Hyperspaces for Object Clustering and Approximate Matching in Peer-to-Peer Overlays

Bernard Wong Ýmir Vigfússon Emin Gün Sirer
Dept. of Computer Science, Cornell University, Ithaca, NY 14853
bwong@cs.cornell.edu ymir@cs.cornell.edu egs@cs.cornell.edu

Abstract

Existing distributed hash tables provide efficient mechanisms for storing and retrieving a data item based on an exact key, but are unsuitable when the search key is similar, but not identical, to the key used to store the data item. In this paper, we present a scalable and efficient peer-to-peer system with a new search primitive that can efficiently find the k data items with keys closest to the search key. The system works via a novel assignment of virtual coordinates to each object in a high-dimensional, synthetic space such that the proximity between two points in the coordinate space is correlated with the similarity between the strings that the points represent. We examine the feasibility of this approach for efficient, peer-to-peer search on inexact string keys, and show that the system provides a robust method to handle key perturbations that naturally occur in applications, such as file-sharing networks, where the query strings are provided by users.

1 INTRODUCTION

Modern P2P substrates do not provide support for efficiently locating objects whose keys are not known precisely. In settings where queries are based on terms provided by users, imprecision stemming from partial specifications of keywords, common variations of search terms and misspellings are unavoidable. For instance, approximately 20% of all Google queries for "Britney Spears" misspell the artist's name [2]. Efficiently routing a query to a set of objects whose keys are close[1] but not identical to the search key is a difficult problem known as "approximate match."

Even though peer-to-peer systems were initially motivated by file-sharing, modern P2P substrates do not provide efficient primitives for approximate matching. Unstructured peer-to-peer systems [1] provide a *search* primitive, which is based effectively on query broadcast[2]. Gnutella nodes receiving the search query match it against their database of known items using a fuzzy similarity metric to yield the approximate matches. Such broadcast-based approaches are inefficient as they may take up to N hops in the worst case where N is the number of hosts, and place a super-linear aggregate load on the network. In contrast, structured peer-to-peer systems [21, 23, 28, 19, 16, 12] provide an efficient *lookup* primitive that can typically locate a target within $\log N$ hops. While these systems provide strong worst-case bounds, the lookup operation does not permit approximate matching. Naive approaches to layer approximate matching on top of a DHT lookup, by inserting each object under all possible key variations and performing every query in parallel with all k variants of the search key, lead to highly inefficient solutions because k is typically on the order of a few hundred even for a moderate length movie title with only two permuted characters. Finally, systems that permit *range lookups* [6, 8] can perform a lookup within a range defined by numeric coordinates, but are difficult to adopt for use with approximate string matching. Overall, existing systems provide inefficient and approximate search or efficient and precise lookup, but not efficient and approximate match.

In this paper, we present e-llama, a scalable peer-to-peer system that can efficiently find the k closest data items for any search key. The central insight behind e-llama is to define a very high-dimensional space (a *hyperspace*) in which every object and node is assigned a virtual coordinate. The bases (axes) for the hyperspace consist of string labels. The virtual coordinate for every object is the tuple created by measuring the edit-distance to each of the axis labels. For instance, for bases **aaa**, **cbc** and **abd**, the keys **abc**, **abd** and **ddd** would map to the points $\langle 2, 1, 1\rangle$, $\langle 2, 2, 0\rangle$ and $\langle 3, 3, 2\rangle$, respectively. This virtual coordinate assignment captures the relative similarities of the strings

[1]Closeness here is defined based on an application-supplied similarity measure, such as edit distance. Our approach requires only that this measure σ obey $\sigma(a, b) \geq 0$, $\sigma(a, b) = \sigma(b, a)$ and $\sigma(a, c) \leq \sigma(a, b) + \sigma(b, c)$. We use edit-distance as a running example without loss of generality.

[2]Optimizations, such as supernodes and expanding ring search, make the broadcast process more efficient, but the primitives are still based fundamentally on flooding.

through the edit-distances to the axis labels.

