Random Projection, Generative Lattices, and Redundancy to Combat Scalability Bounds In Distributed Computing

Lee A. Carraher

School of Electronic and Computing Systems
University of Cincinnati

March 3, 2014

Introduction

- From Cincinnati
- B.S. Computer Engineering (UC 2008)
- M.S. Computer Science (UC 2012)
    - Advisor Prof. Fred Annexstein
- Ph.D Computer Engineering (ongoing)
    - Advisor: Prof. Fred Annexstein

Interests

UNIVERSITY OF Cincinnati

Some research interests:

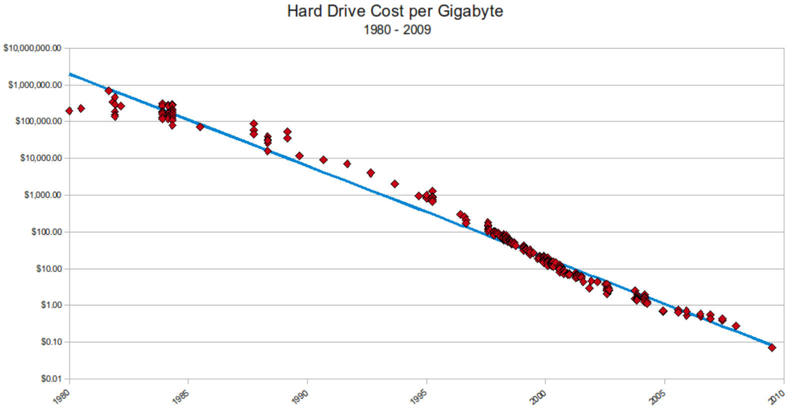
- Machine Learning (bioinformatics, filtering)
- Inverse Problems (min/max problems)
- Parallel Computing (CUDA, MPI)
- Distributed Computing (Mapreduce, Spark)
- Big Data

Motivations

“What are the important problems of your field?”
- Richard Hamming

Data Storage is Cheap

- Store Everything Because it's cheap (NSA...)

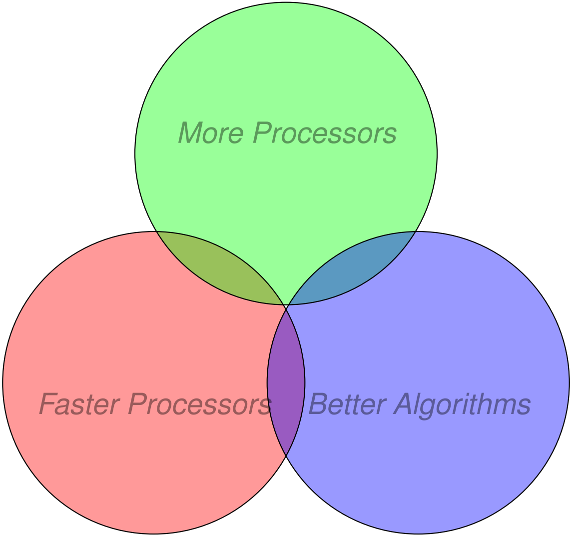


Big Problems in Computing

Two problems:

1. We are storing more data than we can effectively process ($n \to \infty$)
2. Stagnated Clock speeds
   - materials problem
   - energy problem
   - fundamental cooling problem (Landauer's Principle)

Ways To Attack Computing Problems

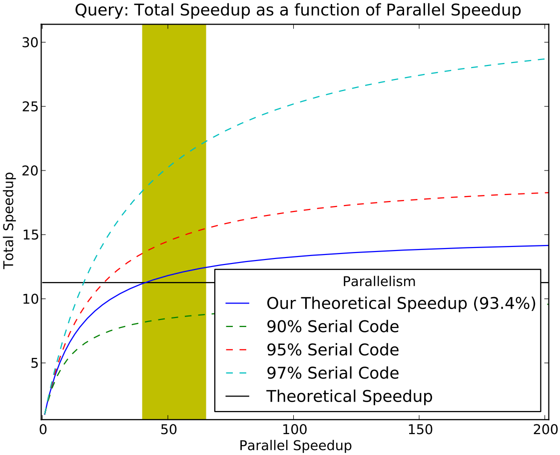


Simple, Add more processors!

