582615 - Overlay and P2P Networks, Spring 2015

Exercise 3 answers

Considering the following points in the answer granted points in the assignment.

Assignment 1 - Content Delivery Networks

What kind of problems can be solved by using a CDN?

A content delivery network is typically a geographically distributed set of datacenters, and a set of software for managing content dissemination. Geographical distribution allows users from all over the world to find a replica of the content from a relatively close location. International / intercontinental links get less loaded when the CDN serves a replica that is already on the correct side of the link.

By serving the content from many datacenters a CDN also works as a load balancer, at least for static content. A lighter setup would suffice for the origin server. A CDN serving static content can keep the content available also when the origin server suffers a failure.

Describe mechanisms used by CDNs to direct users to content

Some mechanisms used in CDN:s

- **DNS Based request routing**

  The CDN has control over the forward DNS requests for a domain. When a user types in a hostname, the CDN will determine the most suitable address to send as a response, based on which nodes have the content, which one is the one closest to the user, which one is under least stress or other such criteria.

- **HTTP Redirection**

  When the content is distributed over HTTP, a point of entry server can use a HTTP redirect (*302 Moved temporarily*) status code and header to redirect the user's browser to the URL where the content can be found in some optimal way.

Describe how Coral delivers content to users.

Coral is a HTTP Proxy -style CDN. Users add a .nyud.net -suffix to the hostname-part of the URL they wish to access to get the content through the Coral network. DNS is used to direct the user to some nearby Coral proxy. The proxy then uses a hash of the URL as a key, checks its own and other Coral nodes' caches for documents by that hash, and if found, returns said content to the user. If the content was not found, a Coral proxy will parse the URL and fetch the document from the original server, then continues as if the content was found.

Assignment 2 - Kademlia

The K-bucket size dictates how many nodes fit in the first and largest bucket in a node's routing table. The following buckets each can contain half of the amount of nodes as the previous one; a k-bucket size of half the number of nodes can lead to a fully connected network.

Increasing the k-bucket size makes the nodes know more about the network, so that they can reach what they are looking for faster. On the other hand it increases the size of the routing table, leading it to consume more memory and require more processing to make lookupts.

Decreasing the k-bucket size makes the nodes know less about the network. The memory footprint of the routing tables is reduced, but as the network is less connected, messages may need to travel longer paths along the network.

Memory

Theoretically the size of the routing tables depends on the k-bucket size and the size of the identifiers. It should not be more than (2 * kbucket_size) * bits_in_identifier. In practice the nodes need connectivity data (like IP-address and port etc.) and probably align the identifier size to the upwards closest multiple of eight bits.

Assignment 3 - Dynamo

Eventual consistency and Shopping carts

One of the design goals in Dynamo was that it must always be possible to insert data into it, and no information must be lost. It must hence write the data to disk somewhere soon. Even though writes must always be available, Dynamo partially sacrifices the ACID guarantee in that reading can sometimes return old data, and there are no isolation guarantees. In terms of CAP Theorem (*Consistency, Availability, Paritition tolerance; choose two*), Dynamo provides Availability and Partition tolerance.

Writes to Dynamo are initially performed on one node, and then propagated to other nodes in the background. All writes are accepted, and possible conflicts between content on the nodes are resolved at read time. During the read, conflicts can be solved by Dynamo if a simple “last-write-wins” strategy is sufficient, or the application can resolve the conflicts itself.

Dynamo versions the shopping carts using vector clocks. On each write/update to a shopping cart, the node must specify which version it is updating. Simultaneous updates can lead to situations where there are multiple different shopping carts with the same version number. Upon read, it is the responsibility of the application to solve such conflicts e.g. by merging them.

Partitioning, scalability and redundancy

Cassandra uses consistent hashing to partition the data, and it is also Cassandra’s main strategy for scalability. Cassandra augments the regular consistent hashing with a load-balancing strategy: the load of each node is analyzed periodically, and lightly burdened nodes are moved on the ring to lighten the load of the more burdened nodes.

To achieve redundancy, Cassandra uses replication. The replication strategy is configurable. The simplest default setting “Rack unaware” simply places the replicas on the next (replication_factor - 1) nodes along the consistent hashing ring. Using this strategy, the keys are already replicated on the nodes that would become responsible for them should the node responsible for that key fail. Cassandra’s membership strategy also plays its role for both features. In Cassandra, every node knows about all other nodes. It is always easy to find a node that has the item and requests never need to travel far. As a downside however the routing tables can be big. Also when a new node is added to an already running system, instead of choosing a random position on the consistent hashing ring, it gets assigned one that splits the responsibility with a heavily loaded one.

Assignment 4 - Future of Overlays and P2P

This assignment was of a more speculative nature than the others. Points were granted for ideas, suggestions and good discussion with the sources.

The idea of a P2P cloud is to offer cloud resources from a possibly heterogeneous collection of ordinary office computers that are interconnected with a network. In this model there would be no need for a centralised data center with dedicated computers, but the normal office computers would participate to produce the same services. If this technology matures, a company might no longer need to buy at least all of their cloud services from a vendor or invest in their own data center.

SDN is a way to control dataflows programmatically. It lets you do fine-grained configuration of switches and routers. With SDN you can, for example, redefine the boundaries of broadcast domains to make computers on separate networks see each other as if they were on the same network. Overlays can benefit from SDN integration e.g. by getting a better ability to learn about the structure and condition of the underlying network and being able to optimize their routing accordingly.

A P2P would benefit from SDN compatible networking in much the same way as any overlay. It could use it to find the fastest paths between two hosts, create isolated virtual networks for the virtual machines, and it could adjust the routing to accomodate moving virtual machines from one host to another.