An efficient algorithm, based on small-worlds [13], for navigating this multi-dimensional hyperspace enables e-llama to quickly identify approximately matching objects. E-llama assigns a random location in hyperspace to each overlay node, and each node maintains the set of objects for which it is the closest node. Every node also keeps track of other nodes in concentric rings of exponential radii. E-llama routes a query to the closest node for a target coordinate by greedily determining the peer in its rings that is closest to the target coordinate and forwarding the query. Each forward brings the query closer to the target coordinate and to a node with more information in the proximity of the targeted region than the previous node. This protocol converges to the closest node in $O(\log N)$ hops with high probability. Once the target node has been located, the search expands in a sphere around the target until k matching objects are found.

Overall, this paper makes three contributions. First, it describes a new technique for constructing a synthetic space in which similar keys are clustered. Second, it describes a scalable and efficient protocol for mapping this space to nodes and routing queries to nodes, yielding a DHT with an approximate match primitive. Finally, it demonstrates the feasibility of the system and analyzes the effects of various system parameters.

2 HYPERSPACE

2.1 Basis Selection

Creating a hyperspace that can provide fine-grain differentiation of different objects requires a careful selection of string labels as dimension bases (axes). In e-llama, the virtual coordinate of every object is the tuple created from its edit-distance to each of the axis labels. In essence, axis labels act as anchor points, and each component of an object's coordinate provides the distance of the object from that anchor point. Much like the Post Office metric on normed vector space [5], the distance from each anchor point clusters similar objects to the extent differentiable by that axis label, assigning them similar coordinates. However, a poor selection or an insufficient number of bases can assign similar coordinates to even dissimilar objects. For example, if axis labels are random strings, it is likely that each label will have a very similar edit-distance to any real English string of the same length. A careful selection of the axis labels is important, as the labels define the hyperspace in which keys will be clustered.

Labels that have some similarity to the actual objects in the system can help accentuate the differences in dissimilar objects. Intuitively, given sufficient labels, each object has a high probability of resembling some labels, the set of which is different and distinct from the set of labels resembling dissimilar objects. This intuition leads to simply selecting axis labels from actual objects. For example, in a deployment with movie titles as objects, a small random sampling of movie titles can serve as a set of axis labels. Once selected, axis labels do not need to change as long as the distribution of object names is relatively stable.

In addition to the selection of axis labels, the number of dimensions also plays an important role in creating an effective hyperspace for differentiating dissimilar objects. Increasing the number of bases should, intuitively, widen the separation of coordinates between dissimilar strings as it becomes increasingly unlikely for them to have the same edit-distance to a large number of independently chosen axis labels. The cost of additional bases is low, requiring only minimal increases in string storage and bandwidth. As we will see later in section 4, increasing the number of bases significantly improves the distinguishing power of the hyperspace yet incurs relatively little overhead.

Note that the Euclidean distance between the coordinates for two strings s_1 and s_2 is loosely related to the edit-distance between these two strings. In the worst case, these two strings might require $n = ||s_1|| = ||s_2||$ many insertions, deletions and replacements from the axis labels, and hence share the same coordinate, and the edit-distance between s_1 and s_2 might be $2n$. This bound forces proximity between related strings, and proper choice of independent axis labels forces dissimilar strings to acquire divergent coordinates in practice, as we show later in section 4.

2.2 Node ID Assignment

Similar to objects, nodes are assigned coordinates in hyperspace. Much like DHTs, each e-llama node is responsible for storing the set of objects for which it is the closest node. The assignment of node coordinates involves a subtlety. While any random assignment of coordinates to nodes will lead to a correct system that will work, the strong result that assures that routing will be performed in O($\log N$) hops [26] requires that nodes be distributed in hyperspace according to the same distribution as objects. To ensure that this is the case, node IDs in e-llama are determined by random sampling. Each node independently selects a random object name, determines its coordinate, and adopts that location as its identifier. Nodes ensure uniqueness by detecting coordinate collisions at join time. As in axis labels, node coordinates do not need to change as long as the distribution of object coordinates remains relatively stable.

