Ranged Hash Functions and the Price of Churn

James Aspnes*[†] Muli Safra[‡] Yitong Yin*[§]

Abstract

Ranged hash functions generalize hash tables to the setting where hash buckets may come and go over time, a typical case in distributed settings where hash buckets may correspond to unreliable servers or network connections. Monotone ranged hash functions are a particular class of ranged hash functions that minimize item reassignments in response to *churn*: changes in the set of available buckets. The canonical example of a monotone ranged hash function is the ring-based consistent hashing mechanism of Karger *et al.* [13]. These hash functions give a maximum load of $\Theta\left(\frac{n}{m}\log m\right)$ when n is the number of items and m is the number of buckets. The question of whether some better bound could be obtained using a more sophisticated hash function has remained open.

We resolve this question by showing two lower bounds. First, the maximum load of any randomized monotone ranged hash function is $\Omega(\sqrt{\frac{n}{m}\ln m})$ when $n = o(m\log m)$. This bound covers almost all of the nontrivial case, because when $n = \Omega(m\log m)$ simple random assignment matches the trivial lower bound of $\Omega(n/m)$. We give a matching (though impractical) upper bound that shows that our lower bound is tight over almost all of its range. Second, for randomized monotone ranged hash functions derived from metric spaces, there is a further trade-off between the expansion factor of the metric and the load balance, which for the special case of growth-restricted metrics gives a bound of $\Omega\left(\frac{n}{m}\log m\right)$, asymptotically equal to that of consistent hashing. These are the first known non-trivial lower bounds for ranged hash functions. They also explain why in ten years no better ranged hash functions have arisen to replace consistent hashing.

1 Introduction

Hash tables are one of the oldest tools in computer science. In the classic model of a hash table, an unknown data set from a large domain is assigned to a fixed number of buckets, through a consistent mapping (the hash function) from the data domain to the buckets. However, in many applications in today's computing systems, such as peer-to-peer systems [22], or Internet routers [3], not only is the current data set (keys, network flows) unknown, but the availability of buckets (peers, peer links) also changes over time. This change in the set of participants of the system is called **churn** [5], and it is a serious concern for any large distributed system.

Hashing has been applied to distributed systems via **ranged hash functions**, introduced by Karger *et al.* [13]. Ranged hash functions are hash functions that depend on the set of available buckets. A typical ranged hash function hashes items to positions in some space, and then assigns each item to the nearest available bucket; as the set of buckets changes, an item may move to a new nearest available bucket. Such hash functions have the property of being **monotone**, [13] meaning that each data item has its own preference list and hashes to the first available bucket in this list. This property minimizes reassignment costs [3], and perhaps because of this, all practical ranged hash functions have this monotonicity property. This includes systems that already do load balancing by, for example, creating multiple virtual locations for each bucket [13, 22].

We are interested in obtaining lower bounds on the maximum load for any these monotone ranged hash functions, parameterized in terms of the number of data items n and the number of available buckets m. Our lower bounds hold even if: (a) the hash function is optimal for the given n and m; (b) the data set is optimal for the given hash function; and (c) the only power given to the adversary is the worst-case choice of available buckets. Furthermore, though our lower bounds are stated for a worst-case choice of buckets, the proof technique used means that they continue to hold with $1 - o(1)$ probability for a uniform random choice of buckets.

Randomization is fundamental to our problem, be-

*Department of Computer Science, Yale University.

[†]Supported in part by NSF grant CNS-0435201. Email: aspnes@cs.yale.edu.

[‡]School of Mathematical Sciences, Tel-Aviv University. Email: safra@post.tau.ac.il.

[§]Supported by a Yale University Fellowship. Email: yitong.yin@yale.edu.

cause no lower bounds have been shown for randomized ranged hash functions. Randomization is also important for practical algorithms. For the deterministic case, it is easy to obtain terrible load balancing. In the full paper, we show that all deterministic monotone ranged hash functions experience a load of $\Omega(\sqrt{n})$ in the worst case when $n = m$. A naive deterministic implementation of consistent hashing would be even worse: an adversary deleting all buckets outside of some very narrow interval can get nearly all n items in the first remaining bucket.

With randomization, the hash function can do much better. Nonetheless, we give two lower bounds. For a general monotone ranged hash function, we show a lower bound of $\Omega\left(\sqrt{\frac{n}{m}\log m}\right)$ when $n = o(m \log m)$; when $n = \Omega(m \log m)$ the trivial lower bound of $\Omega(n/m)$ matches the upper bound obtained from random assignment. We give a tight matching upper bound for the small-n case based on finding nearest neighbors in a hypercube; unfortunately, because of the difficulty of nearest-neighbors in a hypercube, this upper bound does not give a practical implementation of a distributed hash function.

Our second lower bound applies to the more practical setting of ranged hash functions based on assigning each item to the nearest bucket in a metric space, where the choice of how to embed both items and buckets in the space may be based on an arbitrary joint distribution. We show a trade-off between KR-dimension [6, 12, 16], a counting measure of **doubling dimension** and load balance; in a space with KR-dimension K, no randomized monotone ranged hash function—even one where the embedding of items and buckets is chosen according to a non-uniform non-independent distribution—can do better than $\Omega(2^{-2K} \cdot \frac{n}{m} \ln m)$. As with our general lower bound, the result continues to hold with a uniform choice of buckets with probability $1 - o(1)$.

The interesting case is when the KR-dimension is constant, since such a **growth-restricted** metric allows finding nearest neighbors quickly using known techniques [12]. Our lower bound shows that in this case, even a hash function based on a very cleverly chosen metric space, embedding, and probability distribution cannot beat the $O(\frac{n}{m} \ln m)$ upper bound of simple consistent hashing using a uniform independent distribution on a one-dimensional ring. Our lower bound thus hints at the reason for the continued practical use of this technique.

Organization. We give a formal description of our model in Section 2. Previous work is described in Section 3. Our results are described formally in Section 4, and the details are given in the following sections.

2 Model

Items and buckets. Given a domain of *items* $\mathcal{I} = [N]$, and a domain of *buckets* $\mathcal{U} = [M]$, where the buckets may become *available* and *unavailable*, we refer to the set S of available buckets as the **state**. A ranged hash function is a function in the form of $h : 2^{\mathcal{U}} \times \mathcal{I} \to \mathcal{U}$. We denote by h_S the assignment of items to buckets imposed by h for a specific state S. Naturally, we require that $h_S(\mathcal{I}) \subseteq S$ for any $S \subseteq \mathcal{U}$.

For any **data set** $D \subseteq \mathcal{I}$, and any state S, the **load** of bucket $b \in \mathcal{U}$ is denoted as $\ell_S^D(b) = |h_S^{-1}(b) \cap D|$, and we let $\ell_S^D = \max_{b \in S} \ell_S^D(b)$ be the maximum load of state S. If the data set D is the entire domain $\mathcal{I}$, we omit D and write $\ell_S(b)$ for $\ell_S^D(b)$ and ℓ_S for ℓ_S^D.

Besides load balance, another critical performance parameter of ranged hash functions is smoothness, which is characterized by the number of reassigned items going from one state to another. Smoothness represents a natural requirement for a data structure that the maintenance cost should be small when the state of the data structure changes. In [13], the property of being **monotone** is introduced to characterize those ranged hash functions with optimal smoothness. This property says that removing buckets only changes the position of items in the removed buckets: formally, a ranged hash function h is **monotone** if for all $S \subseteq T \subseteq \mathcal{U}$, $h_S(i) = h_T(i)$ if $h_T(i) \in S$. For a monotone ranged hash function, items are reassigned only if necessary.

