# Random Projection Hashing for Scalable Big Data Clustering

A thesis submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

**Doctor of Philosophy**

in the School of Electric and Computing Systems
of the College of Engineering and Applied Sciences

April 06, 2013

by

**Lee A Carraher**
BSCE, University of Cincinnati, 2008 MSCS,University of Cincinnati, 2012

Dissertation Advisor and Committee Chair: Dr. Philip Wilsey

# 1 Project Summary




## Overview:

This proposal presents a distributed algorithm for secure clustering of high dimensional data. A novel algorithm, called Random Projection Hash or *RPHash*, utilizes aspects of locality sensitive hashing (LSH) and multi-probe random projection for computational scalability and linear achievable gains from parallel speed up. The two step approach is data agnostic, minimizes communication overhead, and has a priori predictable computational time. The system is deployable on commercially available cloud resources running the Hadoop (*MR*v2) implementation of MapReduce. The *RPHash* solution will have a wide applicability to a variety of standard clustering applications while this project will focus on a subset of clustering problems in the biological data analysis space. *RPHash* also combats de-anonymization attacks inherently resulting from its algorithmic requirements thus addressing requirements involving the handling and privacy protection of health care data [1] as well as the inherent privacy concerns of using cloud based services. Furthermore, *RPHash* will allow researchers to scale their clustering problems without the need for specialized equipment or computing resources. The proposed cloud processing solution will allow researchers to arbitrarily scale their processing needs using virtually limitless commercial processing resources.

## Intellectual Merit:

A principle driving force in computational progress results from material and architectural advances in microprocessor design. Many of these advances have been stagnated for linear processing however due to thermal dissipation and energy requirements, resulting in a shift toward parallel multi-processing. Though much has been done to adapt current algorithms for this parallel processing landscape, many solutions tend to be a' posteriori methods favoring empirical speedup over long term scalability. In this proposal, we develop a method for an algorithm central to data analysis, designed expressly for parallel multi-processing systems. In addition, our algorithm addresses often overlooked communication bottlenecks in parallel design, through use of side channel synchronization based on mathematically generative groups and probabilistic approximation. A favorable side effect of the probabilistic approximation results in a possible solution to de-anonymization attacks [2] [3] on user data.

## Broader Impacts:

Clustering has long been the standard method used for the analysis of labeled and unlabeled data. Clusterings' effects intrinsically identify the latent models underlying the distributions of objects in a dataset, often unattainable through standard statistical methods. Single pass, data intensive statistical methods are often the primary workhorses for parallel database processing of business logic and other domains, while clustering is often overlooked due to scalability ocncerns and confusion caused by the wide variety of available distributed algorithms [4]. A multitude of surveys [4] have been made available comparing different aspects of clustering algorithms, such as accuracy, complexity and application domain. Fields in health and biology are benefited by data clustering scalability. Bioinformatic problems such as Micro Array clustering, Protein-Protein interaction clustering, medical resource decision making, medical image processing, and clustering of epidemiological events all serve to benefit from larger dataset sizes. Furthermore, the proposed method provides data anonymization through destructive manipulation of the data preventing de-anonymization attacks beyond standard best practices database security methods.

# 1 Project Overview

Data clustering is an inherent problem in data analysis. It is the principle work horse of many types of pilot data analysis and various data retrieval methods. Due to its importance in machine learning, data clustering is studied extensively in computing and mathematics [4–6]. In addition to advancements in data clustering, Many challenges to algorithm design are presented by the changes in computing architectures, resulting from changing use cases, market trends, and underlying physical design limits. Multi-processor architectures and distributed computing are one such area of change, that is rapidly overtaking sequential computing prompting a fundamental restructuring in algorithmic design [7]. Furthermore, optimal coordination of many independent processors is an open problem with fundamental communication scalability issues [8]. Random Projection Hash(*RPHash*) was expressly created for minimizing parallel communication overhead while providing algorithmic scalability. The shortcomings of various other parallel methods will be discussed in this proposal with suggestions on how *RPHash* can overcome them. Many parallel clustering algorithms have been converted to function efficiently on parallel and distributed systems, however they often have potential issues regarding their asymptotic scalability [9], dimensionality in which they are effective [10], and robustness to noise. *RPHASH* is a method proposed for large scale distributed data clustering combining locality sensitive hash functions and multi-probe random projection for distributed approximate data clustering. Clustering is an inherent problem in data analysis in which a set of vectors follow a suspected latent model distribution [11–14]. Although many algorithms have been proposed both parallel friendly and sequential, many issues still are present when applied to very large, high dimensional, distributed datasets [9, 10, 15–17]. The system designed herein will focus on data security, algorithmic and communication scalability as well as the potential gains from parallel speedup. As the system being designed is proposed as a real implementation, beyond a theoretical algorithmic design, another goal is that it is compatible with commercial, cloud-based, distributed computing resources(e.g. Amazon EC2, Google Cloud, and similar offerings from IBM and Intel) as well as private distributed processing systems(Beowulf, Hadoop). Initially the proposed system will provide a straightforward interface running on the popular MapReduce (*MR*v2) [18, 19] parallel processing system. From this point, extension to cloud based resources is straightforward, allowing for arbitrary resource scalability. A system is proposed that will have wide applicability to a variety of standard clustering applications while focusing on a subset of clustering problems in the biological data analysis space. Although many opportunities for clustering in biology exist, the focus of this proposal will be on noisy, high dimensional, genomic expression data gathered from cell samples [20]. Furthermore, due to new requirements involving the handling of health care data, as well as the inherent privacy concerns of using cloud based services, data security is primary to *RPHash*'s interprocess communication architecture.

## 1.1 RPHash: A Secure, Scalable Clustering Algorithm

The *RPHash* algorithm uses various recent techniques in data mining along with a new approach toward achieving algorithmic scalability. A large body of research has been applied to this field and in the past has focused on shared memory architectures with heavy data bandwidth requirements [16, 17, 21]. However, recent architectures changes and market shifts, promoted by the commercial successes of Big Data and cloud based shared commodity hardware, have driven down system bandwidth and reliance on high performance network architectures, in favor of low bandwidth, high data throughput systems [18, 19]. The current set of database and high performance clustering algorithms have had to shift their requirements and take on new responsibilities for these contemporary systems to maintain utility in the "Big Data" clustering space. *RPHash* is an algorithm designed from the ground up with these priorities and responsibility as its principle strength. *RPHash* achieves this by minimizing interprocess communication requirements, via an exchange for some redundant computation and an acceptable level of approximation. As with many parallel computations, and often computing in general, memory access and synchronization bottlenecks often dominate the computational complexity of an algorithm. To an even greater extent, theoretical speedup gains achievable from addition computing resources are negatively effected following Amdahl's Law for scaled speedup [8]. The reasons for this have been addressed since the beginning of parallel computation and are easily evidenced by the polynomial bound of the edges in a completely connected graph. For this reason, redundancy of computation on a per compute node level is suggested as an acceptable trade-off, for both system complexity and scalability, and hopefully

can achieving similar results of other communication intensive clustering algorithms.

Another of the goals of the *RPHash* clustering engine is ease of use. As it is intended for use in fields beyond computer engineering, it must be very easy to deploy. One way of providing this, is to remove the technical hurdle of building and maintaining large scale computing systems. Instead, *RPHash* relies on commercially provided, cloud computing platforms running fully configured MRv2 virtual system nodes. Furthermore, as the distributed processing architecture is a commercially provided system, the companies commercial ambitions will promote ongoing access to state-of-the-art computing resources. Cloud computing provides a pay-per-use computing resources that scale with the computational requirements without burdening researchers or their IT providers.

As *RPHash* is intended for cloud deployment, data security should be a primary concern. Privacy concerns over attacks on de-identification, or de-anonymization [2, 3] of thought to be anonymous data, such as whole genome sequence data [22], has prompted a presidential commission on the subject [1]. The results of which add new restrictions to the way researchers acquire and process data. While attempting to mitigate communication restrictions, *RPHash* intrinsically provides some security in the data it chooses to exchange. Previous attempts at securing data in distributed systems required additional cryptographic steps [23]. Namely, the randomly projected hash IDs, and the aggregation of only the k-largest cardinality vectors sets. Non-distributed data clustering requires the entire dataset to reside on the processing system, while distributed methods often require communication of full data records between nodes. In the subspace projection step of *RPHash*, nearly-orthogonal random projection is utilized as a destructive operation, providing vector anonymity for the data being during interprocess communication steps. Furthermore, fully qualified centroid vectors are communicate however, by construction, these vectors represent the aggregate of many data records. In the case of patient data, an individuals private information is unlikely to be recoverable.

