An Efficient K-means Clustering Algorithm on MapReduce

Q Li, P Wang, W Wang, H Hu, Z Li, J Li, In Proc. of DASFAA 2014

Presented by:

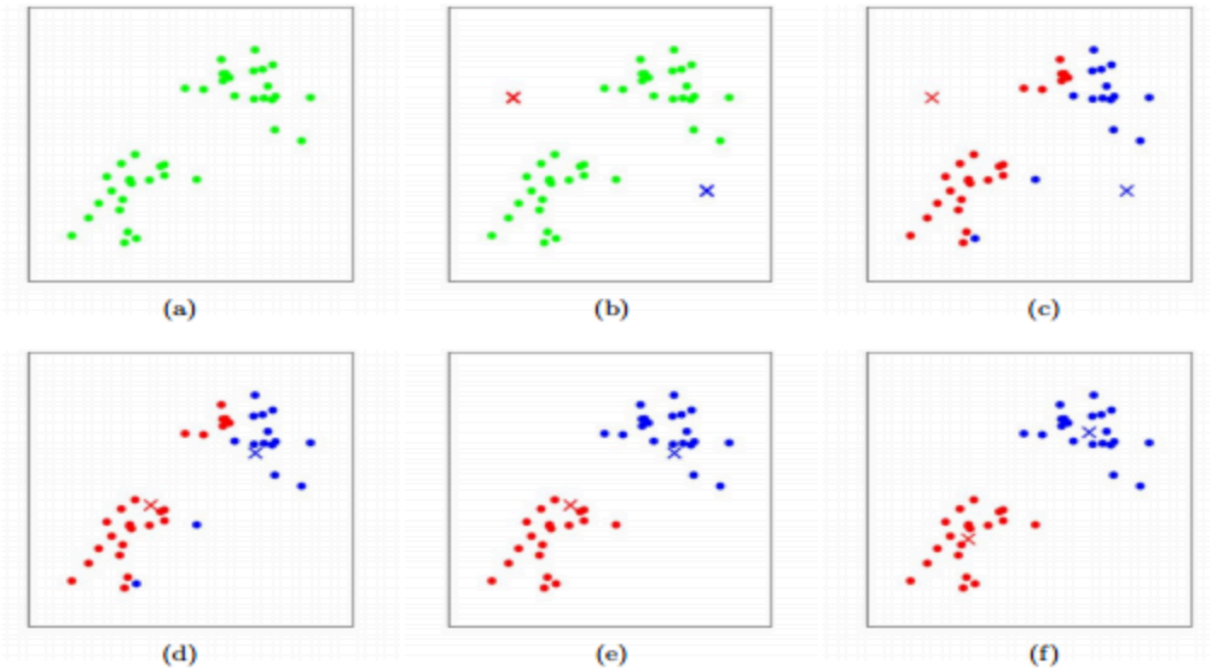
Duan Yurou

17M38118

K-means

- What is K-means

1. Randomly select *k* cluster centers
2. Calculate the distance between Each data point and cluster centers
3. Assign the data point to the most close cluster center
4. Recalculate the new cluster center in each group
5. Recalculate the distance between each data point and new obtained cluster centers
6. If no data point was reassigned then stop, otherwise repeat from step 3



K-means

- Main Disadvantages

1. Requires decide number of cluster centers empirically
2. Sensitive to the selection of initial cluster centers
3. Time consuming in clustering the massive high-dimensional data (O(nkt))

K-means++

- Improve the quality of initial centers

Algorithm 1. k-means++ initialization

Require: X : the set of points, k: number of centers;