Basic Issues

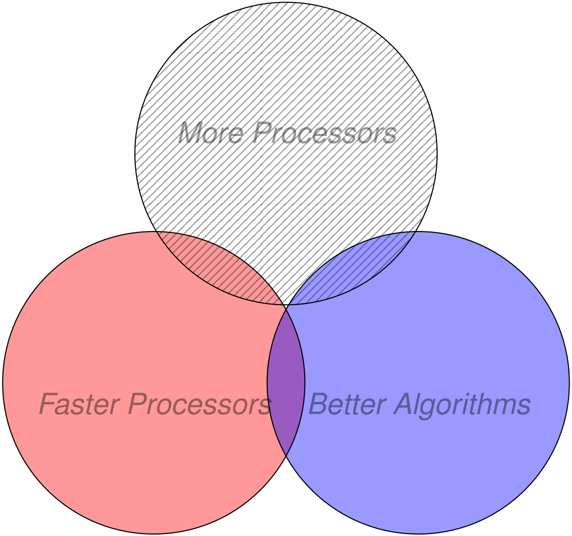
- Communication Bottlenecks
- Algorithmic Bottlenecks

Scaled Speedup Example

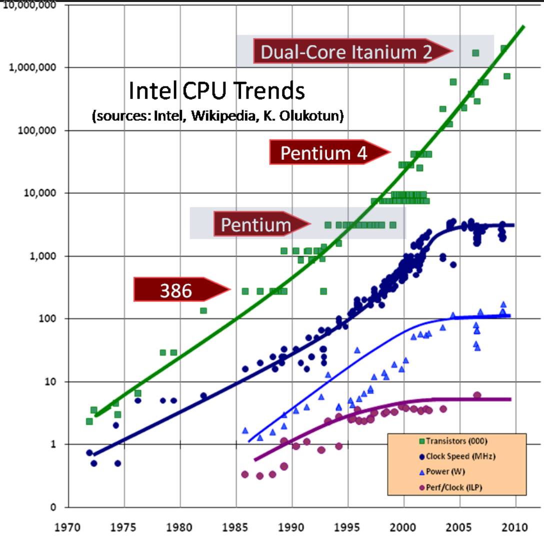


- 80/20 rule (Pareto) isn't even on here!

Ways To Attack Computing Problems

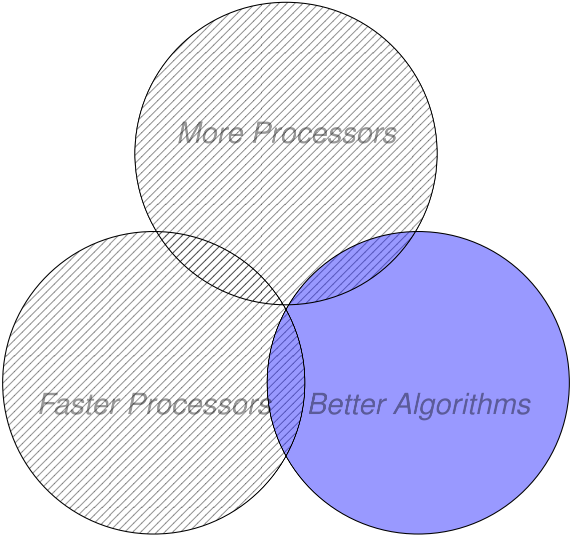


Has Moore's Law Stalled?



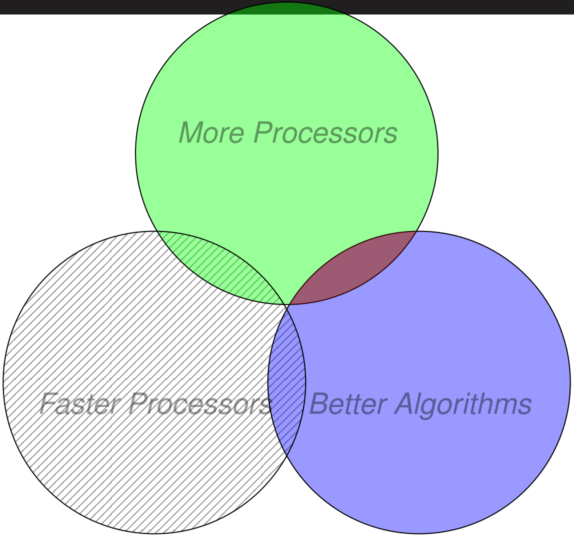
- Clock speed is dead.

