestrator. Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications [5]. Docker Swarm presents native clustering functionality for Docker containers [6].

The volume of healthcare data includes different and variable formats. Also, these data is growing rapidly. To store and process these data is a necessary and challenging issue. Generally, relational databases are not able to handle the massive them. NoSQL databases provide better performance for storing and analyzing large-scale data [7]. NoSQL database has been recommended in this study to handle unstructured data. Schema-less NoSQL database offered because of data format and needs can change very often at a hospital. By storing data as a JSON, data can be stored more flexible than relational databases and avoid unwanted links between tables.

Couchbase Server is a simple, fast, elastic NoSQL database, optimized for the data management needs of interactive web applications [8]. Couchbase Server makes it easy to optimally match resources to the changing needs of an application by automatically distributing data and I/O across commodity servers or virtual machines. Couchbase is document-oriented, schemaless NoSQL database which can store and query unstructured data with low latency. Also, Couchbase can be used as Memcached for key-value needs.

Memory object caching system has been used in system architecture to store authentication/authorization data. Also, cache data shared between components will be stored on this memory object caching system. Couchbase used at our architecture as Memcached.

A message queue service has been recommended to connect different services and components. RabbitMQ offered as Message Queue at our architecture. RabbitMQ is open source messaging broker - an intermediary for messaging. It gives applications a common platform to send and receive messages, and messages a safe place to live until received [9]. RabbitMQ can be run as a cluster and supports multiple queues and routing options.

Object storage has been offered in this study to store all files. Radiography images, voice files, and patient reports will be stored as multiple replicas on object storage. Object storage also will be multiple instances to handle the workload. OpenStack Swift offered as Object Storage. Swift is a highly available, distributed, eventually consistent object/blob store. Swift can be used to store lots of data efficiently, safely, and cheaply [10]. Swift is designed to store files, videos, analytics data, web content, backups, images, virtual machine snapshots and other unstructured data at large scale with high availability and 12 nines of durability.

HAProxy offered as a LoadBalancer at our architecture. HAProxy is a very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications [11]. HAProxy will handle connections between services and redirect to cluster as round-robin. HAProxy also will be used to serve frontend and backend services to the user.

Python and R libraries which languages are widespread for statistical analysis can be used in data analytics component and machine learning component in the architecture. R is a system for statistical computation and graphics. It provides, among other things, a programming language, high-level graphics, interfaces to other languages and debugging facilities [12].
Proposed System Architecture

Proposed system architecture on this study designed as components. There are five components in the system architecture; frontend, backend, rule engine, data analytic engine and machine learning engine (Figure 1).

Frontend component is designed to use by all departments and all officers at the hospital such as nurses, secretaries, physicians, and doctors. Frontend module will provide to users a user-friendly responsive HTML5 interface which can be used with mobile, tablets or PC. Compiled and minified codes will serve behind a web server. Load Balancer will handle a request and redirect it to a web server. Frontend module will be simple and easy to use. Frontend module will reach database over Rest API. OAuth2 [13] will be used for user authentications and API security.

Backend component in the system architecture is designed to combine three different services. This component has been planned to provide a Rest API for the frontend. The first service is API service that will work standalone and provide API only for users. The second service will provide another Rest API for third-party external applications. HMAC or OAuth2 can be used for authorization and security. The third service of the module is WebSocket. WebSocket service will be used to create a two-way connection between user’s browser and backend module. Backend component and related sub-services directly connect to Couchbase NoSQL database over a load balancer. All patients data will be stored on this NoSQL database, and read-write operations will be run over on it. User authentication and authorization operations, session data for OAuth2 will be stored on an in-memory object caching system. Backend component has to use the messaging queue to handle asynchronous operations. By using a messaging queue API service and WebSocket service will be connected to rule-manager service and connection will be over this queue. There are several open source messaging queue project. Any multichannel supported messaging queue can be used in this manner.

An object storage has been recommended for this study to store imaging files such as MRI, tomography, etc. By using an object storage, this files can be accessible for all users who can use the application. Recommended object storage can be work as multiple instances. By this way data security and redundancy provided.

Machine learning component has been designed to analyze data which are stored in NoSQL database. A library will be developed to help users to get data from a database and analyze it. Personal data and healthcare data which can allow identifying patient will be inaccessible in this library. The
library will include deep learning algorithms, supervised and unsupervised machine learning algorithms. The results of the analysis will be imported as a rule to the system.

Data analytics component has designed to make statistical analysis with data are stored on NoSQL database. A library will develop and provide an interface for R Language and Python to query and analyze data.

**Proposed Rule-Based Clinical Decision Support System**

Today’s most of the practically applied decision support systems are designed on rule-based approach. The advanced expert system technologies use AI techniques. These techniques significantly enhance the expert system capabilities. Rule-based CDSS is divided into three distinct parts: inputs, processes, and outputs [14]. It is enclosed by an environment and often include a feedback mechanism. Also, a human decision-maker can be considered part of the system as an expert. Several techniques improving the original ideas of rule-based expert systems are described in [15, 16].

