DISTRIBUTED SYSTEMS: READING QUESTIONS 2

27 OCT. 2016

ALI ABDULMADZHIDOV

---

1. According to [DS], there are three types of system architectures: centralized, decentralized and hybrid. And according to [TAX] there are five of them. How do you think to which type of architecture by [DS] should we correspond the types that were proposed in the [TAX] and why?

Globally centralized client server architecture is centrilized system in terms of DS, cause it consists frp, a single central entity being server.

Locally centralized client server architecture is centralized system cause it just has more then one server, and there are most problems from first architecture.

Globally centralized P2P are hybrid architecture, because it still needed in server that becomes single point-of-failure.

Locally centralised P2P looks like decentralized system, but there're super nodes that brings a bit of centralisation, but they avoid becoming points of failure..

Pure peer-to-peer is fully a decentralised.

2. What is difference between a process and a thread?

Process - instance of a computer program that is beeing executed.

Thread - smallest sequence of instructions that can be managed independently by a OS.

Differences are:

1. **Thread** is part of **process** and process is independent by itself.
2. **Threads** within **process** share *process* state, *memory* and other *resources*.
3. **Processes** have their own separate *address spaces*, that are shared by **threads** of that process.
4. **Processes** can interact with each other only through *OS provided mechanisms*.
5. Switching between **threads** within **process** is faster than between processes.

3. What is an asynchronous (non-blocking) I/O operation?

Asynchronous I/O operation is type of processing of input/output that doesn't block process other processing to continue while transmission has finished.

There're different forms of asynchronous I/O:

1. Separate process:

In this form our processing is separated between several small processes that have their own resources and I/O flow that is connected through pipelines. This type of processing is expensive to create and maintain, and only will work if processes are small. Also coordinating processes between each other make be painfull.

2. Polling

In polling, we poll repeatedly process for I/O operations.

3.Select poll

4. Interrupts

After async I/O is completed it calls interrupt signal and process processes that I/O flow.

5. Callback functions

Same as 4, but there I/O request has it's own completion function for callbacks.

6. Light-weight processes or threads

In this way, we make usuall synchronous I/O on separate thread, which don't block processing of other thread. Communication between threads is far more easier than between processes, that's why this is better then method 1.

4. Does it make sense to use threads on a single-core CPU?

Yes, because there might be some blocking resources and it'll freeze all process waiting for them. In multi-thread process, one thread'll be blocked by that operation, and CPU will switch to work on another thread,

5. VM images such as AMIs can be quite big. How does this impact cloud providers that have many customers creating many different virtual machines all the time?

I think most of that images, like AMI's, are read-only. Many users use single AMI image for every OS or service like in docker, and they don't need to store all variations of them.

6. Are Web servers stateless or stateful?

Web server's are stateless, we store state in cookies and send them in each request.

7. What is the difference in request dispatching for local-area and wide-area clusters? At what point will we need a redirection policy?

In local-area request dispatching one or multiple dispatchers stay in middle o connection and work like NAT. At least request every time goes through them.

In wide-area request dispatching, middleware runs redirection policy to find server that is localy near to user, and connect user with that. For example, dispatcher can be DNS server, and when user send DNS request for server it finds nearest server that can handle that request and send it ip to user.

8. Wide-area redirection requires a method for measuring the distance between two IP addresses. Think of two different methods and discuss pros and cons.

There're two types of measuring distance between two addresses:

1. TTL changing
2. Administrative Distance

TTl changing is easier to implement and can show number of hosts to the neede ip address, but it wouldn't say what protocols was used by middle hops.

In contrary, cisco's way to count administrative distance gives info about number of hosts and protocols are used there, and helps to choose more reliable way.

9. What problems will you need to solve to allow live migration of virtual machines between different wide-area clusters?

There're two problems of migrating virtual machines:

- Migrating memory image
- Migrating bindings to local resources

As to the first problem, there are three ways to handle migration:

1. Pushing memory pages to the new machine and resending the ones that are later modified during the migration process.
2. Stopping the current virtual machine; migrate memory, and start the new virtual machine.
3. Letting the new virtual machine pull in new pages as needed, that is, let processes start on the new virtual machine immediately and copy memory pages on demand.

Concerning local resources, matters are simplified when dealing only with a cluster server. First, because there is a single network, the only thing that needs to be done is to announce the new network-to-MAC address binding, so that clients can contact the migrated processes at the correct network interface. Finally, if it can be assumed that storage is provided as a separate tier, then migrating binding to files

10. According to Fuggetta (Note 3.9) there are three segments in a process. Which segment do you think is typically more difficult to migrate?

There're three segments in process:

1. Code segment
2. Execution segment
3. Resource segment

Resources segment is difficultier to migrate cause there can be many pointers to another process or hardware, that we need to migrate with it, and we can have huge amount of dependecies with that process. And even after that we should assure that we have same addresses for them like in original environment.