PAMAE: Parallel k-Medoids Clustering with High Accuracy and Efficiency

Hwanjun Song
KAIST
songhwanjun@kaist.ac.kr

Jae-Gil Lee*
KAIST
jaegil@kaist.ac.kr

Wook-Shin Han
POSTECH
wshan@postech.ac.kr

ABSTRACT

The k-medoids algorithm is one of the best-known clustering algorithms. Despite this, however, it is not as widely used for big data analytics as the k-means algorithm, mainly because of its high computational complexity. Many studies have attempted to solve the efficiency problem of the k-medoids algorithm, but all such studies have improved efficiency at the expense of accuracy. In this paper, we propose a novel parallel k-medoids algorithm, which we call *PAMAE*, that achieves both high accuracy and high efficiency. We identify two factors—"global search" and "entire data"—that are essential to achieving high accuracy, but are also very time-consuming if considered simultaneously. Thus, our key idea is to apply them *individually* through two phases: *parallel seeding* and *parallel refinement*, neither of which is costly. The first phase performs *global* search over *sampled* data, and the second phase performs *local* search over *entire* data. Our theoretical analysis proves that this serial execution of the two phases leads to an accurate solution that would be achieved by global search over entire data. In order to validate the merit of our approach, we implement PAMAE on Spark as well as Hadoop and conduct extensive experiments using various real-world data sets on 12 Microsoft Azure machines (48 cores). The results show that PAMAE significantly outperforms most of recent parallel algorithms and, at the same time, produces a clustering quality as comparable as the previous most-accurate algorithm. The source code and data are available at https://github.com/jaegil/k-Medoid.

CCS CONCEPTS

•Information systems → Clustering; •Theory of computation → MapReduce algorithms;

KEYWORDS

k-medoids; PAM; parallelization; sampling; Hadoop; Spark

1 INTRODUCTION

The k-medoids algorithm is not as widely used for *big data analytics* as the k-means algorithm mainly because of its high computational complexity, though the k-medoids algorithm is more robust to noises or outliers than the k-means algorithm [2, 12, 23, 26, 28]. The most common realization of k-medoids clustering—the Partitioning Around Medoids (PAM) algorithm [18]—has *quadratic* time complexity [12], whereas the k-means algorithm has *linear* time complexity with respect to the number of objects. In fact, major distributed machine learning platforms such as Apache Mahout [3] and Apache Spark MLlib [4] officially support only the k-means algorithm between the two algorithms.

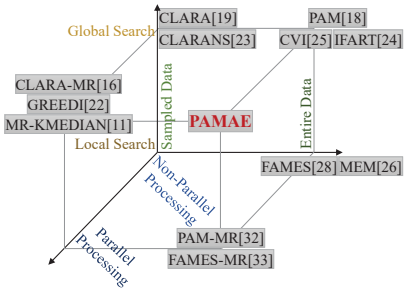


Figure 1: Categorization of the k-medoids algorithms.

1.1 State-of-the-Art Algorithms

Considerable effort has been devoted to solving the efficiency problem of the k-medoids algorithm, which can be categorized into three dimensions as shown in Figure 1. We note that only our algorithm is located at the best position ⟨global search, entire data, parallel processing⟩.

1. **Global search vs. local search**: The set of medoids is repeatedly updated through multiple iterations until a stopping condition is met. Finding a new medoid in the next iteration can be done in two ways. In *global search*, a new medoid can be found from the outside of the corresponding cluster; on the other hand, in *local search*, a new medoid is found only from the inside of the cluster. It is evident that local search improves efficiency because it confines the search scope to a single cluster. However, local search suffers from the local optima problem, possibly decreasing accuracy [27]. FAMES [28] and MEM [26] perform local search instead of global search.
2. **Entire data vs. sampled data**: In general, clustering can be performed against either the entire or a sampled set of objects. Because the size of the latter is much smaller than that of the former, using sampled data significantly improves efficiency. However, if the set of true medoids is far from a sampled set of objects, the result deviates from the optimal clustering. CLARA [19] and CLARANS [23] are the variations of PAM that are based on sampling.
3. **Non-parallel processing vs. parallel processing**: Ever since MapReduce came about, there have been several efforts to extend the existing algorithms for MapReduce. PAM-MR [32], FAMES-MR [33], and CLARA-MR [16] are the extensions of

*Jae-Gil Lee is the corresponding author.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.

DOI: http://dx.doi.org/10.1145/3097983.3098098

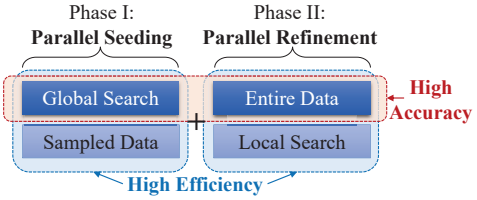


Figure 2: Key idea for the PAMAE algorithm.

PAM, FAMES, and CLARA, respectively. Although these MapReduce algorithms offer higher efficiency than their base algorithms, they do not guarantee high accuracy. FAMES-MR performs local search, and CLARA-MR is based on sampling. PAM-MR performs local search in order to parallelize the medoid update process unlike the original PAM algorithm that performs global search. Recently, two MapReduce algorithms, GREEDI [22] and MR-KMEDIAN [11], have been proposed independently from existing non-parallel algorithms.

Even the two recent algorithms—GREEDI [22] and MR-KMEDIAN [11]—are *not* free from the accuracy and efficiency problems. GREEDI performs global search over each partition which can be regarded as a sample. The accuracy of GREEDI is relatively high compared with the earlier algorithms in a small data set, but it has efficiency issues, as we demonstrate in the experiments. MR-KMEDIAN constructs a single best sample by parallel, iterative sampling. The accuracy of MR-KMEDIAN is acceptable if the number (k) of clusters is high. However, its efficiency degrades as k increases because of a larger sample size.

Overall, as far as we know, *none* of the existing algorithms satisfy both high accuracy and high efficiency. To offer high accuracy, *global search* against *entire data* needs to be secured, as in PAM. Thus, apart from efficiency, PAM has been widely regarded as the most accurate k-medoids algorithm [23]. However, the existing algorithms have opted to improve efficiency at the expense of accuracy by either performing local search instead of global search or using sampled data instead of entire data.

1.2 Contributions

In this paper, we propose a parallel k-medoids algorithm, which we call *PAMAE (PArallel k-Medoids clustering with high Accuracy and Efficiency)*, in order to guarantee *not only high efficiency but also high accuracy*. As stated above, "global search" and "entire data" are the two main ingredients for high accuracy, but using them *simultaneously* harms efficiency. Thus, as in Figure 2, our key idea is to apply these two components *individually* through two phases: *parallel seeding* and *parallel refinement*. In the first phase, the algorithm generates multiple random samples and runs a k-medoids algorithm (e.g., PAM), that performs *global* search, on each *sample* in parallel. Then, it selects the best among the sets of seeds, each of which is obtained from a sample, and feeds it to the next phase. The second phase aims at reducing the possible errors induced by sampling. The algorithm partitions the *entire* data based on the seeds and constructs initial clusters. Then, it updates the medoids in parallel *locally* at each of the initial clusters.

Overall, the contributions of this paper are summarized as follows:

1. We propose an elegant and simple approach to k-medoids clustering that performs parallel clustering in two phases and implement the PAMAE algorithm, achieving both high accuracy and high efficiency. Our novelty lies in the smart integration of sampling and refinement with formal justifications. We believe that the *simplicity* of our algorithm is a strong benefit because simple algorithms often make a big impact and gain widespread acceptance [15].
2. We theoretically prove the soundness of PAMAE from the perspective of accuracy. Our analysis guarantees that the serial execution of the two phases leads to an accurate solution that would be achieved by global search over entire data.
3. We extensively evaluate the accuracy and efficiency of PAMAE using four real-world data sets on 12 Microsoft Azure D12v2 machines (48 cores). PAMAE achieves the lowest error among the *parallel* k-medoids algorithms compared. Furthermore, compared with GREEDI and MR-KMEDIAN, PAMAE improves execution time by orders of magnitude on Hadoop.

2 RELATED WORK

2.1 Non-Parallel k-Medoids Algorithms

PAM was proposed by Kaufman and Rousseeuw [18] and has been regarded as the representative k-medoids algorithm. PAM finds the most centrally-located objects, called *medoids*. It starts from an arbitrary selection of initial medoids and iteratively updates medoids by randomly swapping a medoid object with a non-medoid object. If this swapping reduces clustering error, the current medoid object is replaced with the new one. This procedure repeats until there is no change in the set of medoids. The time complexity is $O(k(N-k)^2)$ [23], where N is the number of objects.

Many variations of PAM have been proposed mostly to improve efficiency. CLARA [19] applies PAM on *five* random samples of $40+2k$ objects to find medoids. Since the results are heavily affected by sampling, CLARA usually repeats this procedure multiple times and selects the best set as the final result. CLARANS [23] considers a graph in which every node is a potential solution and dynamically draws on a sample of neighbors to confine the search space in the graph. MEM [26] stores the distances for all possible pairs of data objects in advance and performs local search to update medoids. CVI [25] uses a density-based selection of initial medoids for reducing the number of iterations. Similar to CVI, IFART [24] uses Fuzzy Art, which is an unsupervised learning technique, to select initial medoids. FAMES [28] quickly finds suitable medoids using geometric computation rather than random selection.