## 1.2 Proposal Objectives

The overall goal of the proposal is to create a secure, and scalable cloud based clustering system of high dimensional data such as genomic data. The following research objectives will be addressed specifically:

1. **Develop the sequential *RPHash* Algorithm:** The first step in developing a parallel system is to develope the initial sequential version of the *RPHash* algorithm. The overall algorithm excluding the decoding and random projection, is fairly straight forward. An existing implementation of the GPGPU parallel Leech Lattice decoder and random projection methods from [24] will be incorporated.

2. **Develop A distributed version of the proposed *RPHash* Algorithm:** The current sequential algorithm for random projection hashing has shown both accuracy and scalability promise, prompting the next step to extend it to the parallel architecture per its original design. The parallel extension is a straightforward application of the naively parallel projection and decoding, along with a the standard log reduction sum of MapReduce [18] and other parallel gather functions [25].

3. **Deploy *RPHash* on Map Reduce(*MR*v2) Hadoop framework for parallel computation:** Building the parallel *RPHash* for the standard open source map reduce implementation will allow scalability testing of both performance and accuracy on local and publicly available cloud resources. *RPHash*'s expandability will be provided through a virtual machine core based system. The virtual machine cores will allow *RPHash* to be deployed on most cloud services that provide virtual machine support.

4. **Compare the Accuracy, Speedup, and Scalability of *RPHash*:** The availability of state of the art data clustering algorithms for Hadoop, provided by the Mahout Data Mining library, will allow comparing the accuracy, performance and scalability of *RPHash* against a variety of optimized alternatives. Comparisons will be made based on overall clustering accuracy, such as precision recall, scaled speed, and overall problem size scalability.

5. **Show *RPHash*'s innate resistance to de-anonymization attacks:** Although proving security without rigorous proof is a difficult problem, This proposal will empirically assert the resistance of *RPHash* to various de-anonymization attacks and data scraping. This rudimentary analysis, will

focus on understanding the amount of destructiveness the random projection method has on data and its resistance to least squares inversions.

### 1.3 Impact

In this proposal an algorithm for scalable data clustering on distributed systems is developed. The algorithm developed combines approximate and randomized methods in a new way to solve issues of scalability and data security for real world distributed data analysis. The focus on a randomized, and thus non-deterministic, clustering algorithm is somewhat uncommon in computing, but common for ill-posed, combinatorially restrictive problems like clustering and partitioning. This assertion is complemented by a similar inversion regarding clustering and computing, in which real world problems tend to converge much faster than adversarial crafted worse-case problems [26]. The principle assumption in this proposal is that approximate and exact clustering, are qualitative similar due to to noise, redundancy, data loss, and the curse of dimensionality. Furthermore, the addition of random noise to the clustering dataset resulting from the random subspace projection requirement, provides some degree of non-deterministic process, so subsequent iterations of the algorithm could potentially find better results. Making the process of finding better clustering results, a matter of available processing time and resources.

Clustering has long been the standard method used for the analysis of labeled and unlabeled data. Clusterings effects intrinsically identify dissimilar and similar objects in a dataset, often unattainable through standard statistical methods. Single pass, data intensive statistical methods are often the primary workhorses for parallel database processing of business logic and other domains, while clustering is often overlooked due to scalability and confusion caused by the wide variety of available algorithms [4]. A multitude of surveys [4] have been made available comparing different aspects of clustering algorithms, such as accuracy, complexity and application domain. Fields in health and biology are benefited by data clustering scalability. Such fields as Micro Array clustering, Protein-Protein interaction clustering, medical resource decision making, medical image processing, and clustering of epidemiological events all serve to benefit from larger dataset sizes.

An objective of this proposal is to create a highly scalable, cloud based, data clustering system for health care providers and researchers. This system will allow researchers to scale their clustering problems without the need for specialized equipment or computing resources. Cloud based clustering will allow researchers to arbitrarily scale processing needs using the virtually limitless commercial processing resources of large organizations.

## 2 Background

Data clustering is a useful tool in pilot data and knowledge engineering. It is used in a range of problems from practical data localization and search, to helping augment the understanding of complex biological, social, and physical structures. Although the problem itself has been shown to be NP-Hard [26, 27], many algorithms exist that usually terminate in polynomial time with good results.

### 2.1 Big Data Clustering

In light of the "Big Data" revolution, many machine learning algorithms have been adapted to work with very large, often distributed datasets. Data storage costs have exponentially decreased, while raw clock speeds, commonly on a similar exponential trajectory, have stagnated [28]. To maintain the expected Moore's law speedup, chip-makers have resorted to increasing the number of processing units per microprocessor. Overall this has resulted in a need for new parallel algorithms that support both massive datasets while also leveraging parallel processing architectures.

The goal of clustering remains the same, however memory and network constraints tend to dominate the running time requirements over algorithmic complexity alone. *RPHash* is a clustering algorithm designed from the ground up to deal with these problems head on. Instead of modifying an existing algorithm to perform better across networks and distributed systems, *RPHash* trades redundant computation for lower memory and network bandwidth requirements through the use of random projections, and universally deterministic hashes.

## 2.2 Random Projection Methods

Random projection is a method where objects are projected to a lower dimensional subspace by a projection matrix composed of Gaussian random variables. Somewhat counter-intuitive is that the random low dimensional subspace embedding is very near the optimal subspace embedding preserving the original data with minimal distortion [29]. The resurgence of the random projection method of Johnson and Lindenstrauss was reinvigorated with the work of Achlioptas on Database Friendly Projections [30], that provided good subspace embeddings requiring minimal computation cost.

**Theorem 2.1** (JL - Lemma [31])**.**

$$(1-\epsilon)\|u-u'\|^2 \leq \|f(u)-f(u')\|^2 \leq (1+\epsilon)\|u-u'\|^2$$

Johnson-Lindenstrauss lemma gives a tight bound for discretion of $n$ vectors subjected to random projections in $\mathbb{R}^d$, $\Theta(\frac{log(n)}{\varepsilon^2 log(1/\varepsilon)})$. Vempala gives a relaxation of the JL-bound of $d \sim \Omega(log(n))$ [31], with a scaling factor of $\frac{1}{\sqrt{n}}$ to preserve approximate distances between projected vectors.

An additional benefit of Random Projection for mean clustering, is that randomly projected asymmetric clusters tend to become more spherical in lower dimensional subspace representations [32]. Mean and mediod clustering algorithms, like *RPHash*, are predisposed to sphereical clusters.

## 2.3 Discrete Subspace Quantizers

Switching from the continuous spaces of random projections, a discussion on discrete space partitioning lattices is pursued. For parallization of the clustering problem, the data space must be partitioned as evenly as possible. Furthermore, a universally generative naming scheme must be established. For known datasets, the perfect partition of the data space is produced by the Voronoi partitioning [33]. In 2-dimensional space the Voronoi Diagram can be generated in $\Theta(nlog(n))$-time [34], however higher dimensional algorithms have less favorable run time complexities [35], making them inefficient for partitioning arbitrary dimensions and consequently, *RPHash*. Instead, a lattice structure is considered, with rotation and shifting transformations to compensate for gaps between spheres.

**Definition 2.2** (Lattice in $\mathbb{R}^n$ [36])**.** *let $v_1, ..., v_n$ be n linear independent vectors where $v_i = v_{i,1}, v_{i,2}, ..., v_{i,n}$ The lattice $\Lambda$ with basis $\{v_1, ..., v_n\}$ is the set of all integer combinations of $v_1, ..., v_n$ the integer combinations of the basis vectors are the points of the lattice.*

$$\Lambda = \{z_1v_1 + z_2v_2 + ... + z_nv_n | z_i \in \mathbb{Z}, 1 \leq i \leq n\}$$

Regular lattices formed from binary codes (such as the the E8 Lattice) have natural enumerations which can be used as labels for the partitioned regions. The mapping from a continuous space region to discrete binary sequence is a hash function. Furthermore, a hash functions with collision probability dependent on a distance function is called a locality sensitive hash function.

**Definition 2.3** (Locality Sensitive Hash Function [37])**.** *let $\mathbb{H} = \{h : S \to U\}$ is $(r_1, r_2, p_1, p_2)-$sensitive if for any $u, v \in S$*

1. *if $d(u,v) \leq r_1$ then $Pr_{\mathbb{H}}[h(u) = h(v)] \geq p_1$*
2. *if $d(u,v) > r_2$ then $Pr_{\mathbb{H}}[h(u) = h(v)] \leq p_2$*

## 2.4 Leech Lattice Decoder

The Leech Lattice is a unique 24 dimensional lattice with many exceptional properties beyond those focused on in this proposal [38,39]. Of particular interest to this proposal, is the Leech Lattice's packing efficiency. The Leech Lattice defines an optimal regular sphere packing of 24 dimensional space [40] and will serve nicely as a space quantizer for *RPHash*. Furthermore, the Leech Lattice, being the cornerstone of many intriguing mathematical concepts, has various relationships with other sub-lattices, which offer some useful algorithmic decompositions.