Monotone ranged hash functions can always be described by a **preference matrix**. A preference matrix π is an $N \times M$ matrix where each row π_i is a permutation of $\mathcal{U}$. In [13], it is shown that a ranged hash function h is monotone if and only if there is a preference matrix π, such that $\pi_i^{-1}(h_S(i)) = \min_{b \in S} \pi_i^{-1}(b)$ for every S and i, i.e., if and only if every item possesses an ordering of buckets, and is always assigned to the first available bucket according to its order. Throughout this paper, we use the notations of a monotone ranged hash function h and its corresponding preference matrix π interchangeably.

In many applications of ranged hash functions, it is impractical to store a complete list of M possible buckets for every item. A general and natural methodology to efficiently represent a monotone ranged hash function is to embed $\mathcal{I}$ and $\mathcal{U}$ in a metric space and assign each item i to the closest available bucket in the current state. An implementation of ranged hash function then involves a specific embedding of items and buckets in a metric space, and a mechanism supporting nearest neighbor search (NNS) [9, 12] in that metric.

Performance measures. We consider the effect on load balance of two important properties of a ranged hash function: (a) the requirement of optimal smoothness (i.e. being monotone); and (b) the expansion properties of the underlying metric space (which may dramatically affect the hardness of searching nearest neighbors).

For this purpose, we define a measure called the **price of churn**. We define this in terms of a class of ranged hash functions sharing a particular property. For a class $\mathcal{H}$ of ranged hash functions, let $\mathcal{P} : \mathcal{H}$ denote an arbitrary probability distribution over hash functions in $\mathcal{H}$. The price of churn for $\mathcal{H}$ is defined formally as

$$\begin{aligned} & \ell_{\mathcal{H}}(N, M, n, m) \\ = & \max_{S \in \binom{\mathcal{U}}{m}} \min_{D \in \binom{\mathcal{I}}{n}} \min_{\mathcal{P}:\mathcal{H}} \sup \left\{ L \;\middle|\; \Pr_{\mathcal{P}}[\ell_S^D \geq L] = 1 - o(1) \right\}. \end{aligned}$$

This represents the lower bound on the maximum load of all randomized ranged hash functions with property $\mathcal{H}$. With $\mathcal{I} = [N]$ and $\mathcal{U} = [M]$, for any randomized ranged hash function with some property $\mathcal{H}$, for any data set $D \in \binom{\mathcal{I}}{n}$, there exists a state $S \in \binom{\mathcal{U}}{m}$, such that the maximum load ℓ_S^D is at least $\ell_{\mathcal{H}}(N, M, n, m)$ with high probability.

Note that in this notion of lower bound, the data set D is fixed by us, and then the state S is chosen by an adversary. This is different from a lower bound against both a worst-case S and a worst-case D. We use this alternative formulation because our purpose is to understand specifically how a changing set of participants may affect a system: thus we focus on the costs caused *solely* by the unknown state, rather than other issues such as an unknown data set (which has been covered by the study of hash tables). Our lower bound states that even if the system is fully optimized towards the current data set, there is still some price we have to pay for unpredictable participation in the system. Giving this additional power to the ranged hash function only strengthens our lower bounds.

3 Previous work

So far the only practical construction of ranged hash function is *consistent hashing*, which was introduced by Karger *et al.* [13] along with the notion of a ranged hash function. Here items and buckets are mapped to a uniformly random place in the continuous *unit circle* $[0, 1)$, and each item is assigned to the closet available bucket. For any data set $|D| = n$ and any state $|S| = m$, the maximum load ℓ_S^D is with high probability $\Theta(\frac{n}{m} \ln m)$.

Consistent hashing was originally designed for web caching; however, through its utility in a wide variety of distributed settings, it has become the foundation of many modern distributed systems [2, 8, 11, 21, 22]. Many systems that implement consistent hashing have tried different ways to improve load balance [1, 4, 14, 15, 17, 19]. All of these methods involve evenly cutting the unit circle by enforcing a certain pattern by which points in the unit circle are occupied. With this method, the systems sacrifice the "off-line" property of a hash function; the location of buckets now depends on the order in which they arrive and depart, requiring coordination mechanisms to manage the assignment.

The question of whether such coordination is necessary has remained open. No new ranged hash function with better performance has been proposed since consistent hashing. No non-trivial lower bound is known to us for ranged hash functions either. Although ranged hash functions as a class of meaningful mathematical objects have been known for a decade, we have little knowledge about their structure, and the relation between their parameters.

4 Our results

We prove that for the class of monotone ranged hash functions $\ell_{\mathtt{monotone}}(N, M, n, m) = \Omega(\sqrt{\frac{n}{m} \ln m})$ for the case that $n = o(m \log m)$ and $\ell_{\mathtt{monotone}}(N, M, n, m) = \Omega(\frac{n}{m})$ for the case that $n = \Omega(m \log m)$. This bound is tight for almost all settings of n and m. This shows a similar property as the price of unknown data set (in general we can do no better than random-balls-into-bins [18] against a worst-case data set): they both approach the asymptotic optimum $O(\frac{n}{m})$ as n grows to $\Omega(m \log m)$. However, the price of churn is much smaller than the skew induced by a random balls-into-bins assignment.

We then look at ranged hash functions arising from metric spaces. We explore the relation between the load balance and the expansion properties of the metric. We adopt a counting measure of doubling dimension introduced in [12], which is called KR-dimension in the literature [6,16], namely, we say the embedding of $\mathcal{U}$ has KR-dimension K if doubling the radius may increase the size of the balls in $\mathcal{U}$ by at most a factor of 2^K. We prove that for the class of ranged hash functions implemented via metric-space embedding where $\mathcal{U}$ has KR-dimension K, $\ell_{K\text{-dim}_{\mathrm{KR}}}(N, M, n, m) = \Omega(2^{-2K} \cdot \frac{n}{m} \ln m)$ if $K \leq \frac{1}{4} \log_2(\frac{n}{m} \ln m)$. Combining with a widely believed conjecture for nearest neighbor search, "the curse of dimensionality" [9], this trade-off provides us a very interesting insight: although dimensionality curses search, it blesses load balance.

When the KR-dimension is $O(1)$, the metric is called **growth-restricted**. In this case, the trade-off becomes $\ell_{O(1)\text{-dim}_{\mathrm{KR}}}(N, M, n, m) = \Omega(\frac{n}{m} \ln m)$, which is exactly the bound for the maximum load of consistent hashing.

Despite the fact that the original $[0, 1)$ metric used in consistent hashing is not in general growth-restricted for uniformly distributed buckets, our lower bound applies to consistent hashing because we can—without changing the structure of the algorithm—replace this metric with a (growth-restricted) discrete counting metric that simply counts the number of bucket locations between any two buckets. Our result thus shows that consistent hashing is optimal for all constructions arising from growth-restricted metrics. This is particularly interesting for distributed computing for two reasons. First, growth-restricted metrics are among the strongest metrics for which efficient exact nearest-neighbor search is currently possible. Second, as argued in [12], a growth-restricted metric holds in the case of many typical Internet applications. Our results thus demonstrate that consistent hashing is optimal for such applications.