$C \leftarrow$ sample a point uniformly at random from X

while $|C| \leq k$ **do**

  Sample $x \in X$ with probability $\frac{d^2(x,c)}{\phi_X(C)}$

  $C \leftarrow C \bigcup x$

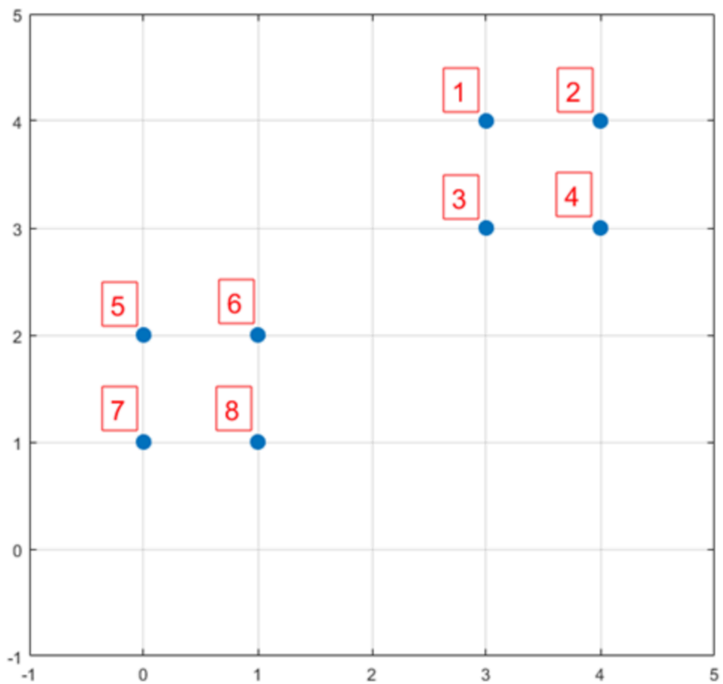
end while

- Main Disadvantages

  Inherent sequential nature

K-means++

- Example



Notice: Point6 is cluster center

序号	①	②	③	④	⑤	⑥	⑦	⑧
$D(x)$	$2\sqrt{2}$	$\sqrt{13}$	$\sqrt{5}$	$\sqrt{10}$	1	0	$\sqrt{2}$	1
$D(x)^2$	8	13	5	10	1	0	2	1
$P(x)$	0.2	0.325	0.125	0.25	0.025	0	0.05	0.025
Sum	0.2	0.525	0.65	0.9	0.925	0.925	0.975	1

K-means||

- Improve parallelism of K-means++

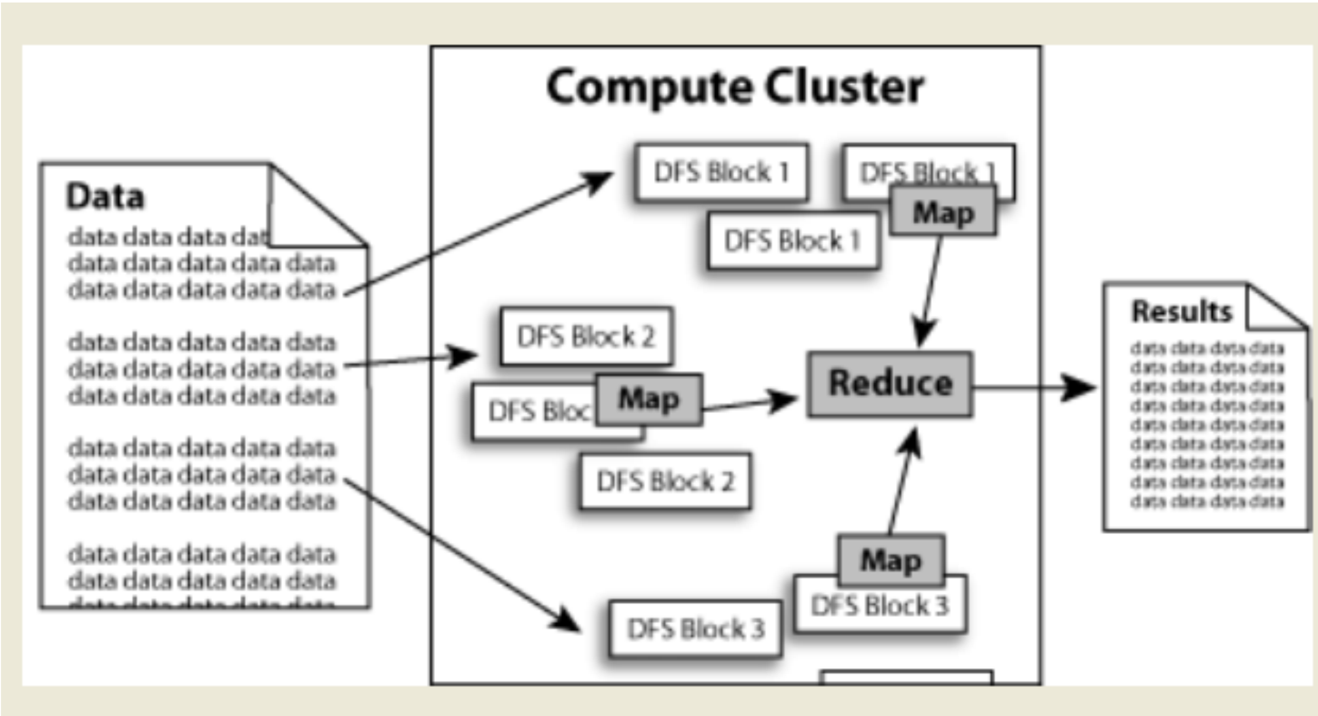
Algorithm 2. k-means|| initialization

Require: X : the set of points,l: the number of centers sampled for a time, k: number of centers;

