Poster: Towards A Fully Distributed N-Tuple Store

Yan Shvartzshnaider
NICTA and The University of Sydney
Sydney, Australia
yan.shvartzshnaider@sydney.edu.au

Maximilian Ott
NICTA
Sydney, Australia
max.ott@nicta.com.au

ABSTRACT

We present our work towards building a novel distributed n-tuple store by extending the Kademlia DHT [1] algorithm to support n dimensional keys as well as an *multi_get* operator, where some of the dimensions of the "query" key can be left unspecified.

Categories and Subject Descriptors

C.2 [**Computer-Communications Networks**]: Distributed Systems

General Terms

Algorithm, Design, Theory

Keywords

Distributed Pattern Matching, Kademlia

1. MOTIVATION

The growing volumes and rapidly changing heterogeneous nature of the data behind popular online services such as Amazon, Google, and Facebook pose new challenges to traditional data storage solutions. A possible alternative comes in the emerging NoSQL (Not Just SQL) systems. This class of systems can be roughly sorted into two categories: some that offer different "schema-less" storage abstractions to capture and describe data, and other that loosen the ACID properties to offer a more horizontally scalable storage. The choice of a particular solution comes down to a trade-off between expressiveness and scalability. This is mainly due to the fact that, many of the highly scalable NoSQL systems use key/value stores such as Distributed Hash Table (DHT) as a building block.

Despite providing features such as self-organisation, high availability, fault and network partition tolerance, DHT-based systems by design operate with single dimensional data, which significantly restricts the expressiveness of queries running on top. In particular, multi-attribute queries are not trivial to implement.

Much of prior work, such as [2], has focused on facilitating a rendezvous between the query and the published data, either through adding a supplementary indices layer or replacing the key elements of the DHT (e.g., hash function) to ensure a non-uniform distribution. The rendezvous approach suffers from load balancing issues and requires extra effort to mitigate the overall impact on the system.

Motivated by these challenges, we present our work towards a novel n-tuple store. We extended the Kademlia DHT [1] algorithm to support n dimensional keys as well as an *multi_get* operator, where some of the dimensions of the "query" key can be left unspecified.

For many of the use cases we are considering, the keys within each dimension would not be uniformly distributed, but there is usually little correlation between dimensions. Concatenation of the n keys from each dimension, followed by a sequence of bit swapping operations leads to fairly well behaved one-dimensional flat keys. In such a scheme, query keys with unspecified dimensions map into bit patterns with wildcards.

In this paper we describe our extensions to the Kademlia algorithm and protocol to support queries with unconstraint wildcard patterns and present preliminary results on its performance.

2. PROBLEM DEFINITION

We first formalise the concepts on which we base our problem.

We define a **key** as a bit vector of length K and a **pattern** as a pair of bit vectors key_p and $mask_p$, where $mask_p$ is also of length K and any "0" denoting a wildcard at that vector index. We further denote a **P-distance** (pd) as the distance of a *key* to a pattern $\mathbb{P}$ that is calculated as $pd = (key \oplus key_p) \wedge mask_p$. The **matching set** for a pattern $\mathbb{P}$ is defined as the set of all keys for which $pd(key, P) = 0$.

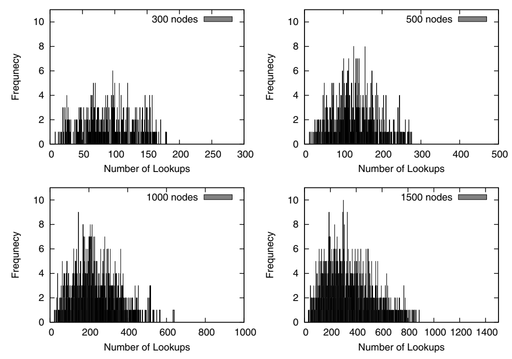
We also define a network $\mathbb{W}$ consisting of N nodes, where each node selects a random key as its ID and "knows" at least one node from any of the $log_2(K)$ regions as long as that region contains a node. Finally, we define a "lookup" as a RPC call for nodes to discover additional nodes from nodes they already know.

Problem Statement: *Define an algorithm which allows a node to obtain the complete matching set for a given pattern $\mathbb{P}$ and network $\mathbb{W}$ with the minimum number of lookups.*

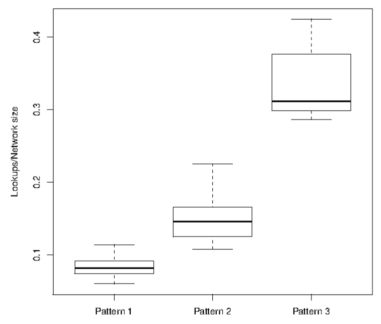
3. APPROACH

We introduce a new *multi_get* operator and a QUERY _PATTERN RPC to the Kademlia DHT protocol.