2.2 Parallel k-Medoids Algorithms

PAM-MR [32] is an extension of PAM for MapReduce. Similar to the extension of the k-means algorithm, it assigns each object to the closest medoid in the map phase and updates the current medoid to the most central object using pair-wise computation in the reduce phase. FAMES-MR [33], an extension of FAMES for MapReduce, is the same as PAM-MR except the reduce phase. FAMES-MR uses a geometric technique for each cluster just like FAMES to find the most central object. CLARA-MR [16], an extension of CLARA for MapReduce, executes two MapReduce jobs that run PAM on samples and select the best set in parallel. The authors chose *five* random samples of $100+5k$ objects according to their experiments.

Recently, GREEDI [22] and MR-KMEDIAN [11] were proposed independently from the existing algorithms. GREEDI runs a greedy algorithm to find a candidate set of k medoids from each partition of entire data, where a partition can be considered as a sample; it

Table 1: Summary of the notation.

Notation	Description
$\mathcal{D}$	the entire set of objects
$\mathcal{S}$	a sample (or subset) of $\mathcal{D}$
o, o_i	an object in $\mathcal{D}$
$\mathrm{dist}(o_i, o_j)$	a distance between o_i and o_j
C	a set of clusters
$\mathbb{C}$, $\mathbb{C}_i$	a cluster of objects
Θ, $\hat{\Theta}$	a set of medoids selected from $\mathcal{D}$ and $\mathcal{S}$
$\phi(\Theta)$, $\phi(\hat{\Theta})$	the clustering error by Θ and $\hat{\Theta}$
N	the total number of objects
k	the number of clusters (medoids)
n	the number of objects per sample
m	the number of samples

runs the greedy algorithm over the set of ($k \times$ number of partitions) medoids to find the final set of k medoids. GREEDI does not completely resolve the efficiency issue, because the number and size of samples are dependent on the size of entire data owing to the requirement of partitioning. MR-KMEDIAN constructs a single best sample by parallel, iterative sampling and runs a weighted k-medoids algorithm over the sample. The iterative sampling is expensive especially in a large data set since it repeatedly accesses entire data to calculate the distances between sampled and unsampled objects. The efficiency of MR-KMEDIAN also degrades as k increases since the number of calculating distances gets higher.

2.3 Limitations of Existing Algorithms

As in Figure 1, none of the existing *parallel* k-medoids algorithms performs global search against entire data. Thus, all the existing algorithms are either only accurate just like PAM or only fast just like CLARA-MR, failing to obtain accurate results within reasonable time for big data. To the best of our knowledge, PAMAE is the only k-medoids algorithm that supports both accuracy and efficiency.

3 OVERVIEW OF PAMAE

The PAMAE algorithm finds a *clustering* (Definition 3.1) that achieves a low *clustering error* (Definition 3.2). Table 1 summarizes the notation used throughout this paper. Especially, C indicates a set of clusters, Θ indicates a set of medoids, and $\phi(\Theta)$ denotes the clustering error of the set of clusters represented by Θ. $\phi_{\mathbb{C}}(\cdot)$ means the clustering error of a specific cluster $\mathbb{C}$. The *optimal* clustering, typically denoted by C_{opt}, minimizes Eq. (1). The clustering result of PAMAE is not necessarily optimal since finding the optimal k-medoids is NP-hard [21].

Definition 3.1. Clustering is to find a set of k clusters, $C = \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$, for a data set $\mathcal{D}$, where $\forall i \neq j, \mathbb{C}_i \cap \mathbb{C}_j = \emptyset$ and $\bigcup_{i=1}^{k} \mathbb{C}_i = \mathcal{D}$. □

Definition 3.2. [12, 16, 19, 23, 26] The *clustering error* is defined to be the sum of the *absolute* error (distance) as in Eq. (1). Here, θ_i denotes the medoid of $\mathbb{C}_i$. □

$$\phi(\Theta) = \sum_{i=1}^{k} \phi_{\mathbb{C}_i}(\theta_i), \text{ where } \phi_{\mathbb{C}_i}(\theta_i) = \sum_{o \in \mathbb{C}_i} \mathrm{dist}(o, \theta_i) \quad (1)$$

Figure 3 as well as Algorithm 1 describe the overall procedure.

Phase I: PAMAE starts by simultaneously performing random sampling with replacement (Step 1, Lines 3–4). That is, it creates m random samples whose size is n. Then, a k-medoids algorithm $\mathcal{A}$ is performed in parallel against each sample (Step 2, Line 5). We note that any existing algorithm of finding medoids globally can be a candidate for $\mathcal{A}$. Among the m sets of medoids, each of which was obtained from each sample, PAMAE searches for the set of medoids that minimizes the clustering error in Eq. (1) (Step 3, Line 6). This set of medoids, denoted by $\{\hat{\theta}_1, \ldots, \hat{\theta}_k\}$, is regarded as the best set of seeds and is fed to Phase II.

Phase II: Using the seeds derived in Phase I, PAMAE simultaneously partitions the entire data set just like the assignment step of the k-means algorithm (Step 4, Line 10). That is, it assigns each of non-medoid objects to the closest medoid. Then, the algorithm updates the medoid of each partition (i.e., cluster) by choosing the most central object (Step 5, Line 11), similar to the update step of the k-means algorithm. Last, the entire set of data objects is partitioned into clusters if needed (Step 6, Line 12).

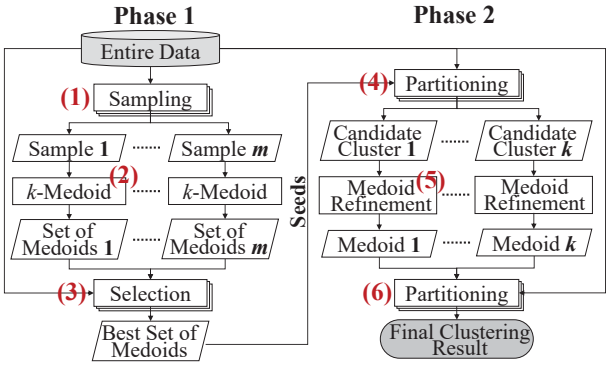


Figure 3: Overall procedure of the PAMAE algorithm.

Algorithm 1 PAMAE (Overall Procedure)

INPUT: A set $\mathcal{D}$ of N data objects, a number k of clusters
OUTPUT: A set Θ of k medoids (and a set C of k clusters)

1: /* PHASE I: PARALLEL SEEDING */
2: /* Steps 1–3 of Figure 3 ⇒ Section 4 */
3: $n \leftarrow 40k$, $m \leftarrow 5$; /* Section 4.2 */
4: $\{\mathbb{S}_1, \ldots, \mathbb{S}_m\} \leftarrow$ Parallel_Sampling($\mathcal{D}$, n, m);
5: $\{\hat{\Theta}_1, \ldots, \hat{\Theta}_m\} \leftarrow$ Parallel_KMedoid($\{\mathbb{S}_1, \ldots, \mathbb{S}_m\}$, k);
6: $\hat{\Theta} \leftarrow$ Parallel_Selection($\mathcal{D}$, $\{\hat{\Theta}_1, \ldots, \hat{\Theta}_m\}$);
7: /* The result of Phase I: $\hat{\Theta} = \{\hat{\theta}_1, \ldots, \hat{\theta}_k\}$ */
8: /* PHASE II: PARALLEL REFINEMENT */
9: /* Steps 4–6 of Figure 3 ⇒ Section 5 */
10: $\{\mathbb{C}'_1, \ldots, \mathbb{C}'_k\} \leftarrow$ Parallel_Partitioning($\mathcal{D}$, $\{\hat{\theta}_1, \ldots, \hat{\theta}_k\}$);
11: $\{\theta_1, \ldots, \theta_k\} \leftarrow$ Parallel_Refinement($\{\mathbb{C}'_1, \ldots, \mathbb{C}'_k\}$);
12: $\{\mathbb{C}_1, \ldots, \mathbb{C}_k\} \leftarrow$ Parallel_Partitioning($\mathcal{D}$, $\{\theta_1, \ldots, \theta_k\}$);
13: $\Theta \leftarrow \{\theta_1, \ldots, \theta_k\}$, $C \leftarrow \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$;
14: **return** Θ (and C) /* medoids (and clusters) */

4 PHASE I: PARALLEL SEEDING

Phase I is intended to quickly retrieve a high-quality candidate set of medoids by sampling. The main technical challenge in Phase I is to *formally* analyze the impact of the parameters on clustering accuracy and to determine their values based on the analysis to achieve a reasonable trade-off between accuracy and efficiency. Additionally, the novelty lies in performing another cheap yet effective phase, using just the result of this phase *as the seeds* of Phase II.

