HW-3: Operating System

Prithul Sarker

1.

Read performance: RAID levels 0, 2, 3, 4, and 5 allow for parallel reads to service one read request. However, RAID level 1 further allows two read requests to simultaneously proceed.

Write performance: All RAID levels provide similar write performance.

Space overhead: There is no space overhead in level 0 and 100% overhead in level 1. With 32-bit data word and six parity drives, the space overhead is about 18.75% in level 2. For a 32-bit data word, the space overhead in level 3 is about 3.13%. Finally, assuming 33 drives in levels 4 and 5, the space overhead is 3.13% in them.

Reliability: There is no reliability support in level 0. All other RAID levels can survive one disk crash. In addition, in levels 3, 4 and 5, a single random bit error in a word can be detected, while in level 2, a single random bit error in a word can be detected and corrected.

2.

(a) For first-come, first-served, the seek time =

10 + 12 + 2 + 18 + 38 + 34 + 32 = 146 cylinders = 876 msec.

(b) For closest cylinder next, the seek time =

0 + 2 + 12 + 4 + 4 + 36 +2 = 60 cylinders = 360 msec.

(c) Using elevator algorithm, the seek time =

0 + 2 + 16 + 2 + 30 + 4 + 4 = 58 cylinders = 348 msec

3.

A layer of software on top of the operating system is one way for a distributed system to achieve some uniformity in the face of varied underlying hardware and operating systems. The name of the layer is called Middleware. This layer provides data structures and activities that enable programs and users on remote machines to work together in a consistent manner.

Three types of middleware technology: Document-based middleware, File-system-based middleware, Object-based middleware

Document-based middleware: The simple and wide known example of this type of middleware is the world wide web. The original paradigm behind the Web was quite simple: every computer can hold one or more documents, called Web pages. Each Web page contains text, images, icons, sounds, movies, and

the like, as well as hyperlinks (pointers) to other Web pages. When a user requests a Web page using a program called a Web browser, the page is displayed on the screen. Clicking on a link causes the current page to be replaced on the screen by the page pointed to. The Web is a great big-directed graph of documents that can point to other documents.

The way the whole system hangs together is as follows. The Web is fundamentally a client-server system, with the user being the client and the Website being the server. When the user provides the browser with a URL, either by typing it in or clicking on a hyperlink on the current page, the browser takes certain steps to fetch the requested Web page.

File-system-based middleware: The idea here is to make a distributed system look like a great big file system Using a file-system model for a distributed system means that there is a single global file system, with users all over the world able to read and write files for which they have authorization. Communication is achieved by having one process write data into a file and having other ones read them back. Many of the standard file-system issues arise here, but also some new ones related to distribution.

There are two types of models associated with it: 1. Upload/download model 2. Remote-access model. In the upload/download model, a process accesses a file by first copying it from the remote server where it lives. If the file is only to read, the file is then read locally, for high performance. If the file is to be written, it is written locally. When the process is done with it, the updated file is put back on the server. With the remote-access model, the file stays on the server and the client sends commands there to get work done there. The advantages of the upload/download model are its simplicity, and the fact that transferring entire files at once is more efficient than transferring them in small pieces. The disadvantages are that there must be enough storage for the entire file locally, moving the entire file is wasteful if only parts of it are needed, and consistency problems arise if there are multiple concurrent users.

Object-based middleware: An object is a collection of variables that are bundled together with a set of access procedures, called methods. Processes are not permitted to access the variables directly. Instead, they are required to invoke the methods. Some examples of object-based middleware: COM+, Java RMI, and CORBA. CORBA (Common Object Request Broker Architecture) is a client-server system, in which client processes on client machines can invoke operations on objects located on (possibly remote) server machines. CORBA was designed for a heterogeneous system running a variety of hardware platforms and operating systems and programmed in a variety of languages. To make it possible for a client on one platform to invoke a server on a different platform, ORBs (Object Request Brokers) are interposed between client and server to allow them to match up. The ORBs play an important role in CORBA, even providing the system with its name. When a CORBA object is created, a reference to it is also created and returned to the creating process. This reference is how the process identifies the object for subsequent invocations of its methods. The reference can be passed to other processes or stored in an object directory.

How the middleware acts as a bridge between the operating system and other applications in a network:

Middleware is software that connects other applications, tools, and databases to provide users with unified services. Middleware functions as a virtual glue that connects disparate heterogeneous apps and databases to provide clients with unified services. In a nutshell, it connects different software platforms and devices. Requests made over the network try to interact with back-end data. This information may be as basic as an image to display or a video to play, or it could be as sophisticated as a bank account history. The requested data can be in a variety of formats and stored in a variety of places, including on a file server, in a message queue, or in a database. Middleware's job is to make those back-end resources accessible and easy to use. Middleware programs, such as simple object access protocol, representational state transfer, or JavaScript object notation, will often provide a messaging service for apps to transmit data. In general, communications platform middleware, whether integrated or external, allows diverse communications tools to function together. These communication mechanisms make it possible for applications and services to engage with one another.

4. The difference between API and middleware:

APIs and middleware enable us to take advantage of interconnection between apps and other apps, as well as between programs and operating systems. The difference between them is given below:

Application Programming Interface (API) is an interface to a programming library (or libraries). It doesn't impose on you a way of doing anything e.g., OpenGL doesn't restrict what you can do with it. An API is designed to solve some specific problem in a particular domain.it contains necessary data structure, classes, methods, interface etc. Such as ADO.net API provides the functionality to connecting Microsoft SQL Server. If the implementation of a particular API requires querying another server to satisfy an incoming request, it would be middleware. There are many APIs which satisfy this criterion (EG a web service which queries one or more databases), but many which do not (EG the Windows API).