1: $C \leftarrow$ sample a point uniformly at random from X
2: $\psi \leftarrow \phi_X C$
3: **for** $o(\log \psi)$ **do**
4: Sample C' each point $x \in X$ with probability $\frac{l \cdot d^2(x,C)}{\phi_X(C)}$
5: $C \leftarrow C \bigcup C'$
6: **end for**
7: For $x \in C$, set w_x to be the number of points in X closer to x than any other point in C
8: Recluster the weighted points in C into k

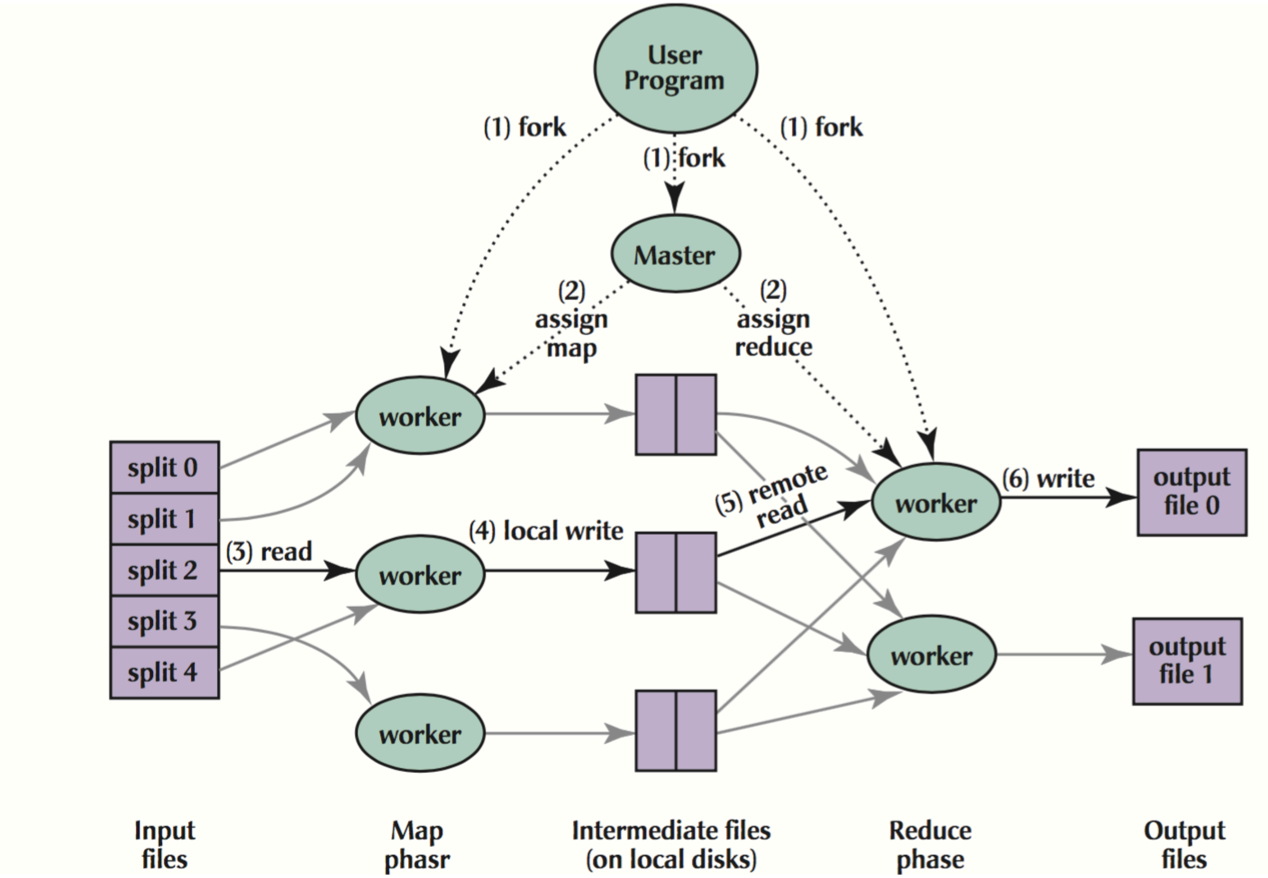
Hadoop

- Apache Hadoop is an open-source software framework
  - Hadoop Distributed File System (HDFS)
  - *Hadoop MapReduce* – a model for large-scale data processing.