Algorithm 2 Parallel Seeding (Phase I) in MapReduce

```
1: /* Steps 1–2 */
2: class PhaseI_Sampling_Mapper
3:     method Map(NULL, object o)
4:         oid ← Pick a random key from [1, N];
5:         emit(oid, o);
6: class PhaseI_Sampling_Reducer
7:     method Reduce(oid, object o)
8:         S_sid ← Read the first n objects;
9:         /* Θ̂_sid = {θ̂_1, ..., θ̂_k} */
10:        Θ̂_sid ← Run A on S_sid to find k medoids;
11:        emit(sid, Θ̂_sid);
12: /* Step 3 */
13: class PhaseI_Selection_Mapper
14:     method Map(NULL, object o)
15:         for each sid in [1, m] do
16:             dist ← Calculate the minimum distance
                       from o to θ̂_i ∈ Θ̂_sid;
17:             emit(sid, dist);
18: class PhaseI_Selection_Reducer
19:     method Reduce(sid, [dist_1, dist_2, ...])
20:         dist_sum ← Σ_i dist_i;
21:         emit(sid, dist_sum);
```

4.1 Algorithm Description

Algorithm 2 represents the detailed procedure of Phase I, which is self-explanatory. For ease of exposition, we describe the procedure in the form of MapReduce. It is straightforward to implement this pseudo code on Spark too. The *sampling* mapper and reducer (Lines 2–11) correspond to Steps 1–2, and the *selection* mapper and reducer (Lines 13–21) correspond to part of Step 3. For sampling, we use a MapReduce version of random sort [30] because of its simplicity of implementation. The sampling mapper assigns a random key to each object and sends these tuples to the sampling reducer with the partitioner function ($key \mod m$). The sampling reducer receives the tuples in the order of keys and reads the first n objects to construct a sample. An algorithm $\mathcal{A}$ is executed on each sample within this reducer (Line 10). The selection mapper and reducer return a list of the clustering errors of each set of medoids. After this reducer is complete, we pick up the set of medoids that has the smallest error and feed it to Phase II.

The time complexity of Phase I is given by Lemma 4.1.

Lemma 4.1. *The time complexity of Step 1 is $O(N \log N)$, that of Step 2 is $O(n^2)$, and that of Step 3 is $O(N)$.*

Proof. Step 1 involves sorting (Line 8). If $\mathcal{A}$ has quadratic complexity, Step 2 needs to check $O(n^2)$ pairs for each sample; m is not included because $\mathcal{A}$ runs for m samples *in parallel*. Step 3 requires reading all objects to calculate the clustering error. □

4.2 Theoretical Analysis

We formally analyze the effect of sampling on the clustering error. Let's consider a subset $\mathcal{S}$ of the entire set $\mathcal{D}$ of objects, i.e., $\mathcal{S} \subseteq \mathcal{D}$. Let $\Theta_{opt} = \{\theta_1, \ldots, \theta_k\}$ be the set of k medoids that minimizes the cluster error, Eq. (1), *when* $\theta_i \in \mathcal{D}$. Let $\hat{\Theta}_{opt} = \{\hat{\theta_1}, \ldots, \hat{\theta_k}\}$ be the set of k medoids that minimizes Eq. (1) *when* $\hat{\theta}_i \in \mathcal{S}$. Practically speaking, Θ_{opt} and $\hat{\Theta}_{opt}$ are the *best* k medoids that can be obtained by running an algorithm $\mathcal{A}$ on $\mathcal{D}$ and $\mathcal{S}$ respectively. Our goal is to present the upper bound of the clustering error by sampling, $\phi(\hat{\Theta}_{opt})$, in terms of the optimal clustering error, $\phi(\Theta_{opt})$. We regard that an object is represented on a complete normed vector space, where the triangle inequality holds by definition.

First, Lemma 4.2 presents the difference between $\phi(\hat{\Theta}_{opt})$ and $\phi(\Theta_{opt})$ when there is only one cluster ($k = 1$), and Lemma 4.3 generalizes Lemma 4.2 for multiple clusters ($k > 1$).

Lemma 4.2. *Let $\mathbb{C}$ be the optimal cluster with the medoid $\Theta_{opt} = \{\theta\}$. Suppose that $\hat{\Theta}_{opt} = \{\hat{\theta}\}$ is selected from $\mathcal{S} \subseteq \mathcal{D}$. Then, $\phi(\{\hat{\theta}\}) \leq \phi(\{\theta\}) + |\mathbb{C}|\mathrm{dist}(\theta, \hat{\theta})$.*

Proof. Eq. (2) holds by the triangle inequality. □

$$\begin{aligned}\phi(\{\hat{\theta}\}) &= \sum_{o \in \mathbb{C}} \mathrm{dist}(o, \hat{\theta}) \ \because \text{Eq. (1)} \\ &\leq \sum_{o \in \mathbb{C}} \Big(\mathrm{dist}(o, \theta) + \mathrm{dist}(\theta, \hat{\theta})\Big) = \phi(\{\theta\}) + |\mathbb{C}|\mathrm{dist}(\theta, \hat{\theta})\end{aligned} \tag{2}$$

Lemma 4.3. *Let $C_{opt} = \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$ be the set of k optimal clusters with the set of k medoids $\Theta_{opt} = \{\theta_1, \ldots, \theta_k\}$. Suppose that $\hat{\Theta}_{opt} = \{\hat{\theta_1}, \ldots, \hat{\theta_k}\}$ is selected from $\mathcal{S} \subseteq \mathcal{D}$. Then, $\phi(\hat{\Theta}_{opt}) \leq \phi(\Theta_{opt}) + \sum_{i=1}^{k} |\mathbb{C}_i|\mathrm{dist}(\theta_i, \hat{\theta}_i)$.*

Proof. Lemma 4.2 holds individually for each cluster $\mathbb{C}_i$, because $\forall i \neq j, \mathbb{C}_i \cap \mathbb{C}_j = \emptyset$ and $\bigcup_{i=1}^{k} \mathbb{C}_i = \mathcal{D}$. The detail is omitted because of lack of space. □

By Lemmas 4.2 and 4.3, $\mathrm{dist}(\theta_i, \hat{\theta}_i)$ determines the difference between $\phi(\hat{\Theta}_{opt})$ and $\phi(\Theta_{opt})$. To calculate its expectation, we need to know the distribution of the objects in a cluster. Following the convention of cluster analysis, we take account of various underlying distributions for clusters, ranging from the *uniform distribution* [7, 17] to a distribution with the *central tendency* [5]. The uniform distribution provides us with the conservative error bound. Then, through the derivation in Appendix A, it is proven that $E[\mathrm{dist}(\theta_i, \hat{\theta}_i)] < \frac{R}{nm}$, where $R = \max_{o \in \mathbb{C}} \mathrm{dist}(o, \theta)$.

Finally, Theorem 4.4 presents the upper bound of $\phi(\hat{\Theta}_{opt})$ in terms of $\phi(\Theta_{opt})$.

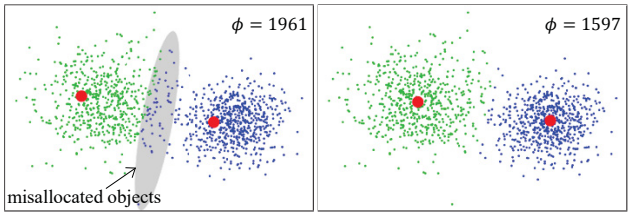
Theorem 4.4. *Let $C_{opt} = \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$ be the set of k optimal clusters with the set of k medoids $\Theta_{opt} = \{\theta_1, \ldots, \theta_k\}$. Suppose that $\mathcal{S}$ consists of m samples whose size is n. Then, $E[\phi(\hat{\Theta}_{opt})] < \phi(\Theta_{opt}) + \sum_{i=1}^{k} \frac{2}{n_i m} \phi_{\mathbb{C}_i}(\theta_i)$, where n_i is the number of objects sampled from $\mathbb{C}_i$.*

Proof. By substituting $\frac{R}{n_i m}$ for $E[\mathrm{dist}(\theta_i, \hat{\theta}_i)]$ in Lemma 4.3, we obtain Eq. (3). By the uniform distribution in Definition A.1 (Appendix A), $E[\mathrm{dist}(o, \theta_i)] = \frac{R}{2}$ for $o \in \mathbb{C}_i$. Then, since $|\mathbb{C}_i| \cdot \frac{R}{2} = \phi_{\mathbb{C}_i}(\{\theta_i\})$, the last equation is derived. □

$$\begin{aligned}E[\phi(\hat{\Theta}_{opt})] &\leq \phi(\Theta_{opt}) + \sum_{i=1}^{k} |\mathbb{C}_i| E[\mathrm{dist}(\theta_i, \hat{\theta}_i)] \\ &< \phi(\Theta_{opt}) + \sum_{i=1}^{k} |\mathbb{C}_i| \frac{R}{n_i m} = \phi(\Theta_{opt}) + \sum_{i=1}^{k} \frac{2}{n_i m} \phi_{\mathbb{C}_i}(\{\theta_i\})\end{aligned} \tag{3}$$

To make Theorem 4.4 more useful in practice, it can be simplified to Eq. (4), provided that $\mathbb{C}_i$'s ($1 \leq i \leq k$) contain the same number of sampled objects, i.e., $n_i = n/k$. This simplification is realistic more often than not, because it is known that the sweet spot for k-medoids and k-means clustering is when the clusters have uniform sizes [2, 12]. We opt to use this simplified form in this paper.

$$E[\phi(\hat{\Theta}_{opt})] < (1 + \frac{2k}{nm})\phi(\Theta_{opt}) \tag{4}$$



(a) Before Phase II. **(b) After Phase II.**

Figure 4: Clustering before and after Phase II (in color).