Besides these specific results, we contribute new techniques to analyzing ranged hash functions. Previous approaches [13, 19] can be seen as bounding the maximum volume of Voronoi cells governed by random centers. This technique works fine for restricted cases such as constructions arising from 1-dimensional metric space, but does not appear to generalize to more general classes of (randomized) ranged hash functions. Our method is to explore directly the structure of the preference matrices. By bounding certain measures of the richness of this structure, we can prove lower bounds for general monotone ranged hash functions. For ranged hash function arising from metric spaces, we consider the connection between load balance and the density of balls in the metric space; using this we can prove a trade-off between load balance and the expansion rate.

5 Domain reduction

In order to avoid considering too many possibilities, it is helpful to be able to reduce the problem of churn to a few representative special cases. In this section, we describe how to do so.

We say a property $\mathcal{H}$ of ranged hash functions is $\mathcal{I}$-hereditary, if for any $h \in \mathcal{H}$, and any $\mathcal{I}' \subseteq \mathcal{I}$, the restriction of h on the domain $\mathcal{I}'$ still belongs to $\mathcal{H}$. We say a property $\mathcal{H}$ of ranged hash functions is probabilistically $\mathcal{U}$-hereditary, if for any $h \in \mathcal{H}$, and any $M' < M$, there is some distribution of $\mathcal{U}' \in \binom{\mathcal{U}}{M'}$ such that the restriction of h on the domain $\mathcal{U}'$ belongs to $\mathcal{H}$ with high probability.

It is easy to see that the property of being monotone is both $\mathcal{I}$-hereditary and $\mathcal{U}$-hereditary, since any restriction of a preference matrix is still a preference matrix.

The following lemma allows reducing the price of churn to a base case with small parameters.

Lemma 5.1. *For any class $\mathcal{H}$ of ranged hash functions and any $n \leq \min(M, N)$ and $m \leq n$,*

1. *if $\mathcal{H}$ is $\mathcal{I}$-hereditary, then $\ell_{\mathcal{H}}(N, M, n, m) \geq \ell_{\mathcal{H}}(n, M, n, m)$;*
2. *if $\mathcal{H}$ is probabilistically $\mathcal{U}$-hereditary, then $\ell_{\mathcal{H}}(N, M, n, m) \geq \ell_{\mathcal{H}}(N, x, n, m)$ for any $m \leq x \leq M$.*

Proof. 1. Observe that for any randomized ranged hash function h with $\mathcal{I}$-hereditary property $\mathcal{H}$, if for some data set $D \in \binom{\mathcal{I}}{n}$ that $\Pr[\ell_S^D < L] = \Omega(1)$ for all $S \in \binom{\mathcal{U}}{m}$, the restriction of h on the new item domain $\mathcal{I}' = D$ is still a randomized ranged hash functions with property $\mathcal{H}$ (since $\mathcal{H}$ is $\mathcal{I}$-hereditary), and it still holds for the new ranged hash function that $\Pr[\ell_S < L] = \Omega(1)$ for all $S \in \binom{\mathcal{U}}{m}$. Therefore, assuming that $\ell_{\mathcal{H}}(n, M, n, m) = L$, i.e. for any randomized ranged hash functions defined on $\mathcal{I}' = [n]$ and $\mathcal{U}' = [M]$ with property $\mathcal{H}$, it holds that $\Pr[\ell_S \geq L] = 1 - o(1)$ for some state $S \in \binom{\mathcal{U}'}{m}$, according to the converse-negative proposition to the above observation, for all randomized ranged hash functions defined on $\mathcal{I} = [N]$ and $\mathcal{U} = [M]$ with property $\mathcal{H}$, for all data sets $D \in \binom{\mathcal{I}}{n}$, there must exist some state $S \in \binom{\mathcal{U}}{m}$ such that $\Pr[\ell_S^D \geq L] = 1 - o(1)$, i.e. $\ell_{\mathcal{H}}(N, M, n, m) \geq L$, which implies that $\ell_{\mathcal{H}}(N, M, n, m) \geq \ell_{\mathcal{H}}(n, M, n, m)$.

2. We assume that $\ell_{\mathcal{H}}(N, x, n, m) = L$, i.e. for any randomized ranged hash functions defined on $\mathcal{I}' = [N]$ and $\mathcal{U}' = [x]$ with property $\mathcal{H}$, for all data set $D \in \binom{\mathcal{I}'}{n}$, it holds that $\Pr[\ell_S^D \geq L] = 1 - o(1)$ for some state $S \in \binom{\mathcal{U}'}{m}$. For any randomized ranged hash functions h defined on $\mathcal{I} = [N]$ and $\mathcal{U} = [M]$ with property $\mathcal{H}$, if $\mathcal{H}$ is probabilistically $\mathcal{U}$-hereditary, we can randomly pick x buckets, say $[x]$, such that the restriction of h on the new bucket domain $\mathcal{U}' = [x]$ has property $\mathcal{H}$ with high probability, therefore according to the previous assumption, for the new ranged hash function, with high probability, for all data set $D \in \binom{\mathcal{I}}{n}$, it holds that $\Pr[\ell_S^D \geq L] = 1 - o(1)$ for some state $S \in \binom{\mathcal{U}'}{m} \subseteq \binom{\mathcal{U}}{m}$, i.e. for h, for all data set $D \in \binom{\mathcal{I}}{n}$, it holds that $\Pr[\ell_S^D \geq L] = 1 - o(1)$ for some state $S \in \binom{\mathcal{U}}{m}$, which means that $\ell_{\mathcal{H}}(N, M, n, m) \geq L$, so we prove that $\ell_{\mathcal{H}}(N, M, n, m) \geq \ell_{\mathcal{H}}(N, x, n, m)$.

6 Load balance vs. monotonicity

In this section, we prove the following theorem.

Theorem 6.1. *For any randomized monotone ranged hash function with domain $\mathcal{I} = [N]$ and $\mathcal{U} = [M]$, for* *any sufficiently large n and m that $N > 2n$, $M > 2m$, and $n \geq m$, for any data set $D \in \binom{\mathcal{I}}{n}$, there exists a state $S \in \binom{\mathcal{U}}{m}$, such that the maximum load $\ell_S^D = \Omega(L)$ with high probability, where*

$$L = \begin{cases} \sqrt{\frac{n}{m}\ln m} & m \leq n < o(m \log m) \\ \frac{n}{m} & n = \Omega(m \log m) \end{cases}.$$

An equivalent statement is that $\ell_{\texttt{monotone}}(N, M, n, m) = \Omega(L)$, where L is defined as above.

According to Lemma 5.1, it is sufficient to consider $\ell_{\texttt{monotone}}(n, m', n, m)$ for $m' = \max(n, 2m)$. For the rest of this section, we assume that $\mathcal{I} = [n]$ and $\mathcal{U} = [m']$, that is, we only consider the $n \times m'$ preference matrices. Since the $\Omega(\frac{n}{m})$ bound is trivial, we are only interested in the case where $m \leq n < o(m \log m)$.

Instead of directly proving the lower bound for randomized ranged hash functions, we look at an arbitrary deterministic π, and show a probabilistic bound on maximum load for a uniformly random state $S \in \binom{\mathcal{U}}{m}$. The same bound for randomized hash functions with worst-case S follows from Yao's min-max principle.