3 ROUTING FRAMEWORK

The basic e-llama routing framework relies on multi-resolution rings to organize peers, a ring membership replacement scheme to maximize the usefulness of ring members, and a gossip protocol for node discovery and membership dissemination. Finding the k data items with keys closest to the search key involves two phases. First, a multi-hop query routing protocol finds the closest node to the search key. Once the closest node is found, it recursively queries its nearby peers to determine the k closest data items.

3.1 Multi-Resolution Rings

The intuitive reason for the multi-resolution ring structure for organizing peers is to provide each node with near authoritative information on nodes that are near it, but also provide a sufficient number of out-pointers to far-away nodes to allow large hops and facilitate rapid search. Each e-llama node organizes its peers into a set of concentric rings centered on itself, where the ring distance is a measure of its Euclidean distance to the node. The ith ring has inner radius $r_i = \alpha s^{i-1}$ and outer radius $R_i = \alpha s^i$, for $i > 0$, where α is a constant, s is the multiplicative increase factor, and $r_0 = 0$, $R_0 = \alpha$ for the innermost ring. Each node keeps track of a finite number of rings; all rings $i > i^*$ for a system-wide constant i^* are collapsed into a single, outermost ring that spans the range $[\alpha s^{i^*}, \infty]$.

3.2 Ring Membership Management

The number of nodes per ring, p, represents a trade-off between accuracy and overhead. A large p allows each node to retain more information for better route selection during query routing, but requires additional overhead in both memory and bandwidth. The utility of a ring member is in relationship to the amount of diversity it can provide to the ring. For each ring, the node retains $p + l$ members, where l is a constant number of additional nodes that serve as potential ring candidates for use in the next ring membership selection process. During ring membership selection, an infrequent periodic event, the subset of p nodes from the $p + l$ members that forms a polytope with the largest hypervolume based on their coordinates are kept as ring members.

3.3 Gossip Based Node Discovery

A standard anti-entropy push protocol [10] provides node discovery and dissemination between e-llama nodes. At each gossip round, an e-llama node collects a random selection of its ring members, and sends this collection to a random member in each of its rings. The receiving node contacts each peer in the collection to discover their coordinates, and these peers are then stored as potential replacement for the node's current primary ring members.

3.4 Query Routing

Locating the closest node to the search key, the first phase in finding the k closest data items, involves a multi-hop search where each hop exponentially reduces the distance to the closest node. On receiving a query, an e-llama node determines its closest peer to the search key's coordinate. If the closest peer is closer to the key than the current node by some threshold, the node forwards the query to the peer. Otherwise, the node selects the closest peer or itself, whichever is closer to the key, as the closest node. This query routing protocol can find the closest node in $O(\log N)$ hops with high probability [25].

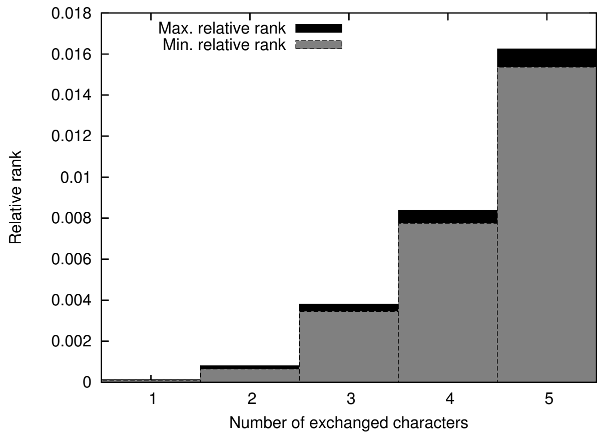

Figure 1: The relative rank of the actual string across different number of character exchanges. Less than 0.02 of the total entries need to be searched on average before finding the actual string for query strings with up to 5 characters exchanged.