Figure 1 gives the parallel decoding algorithm for the Leech Lattice from [41] based on the hexacode decoder of Amrani and Be'ery's '96 [42] publication on decoding the Leech Lattice. The sequential worse-case complexity of this decoder requires 519 floating point operations. This decoder was chosen due to

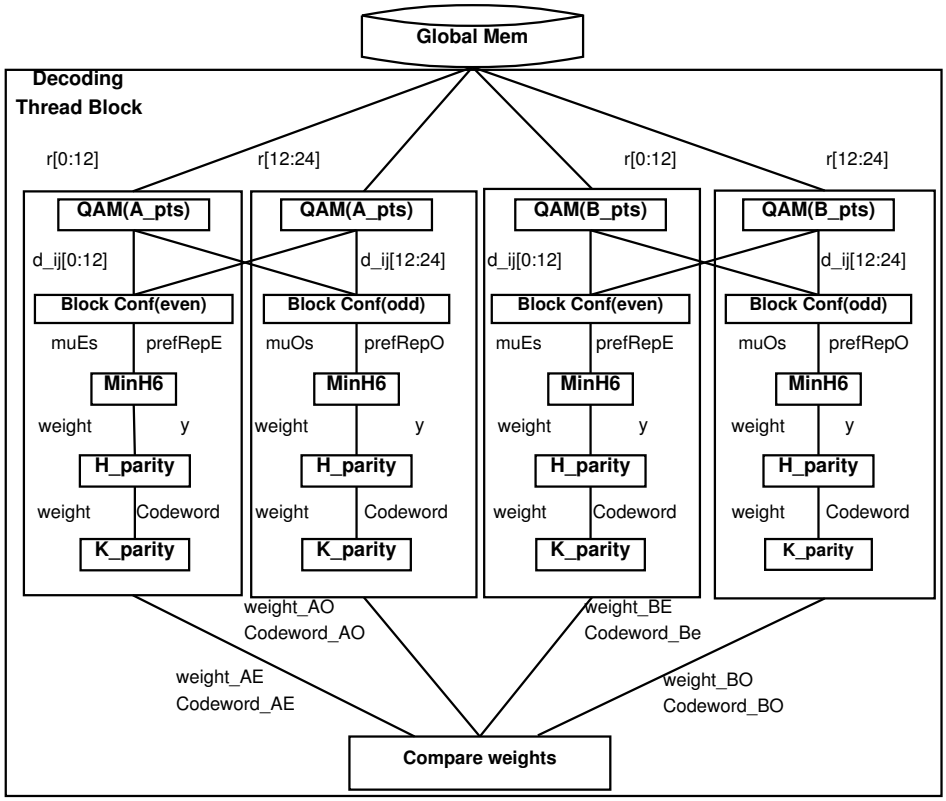


Figure 1: Parallel Leech Decoder [24]

its interesting relationship with the $(24, 12, 8)$ Binary Golay Code, and the decomposition of the Golay Code resulting in very efficient decoding algorithms. Although higher dimensional lattices exist, with comparable packing efficiency, in general decoding complexity scales exponentially with dimension [43,44].

## 2.5 Issues with random projection and high dimensions

The **Curse of Dimensionality** Consider the distance between any two points $x$ and $y$ in $\mathbb{R}^d$. under Euclidean distance: $dist(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$. Now if the vector is composed of values that have a uniformly distribution, according to Ullman [45] the range of distances are all very near the average distance between points. Given this, it becomes difficult to differentiate between near and far points, or in other words, the distance function looses its usefulness for high dimensions [46,47]. The usefulness of this principle metric in clustering suggests that approximate and exact algorithms will have an intersection at some number of dimensions. Although this intersection is not guaranteed to be at a point of favorable clustering accuracy, Vempala suggests random projection actually may solve some of these issues [31,32].

The **Occultation Problem** arises when two discrete clusters in d-dimensional space project in to two non-discrete clusters in a lower dimensional space. Specifically, the probability a u-occultation for d-dimensional space projected to 1-dimensional space given in [48] is: $1 - \frac{2}{\pi} arcos(\frac{r_1+r_2}{d-c})$ where $r_1, r_2$ is the radius of the respective clusters, and $d - c$ is the distance between them. The method employed in Urruty [48] is to repeating the projection until a non-occulting projection is found. The rate of convergence for finding a non-occulting projection is given as:

$$\lim_{d\to\infty} 1 - \left(\frac{2(r_1 + r_2)}{\pi \|d - c\|}\right)^d = 1$$

Which is exponential in $d$. Recognized in [48], this bound is slightly more favorable than the actual convergence rate, as the independence of distinct projection events is not guaranteed. However, for *RPHash*, the required 24-dimensional projection to the Leech Lattice's native subspace, is nearly orthogonal when the projection matrix is constructed from a set of i.i.d. Gaussian distributed random variables [31].

## 2.6 Mapreduce, Hadoop, and Cloud

*RPHash* will utilize state of the art frameworks, built on existing open source cloud technologies using commercial resources. **Cloud** is a somewhat nebulous term as it is often used to describe anything involving web services. In regards to this proposal, cloud will always refer to externally provided computing resources, and in particular Amazon's implementation of Hadoop on their EC2 web service. **Mapreduce** is a framework for data parallel processing using two functional programming mainstays, the map and reduce function, In addition to providing a standard parallel algorithmic structure, mapreduce also performs many of the standard resource management tasks required for parallel processing.

**Hadoop** is a freely distributed, open source implementation of the mapreduce framework. Currently in its second generation utilizing the new Yarn resource manager, Hadoop is a fully developed Map Reduce framework that is highly optimized for parallel data processing. Hadoop will provide the parallel communication and networking infrastructure while the map and reduce abstract methods will provide the parallel design of *RPHash*.

# 3 Proposed Research

Clustering algorithms offer insight to a multitude of data analysis problems. Clustering algorithm are however, often limited in their scalability and parallel speedup due to communication bottlenecks. A variety of methods have been proposed to combat this issue with varying degrees of success. In general, these solutions tend to be parallel patches and modification of existing clustering algorithms. *RPHash* is a clustering algorithm designed uniquely for low interprocess communication distributed architectures. Random projection clustering has been previously explored in [48–51], on single processing architectures.

Despite theoretical results showing that k-means has an exponential worse case complexity [52], many real world problems tend to fair much better under k-means and other similar algorithms. For this reason, clustering massive datasets, although suffering from unbounded complexity guarantees, is still a very active area in computing research. Due to approximate solution's real world proclivity to useful results, randomized methods are commonly used tools for overcoming complexity growth. The concept of random projection clustering is not new, having been explored in a variety papers involving high dimensional data clustering [9]. In the Random Projection Hash(*RPHash*) algorithm, both approximate and randomized techniques are employed to provide a scalable, approximate parallel system for massive dataset clustering. To combat the curse of dimensionality(COD) *RPHash* performs multi-probe, random projection of high dimensional vectors to the unique subset boundaries of the Leech Lattice($\Lambda_{24}$) [53].

## 3.1 Objective 1: Develop the Sequential *RPHash* Algorithm

Although *RPHash* is a parallel algorithm, initially a sequential variant will be developed to show its overall efficacy on various datasets for comparison with common clustering algorithms such as K-means and mean shift. The primary goal of this objective is to develop the sequential per compute node portion of the algorithm. A light sequential wrapper for the parallel centroid aggregation will be created in this step to demonstrate and test.