Given a monotone ranged hash function π, for any bucket $b \in \mathcal{U}$ and any set of items $A \subseteq \mathcal{I}$, we define the A-neighborhood $\Delta_A(b) = \{\pi_{ij} \mid i \in A \text{ and } j \leq \pi_i^{-1}(b)\}$, i.e. $\Delta_A(b)$ is the union of the set of buckets preferred by each item from A over b, including b. We say $|A|$ as the width of the neighborhood. It is easy to see that the concept of neighborhood characterizes the load of a bucket.

Lemma 6.1. *For any state $S \subseteq \mathcal{U}$, the load $\ell_S(b)$ of bucket b at state S is at least L, if and only if there is some $A \subseteq \mathcal{I}$ such that $|A| = L$ and $\Delta_A(b) \cap S = \{b\}$.*

It is thus sufficient to look at the collection of L-wide neighborhoods $\Delta_L = \{\Delta_A(b) \mid b \in \mathcal{U}, |A| = L\}$, which is a set system with ground set $\mathcal{U}$, and show that for the uniformly random $S \in \binom{\mathcal{U}}{m}$, with high probability there exists some $\Delta_A(b) \in \Delta_L$ such that $\Delta_A(b) \cap S = \{b\}$. However, the intersection between neighborhoods can be extremely complicated for arbitrary π, which makes the deviation hard to analyze. To overcome this, we extend the combinatorial **deletion method** introduced in [10,20], namely, we sequentially delete those buckets which cause the correlations, until the remaining family of neighborhoods has a sufficiently large disjoint subfamily. We do this in such a way that, conditioning on the previous deletions, the presence of each deleted bucket can only increase the maximum load.

We first investigate the case where the entries in each column of π do not include many duplicates. We show that in this case we can get a large disjoint subfamily of neighborhoods by applying the Hajnal-Szemeredi Theorem [7]. After that the standard analysis of deviation can be applied.

Let $\lambda_j(b)$ denote the number of b-entries in column j of π, i.e. $\lambda_j(b) = |\{i \mid \pi_{ij} = b\}|$.

Lemma 6.2. *Let $L = \sqrt{\frac{m'}{m}\ln m}$. For any constants $0 < \beta < \frac{1}{2}$ and $0 < \alpha < \sqrt{\frac{1}{2} - \beta}$, if for a $n \times m'$ preference matrix π, it holds that for every $b \in \mathcal{U}$ and every column $j \leq \alpha L$, $\lambda_j(b) = \tilde{O}(n^\beta)$, then*

$$\Pr_{S \in \binom{[m']}{m}}[\ell_S \geq \alpha L] = 1 - o(1).$$

Proof. We first construct a family of disjoint αL-wide neighborhood $\{\Delta_{A_b}(b)\}_{b \in U}$, where $|U| = \tilde{\Omega}(n^{1-2\beta})$, and for any $b \in U$, $|\Delta_{A_b}(b)| \leq \alpha^2 L^2$. Then we apply the second moment method to argue that with high probability, at least one of such neighborhoods has $\Delta_{A_b}(b) \cap S = \{b\}$.

We only consider the first αL columns of π. Since $\lambda_j(b) = \tilde{O}(n^\beta)$ for every b, there are at least $\tilde{\Omega}(n^{1-\beta})$ buckets b, each of which appears at least αL times in the first αL columns; it follows that there is set V of such bs that $|V| = \tilde{\Omega}(n^{1-\beta})$, and for every $b \in V$, there is a A_b that $|A_b| = \alpha L$ and the whole neighborhood $\Delta_{A_b}(b)$ is in the first αL columns. This guarantees that $|\Delta_{A_b}(b)| \leq \alpha L |A_b| = \alpha^2 L^2$.

We define a graph $G(V, E)$ with the above V as the vertex set, and $(a, b) \in E$ if $\Delta_{A_a}(a)$ intersects with $\Delta_{A_b}(b)$. Since $\lambda_j(b) = \tilde{O}(n^\beta)$ and the size of each $\Delta_{A_b}(b)$ is bounded by $\alpha^2 L^2 = O(\ln n)$, the degree of $G(V, E)$ is at most $\tilde{O}(n^\beta)$. According to the Hajnal-Szemeredi Theorem [7], there is an independent set U of size $\tilde{\Omega}(n^{1-2\beta})$. Translating back from the graph to the neighborhood structure, this says that there is a family of disjoint neighborhoods $\{\Delta_{A_b}(b)\}_{b \in U}$ of cardinality $\tilde{\Omega}(n^{1-2\beta})$, such that $|\Delta_{A_b}(b)| \leq \alpha^2 L^2$ for every $b \in U$.

For any bucket $b \in U$, let X_b be the indicator that $X_b = 1$ if $\Delta_{A_b}(b) \cap S = \{b\}$, and $X_b = 0$ if otherwise. Let $X = \sum_{b \in U} X_b$. Due to Lemma 6.1, $\Pr[\ell_S \geq \alpha L] \geq \Pr[\exists b \in U, \Delta_{A_b}(b) \cap S = \{b\}] = 1 - \Pr[X = 0]$.

For any constant $0 < \alpha < \sqrt{\frac{1}{2} - \beta}$,

$$\begin{aligned} E(X) &= |U| \cdot \Pr[\Delta_{A_b}(b) \cap S = \{b\}] \\ &\geq \tilde{\Omega}(n^{1-2\beta}) \binom{m - \alpha^2 L^2}{m-1} \Big/ \binom{m'}{m} \\ &\geq \tilde{\Omega}(n^{1-2\beta-2\alpha^2}) \\ &= \omega(1). \end{aligned}$$

For any $a, b \in U$, $\Delta_{A_a}(a)$ and $\Delta_{A_b}(b)$ are disjoint, thus $cov(X_a, X_b) \leq 0$, therefore $var(X) = \sum_{b \in U} var(X_b) +$

$\sum_{a\neq b} cov(X_a, X_b) \leq \sum_{b\in U} var(X_b) \leq E(X)$. Due to Chebyshev's inequality,

$$\begin{aligned}
\Pr[X=0] &\leq \Pr[|X-E(X)| \geq E(X)] \\
&\leq \frac{var(X)}{E^2(X)} \\
&\leq \frac{1}{E(X)} \\
&= o(1).
\end{aligned}$$

Therefore $\Pr[\ell_S \geq \alpha L] \geq 1 - \Pr[X=0] \geq 1 - o(1)$.

Next we deal with the case where some bucket appears many times in some column, but in all the preceding columns, the number of appearances of any single bucket is small.

Lemma 6.3. *Let $L = \sqrt{\frac{m'}{m}\ln m}$. For any constants $\beta > 0$ and $0 < \alpha < \frac{\sqrt{\beta}}{2}$, if for a $n \times m'$ preference matrix π, it holds that $\lambda_k(b) \geq \alpha L n^{\frac{\beta k}{\alpha L}}$ for some $k \leq \alpha L$ and some $b \in \mathcal{U}$, and $\lambda_j(a) < \alpha L n^{\frac{\beta(k-1)}{\alpha L}}$ for all $j < k$ and all $a \in \mathcal{U}$, then*

$$\Pr_{S\in\binom{[m']}{m}}[\ell_S \geq \alpha L \mid b \in S] = 1 - o(1).$$

Proof. The case for $k=1$ is trivial, so we assume that $k > 1$.