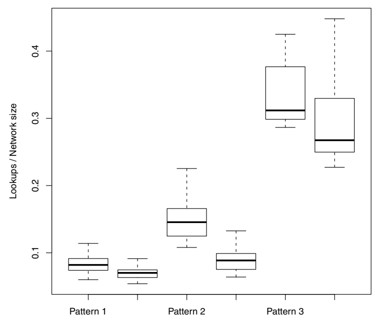
The *multi_get* takes a pattern as a parameter and returns the matching set over all known keys in the systems with their respective values. The QUERY_PATTERN(pattern)



(a) Randomly generated patterns.



(b) The ratio of number of lookups to network size for Pattern 1, 2 and 3



(c) Heuristic vs. Non-heuristic approach.

Figure 1: The experimental results

call returns the k nearest nodes to the specified pattern from the set of nodes known to the call's receiver.

Our algorithm is comprised of the following steps: *1)* Initialise a "target" set with the k nearest nodes from the local routing table. *2)* Select an "unused" node from the "target" set, call QUERY_PATTERN on it, and add all the returned nodes to the set. *3)* Repeat (2) until all nodes in the "target" set have been called.

4. PRELIMINARY RESULTS

We have run several experiments to evaluate our algorithm using the Oversim [3] P2P network simulator with the following Kademlia DHT parameters: $k - bucket$ size 16, id length $160bit$. We create random patterns using the *generate_pattern(msb_window, w)* function, which returns a pattern with w wildcards randomly distributed within the most significant *msb_window* bits. By selecting $msb\_window < log_2(N)$ the probability of any wildcard bit affecting the matching set is high. Throughout the experiments we set $msb\ window = 10$ and $w = 5$. All experiments run without network churn.

We performed three different experiments. In the first experiment, each node generated a random pattern and executed a local *multi_get* operation. We ran this for four different networks comprised of 300, 5000, 1000, 1500 nodes, respectively and after the completion of OverSim's *init* phase. Figure 1(a) depicts the lookup count distribution for the four network configurations. We do observe a fairly high number of lookups in a considerable number of cases which is the result of our overly conservative search strategy. We have begun work on incorporating network size estimations, which will allow us to terminate searches much earlier.

For the second set of experiments, we selected three different patterns 1, 2, and 3, which consistently incurred a minimum, average and maximum number of lookups, respectively. We then constructed ten differently sized networks ranging from 500 to 5000 nodes. For each combination of pattern and network, we randomly picked 20 nodes, which executed a local *multi_get* using the same pattern and reported the number of lookup. Figure 1(b) depicts the box plot of all the measurements for each pattern normalised by the respective network size. This demonstrates that the positioning of wildcard bits has a consistent impact on the number of lookups irrespective of network size.

For the third experiment we are using the same setup as in the previous experiment, but compare the baseline algorithm described above with one using a 'greedy'-type heuristic to determine which nodes to query next. Due to lack of space, we omit details of this approach because we believe there is further room for improvements. Instead, we focus on showing that such improvements are indeed statistically significant. Figure 1(c) shows the same type of box plot as before, but two box plots per pattern, one for the baseline, and one for the heuristic approach. To further validate our claims we performed a statistical analysis of variance (two way ANOVA test) to measure the impact of the two factors: the pattern composition with three levels of treatment (pattern 1, 2, and 3) and the algorithm with two levels (base and heuristics). The ANOVA test results confirm that both factors have statistically significant impact.

5. CONCLUSION

The above preliminary results show that is possible to get all the matching keys in a reasonable number of queries, particularly, while using a specially constructed pattern. We look to build on these results to optimise further the algorithm to develop a distributed n-tuple store.

6. REFERENCES

[1] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Proc. of IPTPS02, Cambridge, USA*, vol. 1, pp. 2–2, 2002.

[2] M. Cai and M. Frank, "RDFPeers: a scalable distributed RDF repository based on a structured peer-to-peer network," in *Proc. of the 13th Int. Conf. on World Wide Web*, 2004, p. 657.

[3] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *IEEE Global Internet Symposium*. IEEE, 2007, pp. 79–84.