Fixing m to be 5, which is a typical setting in other algorithms [16], the decrease of the expected error in Eq. (4) becomes negligible when n is greater than $40k$. To maintain samples as small as possible, we set the parameter values to be $n = 40k$ and $m = 5$. The expected error is only $1.01\phi(\Theta_{opt})$. We empirically confirm that the expected error is very close to the real error in Section 6.2.

5 PHASE II: PARALLEL REFINEMENT

Phase II is intended to further reduce the possible errors induced by sampling. Although the clustering error induced by sampling is very low according to Theorem 4.4, we still need this additional phase of refinement for the following reasons. First, the real-world data may not conform to the uniform distribution or the central tendency which we have used for theoretical derivation. Second, because Theorem 4.4 describes *asymptotic* properties, this tight error bound may not hold in a *specific* execution. Therefore, we propose a cheap and effective phase of refinement, which is similar to one iteration of the k-means algorithm.

Example 5.1. Figure 4 illustrates the effect of this refinement. In Figure 4(a), every cluster was assigned a seed, but the seeds deviated a little from the center of each cluster in the entire data because of the limitation of sampling. In Figure 4(b), the final medoids moved toward the center of each cluster, and thus the objects in the shaded region that belonged to the right cluster now belong to the left cluster. As a result, the clustering error was significantly reduced from 1961 to 1597. □

One might argue that Phase II is not necessary if samples are large enough. However, as we demonstrate in Section 6, the refinement phase of PAMAE enables us to achieve higher accuracy than GREEDI [22] which uses much larger samples (typically, $n = 10,000$ and $m = \lceil N/10,000 \rceil$) than our algorithm. Even worse, using such large samples (i.e., partitions) in GREEDI is shown to drastically degrade the scalability of the algorithm.

5.1 Algorithm Description

Algorithm 3 represents the detailed procedure of Phase II, which is also self-explanatory. The mapper corresponds to Step 4 (or 6), and the reducer corresponds to Step 5. The mapper performs initial clustering based on the seeds. Then, the reducer updates the medoid of each cluster according to Definition 5.2. Intuitively, the updated medoid is the object located at the *very center of a new cluster*, e.g., the red objects in Figure 4(b).

Definition 5.2. For a cluster $\mathbb{C} = \{o_1, \ldots, o_l\}$, suppose that θ^* is the *spatial median* [13] of $\mathbb{C}$, as in Eq. (5). Then, the *medoid* of $\mathbb{C}$ is determined to be the actual object θ closest to θ^*, as in Eq. (6). □

$$\theta^* = \operatorname*{argmin}_{\boldsymbol{x} \in \mathcal{R}^d} \sum_{o \in \mathbb{C}} \mathrm{dist}(o, \boldsymbol{x}) \tag{5}$$

$$\theta = \operatorname*{argmin}_{o \in \mathbb{C}} \mathrm{dist}(o, \theta^*) \tag{6}$$

Algorithm 3 Parallel Refinement (Phase II) in MapReduce

```
1: class PhaseII_Mapper /* Steps 4 or 6 */
2:     method Map(NULL, object o)
3:         cid ← Find the id of the nearest medoid θ̂_cid from o;
4:         emit(cid, o);
5: class PhaseII_Reducer /* Step 5 */
6:     method Reduce(cid, [o_1, o_2, ...])
7:         θ_cid ← Calculate the new medoid of [o_1, o_2, ...]
                   by Definition 5.2;
8:         emit(cid, θ_cid);
```

In Definition 5.2, we derive the medoid from the spatial median instead of directly finding the medoid. The advantage of this approach is that we can exploit several *approximate* algorithms of quickly finding the spatial median. In the Weiszfeld algorithm [6, 31], which we adopt in this paper, the objective function in Eq. (5) is optimized using the gradient descent method. If the number of iterations is regarded as constant, the time complexity is $O(l)$, where l is the number of objects in a cluster [31]. Our experience indicates that a convergence is reached with only a few iterations when the seeds of Phase I are used as an initial solution, because that initial solution is already close to the final solution.

The time complexity of Phase II is given by Lemma 5.3.

Lemma 5.3. *The time complexity of Step 4 (or 6) is $O(N)$, and that of Step 5 is $O(l)$, where l is the number of objects per cluster.*

Proof. Step 4 (or 6) requires reading all objects. Step 5 performs gradient descent, where the time complexity is $O(l)$ [31]. □

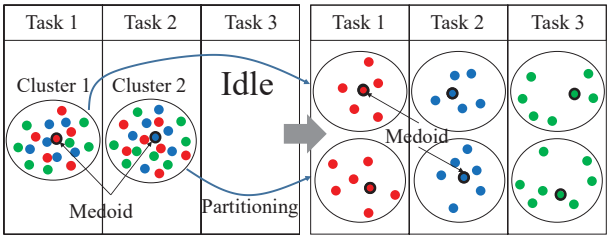
5.2 Theoretical Analysis

The most important issue in Phase II is to guarantee that *every* cluster in C_{opt} contains *only one* medoid in $\hat{\Theta}_{opt}$. Otherwise, Phase II would be trapped in a local optimum. If at least τ (typically, 30) objects are sampled from a cluster $\mathbb{C}_i$, the corresponding sample medoid $\hat{\theta}_i$ should be covered by $\mathbb{C}_i$, because $\hat{\theta}_i$ converges to θ_i by Lemma A.2 in Appendix A. Thus, we make sure that at least τ objects among $|\mathcal{S}| = n \cdot m$ objects are covered by each cluster. Theorem 5.4 presents the probability of such desired coverage.

Theorem 5.4. *Let $C = \{\mathbb{C}_1, \ldots, \mathbb{C}_k\}$ be a set of k clusters. When all $\mathbb{C}_i$'s have the same number of objects, the probability that every cluster $\mathbb{C}_i$ contains at least τ objects of the sample $\mathcal{S}$ of size $n \cdot m$ is formulated as Eq.* (7).

$$\Pr(|\{\mathbb{C}_i \mid |\mathcal{S} \cap \mathbb{C}_i| \geq \tau\}| = k) \geq 1 - \underbrace{\sum_{j=1}^{k-1} \binom{k}{j} \sum_{z=0}^{(\tau-1)(k-j)} \frac{j^{nm-z}(k-j)^z}{k^{nm}}}_{\boldsymbol{B}_j} \tag{7}$$

Proof. Let $\boldsymbol{A}_j$ be the event that only j clusters have at least τ objects. The probability of Eq. (7) is obtained by subtracting $\sum_{j=1}^{k-1} \Pr(\boldsymbol{A}_j)$ from 1. For a specific value of j, the $(k-j)$ "unqualified" clusters can have at most $z = (k-j)(\tau-1)$ objects in total, whereas the j "qualified" clusters have the remaining $(nm - z)$ objects. The denominator of $\boldsymbol{B}_j$ in Eq. (7) is the number of cases that nm objects are allocated to k clusters. In the numerator, j^{nm-z} is the number of cases that $(nm-z)$ objects are allocated to the j qualified clusters, and $(k-j)^z$ is the number of cases that z objects are allocated to the $(k-j)$ unqualified clusters. However, not every possible allocation



(a) Naïve implementation. (b) **Fine-granularity refinement**.

Figure 5: Concept of fine-granularity refinement (in color).

is valid; for example, a case that one of the $(k-j)$ clusters has more than τ objects is invalid. Thus, $\mathbf{Pr}(A_j)$ is no greater than B_j in Eq. (7). This concludes the proof. □

If we assign $n = 40k$ and $m = 5$, as determined in Phase I, as well as $\tau = 30$ [29] to Eq. (7) of Theorem 5.4, the probability becomes almost 1. As a result, *only one* iteration of refinement is enough to improve the quality of clustering. Thus, this phase is not costly (12–40% of the total elapsed time) though its effect is significant.

Overall, putting Theorems 4.4 and 5.4 together, we have theoretically proven that PAMAE achieves an accurate solution that would be achieved by running an algorithm $\mathcal{A}$ with global search over the entire data set $\mathcal{D}$.

5.3 Fine-Granularity Refinement

High performance in parallel processing should be accompanied by high *data parallelism* [14]. According to Figure 3, the numbers of parallel tasks in Steps 2 and 5 are m and k respectively, which are usually much smaller than that of available processors in recent computing environments. Consequently, we may not fully exploit these abundant resources. Step 2 is less problematic than Step 5 since a sample is usually much smaller than a cluster. Therefore, we devise a technique that can improve the data parallelism of Step 5, which we call *fine-granularity refinement*.

