The Case for Designing Data-Intensive Cloud-Based Healthcare Applications

Position Paper

Srini Bhagavan[1,2], Khulud Alsultan[2], and Praveen Rao[2]

[1] IBM, Leawood, KS 66219
srinib@us.ibm.com,
[2] Univ. of Missouri-Kansas City (UMKC), Kansas City, MO 64110
kaq58@mail.umkc.edu,raopr@umkc.edu

Abstract. Cloud computing is one of primary line of business for leading technology companies like Amazon, Google, IBM and Microsoft. There is a growing interest among the healthcare community to adopt cloud services for use cases such as large-scale healthcare data management. One may wonder if cloud computing can meet the requirements of future healthcare applications, which may be data-intensive. In this position paper, we present a cloud computing architecture based on microservices and container technologies to support data-intensive healthcare applications. We propose the use of industry best-practices for managing storage and compute resources, and non-functional requirements such as availability, security, recoverability, and capacity planning. We provide our insights and recommendations for software developers to follow.

1 Introduction

The adoption of cloud computing in healthcare is expected to reach $9.5 billion by 2020 [1]. Cloud services can be used for population health management, care management support, diagnostic support, patient services, lab services, clinical research, and others [5]. Data-intensive healthcare applications are also on the rise. Researchers at Mount Sinai used Amazon Web Services (AWS) to analyze 100+ TB of genomic data for breast and ovarian cancers [4]. Ecosystems like Apache Hadoop [6] and Apache Spark [3] can enable large-scale healthcare data processing. It is possible to rent 100s of (virtual) machines from a cloud provider to run data-intensive healthcare applications. The healthcare use cases using cloud services are endless; the *pay-as-you-go model* of cloud computing is very appealing to hospitals for lowering IT costs.

Motivated by these trends, in this position paper, *we make the case for a cloud computing architecture based on microservices and container technologies to deploy and manage data-intensive healthcare applications in the cloud.* In today's cloud computing world, we are dealing with multiplicity of hardware environments from various vendors and software stacks that deploy and interact with them and with each other. The sheer permutation of deploying and managing them on various platforms is a combinatorial explosion problem at the very least.

Large enterprises also must worry about continuous integration and migrating between environments with least impact. We posit that in the coming years, critical healthcare applications will be hosted in cloud environments – microservices and container technologies are first step in this direction. Unlike prior work that emphasize on microservices for healthcare [2] and bioinformatics [8], we provide recommendations based on industry best-practices and practical experiences.

2 Background

2.1 Containers, Container Orchestration and Microservices

A container is a set of processes that are isolated from the rest of the operating system and can be easily moved to other environments. Containers are very light-weight and can scale efficiently. Multiple containers can be packed on a single host, and hence, share the same OS kernel. Containers facilitate efficient use of resources (e.g., CPU, memory, storage) available on the host machine compared to virtual machines (VMs) that have multiple guest operating systems. Containers increase application development efficiency by enabling continuous integration/continuous delivery (CICD). The same application can be ported across cloud providers. Moreover, containers can be provisioned very quickly, which translates to high application availability for deployment and maintenance.

All complex applications require some degree of orchestration. When there are many machines hosting containers, we need a system to federate multiple hosts into one target. When an application is composed of many containers running on several hosts, we will need to have a mechanism to move containers around when a host is down, for containers on different hosts to communicate with each other, update the application with zero downtime.

Microservice is the new architecture paradigm for cloud applications. The main idea is to decompose an application into many smaller components, where each component has its own responsibilities and by definition, a microservice. Each microservice is loosely coupled with one another. Microservices communicate using a light-weight protocol, can be distributed across different host machines and updated independently of one another. Containers are a natural deployment topology for microservices. In comparison, a monolithic architecture has all its components deployed together in one processing unit rendering it difficult to deploy, manage, upgrade and scale parts of the application stack.

2.2 Microservices Available on Cloud Providers

Infrastructure-as-a-Service (IaaS): IaaS refers to the underlying hardware resources such as network, storage and compute resources (usually with some virtualization technology), which cloud providers host and/or manage. (Depending on the choice of deployment, application developers may be given access to the machines.) For example, storage can be procured in many ways. Object storage can be used for low-cost scenarios. Block storage and file storage can be used for high performance applications with high allocated IOPS (input/output operations per second) per volume. Cloud providers offer predictable high performance storage solutions to support high I/O applications. Individual *performance storage* volumes for block and file storage are allocated specific guaranteed IOPS.

For example, applications can expect to provision storage from 20 GB to 12 TB with IOPS ranging between 100-6000.

Platform-as-a-Service (PaaS): PaaS is a cloud offering that provides to a user a complete stack to develop and deploy software to the cloud in addition to IaaS services. The primary benefit is that the consumer is abstracted from managing the underlying cloud infrastructure including network, servers, operating systems, and storage. Scalability and security are also fully managed by PaaS. This allows the developers to fully focus on what they are really good at, i.e., rapid development and deployment. Over the past few years, several vendors have come up with their PaaS offerings (that also deploy Kubernetes[3] clusters as part of machine provisioning). Kubernetes allows storage to be detached from compute to facilitate faster recovery and storage expansion.

