PARABLE: A PArallel RAndom-partition Based Hierarchical ClustEr ing Algorithm
for the MapReduce Framework

Shen Wang (sw2613@columbia.edu), Haimonti Dutta (haimonti@ccls.columbia.edu)
The Center for Computational Learning System, Columbia University, NY 10115.

Large datasets, of the order of peta- and tera- bytes, are becoming prevalent in many scientific
domains including astronomy, physical sciences, bioinformatics and medicine. To effectively
store, query and analyze these gigantic repositories, parallel and distributed architectures have
become popular. Apache Hadoop is a distributed file system that provides support for data-
tensive applications. It provides an open source implementation of the MapReduce
programming paradigm, which can be used to build scalable algorithms for pattern analysis and
data mining. MapReduce has two computation phases – map and reduce. In the map phase, a
dataset is partitioned into disjoint parts and distributed to workers called mappers. The mappers
implement local data processing and the output of the map phase is of the form < key, value >
pairs. These are passed to the second phase of MapReduce called the reduce phase. The workers
in reduce phase (called reducers) take all the instances that have the same key and do compute-
tensive processing and produce the final result. For a complex computation task, several
MapReduce pairs may be involved.

When processing large datasets, clustering algorithms are frequently used to compress or extract
patterns from the original data. Agglomerative Hierarchical clustering is a popular clustering
algorithm. It proceeds in four steps: (1) At the start of the algorithm, each object is assigned to a
separate cluster. Then, all pair-wise distances between clusters are evaluated using a distance
metric of one’s choice (2) The two clusters with the shortest distance are merged into one single
cluster. (3) The distance between the newly merged cluster and the other clusters are calculated
and the distance matrix is updated accordingly. (4) If more than one cluster still exists, goto step 2.
Hierarchical clustering algorithm is known to have several advantages -- it does not require
apriori knowledge of the number of clusters in the dataset. Furthermore, the distance metric and
cluster cutting criteria can be adjusted easily. However, the time complexity of the hierarchical
clustering algorithm is relatively high. More importantly, to find the two clusters that are closest
to each other, the algorithm needs to know the distances between all the cluster pairs. This
characteristic makes hierarchical clustering algorithm very hard to scale in a distributed
computing framework.

In this abstract, we present a parallel, random-partition based hierarchical clustering algorithm for
the MapReduce framework. The algorithm contains two main components - a divide-and-conquer
phase and a global integration phase. In the divide-and-conquer phase, the data is randomly split
into several smaller partitions by the mapper by assigning each instance a random number as key.
The instances with the same key are forwarded to the same reducer. On the reducers, the
sequential hierarchical clustering algorithm is run and a dendrogram is generated. The
dendrogram, a binary tree organized by linkage length, is built on the local subset of data. To
obtain a global cluster assignment across all mappers and reducers, dendrogram integration needs
to be implemented. Such integration is non-trivial because insertion and deletion of a single
instance changes the structure of the dendrogram. Our approach is to align them by “stacking”
them one on top of another by using a recursive algorithm described in Algorithm 1.

Algorithm 1: Recursive dendrogram aligning algorithm
1: function align(Dendrogram D1, Dendrogram D2)
2:   if similarity(D1.leftChild, D2.leftChild)+similarity(D1.rightChild,D2.rightChild)<
similarity(D1.rightChild,D2.leftChild)+similarity(D1.leftChild,D2.rightChild)
then
3: Switch D2’s two children
4: end if
5: align(D1.leftChild, D2.leftChild)
6: align(D1.rightChild, D2.rightChild)
The following example provides an illustration of the technique:

Example 1: Consider two dendrograms a and b that need to be aligned. Assume a is the template dendrogram -- this means dendrogram b is aligned to a and all structure changes will happen on b only. First, the roots of these two dendrograms (nodes of depth 1) are aligned to each other. Then, nodes at depth 2 need to be aligned. There are two choices -- the first is to align them as they are seen in the figure and thus - align $a_2$ with $b_2$ and $a_3$ with $b_3$. Another choice is the opposite, which aligns $a_2$ with $b_3$ and $a_3$ with $b_2$. The decision is made by comparing similarity $(a_2, b_2) + similarity (a_3, b_3)$ with similarity $(a_2, b_3)+similarity(a_3,b_2)$ and taking the one with higher similarity value. In this case, we will find it more reasonable to align $a_2$ with $b_3$ and $a_3$ with $b_2$. Therefore, $b_2$ and $b_3$ are switched and dendrogram b is transformed to dendrogram c. Then, for each pair of nodes that have been aligned in two dendrograms, say $a_2$ and $c_2$, we repeat the same procedure to align $c_2$’s two children with $a_2$’s two children. This procedure is repeated recursively until it reaches a depth deep enough for labeling.

The cluster labeling comprises of two steps – the first step involves cutting the template dendrogram into subtrees, similar to what a sequential hierarchical clustering algorithm would do during labeling. Each subtree is given a cluster label. Then for the root $a_i$ of each subtree i in the template dendrogram, the algorithm finds out all the nodes in other dendrograms that were aligned with it in the alignment step. Each of these nodes is also a root of a subtree in its own dendrogram and the algorithm will label all the instances belonging to this subtree with the same cluster label as $a_i$. Intuitively, this is like “stacking” all the aligned dendrograms together with the template dendrogram being put on top. Then a knife is used to cut the template dendrogram into pieces. After the template dendrogram is cut, the knife does not stop but cuts all the way to the bottom of the stack. By doing this, the entire stack is cut into several smaller stacks, and instances in each small stack are given the same cluster label.

The algorithm is implemented on an Apache Hadoop framework using the MapReduce programming paradigm. Empirical results on two large data sets from the ACM KDD Cup competition suggests that the PArallel RAndom-partition Based hierarchicaL clustEring algorithm (PARABLE) has significantly better scalability than centralized solutions. Future work involves theoretical analysis of convergence properties and performance benefits obtained from this randomized algorithm and implementation of multiple levels of local clustering.

Acknowledgements: Funding for this work is provided by National Science Foundation award, IIS-0916186.