As illustrated in Figure 5, fine-granularity refinement first divides a cluster into multiple smaller partitions and then performs local refinement *on each partition*. In Figure 5(a), one task remains idle if there are three processors and $k = 2$. On the other hand, in Figure 5(b), each cluster is partitioned and assigned to each processor as indicated by color, where the number of partitions is equal to that of the available processors. Finally, the best among those obtained by multiple tasks is selected just like Step 3.

The medoid of a cluster can be found from the medoids of its partitions if random partitioning is performed on a sufficiently large cluster, which is proven by Theorem 5.5.

Theorem 5.5. *Let $\mathbb{C} = \{o_1, \ldots, o_l\}$ be a cluster with the medoid θ. Suppose that $\mathbb{C}$ is partitioned into $\{P_1, \ldots, P_c\}$. If $l \to \infty$, $\exists j \in [1, c]$, $\theta = \hat{\theta}_{P_j}$, where $\hat{\theta}_{P_j}$ is the medoid of a partition P_j.*

Proof. By Lemmas A.2 and A.3 in Appendix A, for a partition P_j $(1 \le j \le c)$, $\hat{\theta}_{P_j} = \text{argmin}_{o \in P_j} \text{dist}(o, \theta)$; in addition, Eq. (8) holds, where $R = \max_{o \in \mathbb{C}} \text{dist}(o, \theta)$.

$$E[\text{dist}(\theta, \hat{\theta}_{P_j})] \le \frac{R}{|P_j| + 1} \tag{8}$$

In a sufficiently large cluster, because $\lim_{|P_j| \to \infty} \frac{R}{|P_j|+1} = 0$ as $l \to \infty$, the expected distance between the two medoids becomes 0. Thus, there exists a partition P_j whose medoid is the same as θ, and this concludes the proof. □

Table 2: Real-world data sets used for experiments.

Data Set	# Object	# Dim	Size	Type	Central
Covertype [1][1]	581,102	55	71 MB	int	O
Census1990 [1][1]	2,458,285	68	324 MB	int	O
Cosmo50 [20][2]	315,086,245	3	13.6 GB	float	O
TeraClickLog[3]	4,373,472,329	13	300 GB	float	O

The time complexity of Step 5 is now reduced from $O(l)$ to $O(l')$, where l' is the number of objects *per partition*. We note that $l' \ll l$ since $l' = \lceil l/ \text{(number of cores)} \rceil$. The technique has been incorporated into the PAMAE algorithm.

6 EVALUATION

Our evaluation was systematically conducted to support the followings:

- PAMAE is **more accurate** than other algorithms (Section 6.2.1).
- Theorems 4.4 and 5.4 are **valid** (Sections 6.2.2 and 6.2.3).
- PAMAE is **more efficient** than other algorithms (Section 6.3.1).
- PAMAE is **scalable** to the data size (Section 6.3.2).
- The configuration of the sampling parameters is **correct** (Section 6.4).

6.1 Experiment Setting

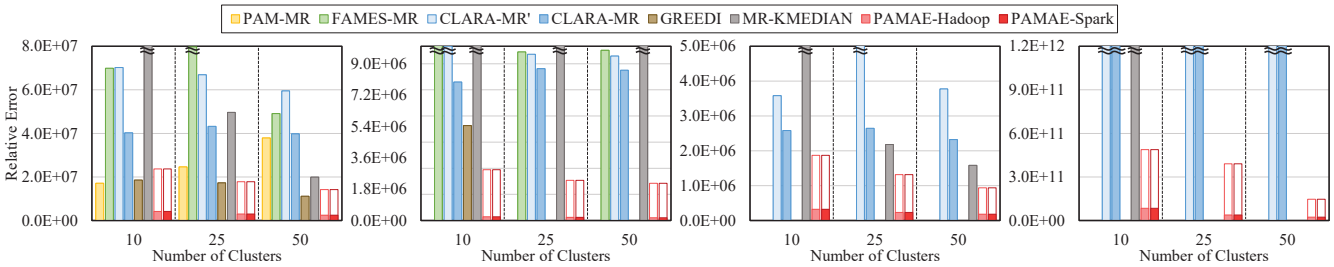
6.1.1 Algorithms. We compared our PAMAE algorithm with PAM and *six* parallel k-medoids algorithms as below. The six existing parallel algorithms (2–7) are implemented on Hadoop only. PAMAE-Hadoop and PAMAE-Spark are the two different implementations of our algorithm on Hadoop and Spark respectively. We use PAM for the algorithm $\mathcal{A}$ in Phase I, because it has been widely regarded as one of the most accurate algorithm among *practical* k-medoids algorithms [19, 23]. Our evaluation is *fully reproducible* since the source code and data are available at https://github.com/jaegil/k-Medoid.

1. PAM [18]: baseline
2. PAM-MR [32]
3. FAMES-MR [33]
4. CLARA-MR′ [16]: $n = 40 + 2k$ and $m = 5$
5. CLARA-MR [16]: $n = 100 + 5k$ and $m = 5$
6. GREEDI [22]
7. MR-KMEDIAN [11]
8. **PAMAE-Hadoop**: implementation on Hadoop
9. **PAMAE-Spark**: implementation on Spark

6.1.2 Data Sets. The real-world data sets used for experiments are summarized in Table 2. We note that TeraClickLog is large enough not to fit in main memory; *TeraClickLog150* means its half. All these data sets are numerical, and the Euclidean distance was used for them. Importantly, all the data sets tend to have *central tendency* because they are positively skewed by Pearson's moment coefficient of skewness [10], satisfying the assumption for the theoretical analysis. Please see Appendix B for the centrality test.

6.1.3 Evaluation Metrics. To measure clustering accuracy, in addition to the *absolute error* ϕ in Eq. (1), we used the *relative error* $\phi^{\mathcal{RE}\mathcal{L}}$ in Eq. (9) by using PAM as the baseline. If a data set was too large to run PAM, ϕ^{PAM} was *conservatively* estimated from the clustering error of PAMAE according to Eq. (4). Using the relative

[1] http://archive.ics.uci.edu/ml/datasets.html
[2] http://nuage.cs.washington.edu/benchmark/astro-nbody/
[3] http://labs.criteo.com/downloads/download-terabyte-click-logs/



(a) Covertype. (b) Census1990. (c) Cosmo50. (d) TeraClickLog150.

Figure 6: Accuracy comparison of eight parallel k-medoids algorithms.

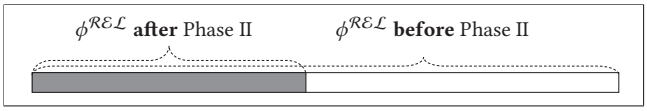
Table 3: *Absolute* error of the parallel k-medoids algorithms in Figure 6.

Data Set	(a) Covertype			(b) Census1990			(c) Cosmo50			(d) TeraClickLog150		
k	10	25	50	10	25	50	10	25	50	10	25	50
PAM-MR	$4.31 \cdot 10^8$	$3.27 \cdot 10^8$	$2.85 \cdot 10^8$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
FAMES-MR	$4.84 \cdot 10^8$	$3.82 \cdot 10^8$	$2.96 \cdot 10^8$	$3.46 \cdot 10^7$	$2.83 \cdot 10^7$	$2.58 \cdot 10^7$	N/A	N/A	N/A	N/A	N/A	N/A
CLARA-MR′	$4.84 \cdot 10^8$	$3.69 \cdot 10^8$	$3.07 \cdot 10^8$	$3.23 \cdot 10^7$	$2.82 \cdot 10^7$	$2.55 \cdot 10^7$	$3.62 \cdot 10^7$	$2.87 \cdot 10^7$	$2.21 \cdot 10^7$	$1.30 \cdot 10^{13}$	$1.20 \cdot 10^{13}$	$7.18 \cdot 10^{12}$
CLARA-MR	$4.54 \cdot 10^8$	$3.45 \cdot 10^8$	$2.87 \cdot 10^8$	$3.02 \cdot 10^7$	$2.73 \cdot 10^7$	$2.47 \cdot 10^7$	$3.52 \cdot 10^7$	$2.61 \cdot 10^7$	$2.06 \cdot 10^7$	$9.76 \cdot 10^{12}$	$8.29 \cdot 10^{12}$	$6.91 \cdot 10^{12}$
GREEDI	$4.32 \cdot 10^8$	$3.19 \cdot 10^8$	$2.58 \cdot 10^8$	$2.77 \cdot 10^7$	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
MR-KMEDIAN	$7.02 \cdot 10^8$	$3.52 \cdot 10^8$	$2.67 \cdot 10^8$	$4.38 \cdot 10^7$	$3.02 \cdot 10^7$	$2.76 \cdot 10^7$	$6.61 \cdot 10^7$	$2.57 \cdot 10^7$	$1.99 \cdot 10^7$	$1.31 \cdot 10^{14}$	N/A	N/A
PAMAE	$\mathbf{4.18 \cdot 10^8}$	$\mathbf{3.05 \cdot 10^8}$	$\mathbf{2.50 \cdot 10^8}$	$\mathbf{2.25 \cdot 10^7}$	$\mathbf{1.88 \cdot 10^7}$	$\mathbf{1.62 \cdot 10^7}$	$\mathbf{3.29 \cdot 10^7}$	$\mathbf{2.37 \cdot 10^7}$	$\mathbf{1.85 \cdot 10^7}$	$\mathbf{8.47 \cdot 10^{12}}$	$\mathbf{3.93 \cdot 10^{12}}$	$\mathbf{2.36 \cdot 10^{12}}$

error makes sense because the *optimal* clustering error that cannot be further reduced takes a significant proportion of ϕ in large-scale data sets. Thus, even *meaningful* differences between algorithms would *not* come out if we compared only the absolute errors.

$$\textit{Relative Error} = \phi^{\mathcal{REL}} = \phi - \phi^{PAM} \tag{9}$$

We show the relative error of PAMAE *separately before and after Phase II* to look into the effect of each phase. In Figure 6, as shown below, the entire region in a bar represents the relative error before Phase II, and the solid region represents the *final* relative error after Phase II.