Ways To Attack Computing Problems



Better Algorithms?

- This attack plan is not well defined
- Some algorithms are optimal



- What about the overlaps?

Random Projection Hashing Goals:

- Scalability
- Minimize Communication Complexity

Tradeoff:

- Redundancy
- Accuracy

UNIVERSITY OF Cincinnati

Database Clustering Methods

- DBScan
- Clique
- CLARANS
- Proclus

Background

- Big Data
- COD
- Locality Sensitive Hash Functions
- Space Partitioning
- Lattices
  - A Decoding Example
  - Leech Lattice
  - Leech Decoder
- Functional Programming
- MR/Hadoop
- Random Projection

Working Definition of “Big Data”

Sales and Commercial hype aside,

Definition (Big Data)

A set of data processing problems in which the required data is too large to reside in main memory.

Thrashing between MM and HD(even solid state) $\triangleright$ unscalable algorithm

- Health Metrics
- DNA Sequences
- Website/Click Metrics

Curse of Dimensionality
COD is sometimes cited as the cause for the distance function loosing its usefulness for high dimensions. This arises from the ratio of metric space partitioning to hypersphere embedding.

$$\lim_{d \to \infty} \frac{Vol(S_d)}{Vol(C_d)} = \frac{\pi^{d/2}}{d 2^{d-1} \Gamma(d/2)} \to 0$$

Given a single distribution, the minimum and the maximum distances become indiscernible. Or the relative majority of space is outside of the sphere

Definition (Locality Sensitive Hash Function)

let $\mathbb{H} = \{h : S \rightarrow U\}$ 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 $Pr_{\mathbb{H}}[h(u) = h(v)] \leq p_2$

For this family $\rho = \frac{\log p_1}{\log p_2}$

An Example Hash Family

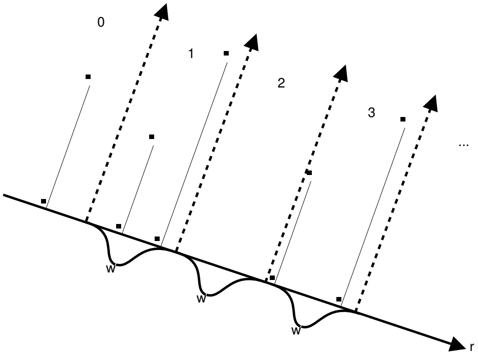


Figure : Random Projection of $\mathbb{R}^2 \rightarrow \mathbb{R}^1$

Voronoi partitioning is optimal in 2D.

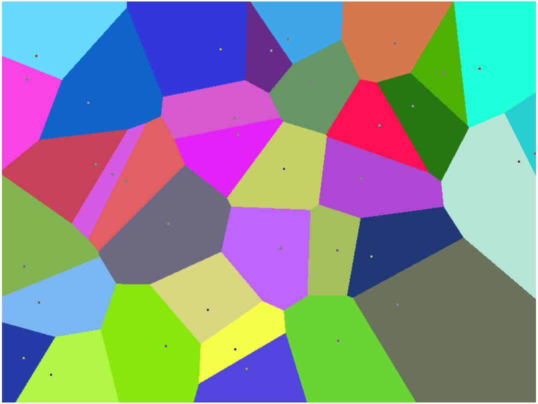


Figure : Voronoi Partitioning of $\mathbb{R}^2$

Voronoi

- Voronoi diagrams make for very efficient hash functions in 2d because, by definition, a point within a Voronoi region is nearest to the regions representative point.
- **Voronoi regions provide an optimal solution to the NN partitioning in 2-d Space!**
- However, for arbitrary dimension d, Voronoi diagrams require $\Theta(n^{d/2})$-space, and no known optimal point location algorithms exists.