### 3.1.1 sequential algorithm

An outline of the steps in *RPHash* is given below, however some of the aspects of its function in regards to randomness and approximation are highlighted here. One way in which the *RPHash* algorithm achieves scalability is through the generative nature of its region assignment. Clustering region assignments are performed by decoding vector points into partitions of the Leech Lattice. The Leech Lattice is a unique lattice that provides optimal sphere packing density among 24 dimensional regular lattices [54]. Although optimal, due to the curse of dimensionality, the overall density is somewhat sparse, requiring that the algorithm apply shifts and rotations to the lattice to fully cover the $\mathbb{R}^{24}$ subspace. Furthermore, in general vectors will be greater than 24 dimensions. The Johnson-Lindenstrauss (*JL*) lemma to provide a solution to this problem (Figure 2). *JL* states that for an arbitrary set of n points in m dimensional space, a projection exists onto a d-dimensional subspace such that all points are linearly separable with $\epsilon$-distortion following $d \propto \Omega(\frac{log(n)}{\epsilon^2 log 1/\epsilon})$. Although many options for projections exists, a simple and sufficient

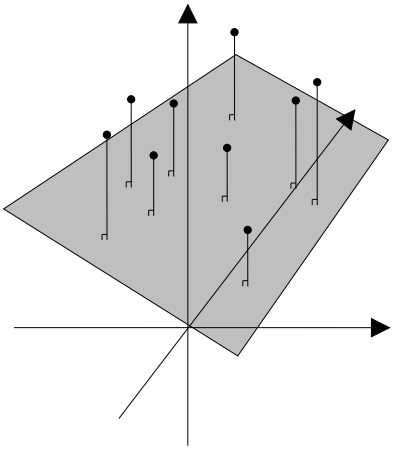

Figure 2: Random projection on a plane

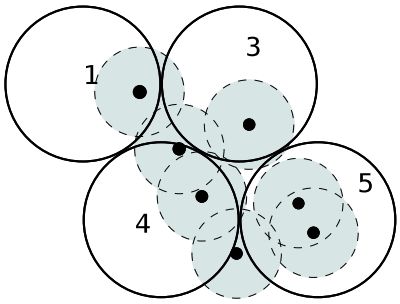


Figure 3: Multiple Projection Region Assignments and Cardinalities

method for high dimensions is to form the projection matrix $r_{ij} \in \mathbf{R}$ is $m \times d$ as follows:

$$r_{ij} = \begin{cases} +1, & \text{with probability } \frac{1}{6} \\ 0, & \text{with probability } \frac{2}{3} \\ -1, & \text{with probability } \frac{1}{6} \end{cases} \quad [30]$$

The approximation of a random projection is computationally efficient for large datasets, and unlike a truly Gaussian projection matrix, yields a semi-positive definite projection matrix, which will likely be useful in proving the convergence of *RPHash*.

In addition to Achlioptas efficient random projection method for databases, a further reduction in the number of operation required for random subspace projection called the Fast Johnson Lindenstrauss Transform(FJLT) [55–57] is currently a very active area of research. FJLT and similar nearly optimal projection methods utilize the local-global duality(Heisenberg Principle) of the Discrete Fourier Transform to precondition the projection matrix resulting in a nearly optimal number of steps to compute an $\epsilon$-distortion random projection [55–57]. A sub-linear bound on the number of operations required for the dominant projection operation may further improve *RPHash*'s overall runtime complexity.

**Overview of the sequential algorithm:** The basic intuition of *RPHash* is to combine multi-probe random projection with discrete space quantization. Following this intuition, near-neighbor vectors are often projected to the same partition of the space quantizer, which is regarded as a hash collision in LSH parlance. As an extension, multi-probe projection ensures that regions of high density in the original vector space are projected probabilistically more often to the same partitions that correspond to density modes in the original data. In other words, partitions with high collision rates are good candidates for cluster centroids. To follow common parameterized, k-means methods, the top k densest regions will be selected. Preliminary results of this process can be found in Figure 4, in which k-means(green) and a simplified *RPHash*(blue) algorithm were compared using 50:50 precision-recall analysis for varying number of Gaussian distributed clusters and number of dimensions respectively. From the result, *RPHash*'s accuracy appears to be a viable, somewhat worse performing alternative to k-means clustering.

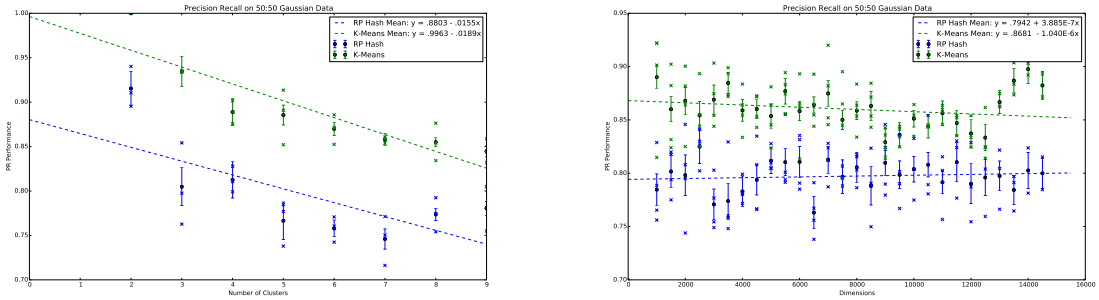


Figure 4: Precision-Recall for K-Means(green) and *RPHash*(blue) varying Vectors and Clusters

According to *JL* lemma, the sub-projections will conserve the pairwise distances in the projected space for points with $\epsilon$-distortion in which the size of the dataset is proportional to the logarithm of the number of dimensions in the randomly projected subspace. In addition to compressing a dataset to a computationally more feasible subspace for performing space quantization, random projection can also make *eccentric* cluster more spherical [31, 58].

Discrete space quantizers play a central role in the *RPHash* algorithm. The sequential implementation of the *RPHash* algorithm will rely on the efficient Leech lattice decoder of Vardy, Sun, Be'ery, and Amrani [42, 59–61] used as a space quantizer. The lattice decoder implementation relies on the decomposition of the binary $(24, 22, 8)$ extended Golay Code into 4 partitions of the $(6, 3, 4)$ quaternary hexacode and its relation to the Leech Lattice as a type B lattice construction. This decomposition and relation to the Golay Code provides a real worse case decoding complexity well below the asymptotically exponential bound for trellis decoders as the dimension $d$ increases [43, 62]. The Leech lattice is a unique lattice in 24 dimensions that is the densest lattice packing of hyper-spheres in 24 dimensional space [39, 40]. The Leech lattice is a unique lattice in 24 dimensional space with many exceptional properties, however of importance to *RPHash*, is that it is the densest regular lattice possible in 24 dimensions and nearly optimal among theoretical non-regular packings [54]. The 24 dimensional subspace partitioned by the Leech Lattice is small enough to exhibit the spherical clustering benefit of random projection. Low distortion random embeddings are also feasible for very large datasets ($n = \Omega(c^{24})$) objects while avoiding the occultation phenomena [48]. Furthermore, the decoding of the Leech lattice is a well studied subject, with a constant worse case decoding complexity of 331 operations [59].

Space quantizers have hard margin boundaries and will only correctly decode points that are within the error correcting radius of its partitions. This is an an issue found in approximate nearest neighbor search [53, 63], and is overcome in a similar way as Panigrahy [63], by performing multiple random projections of a vector then applying the appropriate locality sensitive to provide a set of hash IDs. Using many random projections of a vector allows the high dimensional vector to be represented as 'fuzzy' regions probabilistically dependent on the higher dimensional counterpart. From Panigrahy [63], the requirement of $(\Theta(log(n)))$ random projection probes is given to achieve c-approximate hash collisions for the bounded radius, r-near vectors. Figure 3 shows an example of this process as a set of probabilistic collision regions. Random projection probing adds a $\Theta(log(n))$ -complexity coefficient to the clustering algorithm but is essential for compute node independence as will be addressed in the parallel extension of this algorithm. The intuition of this step is to find lattice hash IDs that on average generate more collisions, than others. The the top $k$ cardinality set of lattice hash ID vector subsets represent regions of high density. The centroids are computed from the means of the non-overlapping subsets of vectors for each high cardinality lattice hash ID. The algorithm used above has a natural extension to the parallel algorithm describe in the following section.

In Figure 5, initial analysis of time complexity for k-means(green) and *RPHash*(blue) on a single processing node, suggest an overall linear complexity for varying number of data vectors and dimensions on Gaussian data. For a single processor, K-means surpasses *RPHash*. The result is not unexpected however, as *RPHash* intentionally trades sequential processing complexity for parallel communication independence.

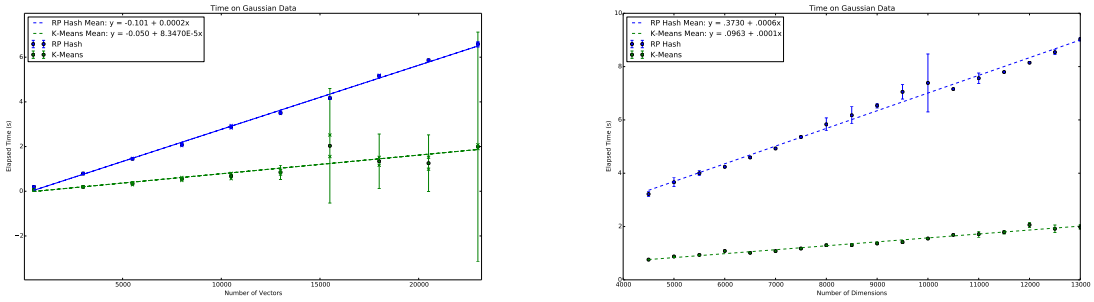


Figure 5: Computation Time for K-Means(green) and *RPHash*(blue) varying Vectors and Dimensions

## 3.2 Objective 2: Develop A distributed version of the proposed *RPHash* Algorithm

A key aspect of generative groups such as the IDs of lattice centers in the Leech Lattice, is that their values are universally computable. For distributed data systems this aspect provides complete compute node independence. Furthermore, r-near neighbor hash collisions for projected vectors do not need to share a particular basis. Instead they need only be randomly projected a bounded number of times according to results found in Panigrahy [63]. Using these tools, the *RPHash* algorithm is mostly naively parallel. The overall parallel portion of the algorithm is in fact no more complicated than a parallel sum algorithm, or reduce operation on Hadoop.