Let $t = \lambda_k(b)$. Without loss of generality, we assume that $\pi_{ik} = b$ for $i = 1, 2, \ldots, t$. Let X_i be the indicator that $X_i = 1$ if $\pi_{ij} \notin S$ for all $j < k$ and $X_i = 0$ if otherwise, and let $X = \sum_{i=1}^{t} X_i$. Note that $\Pr[\ell_S \geq \alpha L \mid b \in S] \geq \Pr[X \geq \alpha L]$.

$$\begin{aligned}
E(X) &= t \cdot E(X_i) \\
&= t \cdot \Pr[\forall j < k, \pi_{ij} \notin S] \\
&= t\binom{m'-k}{m-1}\Big/\binom{m'-1}{m-1} \\
&\geq t\left(1 - \frac{m-1}{m'-k+1}\right)^{(k-1)}.
\end{aligned}$$

Let $G([t], E)$ be the dependency digraph that $(i_1, i_2) \in E$ if $cov(X_{i_1}, X_{i_2}) > 0$. It is easy to see that $(i_1, i_2) \in E$ only if the row i_1 and i_2 of π share some entries in the first $k-1$ columns. Since $\lambda_j(a) < \alpha L n^{\frac{\beta(k-1)}{\alpha L}}$ for all $j < k$ and $a \in \mathcal{U}$, the degree of the dependency graph is at most $k \cdot \alpha L n^{\frac{\beta(k-1)}{\alpha L}}$, thus the number of edges $|E| \leq \frac{t}{2}\alpha k L n^{\frac{\beta(k-1)}{\alpha L}}$, therefore $\sum_{i_1\neq i_2} cov(X_{i_1}, X_{i_2}) \leq 2|E| \leq \alpha t k L n^{\frac{\beta(k-1)}{\alpha L}} = o(E^2(X))$ for any constants $\beta > 0$ and $0 < \alpha < \frac{\sqrt{\beta}}{2}$, hence $var(X) = o(E^2(X))$.

From Chebyshev's inequality, $\Pr[X < \alpha L] \leq \frac{var(X)}{(E(X)-\alpha L)^2} = o(1)$, therefore $\Pr[\ell_S \geq \alpha L \mid b \in S] \geq \Pr[X \geq \alpha L] = 1 - o(1)$.

Combining the two lemmas, we have the following theorem.

Theorem 6.2. *For any monotone ranged hash function π with $\mathcal{I} = [n]$ and $\mathcal{U} = [m']$, where $m' = \max(n, 2m)$, and $n = o(m \log m)$ it holds that,*

$$\Pr_{S\in\binom{[m']}{m}}\left[\ell_S \geq \Omega\left(\sqrt{\frac{n}{m}\ln m}\right)\right] = 1 - o(1).$$

Proof. We denote that $L(x,y) = \sqrt{\frac{x}{y}\ln y}$. Note that $L(m', m) \geq \sqrt{\frac{n}{m}\ln m}$.

For an arbitrary π, we can construct a sequence of buckets $b_1, b_2, \ldots, b_t$ by the following procedure. Initially $t = 1$, the constants are set as $\beta = \frac{1}{3}$ and $\alpha = \frac{1}{4}$.

1. If in the current π, for every $1 \leq j \leq \alpha L(m', m)$ and for any b, $\lambda_j(b) < \alpha L(m', m) n^{\frac{j\beta}{\alpha L(m',m)}}$, then terminate.

2. Pick one b with minimum j such that $\lambda_j(b) \geq \alpha L(m', m) n^{\frac{j\beta}{\alpha L(m',m)}}$. Let $b_t \leftarrow b$ and $t \leftarrow t+1$. Delete b from π, that is, let π become the restriction of π on $\mathcal{U} \setminus \{b\}$.

3. If $t < \log^2 n$ go to Step 1, if otherwise terminate.

Summing the total probability, we have

$$\begin{aligned}
&\Pr_{S\in\binom{[m']}{m}}\left[\ell_S \geq \frac{1}{2}\alpha L(m', m)\right] \\
= &\sum_{k=1}^{t}\Bigg(\Pr_{S\in\binom{[m']}{m}}\left[\ell_S \geq \frac{1}{2}\alpha L(m', m) \mid \{b_i\}_{i<k} \subseteq \overline{S}, b_k \in S\right] \\
&\quad \cdot \Pr_{S\in\binom{[m']}{m}}\left[\{b_i\}_{i<k} \subseteq \overline{S}, b_k \in S\right]\Bigg) \\
&+\Bigg(\Pr_{S\in\binom{[m']}{m}}\left[\ell_S \geq \frac{1}{2}\alpha L(m', m) \mid \{b_i\}_{i\leq t} \subseteq \overline{S}\right] \\
&\quad \cdot \Pr_{S\in\binom{[m']}{m}}\left[\{b_i\}_{i\leq t} \subseteq \overline{S}\right]\Bigg) \\
\geq &\sum_{k=1}^{t}\Bigg(\Pr_{S\in\binom{[m'-k+1]}{m}}\left[\ell_S \geq \alpha L(m'-k+1, m) \mid b_k \in S\right] \\
&\quad \cdot \Pr_{S\in\binom{[m'-t]}{m}}\left[\{b_i\}_{i<k} \subseteq \overline{S}, b_k \in S\right]\Bigg)
\end{aligned}$$

$$+\left(\Pr_{S\in\binom{[m'-t]}{m}} \left[\ell_S \geq \alpha L(m'-t,m)\right] \cdot \Pr_{S\in\binom{[m'-t]}{m}} \left[\{b_i\}_{i\leq t} \subseteq \overline{S}\right]\right)$$

The second inequality is due to fact that $L(m' - O(\log^2 n), m) \geq \frac{1}{2}L(m', m)$ and the observation that conditioning on $T \subseteq \overline{S}$ is equivalent to that π becomes the restriction of π on $\mathcal{U}\setminus T$ and the probability is taken over uniformly random S from $\binom{[m']\setminus T}{m}$.

Due to Lemma 6.3, for the first t terms $\Pr[\ell_S \geq \alpha L(m' - k + 1, m) \mid b_k \in S] = 1 - o(1)$ for every k. If $t < \log^2 n$, all buckets with large number of replications are deleted, thus for the last term, π is as required by Lemma 6.2, hence $\Pr[\ell_S \geq \alpha L(m' - t, m)] = 1 - o(1)$, therefore the total is 1-o(1). Alternatively, if $t = \log^2 n$, then $\Pr[\{b_i\}_{i\leq t} \subseteq \overline{S}] = o(1)$, which means that the total is at least $(1 - o(1) - o(1))$, which is also $(1 - o(1))$.

A direct corollary to Theorem 6.2 is that for any distribution of monotone ranged hash functions that $\mathcal{I} = [n]$ and $\mathcal{U} = [m']$, the same statement still holds for the uniformly random S with size m, thus holds for the the worst-case S with size m, i.e. $\ell_{\texttt{monotone}}(n, m', n, m) = \Omega(L)$ where L is as defined in Theorem 6.1. By Lemma 5.1, we have that $\ell_{\texttt{monotone}}(N, M, n, m) = \Omega(L)$, thus Theorem 6.1 is proved.

7 Tightness of the lower bound

In this section we address the tightness of the lower bound of $\ell_{\texttt{monotone}}(M, N, n, m)$, and show evidence that the lower bound is tight in almost all settings.

