Институт информационных и вычислительных технологий МОН РК

Казахский Национальный Университет имени аль-Фараби

Университет Туран

Люблинский технический университет, Польша

«Ғылым ордасы»

МАТЕРИАЛЫ

IV международной научно-практической конференции "Информатика и прикладная математика", посвященной 70-летнему юбилею профессоров Биярова Т.Н., Вальдемара Вуйцика и 60-летию профессора Амиргалиева Е.Н.
25-29 сентябрь 2019, Алматы, Казахстан

Часть 2

Алматы 2019

30. Guttman, L. A general nonmetric technique for finding the smallest coordinate space for a configuration of points. Psychometrics, 1968.- Vol. 33, N. 4. – P. 469-506.
31. Data text classification [Internet] https://github.com/zamgi/lingvo--classify. (access 10/08/2019)

PARALLEL CLUSTERING USING PARTITIONED GLOBAL ADDRESS SPACE MAPREDUCE MODEL

[1]Shomanov A.S., [2]Mansurova M.E.
e-mail: adai.shomanov@nu.edu.kz, mansurova.madina@gmail.com
[1]Nazarbayev University, Nus-Sultan, Kazakhstan,
[2]Al-Farabi Kazakh National University, Almaty, Kazakhstan

Abstract. *The paper introduces Mapreduce solution to a parallel clustering problem based on partitioned global address space (PGAS) model. In particular, paper discusses some optimization techniques that can lead to a better resource utilization and faster performance. One of the main issues that arises in parallel clustering is how to efficiently distribute the workload and minimize additional overheads of data exchange. Partitioned global address space model utilizes a one-sided communication pattern which outperforms MPI on the bandwidth limited problems and offers a convenient and transparent memory access model. Partitioning a memory into private and shared allows to create communication patterns among threads that enable flexibility in terms of handling data distribution with locality awareness in mind.*

1 Introduction

Clustering problem can be seen as a non-supervised learning approach to find similarity groups inside a dataset. There exist 4 different categories of algorithms for solving a clustering problem: connectivity, centroid, distribution and density based approaches. One of the most useful and efficient category in terms of parallelizability is a centroid-based. Centroid-based clustering works by smoothly moving cluster centers from some initial position to a position where closeness within a single cluster is minimized. Formally, the centroid-based clustering problem can be defined as minimization problem (see Eq. 1), where the goal lies in finding such an assignment S of points to cluster centers such that this assignment minimizes within-cluster squared differences of data points.

$$arg\min_{S} \sum_{i=1}^{K} \sum_{x \in S_i} \|x - \mu_i\| \qquad (6)$$

In terms of Mapreduce model, the problem can be decomposed into 2 or more stages [1]. In the first stage called **map** data needs to be divided into several chunks and distributed among participating threads of execution. The result of the map stage is a set of key-value pairs, where key corresponds to some aggregate feature that can be extracted from the given input. In the second stage called reduce, that follows after the map stage, grouped in an intermediate **shuffle** stage key-value pairs are distributed among parallel threads. These operations of moving and copying the key-value pairs from a memory of a one thread to a memory of another, comes with a set of problems for the runtime environment.

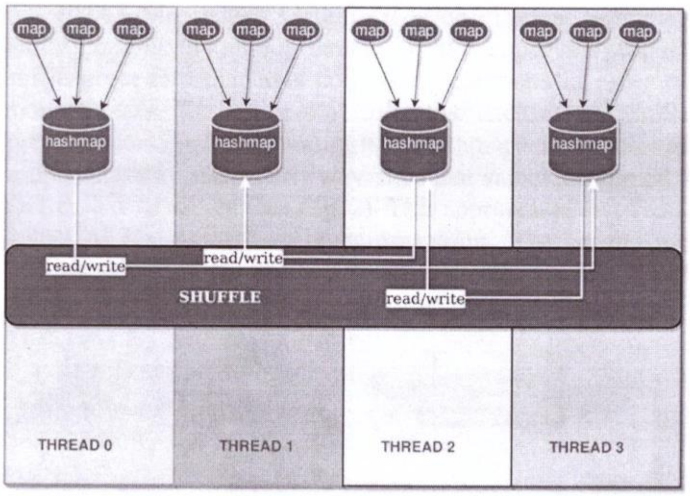

Fig.1 PGAS - based Mapreduce model.

In this work we present a parallel centroid-based clustering algorithm written in a Partitioned Global Address Space based Mapreduce system. PGAS – based Mapreduce system [2] was modified and rewritten to work with arbitrary keys and values. The previous implementation supported only a limited amount of data types. For the underlying mechanism to implement Mapreduce based on partitioned global address space model shared hashmap data structure was selected. Hashmap possesses a number of features that makes it suitable for a Mapreduce system. Operations on hashmap are performed in asymptotic complexity of $O(1)$), which provide fast lookup and insertion operations for the desired key. In our approach, the intermediate key / value pairs stored in a affine hashmap located in a shared portion of thread`s memory (see Fig. 1). Reduce threads are assigned key-value pairs after shuffle collectively gathers and groups keys across hashmap structures.