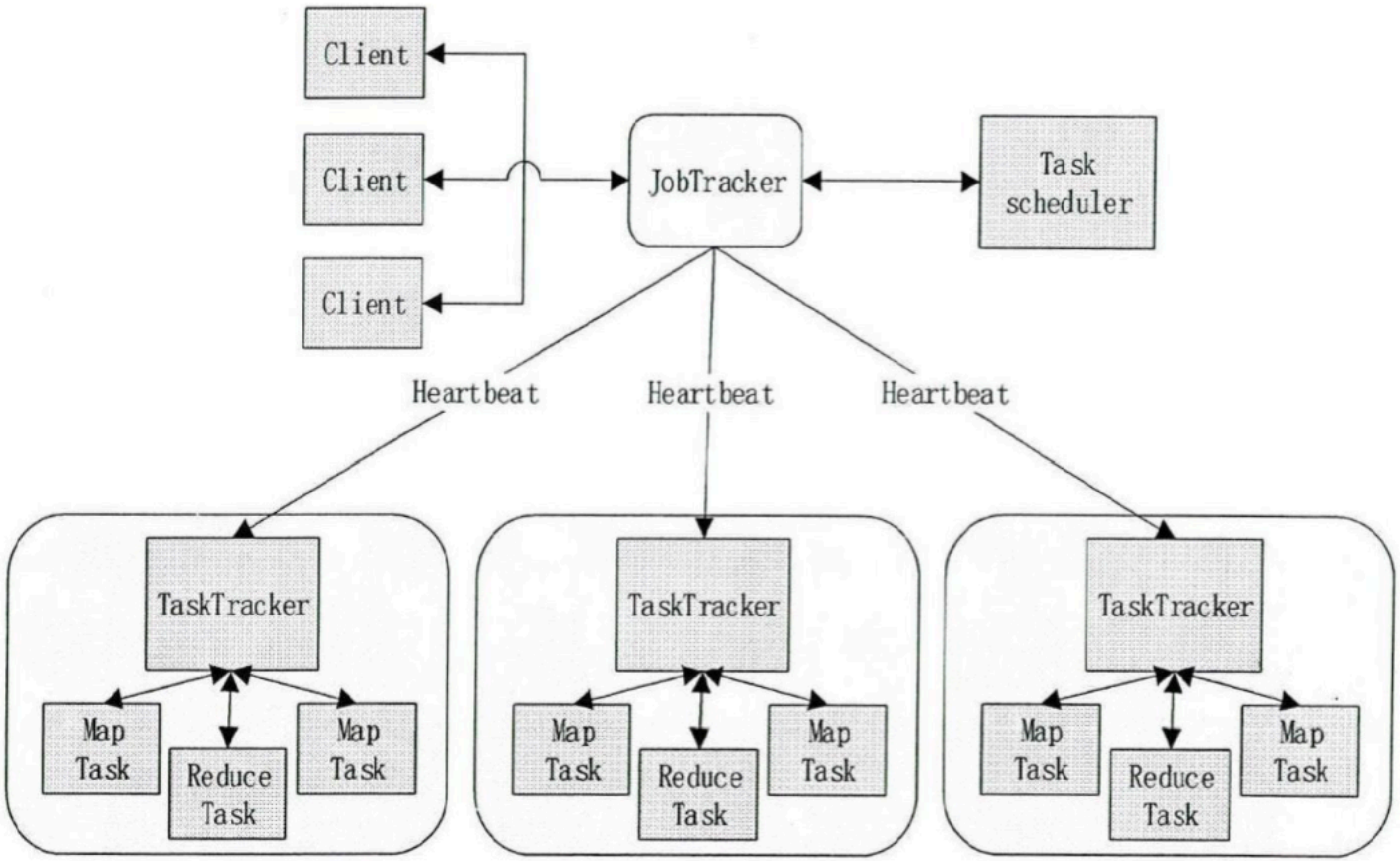
MapReduce

- Structure



MapReduce

- Client
- JobTracker
- TaskTacker
- Task



LSH

- Locality Sensitive Hashing
  - Property: Points in high-dimensional data that are close to each other wil have higher probability to be close after LSH functions
  - Aim: High-dimensional data similarity search

Definition 1. *A function family* $\mathcal{H}=\{h : S \to U\}$ *is called* $(r; cr; p_1; p_2)$*- sensitive for* D *if for any* $v; q \in S$

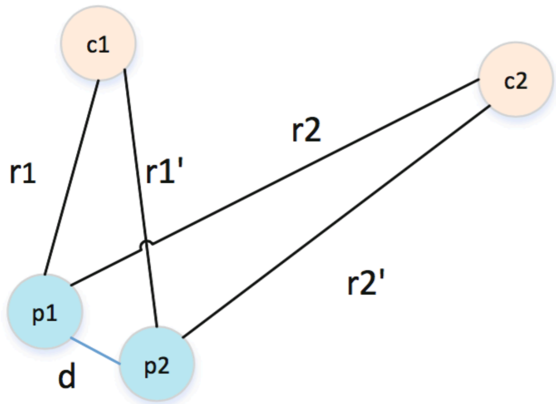
- *if* $v \in B(q, r)$ *then* $P_{rH}[h(q) = h(v)] \geq p_1$,
- *if* $v \notin B(q, cr)$ *then* $P_{rH}[h(q) = h(v)] \leq p_2$.