We first consider a simple construction of (randomized) monotone ranged hash function. For $\mathcal{I} = [N]$ and $\mathcal{U} = [M]$, define a randomized $N \times M$ preference matrix Π as follow: for each $i \in \mathcal{I}$, row Π_i is a uniformly and independently random permutation of $\mathcal{U}$. It is easy to see that for any data set $D \in \binom{\mathcal{I}}{n}$ and any state $S \in \binom{\mathcal{U}}{m}$, the ranged hash function Π assigns each of the n items independently to a uniformly random bucket in S. According to the well-known balls-into-bins result [18], when $n = \Omega(m \log m)$, the maximum load is with high probability $O(\frac{n}{m})$, i.e. the lower bound is tight for the case that $n = \Omega(m \log m)$. However, for the case that n is close to m, there is a gap between the maximum load of Π and the lower bound. The gap is maximized when $n = \Theta(m)$, where the maximum load of Π is $\Theta(\ln n/ \ln \ln n)$ but the lower bound is $\Omega(\sqrt{\ln n})$.

We then introduce a construction that approaches the lower bound when n is close to m. We first deal with the base case where $\mathcal{I} = \mathcal{U} = [n]$. For convenience, we assume that n is power of 2 and $n \geq cm$ for some constant $c > 1$.

The Perturbed Cube. Intuitively, this construction is a Hamming cube with perturbations. Each bucket is mapped to a vertex of the cube; each item is mapped to a vertex of the cube as well. The preference list is determined by Hamming distance plus some perturbation to break ties.[1]

More formally, let $\phi : [n] \to \{0, 1\}^{\log n} \times [0, 1)$ be a mapping from $[n]$ to a $(1 + \log n)$-dimensional vector space X as follow: $\phi(k) = x$ where $x_i = \lfloor k/2^{i-1} \rfloor \bmod 2$ for $i = 1, 2, \ldots, \log n$ and $x_{1+\log n} = k/n$, i.e. the first $\log n$ entries of $\phi(k)$ comprise the binary representation of k, and the last entry is k/n.

For $\mathcal{I} = \mathcal{U} = [n]$, let h be a monotone ranged hash function defined as follows. First, embed $\mathcal{I}$ and $\mathcal{U}$ in the same metric space (X, d) by the same mapping ϕ as defined above, where d is the ℓ_1 distance. Next, for any item $i \in \mathcal{I}$ and any state $S \subseteq \mathcal{U}$, let $h_S(i)$ be some b with the property that $\forall a \in S, d(\phi(i), \phi(b)) \leq d(\phi(i), \phi(a))$. Note that there are at most two such points b for each i; in case of a tie, we pick an arbitrary one. We denote by π the resulting preference matrix.

We now show that the maximum load for the perturbed cube is, with high probability, $O\left(\sqrt{\frac{n}{m}} \ln m\right)$, for the case that n is close to m:

THEOREM 7.1. *Let π be as constructed above. If $n = o\left(\frac{m \ln m}{(\ln \ln m)^2}\right)$ and $n \geq cm$ for some constant $c > 1$, then there is a constant $\alpha > 0$ such that*

$$\Pr_{S\in\binom{\mathcal{U}}{m}} \left[\ell_S \geq \alpha \cdot \sqrt{\frac{n}{m}} \ln m\right] = o(1).$$

Proof. Let $L = \alpha \cdot \sqrt{\frac{n}{m}} \ln m$. Considering the collection of all L-wide neighborhoods $\Delta_L = \{\Delta_A(b) \mid b \in \mathcal{U} \text{ and } A \in \binom{\mathcal{I}}{L}\}$ defined on π, it is sufficient to show that with high probability, none of the neighborhoods $\Delta_A(b) \in \Delta_L$ has $\Delta_A(b) \cap S = \{b\}$. Two key observations are: (1) the chance that any entry after the first L^2 column is ever reached is negligible, thus we only need to consider the neighborhoods contained in the first L^2 columns of π; and (2) for every L-wide neighborhood, $|\Delta_A(b)| = \Omega(L^2)$. Applying these two facts, the theorem is proved. The detailed proof will be given in the full version of the paper.

[1]This perturbed cube construction of ranged hash function does not imply a practical distributed hash table as consistent hashing does. This is because the nearest neighbor search in high dimensional Hamming space is believed to be hard. We propose this construction only to show the tightness of the bound of the price of churn with monotone ranged hash functions. However, it might turn out be useful for more centralized applications of ranged hash functions, as in Internet routers [3], where nearest-neighbor search is unnecessary and it may be feasible to store the whole preference matrix in the routing table.

To construct a randomized ranged hash function against a worst-case S, we randomly rename the buckets by applying a uniformly random permutation to $\mathcal{U}$. With $\mathcal{I} = [n]$ and $\mathcal{U} = [n]$, for any $S \in \binom{\mathcal{U}}{m}$ that $n = o\left(\frac{m \ln m}{(\ln \ln m)^2}\right)$, the maximum load ℓ_S is with high probability $O\left(\sqrt{\frac{n}{m}} \ln m\right)$. We show that the lower bound of $\ell_{\texttt{monotone}}(n, n, n, m)$ is tight when $n = o\left(\frac{m \ln m}{(\ln \ln m)^2}\right)$. This statement can also be easily extended to $\ell_{\texttt{monotone}}(M, N, n, m)$ for $N = O(n)$ and $M = O(n)$.

For general domain sizes N and M, we have a non-uniform construction through reduction to the base case $N = M = [n]$. Given $\mathcal{I} = [N]$ and $\mathcal{U} = [M]$, let $r : [M] \to [n]$ be a uniformly random projection from $[M]$ to $[n]$. We construct a monotone ranged hash function h' by the rule that for each $i \in [n] \subset \mathcal{I}$, and any $S \subseteq \mathcal{U}$, $h_S(i)$ is chosen to be some $b \in r^{-1}(h_{r(S)}(i))$, with arbitrary tie breaking. Intuitively, the first n items in $\mathcal{I}$ are assigned to the buckets in state S according to the perturbed cube h at the state $r(S)$.

For any $S \in \binom{\mathcal{U}}{m}$ with $n = o\left(\frac{m \ln m}{(\ln \ln m)^2}\right)$, we have that with high probability $|r(S)| = \Theta(m)$. Thus the maximum load contributed by these n items is asymptotically the same as the maximum load in the n-vertex perturbed cube, which is $O\left(\sqrt{\frac{n}{m}} \ln m\right)$. (Note that the concentration of r may only decrease the maximum load.) For h', there exists a data set $D \in \binom{\mathcal{U}}{n}$ such that for any state $S \in \binom{\mathcal{U}}{m}$ where $n = o\left(\frac{m \ln m}{(\ln \ln m)^2}\right)$, the maximum load ℓ_S^D is $O\left(\sqrt{\frac{n}{m}} \ln m\right)$ with high probability.

It follows that the lower bound of $\ell_{\texttt{monotone}}(M, N, n, m)$ is tight whenever $n = o\left(\frac{m \ln m}{(\ln \ln m)^2}\right)$ or $n = \Omega(m \log m)$.

8 Load balance vs. expansion rate

General preference matrices create scalability problems, since they require storing $\Omega(M \log M)$ bits of information for each item. Instead, we would prefer a more compact representation that allows each node (bucket) in the system to store at most $m^{o(1)}$ bits of information. A general way to do so is embedding items and buckets into a metric space, and then assign each item to the closest bucket in the current state. Because of this scalability constraint, we are restricted to the cases that $n = m^{1+o(1)}$—if a node cannot handle more than $m^{o(1)}$ bits, we surely do not expect it to store more than that many data items.