The operations on shared hashmap can be performed transparently by any thread, however, only a single, so called affine thread, is assigned to store in its local memory underlying key-value pairs associated with data partition assigned to that thread. The cost of local operation by orders of magnitude faster than remote accesses, therefore, it is important to consider optimal thread-to-data mappings. In PGAS – Mapreduce system we proposed a scheduling approach that assigns threads to data according to an optimality criterion that consists of workload and network latency. The given optimization problem solved by means of a genetic algorithm that tries to iteratively improve the scheduling till the process converges to a particular solution.

Due to large heterogeneity of different HPC platforms and parallel systems there has been some efforts to create Mapreduce systems for specific architectures [3,4] or target specific domain areas [5,6,7].

2 Main part

PGAS Mapreduce system relies on efficient execution of bulk and fine-grained memory operations among threads. In this paper we propose a new implementation of shuffle procedure that uses collective operations in order to exchange key-value pairs among threads. The process of collective exchange is similar to how collective gather operation works in MPI environment. Hashmap entries associated with a particular key are grouped together in a orderly way, such that values are copied in a bulk transfers according to a tree-like topology (see Fig. 2). This approach allows reducing by order by magnitude amount of fine-grained memory operations. The second modification to our previous implementation of PGAS Mapreduce is to add support for arbitrary data types for key-value pairs. The implementation is based on using shared void pointers.

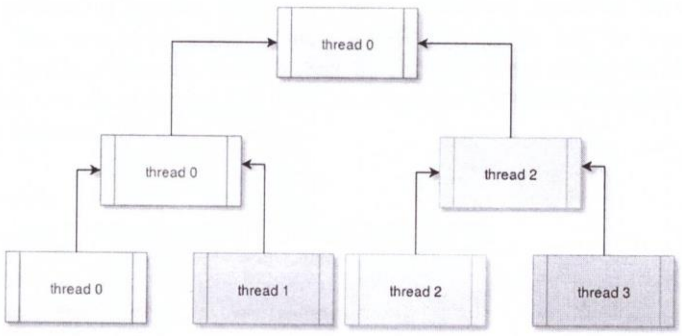

Fig. 2 Collective data exchange for key/values pairs

Mapreduce based parallel clustering algorithm relies on dividing the workload such that map processes are responsible for assigning data points a specific cluster centers by forming corresponding key-value pair of cluster center paired with a data point value (see Fig. 3). Then all data points that were assigned to the same cluster center are grouped together in a shuffle phase. Reduce phase is responsible for modifying cluster centers according to a mean sum of all data points in a cluster (see Fig. 4).

Map processes write generated by **Emit** key-value pairs into a local part of shared hashmap structure. After map processes finish their execution shuffle procedure calls collective merge on all generated keys in order to distribute all values associated with specific keys to their scheduled threads. Reduce processes fetch the data from shared hashmap according to collective exchange algorithm described above.

```
Algorithm: Map
input  : shared void * clusterCenters, string key, void * inputSplit
output: pair(int clusterCenterId,void * pointVal)
void * $points$ = Tokenize (inputSplit);
foreach $point \in points$ do
    $minDist \leftarrow maxDistanceValue$;
    for $clusterID \leftarrow 1$ to $K$ do
        $dist$ = findDistance ($point.clusterID$);
        if $dist < minDist$ then
            $minDist \leftarrow dist$;
            $clusterMinIndex \leftarrow clusterID$;
        Emit ($clusterMinIndex,point$);
    end
end
```

Fig. 3 Map algorithm for centroid-based Kmeans clustering

3 Results

In this section we present a parallel algorithm for a clustering problem of the collection of documents. The aim is to group similar documents given bag of words description of documents inside a dataset. Dataset has been taken from UCI Machine Learning Repository for the testing purposes [8]. Dataset consists of 300000 documents, 102660 unique words and 69679427 words in total.

```
Algorithm: Reduce
input  : shared void * clusterCenters, int clusterID, void * values
output: pair(int clusterCenterId,void * newClusterCentroid)
foreach $point \in values$ do
    $average \leftarrow average$ + findMagnitude($point$);
end
$clusterCenterId = clusterID$;
$newClusterCentroid \leftarrow$ getAverage($average$);
Emit ($clusterCenterId,newClusterCentroid$);
```

Fig. 4 Reduce algorithm for centroid-based Kmeans clustering

Each document and word is encoded by the unique identifier. The input consists of number of rows that are represented by the following tuples: document id, word id, the frequency of the word. Our first step was to transform above given bag of words representation to a sparse matrix form. For this task we created a separate Mapreduce task. The result of this preprocessing Mapreduce task was a vectorized representation of tuples consisting of document identifier as a key and a list of associated word frequencies. In our experiments we used a virtual machine with 64 vCPUs and 240 GB memory from Google Cloud Platform. Experimental setup consisted of Berkeley UPC runtime version 2.28.0, The Berkeley UPC-to-C translator, version 2.28.0.