Software-as-a-Service (SaaS): SaaS refers to the software used on-demand by application developers typically as a subscription-based model. Database-as-a-Service (*DBaaS*) refers to the model where access to the database is provided to the end user through SaaS. Typically, the database software, database configuration, all the physical (or virtual) machines, and storage are managed by the cloud provider. The degree of control over the database (and its host) depends on the cloud provider and the type of DBaaS. The performance and other non-functional requirements (e.g., security, availability) of the database are guaranteed by the cloud provider as per Service Level Agreement (SLA).

3 Our Position

A software developer building healthcare applications that store, manage, and analyze massive amounts of data is always left with the critical decision of choosing the right application stack, the total cost of procurement, and maintenance of the applications. With the entrenchment of cloud providers in the industry and the advent of microservices, architecting healthcare applications and deploying them demands rethinking the traditional development process. We advocate the developer identify/develop the required microservices by following the 12-factor methodology [7]. For example, the developer should containerize the web server and scale it for load balancing and high availability, containerize the data ingest, containerize machine learning, etc., and deploy them on a mature cloud provider runtime of choice.

To orchestrate the microservices, the developer can use Kubernetes, which is a natural fit. However, instead of manually setting up everything ranging from acquiring machines and installing Kubernetes, which is time consuming and requires Kubernetes administration skills, he/she can provision a container service from the cloud provider.

Next the stateful application developer must think about managing storage space and databases. In a typical on-premise environment, the IT team is responsible for taking care of these logistics. However, in a cloud environment, the

[3] https://kubernetes.io

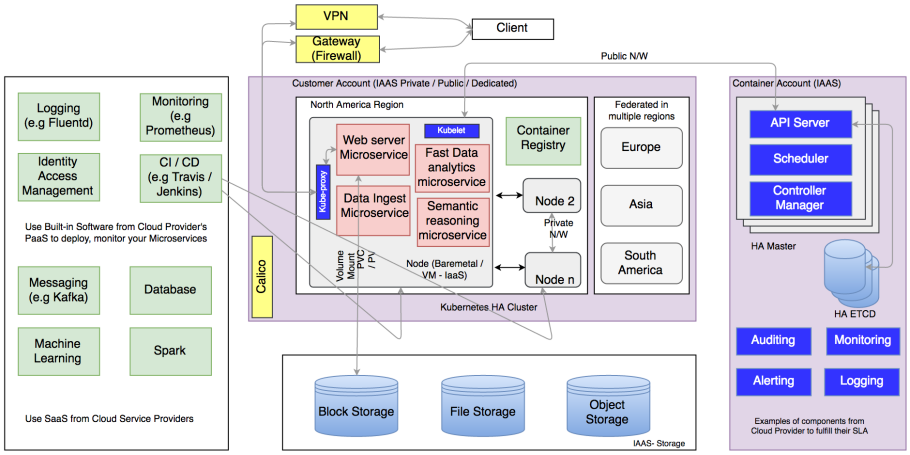

Fig. 1. An architecture for designing healthcare applications in the cloud

developer can take advantage of Storage-as-a-Service and Database-as-a-Service from the cloud provider. The caveat, however, is to pick the correct *SaaS plan* for these services.

Figure 1 shows the main components of our proposed cloud architecture. As shown in the figure, the developer should leverage cloud providers and their IaaS, PaaS, and SaaS instead of having a monolithic, on-premise architecture, wherein all the software and infrastructure are purchased, installed, and managed locally.

This architecture has many critical components which typically come from the cloud providers, and therefore allows the developer to focus on writing their data-intensive health-care application, instead of spending the effort in physically acquiring and administer this infrastructure.

In the Kubernetes HA cluster, there are many worker nodes for each region in the customer account. The worker nodes are where the microservice containers are deployed. We are proposing to architect any health-care application as a collection of microservices which are functionally independent entities. So web server, data ingestion service, infectious-disease-transmission analytics, pathogens-semantic-engine, etc., can all be independently deployed as microservices. To support high availability, disaster recovery, and faster response time for the end-user, the cluster is federated across regions with synchronized data and using a global load balancer.

Data needs to be persisted an IaaS-storage resource outside the Kubernetes cluster to separate storage from compute. The cloud provider provides many different types of storage, at different costs. In this diagram, we are showing block storage, file storage and object storage as examples. Block storage has higher performance than object storage but is more expensive. Data-intensive health-care application developers need to make a critical choice of the type of storage which suits their data needs. For example, electronic medical record data, which is accessed infrequently, can reside in object storage, and more recent

data can reside in block storage for faster retrieval time. By using IaaS-storage resources, developers don't have to worry about setting up storage; they are provided, maintained, backed up by the cloud providers per SLA. All that is required is to create blocks of Persistent Volumes (PVs) and claim them for use.

