

## **CSC547 Homework Assignment #2**

**Akruti Sinha (@asinha6) || Priya Andurkar (@pandurk)**

### **Problem 1:**

#### **Problem 4.1:**

The customer is(n't) always right. Consider the generic business requirements of a cloud consumer we outlined in Section 4.1. Clearly, we need to make a tradeoff between cost and any other requirement. Do we need to make tradeoffs between

1. Availability 24/7 and performance?
2. Accommodating time-varying workloads and performance?
3. Availability 24/7 and avoiding "vendor lock-in"?
4. Availability 24/7, accommodating time-varying workloads and performance?

Explain clearly.

### **Solution 1:**

1. We believe that yes, there would have to be a tradeoff between availability and performance 24 hours a day, seven days a week. To reach and sustain high performance, quantitative or qualitative, we would need to update infrastructure (hardware/software) on a regular basis, which could result in downtime. If updates are performed partially and asynchronously to assure availability, throughput and thus performance may suffer. Performance enhancements might be attempted with existing infrastructure by pushing it to its ultimate limits (compute, storage, etc...), but this would make it vulnerable to failure and would not maintain 24/7 availability.
2. There is no obvious tradeoff between tolerating time-varying workloads and performance. Accommodation of time-varying workloads is really implemented to sustain performance, and these two features are inextricably linked. If time-varying workloads are not accommodated, performance will be inefficient at peak loads, and if performance is tuned for the maximum peak load, there will be resource waste during times of low load.
3. Depending on the vendor, there may be instances where a tradeoff between 24/7 availability and "vendor lock-in" exists. There is no "vendor lock-in" if a service is offered by many cloud providers and their SLAs state 24/7 availability. However, if a specific service is provided by numerous vendors and not all of their SLAs provide 24/7 availability, there will be a tradeoff and the architects may be forced to pick specific vendors that provide 24/7 availability.
4. a) If we want to optimize availability, we will have to compromise performance, as described in part 1, because we will have to update partially and asynchronously to assure availability, which may have an impact on performance. In this scenario, there is no direct trade-off associated with tolerating time-varying workloads.  
b) If we choose to optimize our ability to accommodate time-varying workloads, availability will not be impacted because the two do not have a direct trade-off. As stated

in part 2, performance and accommodating time-varying workloads go hand in hand, so we will not sacrifice performance.

c) If we choose to enhance performance, availability may decrease, as mentioned in Part 1, but accommodation of time-varying workloads is unaffected, as discussed in Part 2.

**Problem 2:**

**Problem 4.3:**

The KISS Principle. In Example 4.15, we have mentioned that the RIP routing protocol is an example of design that applied the KISS principle.

1. In what exact feature of RIP was the principle applied?
2. With respect to this feature alone, would you say that OSPF applied/did not apply the KISS principle?

**Solution 3**

1. Only storing information about its neighbors: RIP only communicates with its immediate neighbors. It recalculates the distance vectors and broadcasts the results to all of its neighbors whenever it receives new information. As a result, it just has to store information about its neighbors rather than the entire network.
2. OSPF does not follow the KISS concept when it comes to this functionality. A node broadcasts its information to all nodes in the network and keeps network-wide information.

**Problem 2:**

**Problem 4.8:**

Treat servers as disposable resources Principle. Consider the second principle in Example 4.16.

1. Describe the principle in your own words.
2. Identify the ?aaS model(s) you could apply this principle to.
3. Identify a business requirement you could apply this principle to

**Solution**

1. Since rapid supply of resources such as servers is a core feature of cloud computing, businesses must abandon the concept of maintaining fixed servers and changing them as new requirements arise. Instead, upgrades should always be executed on fresh servers with the necessary configuration to ensure stability. To automate server instantiation, the client can use a technique known as bootstrapping, using AWS Golden Images or Containers.
2. We may apply this idea to IaaS (infrastructure as a service) because it is the only ?aaS where the client interacts directly with servers and thus has the potential to do so. Provisioning and configuration of servers occurs behind the scenes in all otheraaS models.
3. We may apply this theory to the commercial demand of 24/7 availability because it minimizes the likelihood of errors.

**Problem 3:****Problem 4.2:**

Is the process necessary? Consider the process we outlined in Section 4.4.1. It suggests that six steps are sufficient to create a “good” architecture.

Let’s not question sufficiency. However, are the six steps really necessary?

