System Design for Large Scale

Carlos Baquero
DEI, FEUP, Universidade do Porto

MEIC SDLE 2021

Motivation

- Millions of users have internet access, and more will come. Often, fast adoption of a new service kills the service (e.g. the SlashDot effect).
- Since many users have always-on broadband connections its is tempting to use resources latent at the network edge (e.g. P2P distribution of World of Warcraft patches and video demos by Blizzard).
- The more users adopt a new service, the more power there is to run the service. At least in theory, since in scalability, as in economy, there is always potential for diminishing returns.

law of diminishing returns (Wikipedia)

According to this relationship, in a production system with fixed and variable inputs (say factory size and labor), beyond some point, each additional unit of variable input yields less and less additional output.

Early History

- Altough P2P designs can be tracked in early systems such as Usenet News, DNS, Ficus, Bayou and others; the current expression of the concept arises on the turn of the century with Napster and Seti@Home.
- However, neither Napster nor Seti@Home are purelly P2P.

Early History

SETI@Home

- Do ETs have TV? If so, we might ear them . . .
- Radio signals are collected at the Arecibo Radio telescope.
- These signals are divided in time and frequency creating data buckets.
- Buckets are downloaded and analysed on user nodes and results are uploaded back to the server.
- SETI depicts classical data parallelism and all brokerage is handled by a centralized server.
- No direct contact among peers.

Early History

Napster

- With MP3, music could be efficiently encoded and shared with existing file copying mechanims.
- Centralized serving of MP3, once popular, would induce a huge load in a single point and high data transport costs.
- Napster kept a centralized catalog of music descriptions and references to online users that hosted copies of it.
- Actual downloads are done among peers. They exchange roles, either as providers or as recipients.
- Due to presence of firewalls, ability to comunicate with a server does not imply capacity to accept connections (leading to the double firewall problem).
- Relience on a server proved to be a technological weakness for legal attacks from the music industry.

Gnutella (Early Design)

- Fully distributed solution to P2P file sharing.
- Partially randomized overlay network. Each node i connects to a number k_i of other nodes. This number can vary across nodes, as well as allocated bandwidths.
- Bootstrapping is done by HTTP hosted host caches and by local host caches from previous sessions. Due to high churn, local host caches can quickly become outdated.
- Routing on the overlay is based on flooding and reverse path routing (further data on the paper *Mapping the Gnutella Network* and a topology graph on `http://snap.stanford.edu/data/index.html`).

Gnutella (Early Design)

Protocol

- PING and PONG messages are used to discover new nodes. PINGs are flooded and PONGs are answered by along reverse paths.
- QUERY and QUERY RESPONSES, provide search capabilities on content textual descriptions. Queries are flooded and replies back propagated. The answer set on the requesting node slowly grows with time, until the diameter, or maximum hops, is reached.
- GET and PUSH requests are used to initiate file transfers directly between peers. PUSH is used to circumvent single firewalls that would block a GET in a given direction.
- This early design was found out not to scale, and PING/PONG traffic was dominant in the overlay.

Gnutella (Improved)

- Some nodes had higher maximum connectivity and bandwidth, most were always-on servers. Nodes preferred connections to nodes with more uptime. With time, the notion of super peer was brought to the protocol.
- Super peers act like as in the early Gnutella overlay while shielding traffic and mediating access to client peers. A two-tier architecture is formed.
- Before, contents were not announced, now a digest (using bloom filters) is sent from peers to super peers. Super peers mediate search and only contact target peers with a high likelihood of having the searched for content.
- Gnutella kept scaling and achieved 40% of P2P file sharing, around 2005.

Distributed Hash Tables

- Even in a super-peer architecture search in Gnutella is essentially flooding. Can search, or at least lookup, be done in a more controlled fashion? Can we bound the number of hops transversed to lookup a given target?
- Many alternative solutions to this problem are achieved by Distributed Hash Tables.
- DHTs provide ways of mapping *keys* to network *nodes*. Node joins and leaves should be accounted for in the protocols, in order to preserve some structure in the routing supporting the DHT. In this sense, they can require more maintenance that unstructured approaches.
- Here we will look deeper into *Chord* and Kademlia.

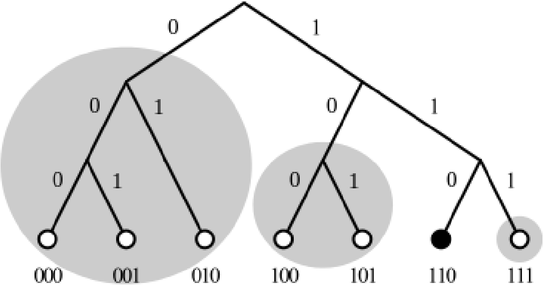
Chord (2001)

- Nodes and keys are assigned probabilistic unique ids in id space from 0 to $2^m - 1$. Both nodes ids (say IPs) and keys are hashed by SHA1 and m bits are taken.
- Keys and nodes are arranged in an ordered circle modulo 2^m. For a given key and a given set of nodes, it is possible to determine the sucessor node (nodeId $\geq$ keyId) of that key position. This node will store the key.
- It must be possible to contact an arbitrary node and ask it to find the sucessor node for an arbitrary key. Keeping a list of all nodes in each node would not scale.
- Each node knows the IP address and id of clockwise m other nodes, and r vicinity nodes in both directions.
- The i^{th} entry in node n indicates the first node s that succeeds n by at least 2^{i-1}.
- Nodes keep $O(\log n)$ knowledge on other nodes and routing takes $O(\log n)$ steps.

Kademlia (2002)

- Nodes and Keys share a 160 bits space of ids. Keys are stored on “close by” nodes.
- Id distance is computed by a XOR metric. XOR is an interesting symmetric distance metric that respects the triangle property.
- Unlike Chord, here routing is symmetric and alternative next hops can be chosen for low latency or parallel routing.
- Routing tables consist of a list for each bit of the node id.
- A node in list position i, must have bits 0 to $i - 1$ identical to the list owner, a different i^{th} bit, and can differ from position i onwards. Its easy to find nodes for the first positions.

Kademlia
Routing tables



For node 110, groups must match initial sequences: $\perp, 1, 11$

Kademlia

- To account for failing nodes and alternative paths in each position up to k nodes are stored. k is about 20.
- Candidate node uptimes is considered when competing for k limited positions.