Algorithmic adaptations to from sequential *RPHash* to a parallel sum problem are yet another common tool for acheiving a work efficient scalable parallel algorithm. Due to the work efficient parallel complexity of the reduce function, the overall complexity of *RPHash* is asymptotically no worse than its per compute node sequential complexity. This complexity, as previously stated is dominated by the number of hash probes required to assure near-neighbor collisions for random projections of the $n$ vectors($\Theta(nlog(n))$).

Another common issue with parallel scalability, is the data transfer bottleneck. Although data transfer overhead cannot be completely eliminate, it can be minimize significantly be decoupling the data using generative groups relying on probable outcomes. *RPHash* achieves this by adopting a two step approach to clustering with little negative effect on accuracy. The one step data transfer requirement for directly applied parallel sum is on the order of the dimensionality of the vectors and the expected number of large clusters which may be parameterized as some $\epsilon k$. Furthermore, the real communication requirement can be further minimize by only gathering the computed IDs, and using the fact that the smallest, top $k$ cluster's cardinality $\hat{c_k}$, must be, at most greater than all other compute node's partial sums. Otherwise, there must exists a cluster $\hat{c}_{k+1}$, where $|\hat{c}_{k+1}| > |\hat{c}_k|$, therefor $\hat{c}_k \nsubseteq C$ and should not be a candidate centroid. The steps will be summarized here, as well as in standard pseudo-code algorithmic form (1 and 2 respectively).

**Phase 1 of Parallel *RPHash*: Maximum Bucket Counting:**

As *RPHash* is a distributed algorithm, the data vectors reside on $D$ independent computing nodes, connected through some non-specific, Hadoop [19] compatible network hardware. The system architecture will utilize the Yarn *MR*v2 resource manager for task scheduling and network communication load balancing. With much of the parallel task scheduling details handled by Yarn *MR*v2, the following algorithm description will focus on the per compute node task level processing.

As the random projections can be performed independently, the compute node interdependency is very low. Due to the Leech lattice's set dimensionality, conversion between vectors of $\mathbb{R}^d$ and $\mathbb{R}^{24}$ will be performed by a random projection matrix of random variables following a nearly Gaussian distribution as in Achlioptas [30]. Therefor the first parallel task is to distribute a random projection matrix to each compute node. The communication load can be further minimized by noting that the random projection matrix, although, stated as random, is actually the result of a pseudo-random computation.

A trade-off exists between the communication cost of distributing a large $\mathbb{P}_{m\rightarrow d}$ and the redundant calculation of such data per compute node. Parallel processing tuning is an effective solution to this

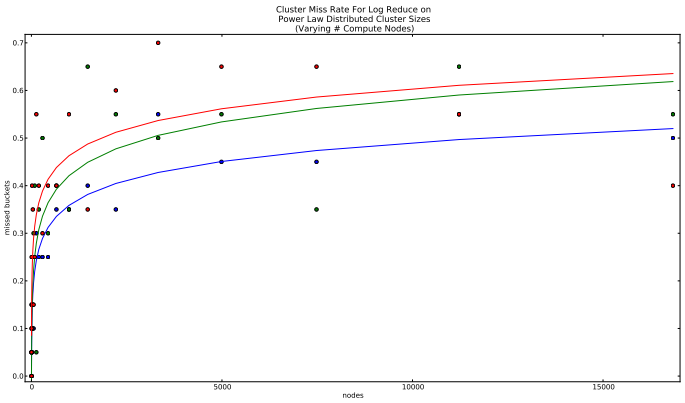


Figure 6: Random projection on a plane

problem, however in *RPHash*, the generative method is chosen, due to the constant scalability barrier that results from network bandwidth saturation. In addition empirical results from Chowdhury [64] suggesting that the ratio of communication to computation increases with the number of MR nodes.

With the initial data distribution requirements taken care of, the focus is on per node computation. Each compute node processes its set of vectors independently; first projecting the vector to a random subspace, then decoding the projected vector to its nearest Leech lattice centroid ID. The lattice region IDs are generative and thus identical across all compute nodes, an aspect discussed further in the parallel algorithm description. In the 'bucket counting' phase, the number of collisions each lattice ID receives is the only needed data. Once all data vectors have been processed and their collisions accumulated, the standard parallel sum reduce algorithm implemented in MapReduce will be used to accumulate the lattice ID counts across all compute nodes. The set of lattice ID counts is sorted, and the $k$-largest Leech lattice IDs are broadcast to all compute nodes for Phase 2 processing.

The bucket count aggregation step is susceptible to missing top k buckets under non-random density mode distribution among computing nodes. Initial testing, Figure 6, on power law distributed bucket cardinality, a common distribution for object-class memberships in natural processes [65], shows diminishing effects on the missed cluster rate for dataset sizes($10^5$(red),$10^{5.5}$(green),$10^6$(blue)) as the algorithm scales to more processors. In general the miss rate seems to converge around 0.5 misses per node aggregation phase. The decreasing missed bucket rate corresponding to the overall dataset size increase, can be attributed to the "law of large numbers", which results in each processing node being assigned a partition of bucket counts that is representational of the overall dataset. Furthermore, Karp et al [66] suggest a simple solution to a similar problem of finding frequent item sets in streams, that would provide a deterministic solution to this problem.

**Algorithm 1** Phase1 *RPHash* Clustering

**Require:** $X = \{x_1, ..., x_n\}$, $x_k \in \mathbb{R}^m$ - data vectors
$D$ - set of available compute nodes
$\mathbb{H}$ - is a d dimensional LSH function
$\widetilde{X} \subseteq X$ - vectors per compute node
$\mathbb{P}_{m \to d}$ - Gaussian projection matrix
$C_s = \{\varnothing\}$ - set of bucket collision counts
**for all** $x_k \in \widetilde{X}$ **do**
    $\tilde{x_k} \leftarrow \sqrt{\frac{m}{d}} \mathbb{P}^\intercal x_k$
    $t = \mathbb{H}(\tilde{x_k})$
    $C_s[t] = C_s[t] + 1$
**end for**
sort($\{C_s, C_s.index\}$)
**return**$\{C_s, C_s.\text{index}\}[0:k]$

**Phase 2 of Parallel *RPHash*:** The two phased approach for clustering is motivated by the work of Panigrahy, Andoni, and Indyk's [53, 63] on linear-space cr-approximate nearest neighbor search(cr-ANN). In their work they showed using time space tradeoff, that moving the multi-random projection hashing phase could be moved to the search portion of the algorithm with no effect on the overall asymptotic complexity of the algorithm. A similar tradeoff will be employ. In the first phase of *RPHash*, the candidate centroid database is constructed of the highest cardinality lattice hash buckets. While actual searching of the database is performed in the second phase. *RPHash* is effectively and inversion of cr-ANN, in the second phase, all vectors of the database are searched, against the small subset of high cardinality lattice hash ID sets. The set of lattice hash IDs corresponding to the $k * log(n)$ largest lattice hash ID subsets are broadcast to all compute nodes through standard MapReduce broadcast methods. In the case of overlapping lattice hash sets, clusters having similar centroids, either a parameterized overlap factor or some reasonable discrepancy value based on the data's information entropy is applied to the parameter $k$. Overlapping subsets can be merged during the gather phase with little effect on the overall computational complexity. An alternative to be experimentally explored, will be to simply assume buckets rarely overlap for clusters. The experimental results of which can be directly applied to the results of Urruty and Florescu, as validation that a 'good' subspace projection, although shown to grow exponentially as d increases [49], is still bounded by the probability bound on the number of multiple random subspace projections, which is also exponential [48].

The per compute node processing in Phase 2 requires $log(n)$ random projections [63] for each vector to achieve lattice hash collisions with high probability for neighboring vectors. While processing the projection and hashes, In addition to the vector counts from phase 1, the centroid partial sums must also be stored. However, instead of storing the partial sums of candidate centroids for all possible hash IDs, *RPHash* only stores the partial vector sums of the highest cardinality, broadcast $k$ - $k * log(n)$ cluster IDs from phase 1. After processing the $log(n)$-projections of $n$ vectors in addition to accumulating the per compute node centroid partial sums ,a parallel log reduce sum is used to accumulate all vector sums and frequency counts. The centroid vectors for each partial vector sum are then scaled using the set of lattice ID collision totals.