Instead we will consider lattices, which provide regular space partitioning and scale to arbitrarily large dimensional space, and have sub-linear nearest center search algorithms associated with them.

Definition (Lattice in $\mathbb{R}^n$)

let $v_1, ..., v_n$ be n linear independent vectors where $v_i = v_{i,1}, v_{i,2}, ..., v_{i,n}$ The lattice Λ 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.

$$\Lambda = \{z_1 v_1 + z_2 v_2 + ... + z_n v_n | z_i \in \mathbb{Z}, 1 \leq i \leq n\}$$

Examples in 2D

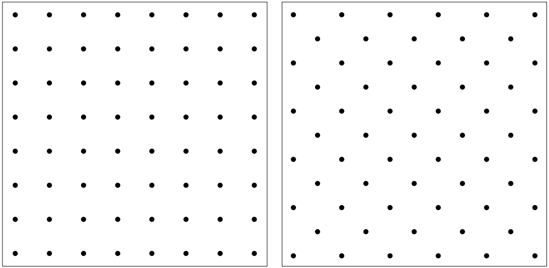


Figure : Square(left) and Hexagonal(right) Lattices in $\mathbb{R}^2$

Constant Time Decoding

- Certain lattices allow us to find the nearest representative point in constant time.
- For example the above square lattice.
  - The nearest point can be found by simply rounding our real valued point to its nearest integer.
- With the exception of a few **exceptional** lattices, more complex lattices have more complex searches (exponential as d increases).

Exceptional Higher Dimensional Lattices

The previous lattices work well in $\mathbb{R}^2$, but our data spaces are in general $\gg 2$.

- fortunately there are some higher dimensional lattices, with efficient nearest center search algorithms.
- E_8 or Gosset's Lattice, is one such lattice in
- it is also the densest lattice packing in $\mathbb{R}^8$.

Example of decoding E_8

E_8 can be formed by gluing two D_8 integer lattices together and shifting by a vector of $\frac{1}{2}$. This gluing of less dense lattices and shifting by a "glue vector" is a common theme in finding dense lattices.

- Decoding D_8 is simple
- $E_8 = D_8 \cap D_8 + \frac{1}{2}$
- both cosets of D_8 can be computed in parallel
- D_8's decoding algorithm consists of rounding all values to their nearest integer value **s.t** they sum to an even number

define $f(x)$ and $g(x)$ to round the components of x, except in $g(x)$ we round the furthest value from an integer in the wrong direction.
let

$$x =< 0.1, 0.1, 0.8, 1.3, 2.2, -0.6, -0.7, 0.9 >$$

then

$$f(x) =< 0, 0, 1, 1, 2, -1, -1, 1 >, sum = 3$$

and

$$g(x) =< 0, 0, 1, 1, 2, \mathbf{0}, -1, 1 >, sum = 4$$

- since $g(x)$ is even, it is the nearest lattice point in D_8

Example of decoding E_8 conti.

.

- Include the coset $\cap D_8 + \frac{1}{2}$.
- We can do this by subtracting $\frac{1}{2}$ from all the values of x.

$$f(x-\frac{1}{2}) =< 0,0,0,1,2,-1,-1,0 >, sum = 1$$

$$g(x-\frac{1}{2}) =< \mathbf{-1},0,0,1,2,-1,-1,0 > sum = 0$$

Now we find the coset representative that is closest to x using a simple distance metric.

$$||x-g(x)||^2 = 0.65$$

$$||x-g(x\frac{1}{2})||^2 = 0.95$$

So this case it is the first coset representative:

$$< 0,0,1,1,2,0,-1,1 >$$

Leech Lattice

By gluing sets of E8 together in a way originally conceived by Curtis' MOG, we can get an even higher dimensional dense lattice called the Leech lattice.