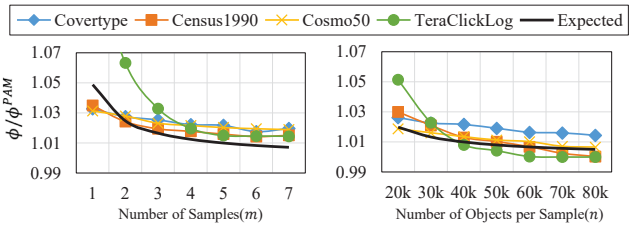


In order to get reliable results, we repeated every test by *five times* and reported the average, except in Section 6.3.2 using TeraClickLog because of its large size. Besides, we stopped executing an algorithm if it did not terminate within *two hours*.

6.1.4 Configuration. We conducted experiments on 12 Microsoft Azure D12v2 instances located in Japan. Each instance has four cores, 28 GB of main memory, and 200 GB of disk (SSD). All instances run on Ubuntu 14.04. We used Hadoop 2.7.1 and Spark 1.6.1 for distributed parallel processing. Ten out of 12 instances were used as worker nodes, and the remaining two instances were used as master nodes. In both systems, all the algorithms were written in the Java programming language and run on JDK 1.7.0_101.

6.2 Accuracy Results

6.2.1 Overall Comparison. Figure 6 shows the relative error of the eight parallel algorithms for the four data sets when k is set to be 10, 25, and 50. Table 3 shows the absolute error for Figure 6. Overall, PAMAE achieved the lowest relative error (the solid region in its bar) and the lowest absolute error. The results of PAMAE-Hadoop and PAMAE-Spark are exactly the same. The results of PAM-MR and FAMES-MR, which are not based on sampling, were not available



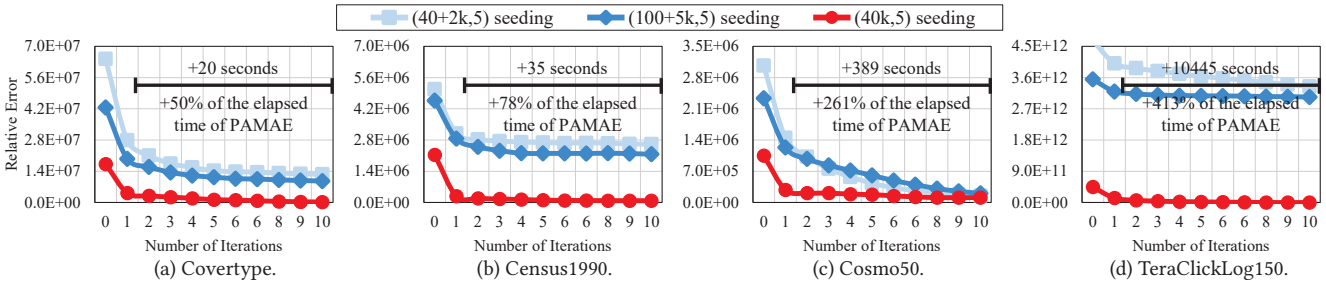
(a) $n = 40k$. (b) $m = 5$.

Figure 7: Sampling errors depending on parameter values.

for large data sets since they did not terminate within two hours. The results of CLARA-MR was better than that of CLARA-MR′ by virtue of larger samples. The results of GREEDI ranged between the relative errors of PAMAE before and after Phase II in the smallest data set (Figure 6(a)) whereas it did not terminate in the larger data sets. The results of MR-KMEDIAN improved as k increased with help from a larger sample, but it did not terminate when $k \geq 25$ in the largest data set (Figure 6(d)).

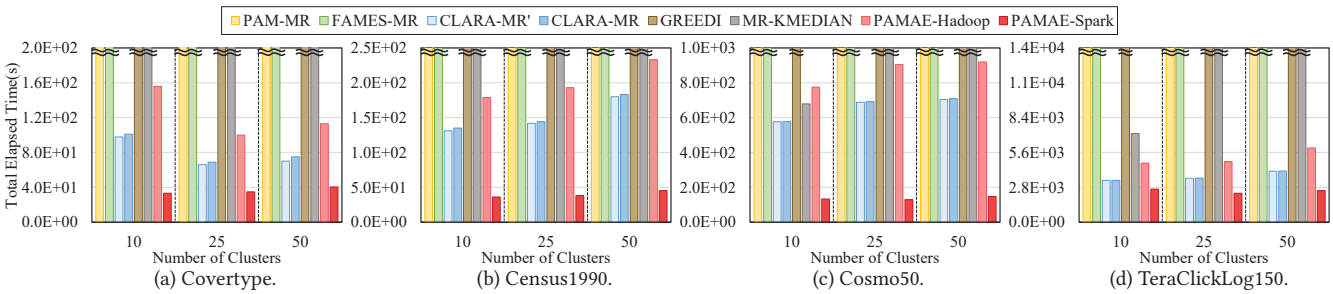
6.2.2 Verification of Theorem 4.4. Figure 7 shows the ratio of the clustering error of PAMAE to that of PAM, ϕ/ϕ^{PAM}, for the four data sets as the sampling parameters n and m vary. In order to focus on the errors caused by sampling, the clustering error ϕ was measured after Phase I finished. Also, in order to validate the simplified form of Theorem 4.4, Eq. (4), we used the preprocessed subsets of the data sets where all clusters contain the same number of objects. We repeated execution by 50 times to get the asymptotic behavior. The *expected* line indicates the theoretical values obtained by Eq. (4). The actual errors from these data sets were very close to the theoretical values. Moreover, with $n = 40k$ and $m = 5$, the error ratio almost converged to 1.01–1.03, thereby showing the reason for our choice of parameter values. Overall, we confirmed that the simplified form of Theorem 4.4 works well for real-world data sets.

6.2.3 Verification of Theorem 5.4. Figure 8 shows the relative error in Phase II of PAMAE as the number of iterations is forced to increase from 1 to 10 when k is 50. Here, we tried three different



(a) Covertype. (b) Census1990. (c) Cosmo50. (d) TeraClickLog150.

Figure 8: Convergence in Phase II depending on seeding strategies.



(a) Covertype. (b) Census1990. (c) Cosmo50. (d) TeraClickLog150.

Figure 9: Efficiency comparison of eight parallel k-medoids algorithms.

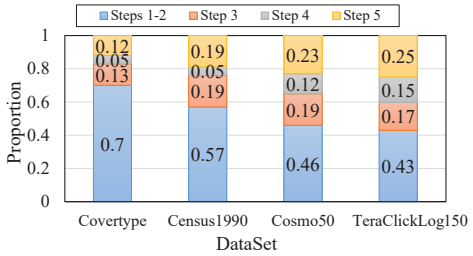


Figure 10: Proportion of the elapsed time in PAMAE-Spark.

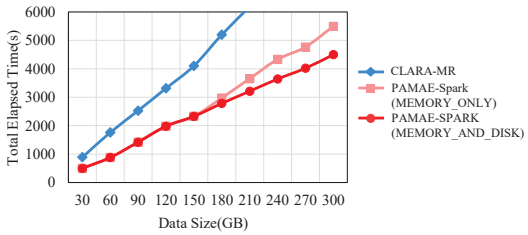


Figure 11: Scalability test on Spark.

