Assignment 2

Problem 1

User-perceived performance of DNS queries should be faster than it was previously. Since the address of these server are unlikely to change, client caching should on average circumvent the additional round trips cost caused by the additional indirection introduced by these servers. In addition, these servers would help reduce load from root servers, which means that the roots server and gTLD servers will be able to process queries faster. In addition, gTLD servers may be close to clients geographically (i.e. having a gTLD server responsible for the “.ca” domain in Canada), which help reduce RTT and thus speed up DNS queries.

Problem 2

The first technique for scaling distributed system is hiding communication latency. The idea of this technique is to void waiting on responses from remote service requests as much as possible. A common way of achieving this is letting the caller do useful computation while waiting on a remote request through asynchronous calling mechanisms or blocking the requesting thread until the result comes back. This complicates somewhat complicates server design in that it requires proper concurrency mechanisms to perform synchronization. In addition, it encourages the system to be design to process in batches or parallel, which may not be suitable for many applications. In such applications, communication latency can be reduced by performing as much work as possible locally on the client. This is certainly simpler from a system design perspective, but may not be acceptable for performance reasons.

The second technique for scaling distributed system is distribution, which involves taking a component, and splitting it into smaller parts and spreading it across the system. This doesn’t really complicate system design and development, except maybe introduce some performance concerns due to having to talk to multiple parts of the system for a request, thus having to pay a price in terms of more RTT’s.

The third technique for scaling distributed systems is replication, which is idea of replicating components across a distributed server. Caching is a special form of replication; the difference is that caching done by the client of a resource, there as replication is done by resource owner. The problem with replication is that if the resource is updated, then there might be inconsistencies in the system if the update is not propagated to all copies immediately. Some application may have tolerance for this kind of inconsistency, but for application that have strong consistency requirement, it become very difficult and require some form of global synchronization mechanism.

Problem 3

Let’s look at what the bandwidth would be for each of the six situations:

A-RAM: This is just the bandwidth of the RAM, which is 20Gbps

Local A-Disk: This is the bandwidth of the disk, which is 80Mbps

B-RAM: even though RAM has a higher bandwidth, the total bandwidth is bottlenecked by the network with rack, which is 128 Mbps

B-Disk: even though network in rack has a higher bandwidth, the total bandwidth is bottlenecked by the disk, which is 80 Mbps

C-RAM: even though RAM has a higher bandwidth, the total bandwidth is bottlenecked by network in DC, which is 26 Mbps

C-Disk: even though disk has a higher bandwidth, the total bandwidth is bottlenecked by network in DC, which is 26 Mbps

Therefore, the order is as follows:

1. A-RAM
2. B-RAM
3. Local A-Disk, B-Disk
4. C-RAM, C-Disk

Let’s look at what the latency would be for each of the six situations:

A-RAM: this is the latency of the RAM, 100ns

Local A-Disk: this is the latency of the disk, 10ms

B-RAM: the latency is the RAM latency + network within rack latency, 100ns + 70us

B-Disk: the latency is the disk latency + network within rack latency, 10ms + 70us

C-RAM: the latency is the RAM latency + network within DC latency, 100ns + 500us

C-Disk: the latency is the disk latency + network with DC latency, 10ms + 500us

Therefore, the order is as follows:

1. A-RAM
2. B-RAM
3. C-RAM
4. Local A-Disk
5. B-Disk
6. C-Disk

Note the above latencies are calculated assuming disk latency accounts for RAM latency, but the order would be the same even if it RAM latency needed to be accounted for separately.

Question 4

Part A

|  |  |
| --- | --- |
| Server queried | NS delegation to |
| e.root-servers.net | a.edu-servers.net.  c.edu-servers.net.  d.edu-servers.net.  f.edu-servers.net.  g.edu-servers.net.  l.edu-servers.net. |
| l.edu-servers.net. | dns2.itd.umich.EDU.  adns3.doit.WISC.EDU.  adns1.doit.WISC.EDU.  adns2.doit.WISC.EDU. |
| adns2.doit.WISC.EDU. | dns2.CS.WISC.EDU.  dns.CS.WISC.EDU.  dns2.itd.umich.EDU.  CS.WISC.EDU. |
| dns2.itd.umich.EDU. | dns.CS.WISC.EDU. hostmaster.CS.WISC.EDU. 2017060601 86400 3600 604800 600 |

Part B

|  |  |
| --- | --- |
| Server queried | NS delegation to |
| e.root-servers.net | a.in-addr-servers.arpa.  b.in-addr-servers.arpa.  c.in-addr-servers.arpa.  d.in-addr-servers.arpa.  e.in-addr-servers.arpa.  f.in-addr-servers.arpa. |
| f.in-addr-servers.arpa. | r.arin.net.  u.arin.net.  x.arin.net.  y.arin.net.  z.arin.net.  arin.authdns.ripe.net. |
| cn-dns-rac.uwaterloo.ca. | ext-ipam-yyz.uwaterloo.ca.  cn-dns-rac.uwaterloo.ca.  cn-dns-mc.uwaterloo.ca. |
| cn-dns-mc.uwaterloo.ca. | connect.uwaterloo.ca. |

The first command I used was:

dig +norecurse @e.root-servers.net PTR 43.208.97.129.in-addr.arpa