Leech Lattice

Here we will state some attributes of the leech lattice as well as give a comparison to other lattices by way of Eb/N_0 and the computational cost of decoding.
Some Important Attributes:

- Densest Regular Lattice Packing in $\mathbb{R}^{24}$
- Lattice Construction can be based on 2 cosets $\mathbb{G}_{24}$
- Sphere Packing Density: $\frac{\pi^{12}}{12!} \approx 0.00192957$
- $K_{min} = 196560$

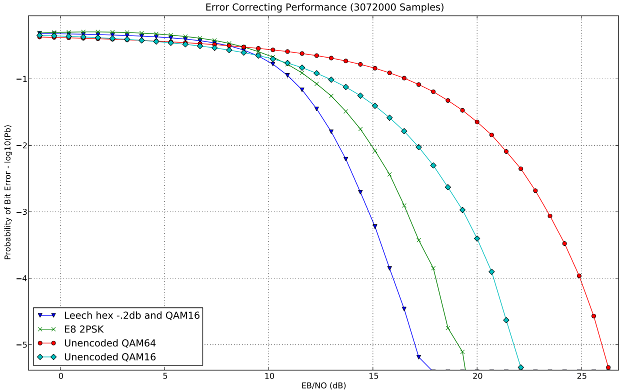


Figure : Performance of Some Coded and Unencoded Data Transmission Schemes

Leech Lattice Decoding

Some information about the decoding algorithm:

- The decoding of the leech lattice is based closely on the Decoding of the Golay Code.
- In general, advances in either Leech decoding or binary Golay decoding imply an advance in the other.
- The decoding method used in this implementation is based on Amrani and Be'ery's '96 publication for decoding the Leech lattice, and consists of around 519 floating point operations and suffers a gain loss of only 0.2bB.
- In general decoding complexity scales exponentially with dimension.

Next is an outline of the decoding process.

Leech Decoder

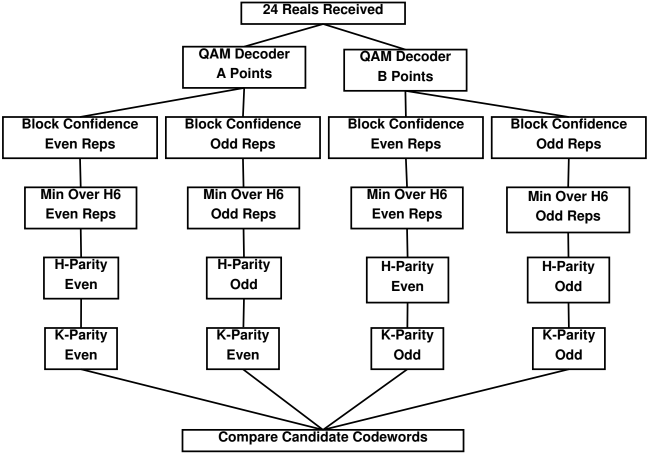


Figure : Leech Lattice Decoder

Functional Programming

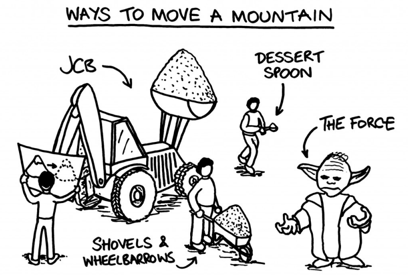


Figure : Scargill, by Tim (timble.me.uk/blog/author/tim)

Parallel and Functional Programming

- Instead of moving the mountain to the people,Move the the people to the mountains
- Where mountains are data and people are functions respectively

Map Reduce Program Design

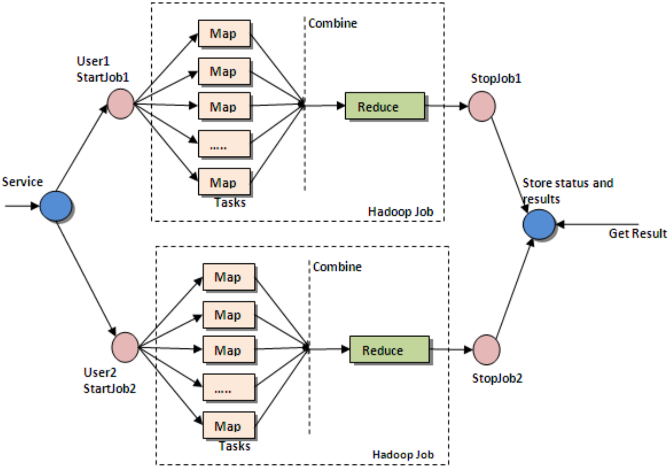


Figure : Map Reduce (courtesy: map-reduce.wikispaces.asu.edu

Hadoop

Hadoop is an open source implementation of the map reduce framework created and maintained by the Apache Software Group. **Benefits**