In this section, we consider monotone ranged hash functions based on this approach, and show a trade-off between load balance and the expansion rate of the underlying metric. Our results demonstrate that consistent hashing on a one-dimensional ring gives optimal load balance among all growth-restricted metrics.

Formally, we start with a metric space (X, d), and assume $\mathcal{I} \subseteq X$ and $\mathcal{U} \subseteq X$. A ranged hash function h can then be defined as follows. For every $S \subseteq \mathcal{U}$ and $i \in \mathcal{I}$, $h_S(i)$ is the nearest neighbor of i in S; specifically, it satisfies the constraint that for any $b \in S$, $d(i, h_S(i)) \le d(i, b)$. In order to make this definition precise, we further require that the embedding is "sparse", that is, the chance that there is more than one nearest neighbor for any given point is negligible. It is obvious that such a ranged hash function is monotone.

The expansion rate of the underlying metric is important because it affects the hardness of finding nearest neighbors. Given any $x \in X$, $Y \subseteq X$, and $r > 0$, we let $B_Y(x, r) = \{b \in Y \mid d(x, b) \le r\}$ be the ball of radius r around x in Y. The KR-dimension [12] of Y, denoted as $\dim_{\mathrm{KR}}(Y)$, is the smallest K such that $|B_Y(x, 2r)| \le 2^K |B_Y(x, r)|$ for all $x \in X$, $r \ge 0$. We say Y is growth-restricted if it has constant KR-dimension, i.e. $\dim_{\mathrm{KR}}(Y) = O(1)$.[2]

For each $x \in X$, let $r_x(t)$ be the smallest value that $|B_Y(x, r_x(t))| = t$. The following lemma captures the connection between the expansion rate of the metric and the density of overlapped balls, which is central to proving the trade-off between expansion rate and load balancing.

LEMMA 8.1. *Let $Y \subseteq X$ with $\dim_{\mathrm{KR}}(Y) = K$. For any $A \subseteq X$, if $\bigcap_{x \in A} B_Y(x, r_x(t)) \neq \emptyset$, then $|\bigcup_{x \in A} B_Y(x, r_x(t))| \le t 2^{2K}$.*

Proof. Consider any $x, y \in X$, and suppose that $r_x(t) \ge r_y(t)$. If there exists some $b \in B(x, r_x(t)) \cap B(x, r_x(t))$, then $B(y, r_y(t)) \subseteq B(x, 3r_x(t))$, since for any $a \in B(y, r_y(t))$, $d(a, x) \le d(a, y) + d(y, b) + d(b, x) \le 3r_x(t)$. For the same reason, if $\bigcap_{x \in A} B(x, r_x(t)) \neq \emptyset$, we can take the largest $r_x(t)$ and so $\bigcup_{x \in A} B(x, r_x(t)) \subseteq B(x, 3r_x(t))$. Because $\dim_{\mathrm{KR}}(Y) = K$, $|B(x, 3r_x(t))| \le 2^{2K} |B(x, r_x(k))| = t 2^{2K}$.

For any monotone ranged hash function π arising from a metric space (X, d), the preference list π_i for each item $i \in \mathcal{I}$ is a list of buckets in $\mathcal{U}$ sorted by increasing distance from i. It is not hard to see that $\{\pi_{i1}, \pi_{i2}, \ldots, \pi_{it}\} = B_{\mathcal{U}}(i, r_i(t))$ for any t, i.e., for each row of π, its first t entries is a t-point ball around i in

[2]In [12], the definition also includes a minimum threshold of the size of balls. For simplicity, we drop this assumption. We can do so because for any size state S, the threshold has to be sufficiently small to enforce the nearest neighbor search, which means that the threshold is independent of $|\mathcal{U}|$.

$\mathcal{U}$. Combining this fact with Lemma 8.1, we have the following proposition.

PROPOSITION 8.1. *If $\dim_{\mathrm{KR}}(\mathcal{U}) = K$, for any $b \in \mathcal{U}$ and any $A \subset \mathcal{I}$, $|\Delta_A(b)| \leq t2^{2K}$ where $t = \max_{i \in A} \pi_i^{-1}(b)$.*

This implies that if a neighborhood $\Delta_A(b)$ is contained in the first t columns of π, the size of the neighborhood is at most $t2^{2K}$ no matter how large A is.

All the above facts intuitively suggest that for ranged hash functions arising from a metric, the KR-dimension of $\mathcal{U}$ strongly affects the structure of the preference matrices, and thus controls load balancing. This intuition is justified by the following lemma.

LEMMA 8.2. *Let π be a monotone ranged hash function with $|\mathcal{I}| = n$, $|\mathcal{U}| = m'$, and $\dim_{\mathrm{KR}}(\mathcal{U}) = K$, where $m' = \max(n, 2m)$, $n = m^{1+o(1)}$, and $K \leq \frac{1}{4}\log_2(\frac{n}{m}\ln m)$. Then for any constant $0 < \alpha < \frac{1}{2}$,*

$$\Pr_{S \in \binom{\mathcal{U}}{m}}\left[\ell_S \geq \alpha 2^{-2K} \cdot \frac{n}{m}\ln m\right] = 1 - o(1).$$

Proof. Let $L = \alpha 2^{-2K} \cdot \frac{n}{m}\ln m$. With the given constraints on the parameters, $L = \Omega(\sqrt{\ln m}) \cap m^{o(1)}$. As in the proof of Theorem 6.1, we consider two cases depending on the number of duplicate entries in each column of π. Recall that for each column j and bucket b, $\lambda_j(b)$ represents the number of copies of b in j.

Case 1: If for each $j \leq L$ and $b \in \mathcal{U}$, $\lambda_j(b) < L^4/\alpha$, applying the same argument as in Lemma 6.2 gives that there exists a family of disjoint L-wide neighborhoods $\{\Delta_{A_b}(b)\}_{b \in U}$ such that (a) $|U| = \tilde{\Omega}(n)$ and (b) for each $b \in U$, the whole neighborhood $\Delta_{A_b}(b)$ is contained in the first L columns of π.

From Proposition 8.1, $|\Delta_{A_b}(b)| \leq L \cdot 2^{2K}$. Again repeating the analysis in Lemma 6.2, we have that $\Pr[\ell_S < L] = \tilde{O}(n^{-1+2\alpha}) = o(1)$ with $\alpha < \frac{1}{2}$.

Case 2: There is some $j \leq L$ and some b, such that $\lambda_j(b) \geq L^4/\alpha$. Without loss of generality, we assume that $\pi_{ij} = b$ for $i = 1, 2, \ldots, t$, where $t = L^4/\alpha$. We only consider the first L^2 columns in these t rows. Because they share the same entry b, according to Proposition 8.1, $|\{\pi_{ij} \mid 1 \leq i \leq t, 1 \leq j \leq L^2\}| = L^2 2^{2K}$, i.e., there are $L^2 2^{2K}$ distinct entries in the first L^2 columns of π in these t rows. Effectively, π assigns $t = L^4/\alpha$ items to only $L^2 2^{2K}$ buckets. The maximum load in these buckets is thus at least $\frac{1}{\alpha}L^2 2^{-2K} \geq L$, as long as none of these rows are totally absent from S. The probability that there is at least one such bad row (containing no element of S in its first L^2 entries) is at most $t \cdot \Pr[[L^2] \subseteq \overline{S}] \leq \frac{1}{\alpha}L^4(1 - \frac{m}{m'-L^2})^{L^2} = o(1)$. Therefore with high probability, the maximum load $\ell_S \geq L$.

