Assignment 119W

Distributed Systems (University of Waterloo)

CS454 Assignment 1
Winter 2019

Due: 8 Feb 2019 (4:00pm)
Returned: 22 Feb 2019
Appeal deadline: within 1 week of return

Question 1 (20%) What factors affect the response time of a client (application) that accesses data managed by a remote server?

Solution:

When the client (application) accesses the server, it invokes operation(s) in a server running on a remote computer. The following can affect the response time.

1. load of the server, e.g., if the server is overloaded.
2. latency in exchanging request and reply messages (due to layers of software, e.g., middleware) between/in client and server.
3. network latency.

Question 2 (16%) If a client and a server are placed far apart, we may see network latency dominating overall performance. How can we tackle this problem?

Solution:

It depends on how the client is organized. It may be possible to divide the client-side code into smaller parts that can run separately. In that case, when one part is waiting for the server to respond, we can schedule another part. Alternatively, we may be able to rearrange the client so that it can do other work after having sent a request to the server. This last solution effectively replaces the synchronous client-server communication with asynchronous one-way communication.

Question 3 (20%) Are there advantages of using synchronous RPC over asynchronous RPC? If so, describe them. If not, provide an explanation of why not.

Solution:
Advantages of synchronous RPC:
(i) Debugging and error handling in synchronous programs is much simpler and therefore less error prone, e.g., it guarantees that the response received by a client corresponds effectively to the request (and not to a former request to which the response might have been lost).
(ii) Faster response time and fewer round-trip messages as the ACK and response can be the same message.
(iii) Synchronous RPC handles dependent operations more cleanly, e.g., to do this with asynchronous RPC, tracking additional contexts and dealing with callbacks/event handlers is usually required.

Question 4 (24%) Identify the main benefits and drawbacks of the following architectures:

1. Client–server
2. Three-tier
3. Unstructured Peer-to-Peer

Solution:

1. Client-server is a very popular architecture for distributed systems and is well-understood. This two-tier approach works well at low scales or with limited complexity applications. The two types of component have different functionality and behaviour; a key aspect that affects performance and scalability is the way in which functionality is divided among the component types.
2. The three-tier architecture facilitates separation of the three strands of functionality (user interface, business logic, data access logic) such that a middle-tier can be dedicated to business logic. This is more flexible and potentially more scalable and robust than the two-tier client-server architecture but is also more complex in terms of structure and behaviour, thus requiring greater design and testing effort.
3. Routing can be done without the need of a global distributed data structure, e.g., hash table, which can be expensive to maintain. Peers can join and leave without the need to keep adjusting a distributed data structure. However, routing requests for items can be costly as it can take a long time to explore neighbourhoods until the item is found.

Question 5 (20%) Assume n computers are interconnected and the availability of every computer is needed to maintain a distributed service, and each of these computers has a probability p $(0 \leq p \leq 1)$ of failing at any time.

1. What is the probability p_s that the service will not be available at any time, assuming that no other components in the distributed system will fail? Express p_s as a mathematical function of n and p?
2. Based on your answer for part (1), what is the probability p_s when computing is not distributed, i.e., for the case where $n = 1$?
3. Based on your answer for part (1), use $p = 0.2$ and $n = 3$ to compute probability p_s. How does this probability compare with the failure probability if the same computing is performed on only one computer?
4. Now assume that the service requires only one of the three computers with the other two computers serving as backups (that is, any of the three computers is capable of providing the service). What is the probability that the service will not be available at anytime, assuming that no other components in the distributed system will fail? How does the failure probability of this system compare with the failure probability if the same computing is performed on one computer only?

Solution:

1. All n computers need to be functioning in order for the service to be available. The probability that each system is functioning is $1 - p$. The probability that all n systems are functioning is $(1 - p)^n$, which is the probability that the service is available. The probability that the service is not available is the complement, or $p_s = 1 - (1 - p)^n$.

2. When $n = 1$, $P_s = 1 - (1 - p)^1 = 1 - (1 - p) = p$.
3. On a single computer, the probability of failing is $p = 0.2$. Using the formula developed in (a), the failure probability is $P_s = (1 - 0.8^3) = 0.488$ on 3 computers. Because all three computers need to be up for the service to be available, the failure probability is greater than when only one computer is needed.
4. The failure probability is p_n in general, or $(0.2)^3 = 0.008$ in this case. Since only any one of the three computers needs to be up for the service to be available, the failure probability is now less than when exactly one single computer is needed.