- Open Source
- Popular, and Maintained
- Free
- Implemented and compatible with Amazon EC2
- Takes care of the networking and fault tolerance drudgery of parallel system programming.

Hadoop Parallel System Design

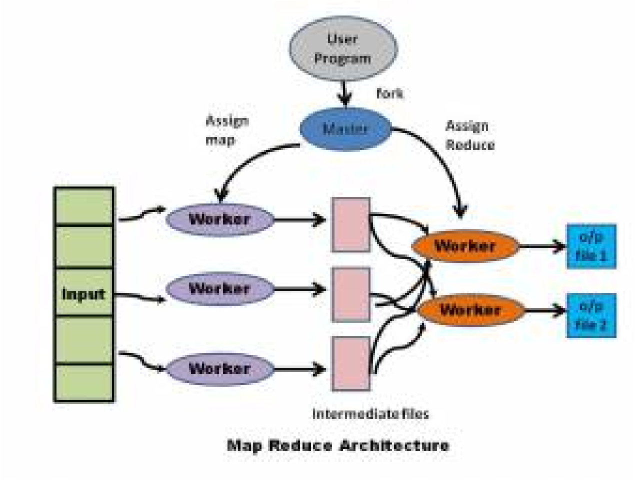


Figure : Hadoop System Design (ibm.com/developerworks)

Cloud and Distributed Services

- Scalable to data processing problems needs
- Very Low Cost Processing Model
- Always Up to Data HW Resources
- Zero HW maintenance and overhead costs

Mahout

Mahout is an open source library of machine learning algorithms made for Hadoop.

Clustering Algorithms:

- Canopy Clustering
- K-Means
- Mean Shift
- LDA
- MinHash

Random Projection

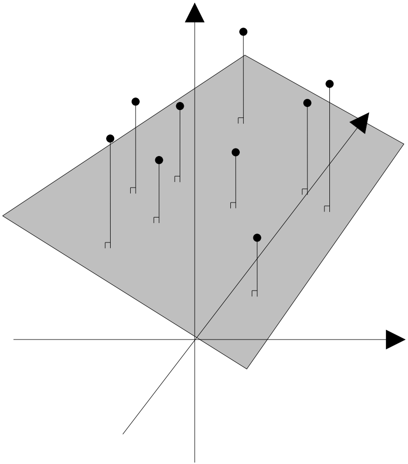


Figure : Random Projection: $\mathbb{R}^3 \rightarrow \mathbb{R}^2$

Theorem (JL - Lemma)

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

ϵ - is an distortion quantity
$u, u' \in U$ - two independent vectors
f - a random projection mapping

$$\mathbb{R}^d \rightarrow \mathbb{R}^l$$
$$l \propto \Theta\left(\frac{log(n)}{\varepsilon^2 log(1/\varepsilon)}\right)$$

Fast Johnson-Lindenstrauss Transform

- New and cool!
- Using Heisenberg Uncertainty in Harmonic Analysis, a spectrum and its signal cannot both be concentrated
- Precondition projection with DFT, (some matrices have very fast DFTs)
- $\approx \Theta(d \log(d) + \epsilon^{-3} \log^2(n))$ vs $\Theta(d\epsilon^{-2} \log(n))$

Motivations of RP Hash

- Parallel Structure of a scalable parallel algorithm (Log Reduce)
- Low Communication Overhead (Hashes)
- Non -Parallel Iterative Structure (Per core redundancy)
- Approximation is usually good enough