1. Set  $k = 1$
2. While  $k < 7$
3. Remove step  $k$  from the process in Section 4.4.1.
4. Consider a new process without step  $k$ . Is it “good”? Justify your answer.
5. Set  $k = k+1$
6. Endwhile
7. Listen to [https://www.youtube.com/watch?v=MAe\\_w9a\\_IN8](https://www.youtube.com/watch?v=MAe_w9a_IN8)

**Answer 3:**

1. When  $k = 1$ , we remove step 1 from the process. This would mean that we are skipping the step where a list of business requirements are summarized. We believe that the new process without step 1 would not be a good process. The first step emphasizes on understanding the problem at hand, breaking it down into business needs, and translating them into well formed english statements describing what the user would need. If we were to skip this part and directly jump into the technical requirements, we would be devoid of any concrete expectations of the solution. For designing an efficient and optimal technical solution, it is pertinent to know the system that is being designed.
2. When  $k = 2$ , we remove step 2 from the process. By eliminating step 2, we are removing the step where we translate our well formed understanding of the problem statement and the solution requirements into technical requirements. We believe that removing step 2 would not be a good process. Every high level business requirement that gets translated to a software solution needs to be broken down into multiple technical requirements. For example, if the business requirement is to ensure high availability of an application, it would consist of multiple technical nitty gritty details such as data replication, failure recovery, monitoring, analysis, fault isolation, geographic distribution, and more. If we don’t break down business requirements into these respective underlying technical requirements, we will not succeed at making optimal decisions on searching for options to address business requirements in step 3.
3. When  $k = 3$ , we remove step 3 from the process. By eliminating step 3 of the process, we are essentially skipping the decision process of choosing options that can be used to address a certain technical and related business requirement. We believe that removing step 3 would not be a good process. There are plenty of ways to achieve the same outcome by designing a certain technical solution. With the advancements in technology, multiple optimal options have been made easily available to consumers. For example, if the technical requirement needs to implement data backup, it can be achieved by using either Cloud Data Backup, On-Premise Data Backup, NAS devices, and more.

Searching for multiple options not only helps the architect understand how a technical requirement can be potentially fulfilled but also effectively achieves the completion of step 4.

4. When  $k = 4$ , we remove step 4 from the process. By eliminating step 4, we are removing the critical step where options are measured with their various pros and cons to ensure choosing the most fitting and optimal way to implement the requirements in the future. We believe that skipping step 4 would not be a good process. We believe that taking tradeoffs in consideration is crucial for the design of any solution. It is very likely that one option, while being the best for a certain use case, may not be the case for some other. Continuing the example of data backup, using a cloud data backup would be easier using a third party provider which reduces the overhead operations such as maintenance, duplication, recovery, or more. However, if the application has extremely classified and sensitive data which needs strong data protection, using On-Premise data backup options with the overhead of maintenance would be better. Hence assessing this tradeoff is very important because different applications have different needs. There is no one size fits all solution and hence skipping this step can result in poor architecture of the final solution.
5. When  $k = 5$ , we remove step 5 from the process. By eliminating step 5, we are removing the decision step from the process. We believe that skipping step 5 would not be a good process. After the due deliberation on options and tradeoffs, it is important to choose the most fitting option for the architecture to be successful. The decision would further help the software programmers to align themselves with the technologies needed to implement the option and they will not need to spend precious time in deciding between multiple available options. Rather they can begin implementing the well thought of option for successful product delivery.
6. When  $k = 6$ , we remove step 6 from the process. By eliminating step 6, we essentially do not document design decisions, design work-flows in the form of architecture diagrams, and more. We believe that skipping step 6 would not be a good process. For any solution to be successful, it is key to document the design, the decisions leading to a certain design, and the architectural diagrams. If documentation is not maintained, it would become difficult to recollect certain design decisions and discussions over a longer period of time. A well documented design can be used to ensure that during the implementation of the final solution, there is clarity of vision between the customers and the technical team. Hence skipping this step would not be the most optimal process.

**Problem 4:**  
**Problem 4.13:**

A best practice from the AWS Well-Architected Framework. Search [1] for a best practice. There are plenty! mention who benefits from the practice.

**Answer:**

There are multiple best practices documented within AWS Well-Architected Framework. This documentation helps describe various key concepts, design principles for cloud best practices and guidance for making improvements in different workloads. These were consolidated by gathering data from real-world case studies. There are 6 major pillars on which they have suggested best-practices. We looked into the cost optimization pillar. According to this pillar, it is a best practice to always perform pricing model analysis. They consider it a high level risk if it is not established during the design of the workload. It emphasizes on the importance of optimizing the costs by selecting appropriate models. It urges consumers to understand Savings Plans, identifying opportunities for cost-effective Spot workloads, and On-Demand Capacity Reservations. It is vital to understand commitment to resources, and the flexibility of their usage without upfront commitment. Understanding consumer workload and the respective best pricing model is the key.

We believe that the cloud consumers would benefit the most from this best practice. Any organization, which would like to host their workloads on AWS, have a large array of services at their disposal. However, after understanding the nature of their workload, it is essential to understand how AWS charges for the usage of their resources and services. This practice urges the user to find the best usage to cost ratio for them.

**Bonus Problem:**  
**Problem 4.9:**

Check <https://aws.amazon.com/architecture/reference-architecture-diagram> it is a repository of reference architecture diagrams. You can filter by technology category or industry.