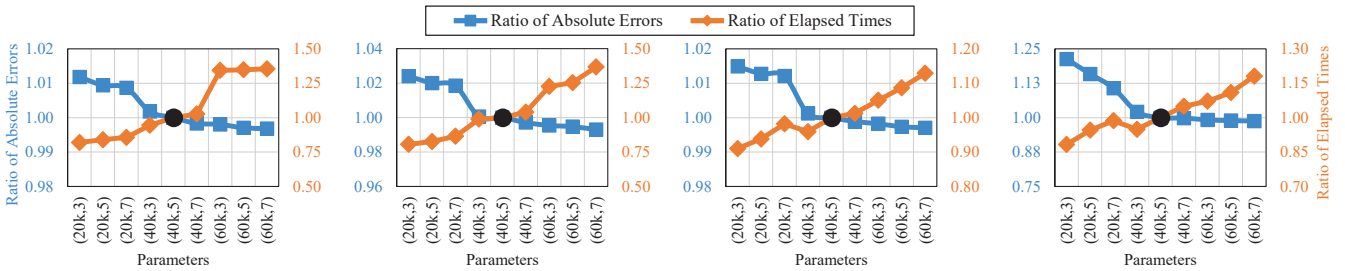
seeding strategies: $(40 + 2k, 5)$ by CLARA-MR′; $(100 + 5k, 5)$ by CLARA-MR; and $(40k, 5)$ by Phase I of PAMAE. Along the x-axis, 0 means the relative error derived by the seeds themselves, and i ($1 \leq i \leq 10$) means the relative error derived after the i-th refinement is complete. In Figure 8, when we used our seeding strategy, $(40k, 5)$, the relative error converged immediately after the first iteration. Subsequent iterations did *not* further decrease the relative error. This quick convergence indeed demonstrates the correctness of Theorem 5.4 that every true cluster is assigned a seed. In contrast, the other seeding strategies did not guarantee high quality of the seeds. Although the other strategies sometimes could achieve an accurate solution at the end, they had to spend quite a lot of *extra time*—i.e., 50–413% of the total elapsed time of PAMAE.

6.3 Efficiency Results

6.3.1 Overall Comparison. Figure 9 shows the total elapsed time of the eight parallel algorithms for the four data sets when k is set to be 10, 25, and 50. In general, PAMAE-Spark is the best, and CLARA-MR (or CLARA-MR′) is the next best. PAMAE-Spark outperformed CLARA-MR by up to 5.4 times. PAMAE-Spark was much faster than PAMAE-Hadoop because of the advantage of Spark over Hadoop. On the same platform, PAMAE-Hadoop spent more time by only 21–31% than CLARA-MR, which was caused by Phase II. We contend that this extra time is very small compared to the significant accuracy gain shown in Figure 6. Moreover, compared with GREEDI and MR-KMEDIAN which achieved higher accuracy than CLARA-MR, PAMAE-Hadoop outperformed them by up to 2.7–110 times.

Figure 10 decomposes the total execution time according to the steps of Figure 3 when k is 50. Step 6 is not included here since it is optional. When a data set is pretty small, e.g., Covertype, the proportion of Steps 1 and 2 was dominant since the *computational* complexity of Step 2 is the highest, i.e., $O(n^2)$. On the other hand, the proportion of Phase II (Steps 4 and 5) increased as the data size did from left to right, because Phase II is sensitive to the data size, not the sample size. Step 4 involves scanning of the *entire* data set, causing high network communication costs.

6.3.2 Scalability on Spark. Figure 11 presents the result of scalability test using TeraClickLog on Spark when k is 50. We used two storage-level options: "MEMORY_ONLY" and "MEMORY_AND_DISK". When main memory is not sufficient to hold a resilient distributed dataset (RDD), the former recomputes the RDD on the fly whereas the latter stores it on local disk. The data set started not to fit in main memory when its size exceeded 144 GB. The latter showed better scalability than the former, since the costs of disk I/O's were not that high because of the high performance of SSDs. Overall, PAMAE-Spark achieves near-linear scalability in that the total elapsed time increased also by 9.1 times when the data size increased from 30 GB to 300 GB by 10 times.



(a) Covertype. (b) Census1990. (c) Cosmo50. (d) TeraClickLog150.

Figure 12: Impact of sampling parameters on accuracy and efficiency (in color).

6.4 Impact of Sampling Parameters

Figure 12 shows the impact of a setting of the sampling parameters on accuracy and efficiency when k is 50. Since the scales of errors and times varied across data sets, we presented the ratio of the result with the other setting to that with the current setting for each data set. The *ratio of absolute errors* and the *ratio of elapsed times* are based on the clustering error and the elapsed time respectively. The lower they are, the better the other setting performs.

In Figure 12, when we tried smaller samples shown in the left of $(40k, 5)$, we could improve the elapsed time by up to 12% in Figure 12(d), but the clustering error increased by up to 21%. The gain in efficiency was not beneficial since accuracy loss was too big. On the other hand, when we tried larger samples shown in the right of $(40k, 5)$, we could improve the clustering error by as low as 0.3–0.7% while the elapsed time increased by up to 37% in Figure 12(b). The gain in accuracy was marginal, considering that the added time was substantial. Overall, we conclude that the current setting provides us with a reasonable trade-off between accuracy and efficiency.

7 CONCLUSION

In this paper, we proposed a novel parallel k-medoids algorithm, which we call *PAMAE*, going through two phases: parallel seeding and parallel refinement. Theoretical foundations of each phase are formally provided by the analysis of clustering error. We implemented our algorithm on Spark as well as Hadoop and then conducted extensive experiments using large-scale data sets on a cluster of 12 Microsoft Azure machines. Our experiment results showed that PAMAE achieved the excellent trade-off between accuracy and efficiency: when PAMAE showed an accuracy comparable to those of the other algorithms such as GREEDI, PAMAE significantly outperformed them; when PAMAE showed an efficiency comparable to those the other algorithms such as CLARA-MR, PAMAE significantly decreased clustering error compared with them. Overall, we believe that our work has significantly raised the usability of the k-medoids algorithm in the era of big data.

A LEMMAS FOR THEOREMS 4.4 AND 5.5

We present the full theoretical analysis of our algorithm. To formally define the properties of the uniform distribution and the central tendency, for a given cluster $\mathbb{C}$, we introduce a random variable X_i that represents the distance from the optimal medoid θ to the i-th *randomly-selected* object o_i from $\mathbb{C}$, as defined by Eq. (10). X_i's are independent and identically distributed (i.i.d.). In addition, R denotes the maximum value of X_i.

$$X_i = \text{dist}(o_i, \theta), \text{ where } o_i \in \mathbb{C} \tag{10}$$

Then, Definition A.1 formally states the properties of the distributions.

Definition A.1. A cluster $\mathbb{C} = \{o_1, \ldots, o_l\}$ follows a *uniform distribution* if $\mathbf{Pr}(X_i \le r) = \frac{r}{R}$ or has a *central tendency* if $\mathbf{Pr}(X_i \le r) > \frac{r}{R}$. □

Lemma A.2 focuses on the relationship between $\hat{\theta}$ and θ, in order to determine $E[\text{dist}(\theta, \hat{\theta})]$. Here, the *spatial median* is an imaginary (not necessarily actual) object that minimizes the sum of the distances from the objects in a cluster [13].

Lemma A.2. *Let θ^* and $\hat{\theta}^*$ be the spatial medians of $\mathbb{C}$ and $\mathcal{S} \subseteq \mathbb{C}$ respectively. Then, $E[\hat{\theta}^*] = \theta^*$.*

Proof. If the sample $\mathcal{S}$ is given with a sufficiently large size and a finite level of variance, the sampling distribution of the spatial median, $\hat{\theta}^*$, becomes approximately d-variate normal, $N_d(\theta^*, \sum)$, by the *central limit theorem* [8, 9]. □

The medoid is the actual object closest to the spatial median by definition. Thus, θ is the closest object to θ^*, and $\hat{\theta}$ is the closest object to $\hat{\theta}^*$. By Lemma A.2, the expectation of $\hat{\theta}^*$ is θ^*. If the size of a sample is large enough that the central limit theorem holds (typically, ≥ 30 [29]), $\hat{\theta} \in \mathcal{S}$ should converge toward θ. Thus, Eq. (11) holds with high probability.

$$E[\text{dist}(\theta, \hat{\theta})] = E[\min_{o_i \in \mathcal{S}} \text{dist}(o_i, \theta)] \tag{11}$$

Lemma A.3 proceeds to the analysis of the right-hand side of Eq. (11).

Lemma A.3. *For a cluster $\mathbb{C}$ which follows the uniform distribution in Definition A.1, let's consider its sample $\mathcal{S} = \{o_1, \ldots, o_n\} \subseteq \mathbb{C}$. Then, $E[\min_{o_i \in \mathcal{S}} \text{dist}(o_i, \theta)] < \frac{R}{n}$.*

Proof. Let $Y = \min\{X_1, \ldots, X_n\}$. The cumulative distribution function of Y is formulated by Eq. (12).

$$\begin{aligned} F(y) = \mathbf{Pr}(Y \le y) &= 1 - \mathbf{Pr}(Y > y) = 1 - \prod_{i=1}^{n} \mathbf{Pr}(X_i > y) \\ &= 1 - (1 - \frac{y}{R})^n, \ y \in [0, R] \end{aligned} \tag{12}$$

Next, the probability density function $f(y)$ is obtained by the derivative of Eq. (12). Then, the expected value of Y is calculated as Eq. (13). □

$$\begin{aligned} E[Y] &= \int_0^R y f(y) dy = \int_0^R y \frac{n}{R} (1 - \frac{y}{R})^{n-1} dy \\ &= \frac{R}{n+1} < \frac{R}{n} \end{aligned} \tag{13}$$

Table 4: The moment coefficients of the real-world data sets.