3.5 K-Clustering

Once the closest node to the search key is found, e-llama determines all objects within a sphere centered around the key's coordinate, expanding the sphere until the k closest data items are inside. The protocol begins with the closest node asking all its neighbors within a distance of h from the search key's coordinate to recursively determine the k closest data items using the same query ID. Given that a node only responds to a search request once per query ID, the recursion terminates when all nodes within the sphere returns what each thinks is the k closest data items. The closest node receives up to k data items from each of its peers within the sphere, and determine the k actual closest items. The protocol repeats with a larger sphere if there are less than k data items within the previous sphere.

4 EVALUATION

We evaluate e-llama by applying it to synthetic searches based on the NetFlix database [3], which consists of a listing of 17770 movie titles. The first test consists of 1000 randomly chosen movie titles with a small number of characters exchanged to simulate typos and spelling variations. For the second test, 6552 randomly selected movie titles were modified with real human typos and misspellings from the Searchspell database [4].

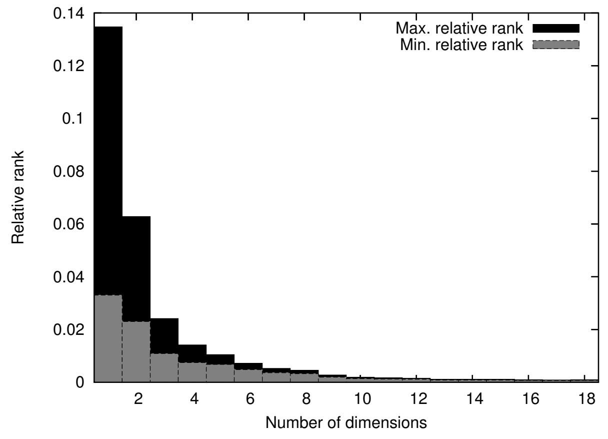

Figure 2: The relative rank of the actual string across different number of dimensions. A small increase in the number of dimensions can dramatically improve the relative rank.

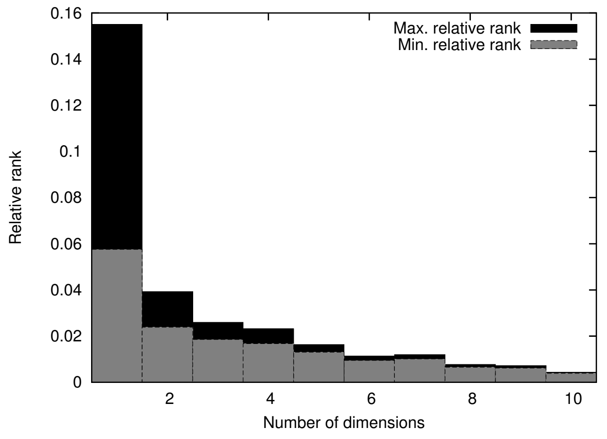

Figure 3: The relative rank of a movie title to one with human typos across different number of dimensions. As before, the relative rank decays rapidly as the number of dimensions increases.

We investigated several schemes for axis label selection which includes using random strings of both short and long length, using random English strings taken from a dictionary, and using randomly chosen titles from the movie database. We found that using movie titles as axis labels consistently outperforms the other two schemes, and use it exclusively for the rest of the evaluation.

We evaluate the effectiveness of e-llama's approximate match using the relative rank of the original string in the search results for the modified query string. We order the results of the approximate match based on the Euclidean distance of coordinates we defined earlier. Thus, if the original string has the lowest Euclidean distance to the query string, its absolute rank is 1; the absolute rank is 2 if one other string is closer to the query string, and so forth. The relative rank is the rank divided by the total number of movie titles in the database. Relative rank captures the average cluster size necessary to contain the desired object and thus the percentage of results that a user must check before locating the intended object. Since several titles commonly share the same distance to a modified query string, we show both the lowest and highest relative rank of the original string.