General Idea of RPHash

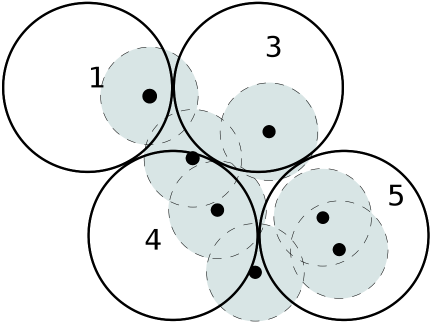


Figure : Multi-Probe Random Projection Intersection Probabilities

- generative space quantization
- random projection
- sequential multi-probe stochastic process

Occultation Problem

The occultation problem is the probability of two or more independent distributions overlapping in projected space.

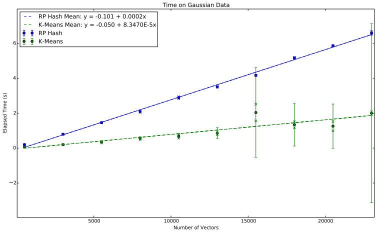
- based on the distribution variance and angle of the projective plane
- Applicable bounds from Urruty '07. *d* is number of probes
  - $\lim_{d \to \infty} 1 - \left(\frac{2(r_1 + r_2)}{\pi \| d - c \|} \right)^d$
- In RPHash, *d* is the dimensionality (24).
- RPHash projections are orthogonal

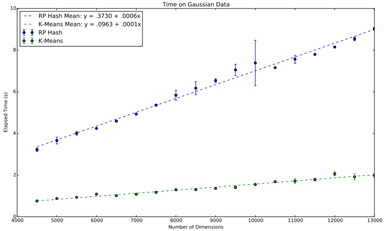
Sequential algorithm

1. Generate Random Projection matrix P
2. Maintain DB_{count} of hash id's
3. Maintain DB_{cent} Array of centroids corresponding to vectors
4. Forall $x \in X$:
    4.1 index $= LatticeDec(xP^{\top})$
    4.2 $DB_{count}[\text{ID}]{+}{+}$
    4.3 $DB_{cent}[\text{ID}] + = x$
5. sort$[DB_{count}, DB_{cent}]$
6. return $DB_{cent}[0 : k]$