Middleware is a vertical slice. If you think of software as layered (e.g., OS, hardware abstractions, utility libraries, etc.), middleware incorporates many of these layers vertically. It provides a full, or partial, solution to an area within your application. E.g., a brokered messaging system, or a rendering library/engine. Middleware supplies more than just the basic library, it also supplies associated tools like logging, debugging and performance measurement. One thing you have to be careful about when using middleware is the DRY principle. Because middleware is vertical system, it may compete or duplicate other parts of your application. In short, Middleware is software logical layer for Integration, while API is dominant trending/recent style/paradigm for Integration.

In a nutshell, Middleware is a software logical layer for integration, whereas API is the most popular and recent style or paradigm for integration.

5. The three most common data formats for APIs are JSON, XML, and YAML.

JSON: Many new APIs have adopted JSON as a format because it's built on the popular Javascript programming language, which is ubiquitous on the web and usable on both the front- and back-end of a web app or service. JSON (JavaScript Object Notation) is a wonderful format when it comes to handling

client-side scripting and is a generally faster counterpoint to other options such as XML. JSON supports a distinction between number, string, and boolean that XML lacks.

XML: XML (EXtensible Markup Language) has been around since 1996. With age, it has become a very mature and powerful data format. Like JSON, XML provides a few simple building blocks that API makers use to structure their data.

YAML: YAML (YAML Ain't Markup Language) is "is a human friendly data serialization standard for all programming languages." Where JSON is lightweight with a somewhat lax feature set, and XML is verbose but often a bit cumbersome, YAML is easy to read, lightweight, and generally "middle of the road". While it's classified as a direct data format due to how it's used to parse configuration settings and relational queries, many systems use it as a basic flat database.

6. Difference between type 2 hypervisors and the docker engine:

Functioning Mechanism

The most significant difference between hypervisors and Dockers is the way they boot up and consume resources.

Hypervisors are of two types – the bare metal works directly on the hardware while type two hypervisor works on top of the operating system. Docker, on the other hand, works on the host kernel itself. Hence, it does not allow the user to create multiple instances of operating systems. Instead, they create containers that act as virtual application environments for the user to work on.

Number of Application Instances Supported

A hypervisor allows the users to generate multiple instances of complete operating systems.

Dockers can run multiple applications or multiple instances of a single application. It does this with containers.

Memory Requirement

Hypervisors enable users to run multiple instances of complete operating systems. This makes them resource hungry. They need dedicated resources for any instance among the shared hardware which the hypervisor allocates during boot.

Dockers, however, do not have any such requirements. One can create as many containers as needed. Based on the application requirement and availability of processing power, the Docker provides it to the containers.

Boot-Time

As Dockers do not require such resource allocations for creating containers, they can be created quickly to get started. One of the primary reasons why the use of Dockers and containers is gaining traction is their capability to get started in seconds. Docker can create containers in seconds, and users can get started in no time.

A hypervisor might consume up to a minute to boot the OS and get up and running.

Architecture Structure

If we consider both hypervisor and Docker's architecture, we can notice that the Docker engine sits right on top of the host OS. It only creates instances of the application and libraries.

Hypervisor though, has the host OS and then also has the guest OS further. This creates two layers of the OS that are running on the hardware.

If you are to run a portable program and want to run multiple instances of it, then containers are the best way to go. Hence you can benefit significantly with a Docker. Dockers help with the agile way of working. Within each container, different sections of the program can be developed and tested. In the end, all containers can be combined into a single program. Hypervisors do not provide such capability.

Security

Hypervisors are much more secure since the additional layer helps keep data safe. Dual OS layers provide extra data security.

Docker security is dependent on supporting linux kernel.

OS Support

Hypervisors are OS agnostic. They can run across Windows, Mac, and Linux.

Dockers, on the other hand, are limited to Linux only. That, however, is not a deterrent for Dockers since Linux is a strong eco-system.

7.

As-a-service types are growing by the day, there are usually three models of cloud service to compare:

1. Infrastructure as a Service (IaaS)
2. Platform as a Service (PaaS)
3. Software as a Service (SaaS)

Examples:

IaaS: DigitalOcean, Linode, Rackspace, Amazon Web Services (AWS), Cisco Metapod, Microsoft Azure, Google Compute Engine (GCE)

PaaS: AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, OpenShift

SaaS: Google Workspace, Dropbox, Salesforce, Cisco WebEx, Concur, GoToMeeting

Advantages of as a service technologies over other technologies:

- Lower up-front cost - As a service technologies are generally subscription-based and have no up-front license fees resulting in lower initial costs. The as a service technology provider manages the IT infrastructure that is running the software, which brings down fees for hardware and software maintenance.
- Quick set up and deployment - As a service technology application is already installed and configured in the cloud. This minimizes common delays resulting from often lengthy traditional software deployment.
- Easy upgrades - The as a service technology providers deal with hardware and software updates, deploying upgrades centrally to the hosted applications and removing this workload and responsibility from you.
- Accessibility - All you need to access a as a service technology application is a browser and an internet connection. This is generally available on a wide range of devices and from anywhere in the world, making as a service technologies more accessible than the traditional business software installation.
- Scalability - As a service technology providers generally offer many subscription options and flexibility to change subscriptions as and when needed, eg when your business grows, or more users need to access the service.

As a service technologies, and more widely cloud computing, can help you make the most of a limited IT budget while giving you access to the latest technology and professional support.