**Algorithm 2** Phase2 *RPHash* Clustering

**Require:** $X = \{x_1, ..., x_n\}$, $x_k \in \mathbb{R}^m$ - data vectors
$D$ - set of available compute nodes
$\{C_s, C_s.\text{index}\}$ - set of $klogn$ cluster IDs and counts
$\mathbb{H}$ - is a $d$-dimensional LSH function
$\widetilde{X} \subseteq X$ - vectors per compute node
$p_{m \to d} \in \mathbb{P}$ - Gaussian projection matrices
$C = \{\varnothing\}$ - set of centroids
**for all** $x_k \in \widetilde{X}$ **do**
  **for all** $p_{m \to d} \in \widetilde{\mathbb{P}}$ **do**
    $\tilde{x_k} \leftarrow \sqrt{\frac{m}{d}} p^\intercal x_k$
    $t = \mathbb{H}(\tilde{x_k})$
    **if** $t \in C_s.\text{index}$ **then**
      $C[t] = C[t] + x_k$
    **end if**
  **end for**
**end for**
**return** $C$

### 3.2.1 Problems with finding good random subspaces

For a fixed number of random projections the probability of finding a good projections converges exponentially to zero as the number of dimensions $d$ increases [49]. In this proposal, the probability of finding 'good' (or low distortion) versus unfavorable (high distortion) subspace projections, is improved by both random probing and the use of a high dimensional projection subspace following the results of Urruty [48].

### 3.3 Objective 3: Deploy *RPHash* to EC2 Cloud's Hadoop Implementation

An important goal of *RPHash* and big data mining for the computing community in general is accessibility to data scientists. *RPHash* is not focused on exotic, difficult to access systems, and instead is intended to be built for cloud processing platforms.

The first requirement of deploying the parallel algorithm on a cloud based platform is to create a simulated environment on a local system. Virtualization will be used to create this simulated environment, and scripts(vagrant, Chef, Puppet) will be written to handle the automatic deployment of virtual machines running the Hadoop *MR*v2 framework. Automatic deployment scripts will help with generating results for the speedup tests in discussed in the experimental design section for both locally and off-site EC2 Cloud, deployments.

After a local deployment system is available, an Amazon EC2 cloud account will be created for further testing. The amazon account will be managed by the researchers using funds from this grant and applied to application server time on Amazon's pay-per-use cloud. Most EC2 testing will be performed around scalability testing on synthetic data. As *RPHash* is a new algorithm, the focus will be more on how it behaves as a distributed system, and less on its qualitative clustering accuracy among the chosen data sets. This requirement will avoid the expenses associated with Amazon's services, while also not violating potential copyright and privacy issues that could arise from uploading to the cloud.

### 3.4 Objective 4: Data Security

Although certain privacies are provided by commercial cloud service providers, security is reliant on the private companies security practices. For certain problems, specifically those involving an patient's health care data, privacy safeguards must undergo rigorous auditing requirements. Such auditing is beyond the scope of many private cloud companies business models, creating a barrier for the health care community in utilizing cloud based resources. Furthermore, recent United States government initiatives pushing for the large scale availability of data resources have made vast quantities of de-identified health information available to the public. These resources however have prompted advances in attacks on de-identification of whole genome sequence data. Such attacks have been used to to associate, thought to be, anonymous medical records with specific individuals [22]. Similar de-anonymization attacks [2, 3] along with a presidential commission (privacy and progress in WGS) have prompted a need for better data security of medical records data. The *RPHash* algorithm provides an intrinsic solution to this problem in both the distribution of data among servers as well as during the communication steps required by the algorithm.

*RPHash* is a distributed algorithm designed to compute clusters on databases that could potentially span over the public internet. As such, data it uses to compute clusters can theoretically reside independently among different health care facilities with no requirement for any single location data storage architecture. As a consequence of projecting the real data vectors to a random subspace via a near, but not completely orthogonal matrix, destructive data loss occurs providing a cryptographic "trap-door' function. The data loss is an intrinsic part of the *RPHash* clustering algorithm that has little adverse effect on its model generation and subsequent recall accuracy.

The only point at which fully qualified vectors are transmitted between compute nodes is during the Phase 2 gather stage. This however is not of concern as the data represents an aggregation of individual data. By definition of a centroid it is the average of only the $k$ largest populations of patients. *RPHash* is a clustering algorithm for very large datasets, where the probability of identifying an individual from Phase 2 vector information is proportional to the size of the smallest identified cluster.

## 4 Experimental Approach

To assure *RPHash*'s accuracy and performance, tests for similarity to other algorithms, as well tests on different datasets will be performed. The experirimental approach for testing *RPHash* will address 3 major areas of *RPHash*'s utility.

- Algorithm Accuracy
- Differing Data Set Accuracy
- Scalability

- Parallel Speedup
- System Security

## 4.1 Algorithm Accuracy

A classic analysis of any clustering algorithm is its ability to correctly categorize unseen data. There are various ways of doing this such as Precision Recall, ROC, and FP/TF with varying applicability to different domain constraints. A primary resource for comparison will be the industry standard learning algorithms found in the Apache Mahout Machine learning for Hadoop Project. The algorithms in Mahout will serve as a comparison set to evaluate the classification accuracy of *RPHash* using the standard win/draw/lose contest metric [67]. In addition, the Mahout algorithms will give a comparison measure for overall speed and scaling complexity of *RPHash*.

## 4.2 Differing Data Set Accuracy

To accurately test *RPHash*, a variety of real and synthetic datasets will be incorporated. The synthetic datasets will consist of varying size and dimension, Gaussian clusters, while the real datasets will be acquired from various bioinformatics databases, as well as semantic data from health care resources. An emphasis on health care related data will be used to establish the distributed database security utility of the algorithm.

The synthetic dataset will consist of $k$ vectors from $\mathbb{R}^d$ chosen from a uniform random distribution. These vectors will form the basis for the cluster centers. For each cluster center an $n/k$ set of vectors will be generated by an additive Gaussian noise permutation added to the center vector. Ultimately this will give $n$ vectors in $k$ Gaussian clusters. From this half of the vectors will be reserved for training, and the other half for testing accuracy.

The real dataset will consist of genomic data and image SIFT vector data. These sets will give us a respective demonstration of *RPHash*'s performance on very high dimensional, and relatively high dimensional datasets.

## 4.3 Scalability

Although *RPHash* is expected to be comparably to other clustering algorithms, the primary goal of the algorithm is not to greater accuracy rather it is scalability on larger data set problems and better computing resource utilization. Standard scalability tests will focus on varying dataset sizes on synthetic data. Although real data would provide more realistic conditions, it is sufficient, in regards to proving scalability bounds, to use synthetic data. Unlike iterative methods, the *RPHash* algorithm's complexity is not adaptive to the data. Communication, and iterations are explicitly set by the size and dimensionality of the data, not the inherent clusters. The theoretical asymptotic complexity of *RPHash* is bound by the required random projection probes to achieve near neighbor bucket collisions $\Theta(nlog(n))$.

## 4.4 Parallel Speedup

Another one of the focused aims of *RPHash*h is the computational gains resulting from the addition of compute nodes. For many algorithms the achievable speedup is restricted by Amdahl's law for scalability. In which the ratio of sequential to parallelizable code reduces the available speed up even for large ratios. For this reason, *RPHash* is an attempt eliminate the sequential bottleneck. Experimentally, the scalability will be demonstrated using a varying number of nodes in the EC2 cloud. Although scaling will be performed across a large set of nodes, extrapolation will also be used to characterize the experimental scalability with the theoretical scalability. Furthermore, these results will validate the claims for *RPHash*'s overall theoretical parallel complexity and scalability while accounting for communication overhead.

An as of yet, uncertain aspect of the *RPHash* algorithm, is to avoid overlapping clusters and not disregard cluster that could potentially garner enough support to become clusters when accumulated over the many nodes in the parallel system. The potential options change the communication overhead, and therefor tests of the various implementations will be compared to find the best overall parallel speed up.

In the untested collision variant, the communication overhead is the complexity of the gather operation over the nodes, number of clusters and dimension of the data vectors ($\Theta(kdlog(N))$).