Data Set	Covertype	Census1990	Cosmo50	TeraClickLog
$k = 10$	0.087	0.16	0.012	0.24
$k = 25$	0.061	0.14	0.014	0.25
$k = 50$	0.031	0.11	0.011	0.25

Lemma A.4. *For a cluster $\mathbb{C}$ which follows the uniform distribution in Definition A.1, let $\mathcal{S} = \mathbb{S}_1 \cup \ldots \cup \mathbb{S}_m$ be a set of m samples whose size is n. Then, $E[\min_{o_i \in \mathcal{S}} \text{dist}(o_i, \theta)] < \frac{R}{nm}$.*

Proof. Since each sample is independent with each other, the cumulative distribution function of Y is calculated by multiplying that of each sample, as in Eq. (14).

$$F(y) = 1 - \prod_{i=1}^{m}(1 - \frac{y}{R})^n = 1 - (1 - \frac{y}{R})^{nm},\ y \in [0, R] \tag{14}$$

Thus, Lemma A.4 holds in the same way as Lemma A.3. □

B CENTRALITY IN REAL-WORLD DATA SETS

We verify that the clusters of the data sets in Table 2 have the *central tendency* of Definition A.1. For this purpose, we adopt Pearson's moment coefficient of skewness [10] in Eq. (15), which is commonly used to identify the dispersion of a given data set.

$$\gamma = \frac{E[(\text{dist}(o_i, \theta) - E[\text{dist}(o_i, \theta)])^3]}{E[(\text{dist}(o_i, \theta) - E[\text{dist}(o_i, \theta)])^2]^{3/2}},\ \text{where } o_i \in \mathbb{C} \tag{15}$$

According to the design of the random variable based on the *distance* from the medoid, a positive coefficient value (i.e., positive skew) implies that the mass of the distribution is concentrated on its medoid. Hence, a positive coefficient value is a suitable necessary condition to check the central tendency of a cluster. To evaluate the coefficient values, we chose 10,000 random samples, found the most accurate k medoids using PAM, and calculated the distances between the medoid and the belonging objects for each cluster. The distances were normalized to the range of 0 to 1. Then, we averaged the coefficient values obtained from all clusters. As a result, all the coefficient values are shown to be *positive*, as in Table 4. Therefore, Lemmas A.3 and A.4 are likely to hold for our real-world data sets, because $E[Y] < \frac{R}{n+1}$ in Eq. (13) when the central tendency is satisfied.

ACKNOWLEDGMENTS

This work was partly supported by the MOLIT (The Ministry of Land, Infrastructure and Transport), Korea, under the national spatial information research program supervised by the KAIA (Korea Agency for Infrastructure Technology Advancement) (17NSIP-B081011-04) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2015R1A1A1A05001475). Also, this project was supported by Microsoft Research through "Azure for Research" global RFP program.

REFERENCES

[1] Marcel R. Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. 2012. StreamKM++: A Clustering Algorithm for Data Streams. *J. of Experimental Algorithmics* 17, 2 (2012).
[2] Charu C. Aggarwal. 2015. *Data Mining: The Textbook.* Springer.
[3] Apache Software Foundation. 2017. Apache Mahout: Scalable Machine Learning and Data Mining. http://mahout.apache.org/. (2017). Accessed: 2017-02-01.
[4] Apache Software Foundation. 2017. Apache Spark MLlib. http://spark.apache.org/mllib/. (2017). Accessed: 2017-02-01.
[5] Jeffrey D Banfield and Adrian E Raftery. 1993. Model-based Gaussian and non-Gaussian Clustering. *J. of Biometrics* (1993), 803–821.
[6] Amir Beck and Shoham Sabach. 2015. Weiszfeld's Method: Old and New Results. *J. of Optimization Theory and Applications* 164, 1 (2015), 1–40.
[7] Christophe Biernacki, Gilles Celeux, and Gérard Govaert. 2000. Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 22, 7 (2000), 719–725.
[8] Probal Chaudhuri. 1992. Multivariate Location Estimation Using Extension of R-estimates through U-statistics Type Approach. *Ann. of Statistics* 20, 2 (1992), 897–916.
[9] Probal Chaudhuri. 1996. On a Geometric Notion of Quantiles for Multivariate Data. *J. the American Statistical Association* 91, 434 (1996), 862–872.
[10] David P Doane and Lori E Seward. 2011. Measuring Skewness: a Forgotten Statistic. *J. of Statistics Education* 19, 2 (2011), 1–18.
[11] Alina Ene, Sungjin Im, and Benjamin Moseley. 2011. Fast Clustering using MapReduce. In *Proc. 17th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*. 681–689.
[12] Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques.* Morgan Kaufmann.
[13] Thomas P Hettmansperger and Joseph W McKean. 2010. *Robust Nonparametric Statistical Methods.* CRC Press.
[14] W. Daniel Hillis and Guy L. Steele, Jr. 1986. Data Parallel Algorithms. *Comm. of the ACM* 29, 12 (1986), 1170–1183.
[15] IBM. 2014. "Simple" Threshold Algorithm Earns Gödel Prize. https://www.ibm.com/blogs/research/2014/05/simple-threshold-algorithm-earns-godel-prize/. (2014). Accessed: 2017-02-01.
[16] Pelle Jakovits and Satish Narayana Srirama. 2013. Clustering on the Cloud: Reducing CLARA To MapReduce. In *Proc. 2nd Nordic Sympo. on Cloud Computing and Internet Technologies*. 64–71.
[17] A.H.G. Rinnooy Kan and Gerrit T Timmer. 1987. Stochastic Global Optimization Methods Part I: Clustering Methods. *Mathematical Programming* 39, 1 (1987), 27–59.
[18] Leonard Kaufman and Peter J. Rousseeuw. 1987. *Clustering by Means of Medoids.* North Holland.
[19] Leonard Kaufman and Peter J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley.
[20] YongChul Kwon, Dylan Nunley, Jeffrey P. Gardner, Magdalena Balazinska, Bill Howe, and Sarah Loebman. 2010. Scalable Clustering Algorithm for N-body Simulations in a Shared-Nothing Cluster. In *Proc. 22nd Int'l Conf. on Scientific and Statistical Database Management.* 132–150.
[21] Nimrod Megiddo and Kenneth J. Supowit. 1984. On the Complexity of Some Common Geometric Location Problems. *SIAM J. on Computing* 13, 1 (1984), 182–196.
[22] Baharan Mirzasoleiman, Amin Karbasi, Rik Sarkar, and Andreas Krause. 2013. Distributed Submodular Maximization: Identifying Representative Elements in Massive Data. In *Proc. 27th Annual Conf. on Neural Information Processing Systems*. 2049–2057.
[23] Raymond T. Ng and Jiawei Han. 2002. CLARANS: A Method for Clustering Objects for Spatial Data Mining. *IEEE Trans. on Knowledge and Data Engineering* 14, 5 (2002), 1003–1016.
[24] Sevinc Ilhan Omurca and Nevcihan Duru. 2011. Decreasing Iteration Number of k-Medoids Algorithm With IFART. In *Proc. 7th Int'l Conf. on Electrical and Electronics Engineering*. 454–456.
[25] Bharat Pardeshi and Durga Toshniwal. 2010. Improved k-Medoids Clustering Based on Cluster Validity Index and Object Density. In *Proc. IEEE 2nd Int'l Conf. on Advance Computing*. 379–384.
[26] Hae-Sang Park and Chi-Hyuck Jun. 2009. A Simple and Fast Algorithm For K-Medoids Clustering. *Expert Systems with Applications* 36, 2 (2009), 3336–3341.
[27] Hae-Sang Park, Jong-Seok Lee, and Chi-Hyuck Jun. 2006. A K-means-like Algorithm for K-medoids Clustering and Its Performance. In *Proc. 2006 Int'l Conf. on Computer and Information Engineering*. 102–117.
[28] Adriano Arantes Paterlini, Mario A Nascimento, and Caetano Traina, Jr. 2011. Using Pivots to Speed-Up k-Medoids Clustering. *J. of Information and Data Management* 2, 2 (2011), 221–236.
[29] Sheldon M. Ross. 2014. *Introduction to Probability and Statistics for Engineers and Scientists.* Academic Press.
[30] A. B. Sunter. 1977. List Sequential Sampling with Equal or Unequal Probabilities without Replacement. *Applied Statistics* 26, 3 (1977), 261–268.
[31] H. Üster and R. F. Love. 2000. The Convergence of the Weiszfeld Algorithm. *J. of Computers & Mathematics with Applications* 40, 4 (2000), 443–451.
[32] Xianfeng Yang and Liming Lian. 2014. A New Data Mining Algorithm Based on MapReduce and Hadoop. *Int'l J. of Signal Processing, Image Processing, and Pattern Recognition* 7, 2 (2014), 131–142.
[33] Ying-ting Zhu, Fu-zhang Wang, Xing-hua Shan, and Xiao-yan Lv. 2014. K-Medoids Clustering Based on MapReduce and Optimal Search of Medoids. In *Proc. 9th Int'l Conf. on Computer Science and Education*. 573–577.