Name: Jian-Heng Tang
Login Name: jianhengt
Student ID: 764555

There are many challenges associated with distributed systems - Heterogeneity, Openness, Security, Scalability, Failure handling, Concurrency, Transparency.

Heterogeneity

Heterogeneity is the feature that havi the differences between two computers of distributed system in software or hardware. The different softwares in two separate computers are operating systems, programming languages or compilers, whereas the hardwares part is in different devices, hardware architecture or CPU. To communicate with each computer, many approaches are adopted to connect each computer together. One of the approaches, for example, is that Javascript, HTML and CSS are the common programming langaue for different browser. The derived challenges, however, is the security issues. Malicious code can still be executed when transfering the message between computers, even though the web standards are set nowaday.

Openness

Openness is described as the capacity of allowing system more extensible by means of resource allocation in software or hardware. Google document, for example, allows users to access to same office document and edit them at the same time. This is the openness of the distributed system. But security is an important issue when the hacker retrive the content from the document, this will endanger the privacy of users.

Security

The common method to enhance the security is public-key cryptography. This method can provide security and convenience because the secret-key can not be transmitted or revealed to anyone else. A serious concern, however, is that the speed is limited when using public-key cryptography for encryption.

Scalability

Whether a system to be considered as better scalability depends on the usage of algorithm. For approximate matching algorithm, for example, if a system make a comparison of two strings by naive matching, it takes $O(n^2)$ for comparing each character. Nevertheless, n-gram algorithm requires O(n) for a suffix tree, offering an effective method to deal with matching by searching for correct characters. The exception is that when a query is a boken string, it needs to repair first instead of matching due to worse time complexity.

Failure handling

One of the approaches to handle failure is redundant. When a packet needs to be transmitted from one computer to the other, the mechanism is to cnoduct the shortest path. But the problem is that if there is a broken router in the middle of the path, the source computer need to switch to other multiple routes to reach to destination. Heuristic routing, for instance, determines how data is delivered to the destination by deciding best, although not the optimal, path when an interruption in a network topology occurrs.

Concurrency

Concurrency allows computer to execute program in parallel, which means that two statements in different program can run at the same time. In this case, a statement can be interrupted by unexpected statement. This will cause the first statement execute as the wrong result. To solve this problem there is a mechenism named mutual exclusion to solve this problem. The mutual exclusion allows a statement to execute completely by adding a key as Semaphore or Lock without being interrupted by another process. This procedure can be called atomic operation. The drawbacks, however, the overall execution time is longer than program run in parallel. All statement can not be executed until the atomic operation is done. Although it takes time, this is still a trade-off to allow our computer run more correct.

Transparency

Transparency means that computers execute function or make some modifications without being known by users. Such modification, for example, can either handle routing failure on the Internet, or copy one file in a computer to another. For users, they do not know exactly the computers conduct on the backgroud so as to have a better user experiences. For example, cloud service as a software solution provide a sharing space to synchronize a variety of computers in a common storage. The computers can be seen as client side, whereas the common storage, or explicitly the database on the back-end, can play as a server. Albeit one device is a smartphone and the other is personal computer, they can share document by sending a file to cloud storage and other side can automatically receive the file from the cloud. As a result, the users would not know the changes via cloud service. The disadvantages, however, is that cloud service is heavily dependent on the Internet. When device disconnected to the Internet, the file can not synchronize to each computers, which means that files are not able to access to other computers.

Describe how a multi-server architecture could be implemented.

Multi-server architecture is the client-server architecture with more than one server. Different server provides a variety of functionality based on the different services, for those services can be file systems or process management. The service actions are controlled by the component, called coordinator, that is logically located in the middle of client and server. This coordinator is a virtual service to conduct the message-forwarding between client and server by using different protocols, such as TCP or UDP. An illustrating example of the message mechanism is that when client sends an message, the coordinator receives message and then forwards them to the different functional servers. After that when the server replies, coordinator forwards the message from server to the client.

The architecture can be roughly narrated by two parts: client and server. First of all, an emulation in the client process provides a service to generate the semantics of an standard OS. This service is also to make parts of OS reusable for other OS's when communicating with servers. It can be invoked either by user process directly or by system call. There has been no demand, however, for calling by user process but instead invoking by the syscall redirection nowaday. The implementation of an emulation is UNIX API syscall, and its basic responsibilities are Syscall, User State Management, API Value Translations, API Semantics Objects, Name Resolution, and Forking. In order to communicate with servers across different OSs, these basic operations serves as an essential role to satisfy the requirement for sending messages to the server.

Second, the role of the server is to provide their individual functionality so that they can supply a set of OS-items. The functionality is like the methods in C language, such as strcpy, ctime or printf. The C code is implemented by programming multi-thread mechenism. Each server is multi-threaded, for each thread to service individual client requests and to control activities. In addition, the specific system servers supplies different kind of services. For instance, the path name service is for PathName server, network endpoints and communication are for Net server, and process tasks for ProcessMgr server. The various servers also supports a combination of different interfaces, such as naming, I/O or async notification, to do their job. Thus, the servers supply their individual functionality in response to services and interfaces to allow an emulation library uses them to emulate the target OS.

Hence, heterogeneity is essential for clients to change their environment from one OS to another. As the coordinator layer from multi-server architecture, the coordinator can unify the formal message format when receving messages from different OSs. When one client would like to use different platforms, in the case of from Windows to Mac OS, clients still can access to the server and retrive the information they wish.

Describe the messages that would be used between the servers.

Likewise, the message can be controlled by the coordinator. When server sends message to the coordinator, the coordinator will forwards messages to other servers. Then other servers will send these messages to the clients that they belong to.

How would the data be distributed?

The data is mainly stored in the database. Individual servers can access to the database to store information depend on different services, whereas the client can use browser to absorb the information retrieved from the database. But how can client access to different servers? In my opinion, the server need to provide transparancy. When clients need to look up or update information but they are away from the server with their data, the nearest server to clients needs to backup the data from the previous server so that client will absorb information fluently. The security issues, however, will be arised from this senario. More data sharing between different server means more tendency to expose client’s personal privacy.

How would the client change?

Once the client change the data, the project that correspond to the code or program needs to change as well to ensure the data consistency. For example, when the client who is using an application, like a chatroom, would like to change the data on different database from different servers, the server which receives messages from client’s request will broadcast the data not only to other clients who is using the same applcation, but other servers which has stored the related information.

Reference

Client-Server Interactions in Multi-Server Operating Systems: The Mach-US Approach. J. Mark Stevenson, Daniel P. Julin. CMU Technical Report: CMU-CS-94-191, Sep. 1994.