Beyond compute and storage, the diagram shows other services from the cloud providers that would help with many other non-functional requirements. Monitoring service can notify the Operation teams if there are issues with the applications. Identity Access Management, Logging, Auditing services can help with the health-care applications achieving the required certifications regarding sensitive data like HIPPA, SOC 2, etc. Audit logs track authorized and unauthorized access to records which can trigger alerts for prompt action. Developers can focus on only their application to adhere to desired compliance standards and piggy-back on the cloud provider certifications for the deployed stack.

Besides the core components of the applications and the NFRs, there are data analytics SaaS such Machine Learning, Streams, Health Cognitive which can help researchers gain insights from their proprietary data to improve patient care, disease management, discover new cures, etc. These expertise-intensive data analytics services are provided and managed by cloud providers. Furthermore, many of these SaaS services can also be scaled independently (as they are microservices) to better process the big data on demand. So if there is an outbreak of cholera in a region, epidemiologists can run analytics on patient data and determine hot-spots and determine the course of action to mitigate further spread of the disease. In order to support the sudden influx of intensive analytics needs, perhaps we could scale the number of pathogens-semantic-engine microservice instances (replica pods) independently or add more computing resources to the existing microservice instances. It can be argued that this type of scaling would be time/cost prohibitive in a monolithic architecture. Finally, healthcare applications that deploy in such environments can also take advantage of future inventions at record pace as they are made available by the cloud provider.

Below are the recommendations for key NFRs along with our rationale.

Availability: The developer should use container orchestration, which can avoid single point of failures. Kubernetes container orchestration also supports multi-node clusters with cross data center federation capability. Application workload pods immediately re-provisioned on other available nodes on failure. A container automatically restarts when any of the services components goes down. Kubernetes supports horizontal scaling, so when the load on an application increases, it automatically starts a number of containers to support the load. Several tools are available to monitor container services like Prometheus and New Relic. These tools provide alert mechanism when services go down. Many of the persistence storage support snapshot backup and restore functionality, which are very efficient to restore service data in case system recovery is required.

Security: Docker containers are, by default, very **secure**; especially when processes are run as non-privileged users inside the container. An extra layer of safety can be achieved by enabling tools like SELinux. By enabling data encryption on the disk, data cannot be read with out knowing the key (e.g., Linux Unified Key

Setup (LUKS) encryption). All cloud providers and databases support data encryption on storage using encrypted keys or a Vault scheme for secrets especially as we are in the age of massive data processing, auditing, protecting customer sensitive data from cloud providers, etc. Cloud providers support Vault store to store the encrypted keys/secrets and provide APIs to access them.
Scalability: Kubernetes container orchestration provides auto scaling feature that adds/deletes nodes when required automatically. It also provides horizontal scaling of the containers when required. This translates to the ability for applications to scale and shrink on demand automatically without/minimal human intervention or downtime.
Capacity Planning: Forecasting capacity requirement for data-intensive healthcare applications can be characterized "chaotic" at best. Baseline assumptions for compute, storage, IPs, memory, etc., are quickly invalidated given that the volume of data is unpredictable. Kubernetes orchestration coupled with cloud provider capabilities, efficiently accounts for hardware failure, rollover upgrades, and node/storage expansion scenarios. For example, an impacted node is quarantined and workload pods are drained and redeployed on a healthy available node. Latency is in milliseconds given a cloud provider maintains several classes of managed hardware pools, which are bootstrapped to the Kubernetes cluster. Furthermore, some cloud providers have support for multiple AZs in a region/data center. Applications may be federated on multiple AZs/regions to increase availability and perhaps load balance.

Concluding Remarks: We believe microservices and container orchestration are two synergistic technologies that will organically solve majority of cloud-based health-care application development challenges around scalability, availability, security, isolation, and performance in a cost-effective manner. We hope this paper will inspire the healthcare community to use industry best-practices for designing next-generation health-care applications.

Acknowledgments: The first author was supported by UMKC Provost's Strategic Funding and School of Graduate Studies Travel Grant.

References

1. http://www.healthcareitnews.com/news/hospitals-triple-use-cloud-services.
2. https://www.cio.com/article/3159071/innovation/microservice-ecosystems-for-healthcare.html.
3. Apache Spark. http://spark.apache.org.
4. Icahn School of Medicine at Mount Sinai Case Study. https://aws.amazon.com/solutions/case-studies/mt-sinai.
5. Impact of Cloud Computing on Healthcare. http://www.cloud-council.org/deliverables/CSCC-Impact-of-Cloud-Computing-on-Healthcare.pdf.
6. The Hadoop Project. http://hadoop.apache.org.
7. The Twelve-Factor App. https://12factor.net/.
8. C. L. Williams, J. C. Sica, R. T. Killen, and U. G. J. Balis. The growing need for microservices in bioinformatics. *Journal of Pathology Informatics*, 7:45, 2016.