We first examine the change in relative rank from different numbers of perturbations to the original movie title. In this experiment, the number of dimensions is fixed at 20. Figure 1 shows that increasing the number of modifications significantly increases the difficulty of the problem. With five characters exchanged, the actual object has an average relative rank less than 0.02. In other words, the object resides within a cluster that contains less than two percent of the total number of movies on average. For perturbations of only one character, the average cluster size is less than 0.01 percent of the number of movie titles. These results suggest that the e-llama technique can return a very small cluster that nevertheless contains the desired object, even with very few dimensions in the hyperspace. A small cluster size limits the number of results the users must manually look through to find their actual object. A less encouraging result is the exponential growth in the relative rank from the linear increase in the number of perturbations.

Fortunately, Figure 2 shows that the relative rank also decreases exponentially with a linear increase in the number of dimensions. In this experiment, we fix the number of perturbations at two. With only one dimension, the relative rank is nearly 0.14. This suggests that the approximate string matching problem is not well suited to solutions using distributed hash tables which are restricted to a one dimensional space. However, with a modest increase from one to five dimensions, the relative rank falls significantly to approximately 0.01. The same phenomenon can be seen in Figure 3. The experiment consists of 6552 randomly selected movie titles, each of which has 80% of its words replaced by a mistyped or misspelled version from the Searchspell database [4] of human typos and spelling errors. On average, the edit distance between the queries and the actual titles is 4.9. In this setting, an increase from one to five dimensions decreases the relative rank to below 0.02. Additional dimensions require only small and inconsequential increases in bandwidth and computational overhead for most practical applications. Applications that receive search strings with a high number of perturbations can simply configure their deployment with a higher number of dimensions in order to arrive at their desired average relative rank.

5 RELATED WORK

E-llama is a distributed hash table that provides a novel approximate match primitive. It differs from previous DHTs [21, 23, 28, 19, 16, 12], which support only precise lookups.

The high dimensional hyperspace in e-llama is similar conceptually to virtual coordinate schemes for estimating inter-node latencies [17, 9, 14, 24, 22, 18]. However, increasing the number of dimensions entails very little associated cost in e-llama as the coordinates are completely synthetic and do not require network pings for assignment. The low cost of coordinate assignment and comparison also renders expensive techniques for reducing dimensions unnecessary.

Query routing in e-llama most closely resembles routing in Meridian [25], Small-World networks [13], and CAN [20]. In CAN, each node knows its immediate closest neighbor in each of the dimensions and greedily routes to the destination. However, border cases in dealing with churn makes CAN difficult to implement and deploy in practice. Small-World networks introduce long links between peers to reduce the number of routing hops to $O(\log^2 N)$. The query routing in e-llama is similar to Small-World network routing but reduces the number of hops to $O(\log N)$ by introducing additional structure. Meridian uses a similar multi-resolution ring structure as e-llama, but focuses on operating in non-grid like metric spaces and has no notion of absolute position of any of the nodes.

Efficient similarity comparison of strings are typically based on sampling techniques [11, 7, 15], where portions of a string represent the entire string. In these techniques, each string is broken down into a number of overlapping sub-strings of fixed length and each individual sub-string is hashed. A consistent sampling of approximately k sub-string hashes is taken from each string as fingerprints, and the *resemblance* of two strings is the number of shared sub-strings in the fingerprint, divided by the total number of unique sub-strings from the fingerprints. These schemes differ from e-llama as they require a centralize system for performing string comparison, where e-llama provides a distributed and peer-to-peer solution.

In [27], the authors use the Soundex algorithm to encode keywords by their phonemes before indexing them in a DHT. Unlike edit distance, Soundex is appropriate only for English keywords and is not effective against typing errors.

6 SUMMARY

In this paper, we described a new technique for efficient approximate matching in peer-to-peer overlays. The technique is scalable, efficient, and of immediate applicability to domains, such as peer-to-peer filesharing, where query terms are provided by users and require an approximate match against objects in the system. More generally, we presented an object clustering technique based on creating a synthetic, high-dimensional space and assigning coordinates to objects in this space where Euclidean distances capture similarity. Given appropriately chosen axis labels, such a mapping can facilitate the identification of similar objects. We showed how coupling such a space with an efficient search function based on small-worlds can yield an efficient and scalable system. This overall approach may be applicable to other domains where a similarity-based clustering of objects is desired.