- different distance functions
  - Eclidean distance

$$h_{a,b}(v) = \left\lfloor \frac{a.v + b}{r} \right\rfloor$$

LSH for Data Skeleton

- Group similar points together



Theorem 1. *Given c_1 and c_2 as two centers, p_1 and p_2 as two points with the distance d, r_1, $r_{1'}$, r_2 and $r_{2'}$ are the distances between p_1, p_2 and c_1, c_2 respectively. If $r_1 < r_2$ and $r_2 - r_1 > 2 * d$, then it holds that $r_{1'} < r_{2'}$.*

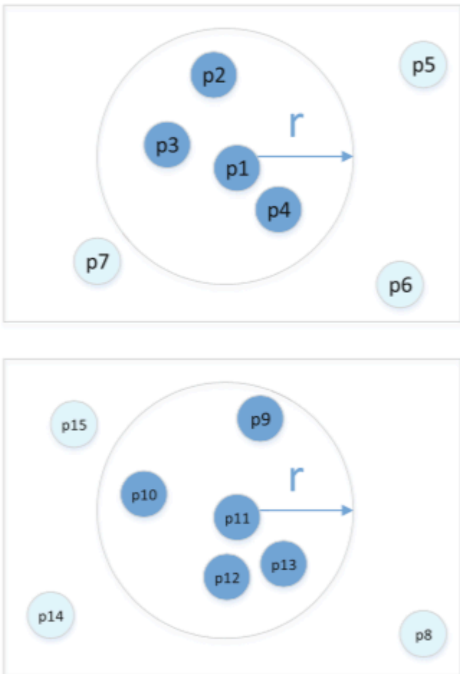
Proof. According to triangle inequality, we have $r_1 - d < r_{1'} < r_1 + d$ and $r_2 - d < r_{2'} < r_2 + d$. Therefore, we have

$$\begin{aligned} r_{1'} &< r_1 + d \\ &< r_2 - 2d + d && (r_2 - r_1 > 2 * d) \\ &= r_2 - d \\ &< r_{2'} && (r_2 - d < r_{2'}) \end{aligned}$$

LSH for Data Skeleton

- Representative data point

$$< p_r, L_p, weight >$$



	Data Skeleton	Weighted Point
Bucket (1,0,6)	$(p_1,<p_2,p_3,p_4>,4)$	$(p_1,4)$
	$(p_5,null,1)$	$(p_5,1)$
	$(p_6,null,1)$	$(p_6,1)$
	$(p_7,null,1)$	$(p_7,1)$
Bucket (4,6,9)	$(p_{11},<p_9,p_{10},p_{12},p_{13}>,5)$	$(p_{11},5)$
	$(p_8,null,1)$	$(p_8,1)$
	$(p_{14},null,1)$	$(p_{14},1)$
	$(p_{15},null,1)$	$(p_{15},1)$

Improve center initialization

- Data partitioning and weight initialization
  - Divide the points in data skeleton into |B| blocks
  - Map phase: Assign each point to a block randomly; Calculate the distances between these points to current Centers
  - The sampling weight for a weighted point

    $< x, w_x >$ is $w_x * d^2(x, C)$, denoted as wp_x.
  - Reduce phase: Compute the sum of the weight in each block

    $\sum_{x \in B_i} wp_x$, denoted as wb_i

- Sampling L centers in K-means||

- Update the weights
  - Map phase

Pruning Strategy based on LSH

- First Strategy based on Theorem 1
  - To adjust the centers, we don't need to compute the distance between centers and all points
  - C1 is the nearest center for all points represented by pr

$$d(p_r, c_2) - d(p_r, c_1) > 2\varepsilon$$

Theorem 1. *Given c_1 and c_2 as two centers, p_1 and p_2 as two points with the distance d, r_1, $r_{1'}$, r_2 and $r_{2'}$ are the distances between p_1, p_2 and c_1, c_2 respectively. If $r_1 < r_2$ and $r_2 - r_1 > 2 * d$, then it holds that $r_{1'} < r_{2'}$.*