In this study, a rule-based clinical decision support system architecture proposed to help doctors’ decisions. In this way, activated rules will be work with read-or-write action and will show suggestions very fast. The syntax should be developed to define actions when creating rules.

The architecture of the proposed rule-based system has three layers. These layers are data, rule, and knowledge. The data layer includes data sources. Data sources use Hospital Information System (HIS), guidelines, drug interactions, etc. The rule layer provides the engine that works the rules for input and produces output. The knowledge layer has decision rules, expert decisions, and clinical recommendations. Also, this layer interacts with machine learning engine in the system (Figure 2).

![Figure 2. The architecture of rule-based system.](image)

The design of rule layer in the proposed CDSS is based on two main components: rule manager and rule worker.

Rule manager has been designed to run rule-set include which rules submitted when an operation applied to a patient and redirect results to backend module. When a read or write operation executed for a patient, backend module will find required rules for related operation. This rule set will be sent to Rule Manager module over the messaging queue. Rule Manager gets rule-set and split them to sub-sets and redirect to free Rule Workers over messaging queue. Rule Manager redirects all rules to Rule Workers as round-robin.

Rule worker has been formed to listen to messaging queue and run the rules. Rule worker gets applied rules. Rule Worker connects to NoSQL database and read data for each rule. Then Rule worker runs the rule with data and gets the result as boolean. When Rule Worker finishes applied rule set, write results to the in-memory object caching system. Then inform to Rule Manager over the messaging queue.

Rule manager has to send all required rules to Rule Worker and get results. After the operation, the summary of run rules and results will be serialized and send to WebSocket service over the messaging queue. WebSocket service will show to the user if any action fires after rules.
Machine learning component will be work on the background to analysis together between on-demand rule engine analysis. Approved results of machine learning by doctors will be converted to rule and import to rule engine.

Summary
In this study, a system architecture for rule-based decision support system has been described which is scalable for clinics to big hospitals. The recommended architecture has been designed over Linux Containers to run distributed. All recommended software and libraries at this study can be found on as open source applications. When the architecture is describing, data input and outputs, decision support, statistical analysis and machine learning modules have been designed as holistic. By this way, a system architecture has been developed that will provide all the needs of the hospital and help doctors from the end to end.

Acknowledgement
This research was financially supported by the Necmettin Erbakan University BAP Office.

References
[1] L. Griebel, H.-U. Prokosch, F. Kopcke, D. Toddenroth, J. Christoph, I. Leb, I. Engel and M. Sedlmayr, A scoping review of cloud computing in healthcare, BMC Medical Informatics and Decision Making (2015) 15:17.
[2] A. Bahga and V.K. Madisetti, A Cloud-based Approach for Interoperable Electronic Health Records (EHRs), IEEE Journal of Biomedical and Health Informatics, Vol. 17, No. 5 (2013).
[3] RA. Greenes, Clinical decision support: the road ahead. Burlington, Elsevier, Inc., (2007) 3–29.
[4] S. Oh, J. Cha, M. Ji, H. Kang, S. Kim, E. Heo, J.S. Han, H. Kang, H. Chae, H. Hwang, S. Yoo, Architecture Design of Healthcare Software-as-a- Service Platform for Cloud-Based Clinical Decision Support Service, Healthcare Informatics Research, 21 (2) (2015) 102-110.
[5] Kubernetes, https://kubernetes.io/
[6] Docker Swarm, https://docs.docker.com/swarm/
[7] Z. Goli-Malekabadi, M. Sargolzaei-Javan, M. K. Akbari, An effective model for store and retrieve big health data in cloud computing, Comput Methods Programs Biomed, V: 132, (2016).
[8] Couchbase, https://www.couchbase.com
[9] RabbitMQ, https://www.rabbitmq.com/
[10] OpenStack Swift, https://www.swiftstack.com/product/openstack-swift
[11] HAProxy, http://www.haproxy.org
[12] http://web.mit.edu/~r/current/arch/amd64_linux26/lib/R/doc/manual/R-lang.pdf
[13] OAuth2, https://oauth.net/2/
[14] E. Turban, J.E. Aronson, and T.-P. Liang with contr. by R.V. McCarthy, Decision Support Systems and Intelligent Systems, Prentice-Hall, 2005.
[15] L. Lhotska, V. Marik, T. Vlcek, Medical applications of enhanced rule-based expert systems, International Journal of Medical Informatics 63 (2001) 61–75.
[16] K.-L. Kuo and C.-S. Fuh, A Rule-Based Clinical Decision Model to Support Interpretation of Multiple Data in Health Examinations, Journal of Medical Systems, V: 35-6, (2011) 1359-1373.