References

[1] Gnutella. http://www.gnutella.com/.

[2] Britney Spears spelling correction. http://www.google.com/jobs/britney.html.

[3] Netflix Prize. http://www.netflixprize.com.

[4] Searchspell. http://www.searchspell.com/typo/.

[5] Metric space. http://en.wikipedia.org/wiki/Metric_space.

[6] A. Bharambe, M. Agrawal and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *SIGCOMM*, Portland, Oregon, August 2004.

[7] A. Broder, S. Glassman and M. Manasse. Syntactic Clustering of the Web. In *World Wide Web Conference*, Santa Clara, California, April 1997.

[8] A. Crainiceanu, P. Linga, J. Gehrke and J. Shanmugasundaram. Querying Peer-to-Peer Networks Using P-Trees. In *WebDB*, Paris, France, June 2004.

[9] F. Dabek, R. Cox, F. Kaashoek and R. Morris. Vivaldi: A Decentralized Network Coordinate System. In *SIGCOMM*, Portland, Oregon, August 2004.

[10] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC*, Vancouver, Canada, August 1987.

[11] N. Heintze. Scalable Document Fingerprinting. In *Workshop on Electronic Commerce*, Oakland, California, November 1996.

[12] F. Kaashoek and D. Karger. Koorde: A Simple Degree-Optimal Distributed Hash Table. In *IPTPS Workshop*, Berkeley, California, February 2003.

[13] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *STOC*, Portland, Oregon, May 2000.

[14] H. Lim, J. Hou and C. Choi. Constructing Internet Coordinate System Based on Delay Measurement. In *IMC*, Miami Beach, Florida, October 2003.

[15] U. Manber. Finding Similar Files in a Large File System. In *Winter Technical Conference*, San Francisco, California, January 1994.

[16] P. Maymounkov and D. Mazieres. Kademlia: A Peer-to-Peer Information System Based on the XOR Metric. In *IPTPS Workshop*, Cambridge, Massachusetts, March 2002.

[17] T. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *INFOCOM*, New York, New York, June 2002.

[18] M. Pias, J. Crowcroft, S. Wilbur, T. Harris and S. Bhatti. Lighthouses for Scalable Distributed Location. In *Intl. Workshop on P2P Systems*, Berkeley, California, February 2003.

[19] S. Ratnasamy, P. Francis, M. Hadley, R. Karp and S. Shenker. A Scalable Content-Addressable Network. In *SIGCOMM*, San Diego, California, August 2001.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker. A Scalable Content-Addressable Network. In *SIGCOMM*, San Diego, California, August 2001.

[21] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, Heidelberg, Germany, November 2001.

[22] Y. Shavitt and T. Tankel. Big-Bang Simulation for Embedding Network Distances in Euclidean Space. In *INFOCOM*, San Francisco, California, March 2003.

[23] I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *SIGCOMM*, San Diego, California, August 2001.

[24] L. Tang and M. Crovella. Virtual Landmarks for the Internet. In *IMC*, Miami Beach, Florida, October 2003.

[25] B. Wong, A. Slivkins and E. G. Sirer. Meridian: A Lightweight Network Location Service without Virtual Coordinates. In *SIGCOMM*, Philadelphia, Pennsylvania, September 2005.

[26] B. Wong, A. Slivkins and E. G. Sirer. A Framework for Network Location-Aware Node Selection. *TOCS (in submission)*.

[27] M. Zaharia, A. Chandel, S. Saroiu and S. Keshav. Finding Content in File-Sharing Networks When You Can't Even Spell. In *Intl. Workshop on P2P Systems*, Bellevue, Washington, February 2007.

[28] B. Zhao, J. Kubiatowicz and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, Berkeley, California, April 2001.