Proof. According to triangle inequality, we have $r_1 - d < r_{1'} < r_1 + d$ and $r_2 - d < r_{2'} < r_2 + d$. Therefore, we have

$$\begin{aligned} r_{1'} &< r_1 + d \\ &< r_2 - 2d + d && (r_2 - r_1 > 2 * d) \\ &= r_2 - d \\ &< r_{2'} && (r_2 - d < r_{2'}) \end{aligned}$$

Pruning Strategy based on LSH

- Second Strategy based on Theorem 2
  - Reduce the centers to be compared
  - Only compute the distance between a point and it's nearby buckets' centers
  - We only compute the distance between p and centers in set:

$$\{c||h(c) - h(p)| < \delta_h\}$$

Theorem 2. *Given a LSH function:* $h_{a,b}(v) = \lfloor \frac{a \cdot v + b}{r} \rfloor$. *If* $| h_{a,b}(v_1) - h_{a,b}(v_2) | \geq \delta_h$*, then we have* $d(v_1 - v_2) \geq \frac{(\delta_h - 1) \cdot r}{|a|}$.

Proof. *According to definition of LSH, we have* $| h_{a,b}(v_1) - h_{a,b}(v_2) | = | \lfloor \frac{a \cdot v_1 + b}{r} \rfloor - \lfloor \frac{a \cdot v_2 + b}{r} \rfloor | \geq \delta_h$. *We can conclude that* $| \frac{a \cdot v_1 + b}{r} - \frac{a . v_2 + b}{r} + 1 | \geq \delta_h$. *Therefore, we have* $| \frac{a \cdot (v_1 - v_2)}{r} | \geq \delta_h - 1$. *We have* $| v_1 - v_2 | \geq \frac{r(\delta_h - 1)}{|a \cos \theta|} \geq \frac{r \cdot (\delta_h - 1)}{|a|}$. *Here* θ *is the angle between point* a *and vector* $v_1 - v_2$. ∎

Pruning Strategy based on LSH

- Combine the two Strategy
  - Compute threshold δ_h

Require: Set[1:k] C, parameter a, b, r, ε

Pruning-Map(Key k, Value v)

begin

    Set $< p_r, L_p, weight > \leftarrow v$

    $hash1 = h(p_r)$

    get $so-far-closest$ from the closest bucket from $hash1$ using binary search;

    $dis = d(p_r, so-far-closest)$

    $\delta_h = \frac{|a| \cdot dis}{r} + 1$

Pruning Strategy based on LSH

- Compute Strategy 2

Set $C' = null$
for c in C **do**
    $hash2 = \left\lfloor \frac{a \cdot c + b}{r} \right\rfloor$
    Set $diff \leftarrow abs(hash1 - hash2)$
    if $diff \leq \delta_h$ **then**
        $C' = C' + \{c\}$
    end if
end for

Pruning Strategy based on LSH

- Compute Strategy 1

get closest center c' for p_r in C'
$min = distance(p_r, c')$
$closeSet = null$
for c in C' **do**
    $dis2 = d(p_r, c)$
    if $\mid dis2 - min \mid \leq 2\varepsilon$ **then**
        $closeSet = closeSet + \{c\}$
    end if
end for

Pruning Strategy based on LSH

- Find closest center for Pr

```
    if closeSet = null then
        for p in L_p do
            Output(c', p)
        end for
    else
        for p in L_p do
            get closest center cen' from closeSet
            Output(cen', p)
        end for
    end if
    Output(c', p_r)
end
```

Pruning Strategy based on LSH

- Reduce: to calculate new center

Pruning-Reduce(Key k, Set $values$)
begin
    $mean = (\sum_{v \epsilon values} v)/sizeof(values)$
    $center =$ nearest point from $mean$
    $Output(center,$ null$)$
end