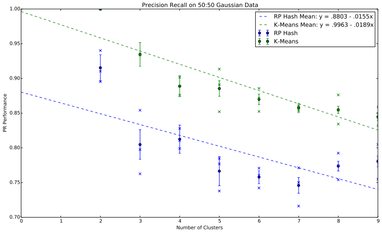
Comparison with standard k-means

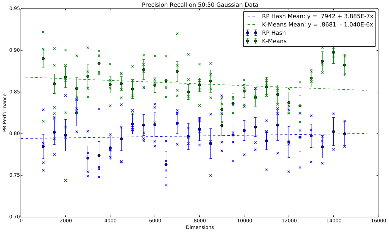
Sequential Algorithm Time Results





Sequential Algorithm Accuracy Results

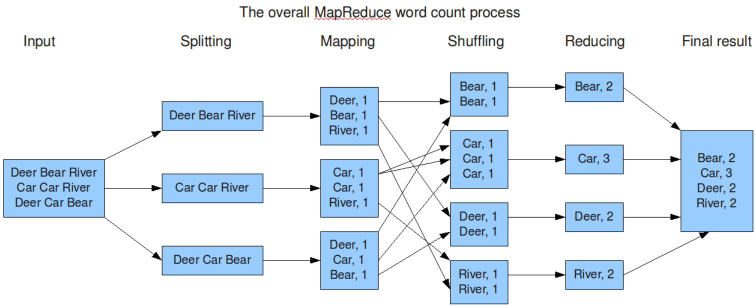




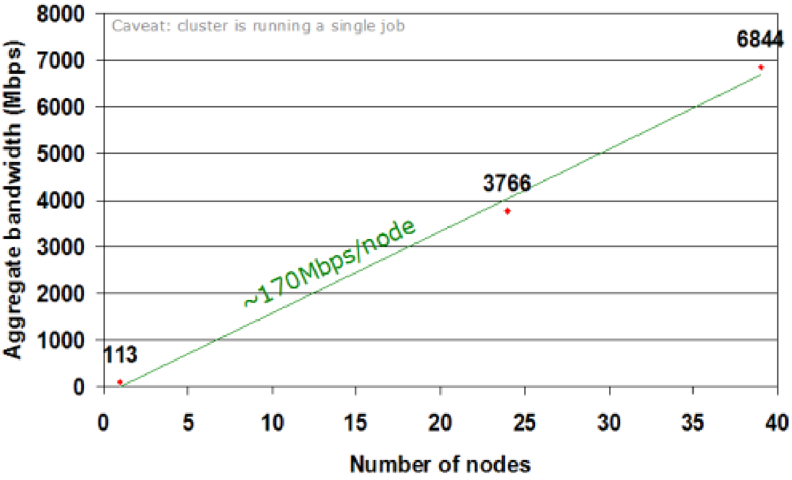
Scalability Goal

- It would be nice if our algorithm scaled with the number of processing nodes
- lets look at an algorithm that scales well and try to apply it to our problem
- the canonical hadoop ”Hello World” simulacrum ”Word Count” is a good place to start

Hadoop Word Count



Hadoop Word Count Scalability



Basic Outline

- Each processing node computes hashes and counts
- Share the top k buckets to all computers
- Aggregate all centroid averages

A Trick to minimize communication and storage reqrmnt.

- Two Phase:
- Phase 1: Only store counts and communicate IDs
- Phase 2: Only accept hash collisions with Phase 1's top IDs for all clusters

UNIVERSITY OF Cincinnati

begin

$X = \{x_1, ..., x_n\}$, $x_k \in \mathbb{R}^m$ - 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

foreach $x_k \in \widetilde{X}$ **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

sort($\{C_s, C_s.index\}$) **return** $\{C_s, C_s.\mathrm{index}\}[0 : k\log(n)]$

end

Parallel Algorithm Phase 2

begin
- $X = \{x_1, ..., x_n\}$, $x_k \in \mathbb{R}^m$ - 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
- **foreach** $x_k \in \widetilde{X}$ **do**
  - **foreach** $p_{m \to d} \in \widetilde{\mathbb{P}}$ **do**
    - $\tilde{x}_k \leftarrow \sqrt{\frac{m}{d}} p^\intercal x_k$
    - $t = \mathbb{H}(\tilde{x}_k)$ **if** $t \in C_s.index$ **then**
      - $C[t] = C[t] + x_k$
    - **end**
  - **end**
- **end**
- **return** C

end

What to Use this For?

UNIVERSITY OF Cincinnati

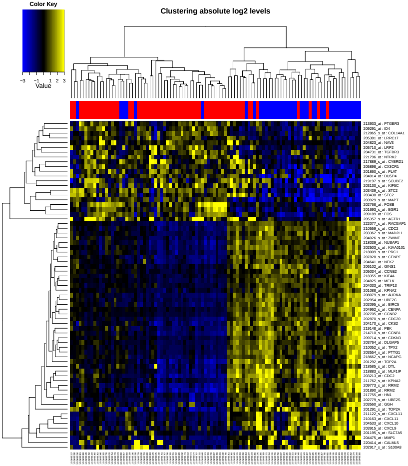
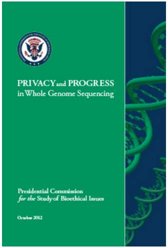


Figure : Gene Expression Levels in Primary Breast Cancer Tumor Samples

Data Security in BioInformatics



- New attacks on anonymized data present a risk to patients privacy.
- Very few cloud services guarantee secure processing.
- Highly distributed systems add even more attack vectors.
- Attacks prompted a Presidential commission on WGS privacy.

Data Security: A Freebie!

- Random Projection Offers Some Protections
- The only full vectors transmitted are cluster centroids, which by definition are an aggregate of many vectors.
- Showing dissimilarity should be somewhat straightforward
  - $v = \sqrt{\frac{n}{k}} R^T u, v' = \sqrt{\frac{k}{n}} v^T \hat{R}^{-1}$
  - $similarity = ||v, v'||_2$

Questions ?