Proposal

Andrew Rosen

September 22, 2014

Contents

Chapter 1

Hypotheses

1.1 DHTs are better for distributed computing under many circumstances

Distributed Hash Tables (DHTs) are traditionally used as the backbone of P2P file-sharing applications.

However, they are seeing increasing use in other applications

- Citation(Hybrid computation)
- Cloud Provisioning (P2P cloud Provisioning)

1.1.1 Really Cool Qualities

Many DHTs are built upon the assumption that they will be used in some kind of P2P application.

Robustness and Fault-Tolerance

One of the most important assumptions of DHTs is that they are deployed on a non-static network. DHTs need to be built to account for a high level what is called *churn*. Churn refers to the disruption of routing caused by the constant joining and leaving of nodes.

This is mitigated by a few factors. First, no single node is a single point of failure for the network; this is accomplished by each node in the routing table having a small portion of the both the routing table and the information stored on the DHT (see the Load Balancing property below). Second is DHTs have a cheap repair or maintenance process that takes care of damage and integrating nodes that affects a minimal number of nodes, minimizing overhead. Third, DHTs often integrate a backup process into their protocols so that when a node goes down, one of the neighbors immediately take over its responsibility. The last property is that the Hashing Algorithm used to distribute content evenly across the network(again see load balancing) also distributes nodes evenly across the DHT. This means that nodes in the same geographic region occupy vastly different positions in the keyspace. If an entire geographic region is affected by a network outage,

Size	Speed	DHTs	Comments
$O(d)$	$O(n^{\frac{1}{d}})$	CAN [2]	d is the number of dimensions
$O(\log n)$	$O(\log n)$	Chord [3], Kademlia [4]	This is where most DHTs fall
$O(n)$			

Table 1.1: The different ratios and their associated DHTs

this damage is spread evenly across the DHT, which can be manageable This property is the most important, as it deals with failure of entire sections of the network, rather than a single node.

Recent research in using DHTs for High End Computing [1] shows what can happen if we remove this assumption by placing the network that is almost completely static.

Load Balancing

All Distributed Hash Tables associate nodes and file identifiers with keys. These keys are generated by passing the identifiers into a hash function, typically SHA-160. The chosen hash function is typically large enough to avoid hash collisions and generates keys in a uniform manner. The result of this is that as more nodes join the network, the distribution of nodes in the keyspace becomes more uniform, as does the distribution of files.

However, because this is a random process, it is possible that loads can become imbalanced. Solutions exist []

While most DHTs follow this scheme, there is some minor variation on how keys are generated. Some DHTs can or do use geographic information to generate keys. In VHash, keys are not static, but move according to a simplified spring model.

Scalability

There have been multiple proposed strategies for tackling scalablity, and it is these strategies which play the greatest role in driving the variety of DHT architectures. Each DHT must strikes a balance between memory cost of the peerlist and lookup time. The vast majority of DHTs balance the size of the lookup table Table 1.1 lists the

Other factors: Bases, maintenance process, maintenance intervals,

Heterogeneity

Ease of Adding, Removing, Maintaining, and money related factors

1.1.2 The Takeaway

The consequences of the Properties

So what are the consequences of these properties?

- DHTs are highly resilient to damage and often at higher rates of disruption that is normally.
-

•

The consequences of the Properties

1.2 Different or subproblem: Certain DHTs are better at one application than another due to differences

1.2.1 Design Differences Impacts

1.2.2 Geometries

1.2.3 Routing Table Construction

1.2.4 Implementation Differences Impacts

Recursive or iterative seek

1.2.5 MANETS

Chapter 2

Justification and Why I Think It's Cool

2.1 Why DHTs for distributed computing?

2.1.1 DHTs well understood

2.1.2 DHTs are Highly used for their intended purposed

Bittorrent, WoW

2.2 DHTs are being effectively leveraged for other things besides file sharing already

PaaS

Load Balancing in the cloud

Computing is a natural extension

Chapter 3

Possible Experiments and Applications

3.1 MapReduce

3.1.1 Current MapReduce DHT/P2P combos

ChordReduce

3.1.2 Experiment: Comparison of MapReduce paradigm on different DHTs

3.2 High End Computing

3.2.1 Metadata Management

3.2.2 Robustness

3.3 Graph Processing on a DHT

3.3.1 Embedding

3.3.2 Distribute the work for solving a graph on a DHT

3.3.3 Comparison to well established or state of the art methods

3.4 Machine Learning Problems on A DHT

Bayesian Learning

3.5 DHTs as a volunteer Platform

Chapter 4

DHT Background

Bibliography

[1] T. Li, X. Zhou, K. Brandstatter, D. Zhao, K. Wang, A. Rajendran, Z. Zhang, and I. Raicu, "Zht: A light-weight reliable persistent dynamic scalable zero-hop distributed hash table," in *Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on*, pp. 775–787, IEEE, 2013.

[2] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," 2001.

[3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," *SIGCOMM Comput. Commun. Rev.*, vol. 31, pp. 149–160, August 2001.

[4] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems*, pp. 53–65, Springer, 2002.