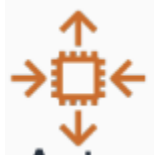



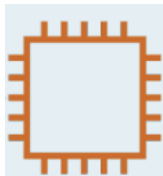
1. Select an architecture diagram of interest to you.
2. Describe, in your own words, the problem solved (what were the objectives?)
3. The diagram uses “standardized” AWS icons. Explain them.
4. List all the services involved.
5. Describe the services.
6. Comment on features of the design - e.g., is it scalable? highly available? secure? etc.




**Answer:**

1. We have chosen the architecture diagram for Adobe Commerce and Magento Open Source on AWS as mentioned below:

[https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/magento-on-aws-r-a.pdf?did=wp\\_card&trk=wp\\_card](https://d1.awsstatic.com/architecture-diagrams/ArchitectureDiagrams/magento-on-aws-r-a.pdf?did=wp_card&trk=wp_card)

2. This architecture provides one of the most popular solutions for a fully managed hosting platform for online sales and retails. It contains an array of development and deployment features for efficiently developing and continuously deploying applications on a PaaS environment. Apart from the ease of deployment it also provides solutions to common e-commerce application problems such as scalability, reliability, security and more.
3. The diagram consists of multiple standard AWS icons which are used throughout the diagram to signify the following:

	This icon represents the AWS Cloud environment and is used in the diagram to showcase that all components are encapsulated within it.
	This icon represents the auto-scaling group that enables flexibility in scaling.
	This icon represents an internet gateway which plays a role in enabling public internet access to the Virtual Private Cloud.
	This icon represents the Application Load Balancer which helps in routing traffic within multiple running instances of the applications for good performance.
	This icon represents a NAT gateway. It is used as a bridge between the private and the public subnet to prevent direct access.
	This icon represents the multiple third party services that are hosted within AWS and used in an architecture. In this diagram it represents the Varnish Cache and Magento instances.

	<p>This icon represents the public subnet. They are configured to allow instances within them to send and receive traffic from the public internet.</p>
	<p>This icon represents the private subnet. These sets of components are isolated from the public internet to restrict direct access from the public internet.</p>
	<p>This icon represents an EFS Mount Target. They provide network endpoints to EC2 instances to connect and access the EFS file system.</p>

4. All the services involved in this architecture are as follows:

- a. Amazon Route 53
- b. AWS WAF
- c. Amazon CloudFront
- d. AWS Auto Scaling
- e. AWS Marketplace
- f. Varnish Cache
- g. Magento Instance
- h. Amazon EC2
- i. Amazon OpenSearch
- j. Amazon ElasticCache for Redis
- k. Amazon Aurora
- l. Amazon Elastic File System
- m. Application Load Balancer
- n. Internet Gateway

5. Description of the services is as follows:

- a. Amazon Route 53 : A scalable DNS to help route web traffic to different AWS resources. It also helps in ensuring availability and reliability of applications.
- b. AWS WAF : A managed firewall service which protects web applications. It includes security against common malicious traffic and web exploitations.
- c. Amazon CloudFront : A CDN service which helps in accelerating the distribution of content of your website. It helps with reduction of latency and improvement of performance.
- d. AWS Auto Scaling : Helps in automatically upscaling or downscaling the number of EC2 instances present in an Auto Scaling group.

- e. Varnish Cache : An open-source accelerator for web applications. It also allows caching HTTP reverse proxy.
  - f. Magento Instance : These instances are deployed on AWS for building and managing online stores.
  - g. Amazon EC2 : Service that provides a resizable compute capacity. Helps with launching and managing virtual servers.
  - h. Amazon OpenSearch : A service which helps in managed search and analytics. Works optimally with very large data that needs to be analyzed or searched in real-time.
  - i. Amazon ElasticCache for Redis : A Redis caching engine for storing cache and improving performance. It is an in-memory data storage service.
  - j. Amazon Aurora : A relational database service with high performance. It is fully managed and compatible with PostgreSQL and MySQL.
  - k. Amazon Elastic File System : A file store service which is capable of being shared across multiple EC2 instances.
  - l. Application Load Balancer : Service that helps in distributing incoming traffic across multiple instances to optimize the performance of the application.
  - m. Internet Gateway : A gateway helps establish a bridge between a virtual cloud provider and the internet and vice versa.
6. The architecture is such that it provides an array of features for successfully running an online store. Services such as Amazon EC2 help with scaling up and down based on the incoming traffic for the application. Furthermore, from the diagram we can infer that the solution is also highly available. It uses multiple availability zones to deploy the EC2 instances which in turn provides robust fault tolerance. The application is also made secure by the usage of AWS WAF service and augments the high availability feature by reducing malicious usage of application resources. The architecture also employs multiple components for optimum performance using CDNs, caching and high performance databases.

### **References:**

[1]

<https://docs.aws.amazon.com/whitepapers/latest/migrating-magento-open-source-adobe-commerce-to-aws/magento-commerce-cloud-by-adobe.html>

[2]

<https://blogs.cisco.com/security/the-kiss-principle-for-successful-expansion-into-the-cloud-keep-it-simple-and-secure>