### 4.5 Security Tests

As the main goal is to show *RPHash*'s resistance to de-anonymization attacks by use of destructive projection functions, the security demonstration will be predominantly empirical. Proving the security of a system is often not possible, and stating such would be misleading. Instead, showing that no single, fully qualified record can be associated with the data sent between Hadoop nodes will suffice to show *RPHash*'s added data security feature. Fully auditing security is a difficult task, and is somewhat beyond the scope of this proposal, Furthermore, the final Hadoop implementation may expose side channel attacks, which would be beyond the scope of the *RPHash* algorithm.

Demonstrating the level of obfuscation of the data communicated between nodes running *RPHash* will be used to suggest its level of security. A common method for reconstructing a vector from its projection would be to invert the projection matrix, and apply it to the projected vector. This of course is not feasible for singular matrices and because the projections are specifically subspace projections, they will not be square. However there is some data leakage in the form of communicating the Lattice hash IDs for each vector. The hash ID represents a 24 dimensional vector, and having multiple of which could possibly allow for a least squares pseudo inverse of the projection matrix that could be used to recover information. Comparing known, correlated, fully qualified vector $u$ and the randomly projected counterpart vector $v$ and its projected inverse, using the least squares pseudo-inverse of the projection matrix $\hat{R}^{-1}$ as the mapping back to the original space, will give a qualitative measure of data obfuscation. $v = \sqrt{\frac{n}{k}}R^T u$, $v' = \sqrt{\frac{k}{n}}v^T\hat{R}^{-1}$, $similarity = ||v, v'||_2$.

For orthogonal projection matrices the least squares solution serves as a suitable inverse for the projection function. However, for non-orthogonal projection matrices the least-squares solution is over-determined, and the basis vectors will overlap, causing unrecoverable data loss. The goal in this test of *RPHash* is to show that the data loss is enough to make it very difficult to re-associate vectors with any projected data shared among Hadoop Nodes while also maintaining enough internal correlation to allow for accurate clustering.

## 5 Related Work

Clustering is a fundamental machine learning algorithm and therefor has been extensively applied to a variety of data analysis problems on a wide variety of platforms. Vector distance computation and near neighbor search, requirements innate to data clustering, are naively parallelizable. However scalability of clustering in general despite parallelization of these aspects is none-the-less dominated by communication bottlenecks. Due to the inherent scalability issues of many sequentially designed algorithms, parallel algorithms have also been developed to mitigate these issues.

### 5.1 Density Scan Database Clustering Methods

The first set of parallel clustering algorithms began with density based scanning methods. These methods tend to work well on spatially separated datasets with relatively low dimension. A very common clustering problem for these types of algorithms would be on geo-spatial data, such as large maps and image segmentation. The algorithms DBScan [68], Clique [69], and CLARANS [10], respectively, represent a successful progression of the density scanning techniques. Although density scan algorithms are an example of parallel designed algorithms like *RPHash*, they often show weaknesses in accuracy when scaling the number of dimensions. A proposed solution mentioned below to this problem is PROCLUS [9].

### 5.2 Random Projection Techniques

Another important feature of *RPHash* is the use of random projection clustering. A proof of the convergence of projected k-means clustering is given in Boutsidis [70]. Proclus [9] was an even earlier application that explored random projection for clustering. Proclus used an assumption similar to *RPHash* regarding high dimensional data's sparseness. Feature subset selection offers a way of compacting the problem

as well as removing artifacts. Many subset selection algorithms have been explored [71–74], they often require difficult to parallelize iterative gradient descent [8] or tree traversal [75]. Random projection is performed on the data to compress it to spherical clusters [58] in a dense subspace. An iterative method for k-medoid search, similar to CLARANS [10] is then used to identify clusters. Although the proposed method is successful in accelerating larger dimensionality problems ( $d = 20$) it does not have the overall proposed scalability of *RPHash*. This is due to Proclus being based on an ultimately iterative algorithm, containing inevitably sequential code, and therefor lacking indefinite scalability [8].

In addition to PROCLUS, various other methods and analysis have been performed on clustering with random projections that provide some bounds on the convergence and extensibility of random projection clustering. Florescu gives bounds on the scaling and convergence of projected clustering [49]. Their results closely follow the logic of Urruty [48], and find that the number of trial projections, or or more favorably, the number of orthogonal, projected dimensions, effects the convergence rate exponentially. Where the base is related to the arctangent of the angle between two distinct clusters. They also assert the probability of random projection offering a good partitioning plane decreases exponentially as the number of dimensions increases. This suggests *RPHash* may have to find a way to overcome this issue, or show empirically that the asymptotic behavior of increasing projected dimensions, dominates the increase in $d$.

Other clustering by random projection algorithms have been explored that are very similar to the proposed method, but for projections on the line. These so called cluster ensemble approaches [50,51,76] use histograms to identify cluster regions of high density much like *RPHash*. Although, as suggested in Florescu and Urruty, the single dimension approach may be plagued by issues of occultation, and exponential convergence as $d$ increases.

# 6 Proposed Research Schedule

The proposed project duration is 2 years. Project development will be performed at the University of Cincinnati (UC), following the schedule below in 6 month increments.

- **First**: Initial development of the sequential *RPHash* will be redesigned for efficiency, readability, and adaptability to parallel and cloud based resources. Sequential algorithm development will be transfered from its current structure to a Map and Reduce architecture used in MRv2. The algorithmic accuracy and theoretical scalability will be performed during this period on synthetic and real world data.

- **Second**: Parallel implementations of variants of the *RPHash* algorithm will be developed at the University of Cincinnati. Parallel system architecture and setup will be experimented with during this period, and an optimal set of variants will be chosen to continue with the full scale scalability and real world data testing on cloud resources.

- **Third**: Cloud resources will be acquired, and the parallel *RPHash* algorithm will be deployed for initial testing on synthetic data. Scalability testing will be performed on synthetic data prior to real world biological data analysis. Due to the lack of data security features in the set of comparison algorithms, comparison data will only be performed on synthetic data during this step.

- **Forth**: Due to privacy concerns regarding biological data on cloud resources, the secure data handling of *RPHash* will be assessed prior to any cloud processing. Following favorable assessment of data security, real world biological data will be processed using the *RPHash* algorithm on commercially available cloud resources.

# References


[1] "Privacy and progress in whole genome sequencing," report to the president, Presidential Commission for the Study of Bioethical Issues, Washington, October 2012.

[2] X. Ding, L. Zhang, Z. Wan, and M. Gu, "A brief survey on de-anonymization attacks in online social networks," in *Computational Aspects of Social Networks (CASoN), 2010 International Conference on*, pp. 611–615, 2010.

[3] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pp. 111–125, 2008.

[4] R. Xu and . I. I. Wunsch, D., "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, pp. 645–678, May 2005.

[5] M. R. Anderberg, "Cluster analysis for applications," tech. rep., 1973.

[6] H. Samet, *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann, 2006.

[7] J. Waldo, G. Wyant, A. Wollrath, and S. Kendall, *A note on distributed computing*. Springer, 1997.

[8] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), (New York, NY, USA), pp. 483–485, ACM, 1967.

[9] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park, "Fast algorithms for projected clustering," in *ACM SIGMOD Record*, vol. 28, pp. 61–72, 1999.

[10] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 5, pp. 1003–1016, 2002.

[11] J. Magidson and J. Vermunt, "Latent class models for clustering: A comparison with k-means," *Canadian Journal of Marketing Research*, vol. 20, no. 1, pp. 36–43, 2002.

[12] T. Hofmann, "Unsupervised learning by probabilistic latent semantic analysis," *Mach. Learn.*, vol. 42, pp. 177–196, Jan 2001.

[13] U. V. Luxburg and S. Ben-david, "Towards a statistical theory of clustering," in *In PASCAL workshop on Statistics and Optimization of Clustering*, 2005.

[14] R. Zass and A. Shashua, "A unifying approach to hard and probabilistic clustering," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1, pp. 294–301, Oct 2005.

[15] I. Dhillon and D. Modha, "A data-clustering algorithm on distributed memory multiprocessors," in *Large-Scale Parallel Data Mining* (M. Zaki and C.-T. Ho, eds.), vol. 1759, ch. Lecture Notes in Computer Science, pp. 245–260, Springer Berlin Heidelberg, 2000.

[16] B.-H. Park and H. Kargupta, "Distributed data mining: Algorithms, systems, and applications," pp. 341–358, 2002.

[17] T. Tamura, M. Oguchi, and M. Kitsuregawa, "Parallel database processing on a 100 node pc cluster: Cases for decision support query processing and data mining," *SC Conference*, vol. 0, p. 49, 1997.

[18] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan 2008.

[19] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, MSST '10, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2010.

[20] S. Anders and W. Huber, "Differential expression analysis for sequence count data," *Genome biol*, vol. 11, no. 10, pp. –106, 2010.

