International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 12 | Dec 2018 www.irjet.net p-ISSN: 2395-0072

Estimating various DHT protocols

Vishal Wadhwani[1], Nayan Jain[2]

[1,2]*B.E.:Computer Engineering Dept., VESIT, Mumbai, India.*

Abstract - *This project involves the use of Peer-to-Peer networks to facilitate premium digital content streaming. Clients seeking this service will not have to rely upon a third party to ensure the safety of the data. Our system mitigates the common problems associated with centralized systems such as trust issues, data failures, outages and traditional security concerns. Unlike the non-standard encryption in centralized systems, we use client-side encryption to ensure data privacy against the nodes storing the data. Data availability is a major concern associated with the Peer-to-Peer networks. We seek to solve this problem with a verification system proposed by Storj[1] which involves a challenge-response protocol. We further propose a model for distributing and accessing the data referencing the BitTorrent 'Distributed Sloppy Hash Table' protocol[4] based upon the Kademlia protocol[2]. Each node in our network is a valid Ethereum address. To implement the above-stated project we had to compare the below three protocols.*

Keywords—Distributed hash table, Tapestry, Chord, Kademlia.

1. INTRODUCTION

Distributed hash table protocols might look identical in some aspects such as node forward lookups, but they have their share of discrepancies in terms of the amount of state they possess, search for low latency routes, selecting between alternatives for parameters like frequency, etc.

Instead of the traditional approach of comparing DHT's with focus on hop-count latency, or routing size table we are assessing the protocols in the face of joining and leaving protocols. It will make it simple to examine tradeoffs between state maintenance detriments and lookup performance. In this paper, we will be analysing three Distributed Hash Table protocols namely, Kademlia, Tapestry and Chord as they exhibit a wide variety of design choices for Distributed Hash Table protocols.

The illustration of inter-node latencies has been realised using a simple simulator called King method. The King method overlooks the effects of congestion. Using lookup operations and churn we distinguish the performance of the protocols. We examine these protocols under varying settings of the parameters and come up to a conclusion that the protocols can have similar or different performance depending on the parameters.

In our framework, we are comparing different cost and performance measures. Instead of conventional cost measures of CPU time or memory we are using the number of bytes of the message sent. For performance, we are using lookup latency as the performance metric in place of success rate or hop-count. We penalise or award toward the cost of the framework depending upon the efficacy of the routes taken by the protocol. We have delineated the model to highlight the discrepancies solely based upon our choice of parameters. We assess each DHT over a range of parameter values, planning a performance envelope from which we can extrapolate an optimal cost-performance tradeoff curve.

2. Protocol Overview

In this paper, we estimate the performance of three Distributed Hash Table protocol(Tapestry, Chord, and Kademlia) using the mentioned framework. In this section, we try to present a concise summary of each of the above-mentioned DHT and discern the tunable parameters.e.

A.Tapestry

Tapestry is a peer-to-peer overlay network which provides a distributed hash table, routing, and multicasting infrastructure for distributed applications.

Tapestry has a large identifier space. This identifier space is produced using the SHA-1 algorithm. This is a 160-bit identifier space represented by 40 digit hex key. In this large identifier space, each node is assigned an unique node id. NodeIDs and GUIDs are roughly evenly distributed in the overlay network with each node storing several different IDs. Multiple applications sharing the same overlay network increases efficiency as tapestry's efficiency increases with network size.

B. Chord

To locate a piece of data in the P2P system we use Chord where every node, where data is stored, is mapped to a key in a distributed system in a scalable manner.

Using SHA1 Cryptographic Hash Algorithm chord uses uniform hashing to map every node and data to an m bit identifier. The address of both the node and data item is the SHA1 of IP in the case of node and content in the case of the data item. Considering identifiers are arranged in a circular manner and every data item is mapped to a node whose id is equal to data item's id. Every node stores routing information about nodes 1, 2, 4, 8 ... hops away. This information is stored in the finger table.

On receiving a key to lookup, a node reroutes it to another node close to the key's id. Stabilization protocol keeps the finger table correct in case of failure. Successor list contains information about r next successors in the circle. On failure of a node, it will be replaced with the next live node in its successor table.

C. Kademlia

One of the most popular peer-to-peer (P2P) architecture in use today is Kademlia. Kademlia has many advantages over other DHTs. These advantages make it a more preferable choice over other DHTs. These advantages include:

It minimizes the number of inter-node introduction messages.

Configuration knowledge such as nodes on the network and neighbouring nodes spread automatically as a result of key lookups.

Nodes in Kademlia are aware of different nodes. This enables routing queries through low latency paths.

Kademlia avoids timeout delays from failed nodes using parallel and asynchronous queries.

Kademlia is immune to some DOS attacks.

To give a simplified version of Kademlia we can delineate it as a binary tree where the leaves represent the nodes. In this tree, by tracing the bit value of the edges of the node we can find the participating key. Over this, the Kademlia protocol ensures that every node knows at least one node in each of its subtrees if that subtree contains a node.