Naturally, the same bound holds for randomized ranged hash functions against a worst-case S. Specifically, for any family of ranged hash functions with $\dim_{\mathrm{KR}}(\mathcal{U}) = K$, $\ell_{K\text{-}\dim_{\mathrm{KR}}}(n, m', n, m) = \Omega(2^{-2K} \cdot \frac{n}{m}\ln m)$, where $K \leq \frac{1}{4}\log_2(\frac{n}{m}\ln m)$.

In [12], Karger and Ruhl show that a uniformly selected subset of $\mathcal{U}$ has essentially the same KR-dimension with high probability,[3] i.e. K-$\dim_{\mathrm{KR}}$ is probabilistically $\mathcal{U}$-hereditary. It is easy to see that K-$\dim_{\mathrm{KR}}$ is also $\mathcal{I}$-hereditary, since the KR-dimension of $\mathcal{U}$ is irrelevant of $\mathcal{I}$. By Lemma 5.1, for all ranged hash functions with the same KR-dimension of $\mathcal{U}$ it holds that $\ell_{K\text{-}\dim_{\mathrm{KR}}}(N, M, n, m) \geq \ell_{K\text{-}\dim_{\mathrm{KR}}}(n, m', n, m)$. This gives us the following theorem.

THEOREM 8.1. *For any distribution of ranged hash functions with $|\mathcal{I}| = N$, $|\mathcal{U}| = M$, and $\dim_{\mathrm{KR}}(\mathcal{U}) = K$, for any sufficiently large n and m with $N > 2n$, $M > 2m$, $n \geq m$, $n = m^{1+o(1)}$, and $\frac{1}{4}\log_2(\frac{n}{m}\ln m) \geq K$, it holds that for any data set $D \in \binom{\mathcal{I}}{n}$, there exists a state $S \in \binom{\mathcal{U}}{m}$, such that $\ell_S^D = \Omega(2^{-2K} \cdot \frac{n}{m}\ln m)$ with high probability. Equivalently, under these conditions, $\ell_{K\text{-}\dim_{\mathrm{KR}}}(N, M, n, m) = \Omega(2^{-2K} \cdot \frac{n}{m}\ln m)$.*

This justifies our previous observation that dimensionality helps load balance even though it hurts searching.

In particular, if $\mathcal{U}$ is growth-restricted, $\ell_{O(1)\text{-}\dim_{\mathrm{KR}}}(N, M, n, m) = \Omega(\frac{n}{m}\ln m)$ for $N > 2n$, $M > 2m$, $n \geq m$, and $n = m^{1+o(1)}$. This bound is tight because it is achieved by standard consistent hashing, which (as discussed in Section 4) can be modeled using a discrete growth-restricted metric in place of the usual continuous $[0, 1)$ metric. This lower bound explains where the $\Theta(\frac{n}{m}\ln m)$ bound for consistent hashing comes from: it is the best we can get from any growth-restricted metric. It also explains why no more sophisticated growth-restricted metric has arisen to displace consistent hashing.

Unlike the case for general monotone ranged hash functions, in a growth restricted metric, the price of churn dominates the balls-into-bins skew, and unlike the balls-into-bins bound, the price of churn does not approach $O(\frac{n}{m})$ as n grows to $\Omega(m \log m)$. The price of churn in this setting is also robust for all values of n and m satisfying the scalability condition $n = m^{1+o(1)}$.

[3]In fact, they show that a set S of uniformly random m points from $\mathcal{U}$ with $\dim_{\mathrm{KR}}(\mathcal{U}) = K$ has $\dim_{\mathrm{KR}}(S) \leq K + 1$ with probability $1 - \exp(-\Omega(t))$ if we only care about balls larger than a threshold t. Because all balls in our proofs have size $L = \Omega(\sqrt{\ln m})$, we can easily make the probability $(1 - o(1))$ without affecting our argument. We also ignore the $+1$ on the KR-dimension, because it can only affect the maximum load by a constant factor.

These facts suggest that in systems that yield growth-restricted metric, which is the typical case for Internet applications, the unreliability of nodes is more critical than the uncertainty of data.

References

[1] M. Adler, E. Halperin, R. M. Karp, and V. V. Vazirani. A stochastic process on the hypercube with applications to peer-to-peer networks. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 575–584, 2003.

[2] J. Aspnes and G. Shah. Skip graphs. In *Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, pages 384–393, Jan. 2003.

[3] J. Aspnes, Y. R. Yang, and Y. Yin. Path-independent load balancing with unreliable machines. In *Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 814–823, Jan. 2007.

[4] G. Giakkoupis and V. Hadzilacos. A scheme for load balancing in heterogenous distributed hash tables. In *Proceedings of the 24th Annual ACM Symposium on Principles of Distributed Computing (PODC 2005)*, pages 302–311, 2005.

[5] P. Godfrey, S. Shenker, and I. Stoica. Minimizing churn in distributed systems. *Proceedings of the ACM SIGCOMM 2006*, pages 147–158, 2006.

[6] A. Gupta, R. Krauthgamer, and J. Lee. Bounded geometries, fractals, and low-distortion embeddings. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pages 534–543, 2003.

[7] A. Hajnal and E. Szemeredi. Proof of a conjecture of Erdos. *Combinatorial Theory and its Applications*, 2:601–623, 1970.

[8] K. Hildrum, J. Kubiatowicz, S. Rao, and B. Y. Zhao. Distributed object location in a dynamic network. In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2002)*, pages 41–52, 2002.

[9] P. Indyk, J. Goodman, and J. O'Rourke. Nearest neighbors in high-dimensional spaces. *Handbook of Discrete and Computational Geometry, chapter 39*, 2004.

[10] S. Janson and A. Rucinski. The infamous upper tail. *Random Structures and Algorithms*, 20(3):317–342, 2002.

[11] M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *The 2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003)*, pages 98–107, 2003.

[12] D. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. *Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 741–750, 2002.

[13] D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing (STOC 1997)*, pages 654–663, 1997.

[14] D. R. Karger and M. Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proceedings of the 16th Annual ACM Symposium on Parallel Algorithms (SPAA 2004)*, pages 36–43, 2004.

[15] K. Kenthapadi and G. S. Manku. Decentralized algorithms using both local and random probes for p2p load balancing. In *Proceedings of the 17th Annual ACM Symposium on Parallel Algorithms (SPAA 2005)*, pages 135–144, 2005.

[16] R. Krauthgamer and J. Lee. Navigating nets: simple algorithms for proximity search. *Proceedings of the fifteenth annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004)*, pages 798–807, 2004.

[17] G. S. Manku. Balanced binary trees for id management and load balance in distributed hash tables. In *Proceedings of the 23rd Annual ACM Symposium on Principles of Distributed Computing (PODC 2004)*, pages 197–205, 2004.

[18] R. Motwani and P. Raghavan. *Randomized algorithms.* Cambridge University Press, New York, NY, USA, 1995.

[19] M. Naor and U. Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *Proceedings of the 15th Annual ACM Symposium on Parallel Algorithms (SPAA 2003)*, pages 50–59, 2003.

[20] V. Rodl and A. Rucinski. Threshold Functions for Ramsey Properties. *Journal of the American Mathematical Society*, 8(4):917–942, 1995.

[21] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware 2001, IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, 2001.

[22] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM 2001*, pages 149–160, 2001.