[21] M. J. Zaki and C.-T. Ho, *Large-scale parallel data mining*. No. 1759, Springer, 2000.

[22] G. D. H. E. Gymrek M, McGuire AL and E. Y, "Identifying personal genomes by surname inference," *Science*, no. 339, pp. 321–324, 2013.

[23] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Advances in Cryptology — CRYPTO 2000* (M. Bellare, ed.), vol. 1880, ch. Lecture Notes in Computer Science, pp. 36–54, Springer Berlin Heidelberg, 2000.

[24] L. Carraher, "A parallel algorithm for query adaptive, locality sensitive hash search," 2012.

[25] T. M. Forum, "Mpi: A message passing interface," 1993.

[26] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *WALCOM: Algorithms and Computation* (S. Das and R. Uehara, eds.), vol. 5431, ch. Lecture Notes in Computer Science, pp. 274–285, Springer Berlin Heidelberg, 2009.

[27] S. Dasgupta, *The hardness of k-means clustering*. Department of Computer Science and Engineering, University of California, San Diego, 2008.

[28] I. C. James Cownie, "Multicore: The software view," June 2007.

[29] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, no. 1-2, pp. 46–52, 1985.

[30] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 274–281, 2001.

[31] S. S. Vempala, *The Random Projection Method*. DIMACS Series, American Mathematical Society, 2004.

[32] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," in *in Knowledge Discovery and Data Mining*, pp. 245–250, ACM Press, 2001.

[33] R. Klein, "Abstract voronoi diagrams and their applications," in *Computational Geometry and its Applications* (H. Noltemeier, ed.), vol. 333, ch. Lecture Notes in Computer Science, pp. 148–157, Springer Berlin Heidelberg, 1988.

[34] S. Fortune, "A sweepline algorithm for voronoi diagrams," in *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, (New York, NY, USA), pp. 313–322, ACM, 1986.

[35] M. L. Gavrilova, "An explicit solution for computing the euclidean d-dimensional voronoi diagram of spheres in a floating-point arithmetic," in *Computational Science and Its Applications — ICCSA 2003* (V. Kumar, M. Gavrilova, C. Tan, and P. L'Ecuyer, eds.), vol. 2669, ch. Lecture Notes in Computer Science, pp. 827–835, Springer Berlin Heidelberg, 2003.

[36] P. V. Huffman W., *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 1 ed., 2003.

[37] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, SCG '04, (New York, NY, USA), pp. 253–262, ACM, 2004.

[38] R. T. Curtis, "A new combinatorial approach to m24," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 79, no. 01, pp. 25–42, 1976.

[39] J. Conway and N. Sloane, *Sphere Packings Lattices and Groups Third Edition*. New York: Springer, 1998.

[40] J. Leech, "Notes on sphere packings," *Canadian Journal of Mathematics*, 1967.

[41] L. A. Carraher, P. A. Wilsey, and F. S. Annexstein, "A gpgpu algorithm for c-approximate r-nearest neighbor search in high dimensions," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pp. 2079–2088, May 2013.

[42] O. Amrani, Y. Be'ery, A. Vardy, F.-W. Sun, and H. C. A. van Tilborg, "The leech lattice and the golay code: bounded-distance decoding and multilevel constructions," *Information Theory, IEEE Transactions on*, vol. 40, pp. 1030–1043, Jul 1994.

[43] V. Tarokh and I. F. Blake, "Trellis complexity versus the coding gain of lattices. i," *Information Theory, IEEE Transactions on*, vol. 42, pp. 1796–1807, Nov 1996.

[44] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *Information Theory, IEEE Transactions on*, vol. 48, pp. 2201–2214, Aug 2002.

[45] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2011.

[46] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?," in *In Int. Conf. on Database Theory*, pp. 217–235, 1999.

[47] J. Han, M. Kamber, and J. Pei, *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[48] T. Urruty, C. Djeraba, and D. Simovici, "Lecture notes in computer science," in *Advances in Data Mining. Theoretical Aspects and Applications* (P. Perner, ed.), vol. 4597, ch. Clustering by Random Projections, pp. 107–119, Springer Berlin Heidelberg, 2007.

[49] I. Florescu, A. Molyboha, and A. Myasnikov, "Scaling and convergence of projection sampling," 2009.

[50] X. Z. Fern and C. E. Brodley, "Random projection for high dimensional data clustering: A cluster ensemble approach,"

[51] R. Avogadri and G. Valentini, "Fuzzy ensemble clustering based on random projections for dna microarray data analysis," *Artificial Intelligence in Medicine*, vol. 45, no. 2, pp. 173–183, 2009.

[52] A. Vattani, "k-means requires exponentially many iterations even in the plane," in *Proceedings of the 25th annual symposium on Computational geometry*, SCG '09, (New York, NY, USA), pp. 324–332, ACM, 2009.

[53] A. Andoni, *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, Massachusetts Institute of Technology, September 2009.

[54] H. Cohn and A. Kumar, "Optimality and uniqueness of the leech lattice among lattices," tech. rep., 2004.

[55] N. Ailon and B. Chazelle, "Approximate nearest neighbors and the fast johnson-lindenstrauss transform," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pp. 557–563, 2006.

[56] A. Dasgupta, R. Kumar, and T. Sarlos, "A sparse johnson: Lindenstrauss transform," in *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, (New York, NY, USA), pp. 341–350, ACM, 2010.

[57] N. Ailon and E. Liberty, "An almost optimal unrestricted fast johnson-lindenstrauss transform," *ACM Trans. Algorithms*, vol. 9, pp. 21–1, Jun 2013.

[58] S. Dasgupta, "Experiments with random projection," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, UAI'00, (San Francisco, CA, USA), pp. 143–151, Morgan Kaufmann Publishers Inc., 2000.

[59] A. Vardy, "Even more efficient bounded-distance decoding of the hexacode, the golay code, and the leech lattice," *Information Theory, IEEE Transactions on*, vol. 41, no. 5, pp. 1495–1499, 1995.

[60] F.-W. Sun and H. C. A. van Tilborg, "The leech lattice, the octacode, and decoding algorithms," *Information Theory, IEEE Transactions on*, vol. 41, pp. 1097–1106, Jul 1995.

[61] O. Amrani and Y. Beery, "Efficient bounded-distance decoding of the hexacode and associated decoders for the leech lattice and the golay code," *Communications, IEEE Transactions on*, vol. 44, pp. 534–537, May 1996.

[62] V. Tarokh and I. F. Blake, "Trellis complexity versus the coding gain of lattices. i," *Information Theory, IEEE Transactions on*, vol. 42, pp. 1796–1807, Nov 1996.

[63] R. Panigrahy, "Entropy based nearest neighbor search in high dimensions," in *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, SODA '06, (New York, NY, USA), pp. 1186–1195, ACM, 2006.

[64] M. Chowdhury, M. Zaharia, and I. Stoica, "Performance and scalability of broadcast in spark,"

[65] W. J. Reed and B. D. Hughes, "From gene families and genera to incomes and internet file sizes: Why power laws are so common in nature," *Physical Review E*, vol. 66, no. 6, p. 067103, 2002.

[66] R. M. Karp, S. Shenker, and C. H. Papadimitriou, "A simple algorithm for finding frequent elements in streams and bags," *ACM Trans. Database Syst.*, vol. 28, pp. 51–55, mar 2003.

[67] G. Webb, "Multiboosting: A technique for combining boosting and wagging," *Machine Learning*, vol. 40, no. 2, pp. 159–196, 2000.

[68] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *KDD*, vol. 96, pp. 226–231, 1996.

[69] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, *Automatic subspace clustering of high dimensional data for data mining applications*, vol. 27. ACM, 1998.

[70] C. Boutsidis, A. Zouzias, and P. Drineas, "Random projections for $k$-means clustering," *CoRR*, vol. abs/1011.4632, 2010.

[71] R. Kohavi and D. Sommerfield, "Feature subset selection using the wrapper method: Overfitting and dynamic search space topology.," in *KDD*, pp. 192–197, 1995.

[72] P. S. Bradley, O. L. Mangasarian, and W. N. Street, "Clustering via concave minimization," *Advances in neural information processing systems*, pp. 368–374, 1997.

[73] Y. Yang and K. Chen, "Temporal data clustering via weighted clustering ensemble with different representations," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 23, pp. 307–320, Feb. 2011.

[74] S. Kaski, "Dimensionality reduction by random mapping: Fast similarity computation for clustering," 1998.

[75] J. Freeman, "Parallel algorithms for depth-first search," Tech. Rep. MS-CIS-91-71, October 1991.

[76] M. Al-Razgan and C. Domeniconi, "Weighted clustering ensembles," in *Proceedings of the SIAM international conference on data mining*, pp. 258–269, 2006.