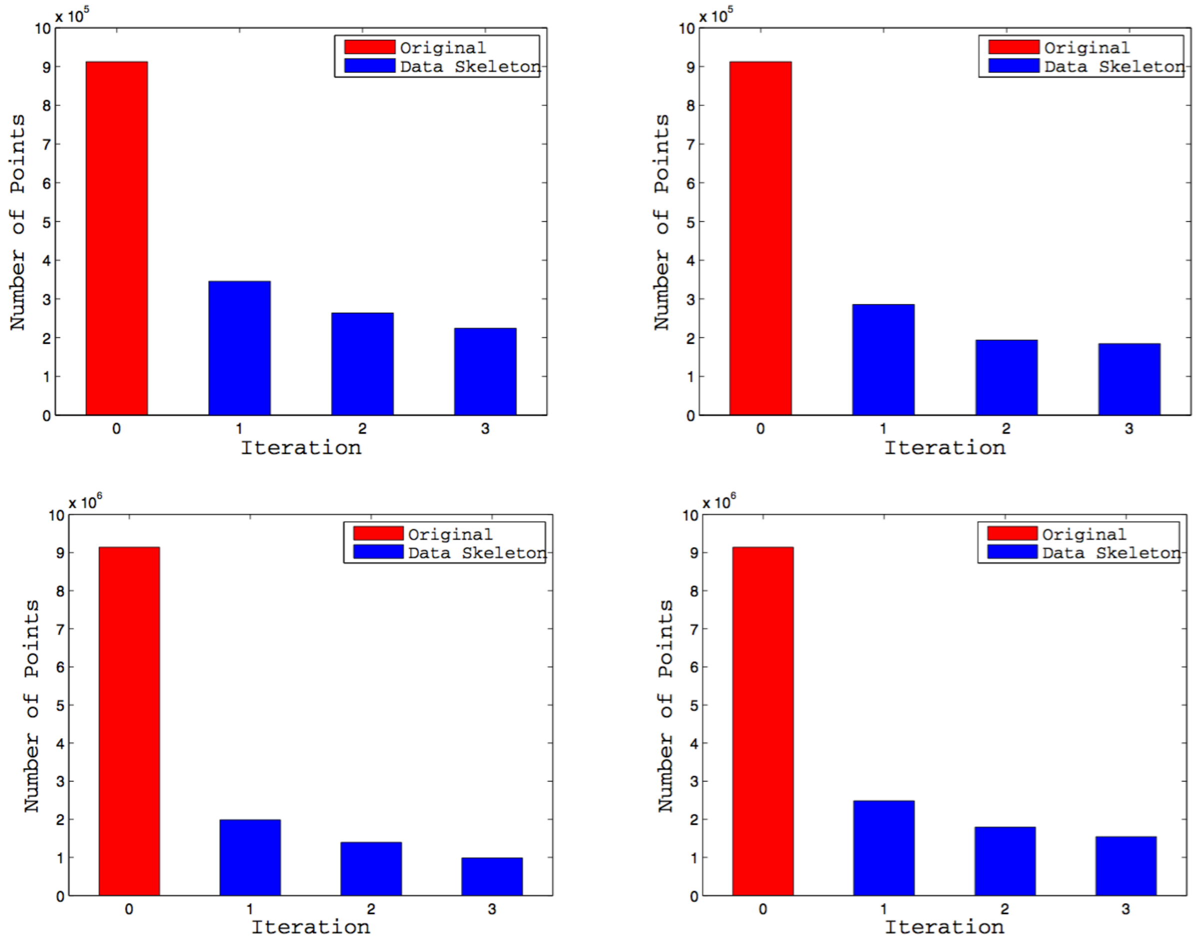
Experiment

- Environment
  - a cluster of 14 computers
  - Two Pentium(R) Dual-Core (2.70GHz) CPU E5400 and 4GB of memory
  - Linux. Hadoop version 0.20.3 and Java 1.6 are used as the MapReduce system.

- Dataset
  - KDDCUP1999
  - Self-build music database
    - 919711 Mp3 songs
    - POP, classical and folk music
    - 26-dimension represents a frame of the song

Results

- Data Reduction of Data Skeleton
  - For KDDCUP1999 60s vs Above 600s for k=1500
  - For Music Frames 130s vs Above 1567s for k=1500