The closeness of keys x and y are taken by computing the XOR value of the two keys. Kademlia converges to the lookup target in many steps by finding the successively closer nodes to the desired ID. Using a replication parameter Kademlia specifies the number of nodes on which a data should be replicated. Routing table and network information of a node are stored at the node state and at each remaining node the routing table containing contacts of other DHT nodes is stored. This routing table is made up of 160 buckets with each bucket storing k contacts at different distances away from the node for some notion of distance.

Kademlia has four RPCs: PING, STORE, FIND_NODE and FIND_VALUE. The PING remote procedure call examines a node to see if it's online. The STORE remote procedure call directs a node to store a [key, value] pair for later retrieval. The FIND_NODE remote procedure call takes a 160-bit key as an argument, the recipient of the FIND_NODE remote procedure call returns knowledge for the k nodes nearest to the target id. The FIND_VALUE remote procedure call behaves like FIND_NODE returning the k nodes nearest to the target Identifier with one anomaly – on accepting a STORE key, the remote procedure call just returns the stored value.

3. Evaluation

The DHTs have been implemented in P2PSIM, a discrete-event packet-level simulator. The simulated network consists of 1,024 nodes with inter-node latencies derived from measuring the pairwise latencies of 1,024 DNS servers using the King method. The round-trip delay is 152 milliseconds. The simulator does not simulate link transmission rate or queuing delay. Only key lookup is involved.

Nodes issue lookups for random keys at interims exponentially diffused with a mean of ten minutes and nodes crash and rejoin at exponentially diffused intervals with a mean of one hour. This choice of mean session time is uniform with past studies, while the lookup rate guarantees that nodes perform numerous lookups per session. All experiments run for a duration of six hours with nodes keeping their IP address and ID for the time span of the experiment.

A. Protocol Comparison

A protocol has numerous parameters that influence its cost and performance. There is no single reliable aggregate of parameter values. Instead, there is a set of best possible cost-performance sequences: for each given cost, there is a least feasible latency, and for each latency, there is a least feasible cost.

B. Parameter Exploration

The first question to be answered is what parameters do we choose who have relative importance on the performance tradeoff for a single protocol and whether similar parameters have a similar effect on the performance of different protocols. Because of the interaction between parameters, the question is quite complex to be answered easily. After careful examination, we have chosen the below parameters for each of our protocols.

Tapestry: Base (2 – 128), Stabilization interval (36 sec – 19 min), Number of backup nodes (1–4), Number of nodes contacted during repair (1 – 20).

Chord: Number of successors (4 – 32), Finger base (2 – 128), Finger stabilization interval (40 sec – 19 min), Successor stabilization interval (4 sec – 19 min).

Kademlia: Nodes per entry (4, 8, 16, 32), Parallel lookups (1 – 10), Stabilization interval (20 min – 1 hour).

Keeping the above parameters in consideration we then plot the average lookup latency to average live bandwidth. To separate the influence of a single parameter, we calculate the convex hull segment for every value of a parameter while varying all the other parameter values. This helps in tracing the full convex hull.

Chord:

In Chord finger and successors are stabilised separately. By our observation faster rates result in wasted bandwidth while slower rates result in a greater number of timeouts during lookups. The finger stabilization interval is independent of success rate in term of performance so its value must be varied to achieve the best tradeoff. Faster finger stabilization results in lower lookup latency due to fewer timeouts, but at the at a much higher communication cost. In Chord, there is no particular best base value like in Tapestry. In Chord, there is a more involved join algorithm that samples a larger number of candidate neighbours during PNS.

Kademlia:

In Kademlia the parameter parallel lookup causes a decrease in latency at higher values but at the cost of more lookup traffic. Our observation also shows that the Kademlia stabilization interval has little impact on latency, but it does increase communication cost. Though it does decrease the number of routing table entries pointing to dead nodes and thus limits the number of timeouts during lookups. However, parallel lookups already assure that these timeouts are not on the critical path for lookups, so their exclusion does not decrease lookup latency

Tapestry

The latency results in Tapestry are a bit counter-intuitive: every value of base is able to achieve the identical lookup performance, even though a meagerer base results in more hops per lookup on average. This response is due to Tapestry's contiguity routing. The starting hops in every lookup tend to be to nearby neighbours, and so the time for the lookup becomes overshadowed by the last hop, which is vital to a random node in the network. Therefore, in a protocol with proximity routing, the base can be set to be a small value in order to preserve bandwidth costs due to stabilization.

As more and more nodes stabilize, they achieve lower latencies by evading more timeouts on the critical path of a lookup. Although this development comes at the cost of bandwidth, the outcomes show that the cost in bandwidth is peripheral when compared to the savings in lookup latency. Thus, along with setting the base low, we can also frequently stabilise to keep routing table up to date.

CONCLUSION

In this paper, we present a unified framework for studying the cost versus lookup latency tradeoffs in various DHT protocols and evaluates Tapestry, Chord, and Kademlia in that framework. Under the discussed conditions, these protocols can achieve identical performance if parameters are adequately well-tuned. However, because of the interaction of parameters within a protocol, the cost versus performance tradeoff can be affected, but similar parameters in other protocols, such as base and stabilization interval, can behave conversely.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. (references)

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in the press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.