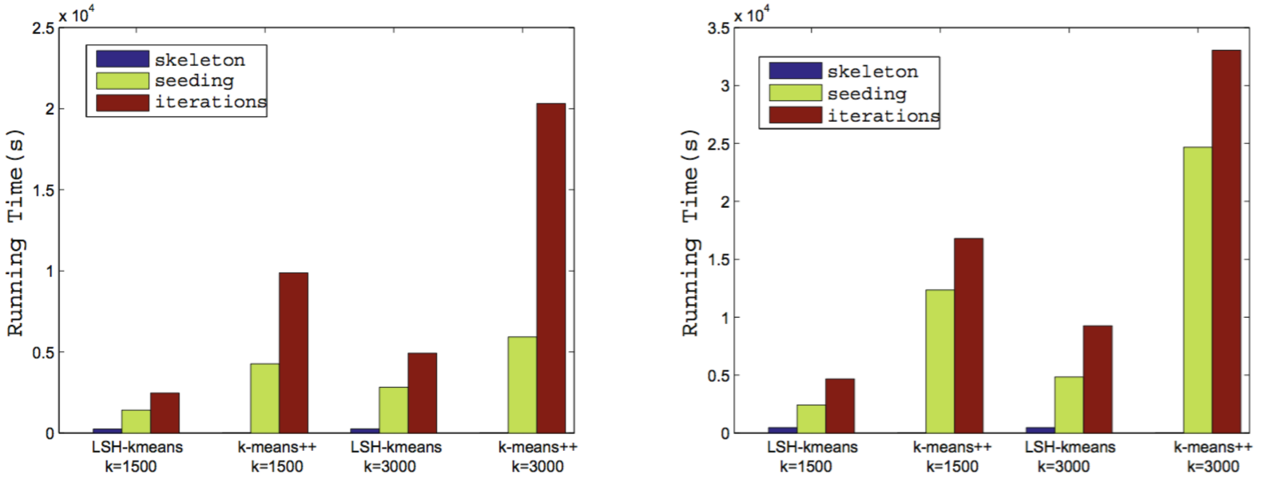
KDDCUP1999

Music Frames

(a) r=0.05 (b) r=0.01

Results

- The Center Initialization
  - The time for using LSH-kmeans is about 1/3 that of k- means++
  - Cost comparison



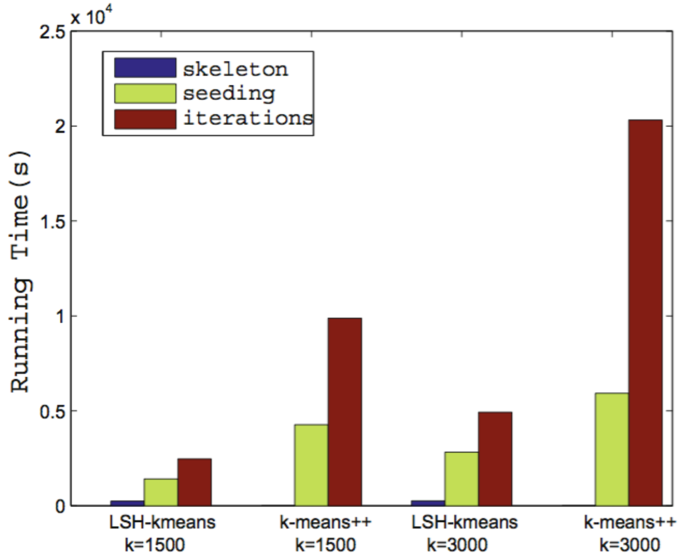
(a) KDDCUP1999 (b) Music Frames

Table 1. Comparison of Clustering Cost (k=3000)

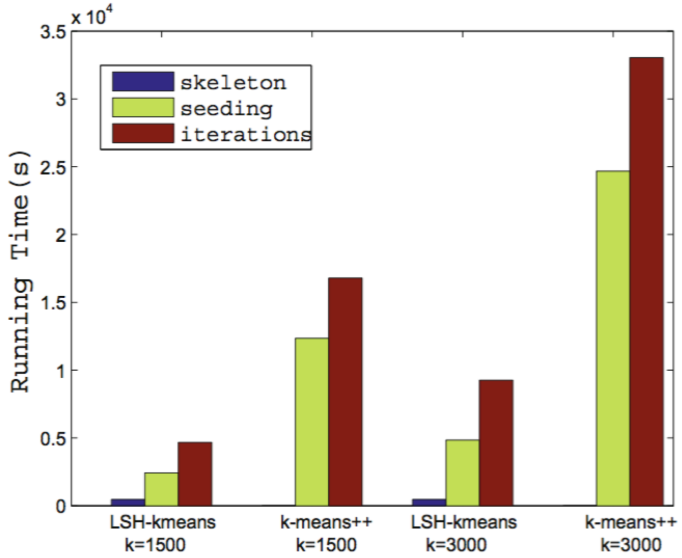
Iteration	Cost of Original Dataset	Cost of Data Sleleton
1	47824.77	47664.18
2	40292.91	40200.01
3	38318.60	38222.02
4	37474.73	37355.58
5	37019.76	36950.85
6	36714.02	36672.52

Results

- The Overall Performance Comparisons
  - KDDCUP1999 : The time cost is reduced by 67% when k is 1500, and 76% when k is 3000
  - Music frame: The time cost is reduced by 57% when k is 1500, and 64% when k is 3000.



(a) KDDCUP1999



(b) Music Frames

Conclusion

- Cluster high- dimensional data on MapReduce with the LSH technology
- Evaluate its performance on several datasets
- Improve the clustering performance dramatically without decreasing the quality.

Thank you!