Results of the first run were specified as an input to a second Mapreduce task. This Mapreduce task corresponds to a clustering algorithm described in the previous section.

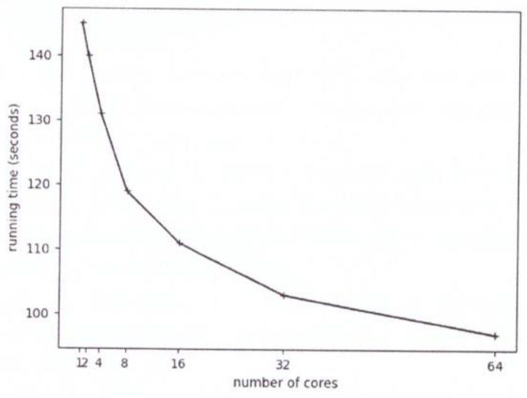

Fig. 5 Scalability of parallel Kmeans algorithm

Presented parallel Mapreduce algorithm shows good scalability which can be seen from Fig. 5.

4 Conclusion

This paper presented Mapreduce solution to a parallel centroid-based clustering problem based on partitioned global address space (PGAS) model. Several optimizations to a previous implementation of Mapreduce system have been introduced: a support for an arbitrary data types for key-value pairs and optimized tree-based collective exchange for a shuffle procedure.

5 Acknowledgement

This work was supported in part under grant of Foundation of Ministry of Education and Science of the Republic of Kazakhstan "System development for knowledge extraction from heterogeneous data sources to improve the quality of decision-making" (2018-2020) under project ID no. AP05132933.

References

1. Jeffrey, D., Ghemawat, S.: MapReduce: simplified data processing on large clusters. In: Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation, December 2004, vol. 6, pp. 10-10.

2. Shomanov, A., Akhmed-Zaki, D., Mansurova, M.: PGAS Approach to Implement Mapreduce Framework Based on UPC Language. In: Malyshkin V. (eds) Parallel Computing Technologies. PaCT 2017. Lecture Notes in Computer Science, vol 10421. Springer, pp. 342-350, Cham (2017)

3. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G. & Kozyrakis, C.: Evaluating MapReduce for multi-core and multiprocessor systems. In: *Proceedings - International Symposium on High-Performance Computer Architecture*, 2007, pp. 13.

4. He, B., Fang, W., Luo, Q., Govindaraju, N.K. & Wang, T.: Mars: A MapReduce framework on graphics processors. In: *Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT*, 2008, pp. 260.
5. Engelmann, C. & Böhm, S.: Redundant execution of HPC applications with MR-MPI. In: *Proceedings of the 10th IASTED International Conference on Parallel and Distributed Computing and Networks, PDCN 2011*, pp. 31.
6. Hoefler, T., Lumsdaine, A. & Dongarra, J. 2009, *Towards efficient mapreduce using MPI*. In: Ropo M., Westerholm J., Dongarra J. (eds) Recent Advances in Parallel Virtual Machine and Message Passing Interface. EuroPVM/MPI 2009. Lecture Notes in Computer Science, vol 5759. Springer, Berlin, Heidelberg
7. Kang, U., Tsourakakis, C.E. & Faloutsos, C.: PEGASUS: A peta-scale graph mining system - Implementation and observations", *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2009, pp. 229.
8. Dua, D., Graff, C. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2019.

ЖАРЫҚ КӨЗІ

Ауэзова А.М., Алибиева Н.М.
a.auezova@mail.ru, alibieva_n_85@mail.ru
«ҚДСДЖМ» Қазақстандық медицина университеті,
әл – Фараби атындағы Қазақ ұлттық университеті

Аңдатпа. *Халық санының өсуімен және оның әл-ауқатының жақсаруымен энергия ресурстарына деген қажеттілік өсуде. Ғаламшардың күшеюі - 2 миллиард, жаһандық экономика – 140% - ға, энергияны тұтыну тиімділігі - 45%-ға өсті. "Жасыл экономика" жаһандық қауіптерден Қазақстанның тәуекелдерін төмендетуге көмектеседі. Мысалы, климаттың өзгеруі, пайдалы қазбалардың сарқылуы, су ресурстарының тапшылығы. сонымен қатар, бірқатар сарапшылар мен мемлекеттік қызметкерлердің пікірінше, "жасыл" үлгіге көшу ел үшін қосымша мүмкіндіктер ашады [5].*

Түйін сөздер: *жарық көзі, сәуле шығаратын диодтар (СШД), жасыл экономика.*

Кіріспе

Тұңғыш Президентіміз Нұрсұлтан Әбішұлы Назарбаев мемлекет алдына барлық шаруашылық салаларында энергияны тұтынуды айтарлықтай үнемдеу міндетін қойды. Отандық кәсіпорындар жинақтаған тәжірибеге сүйеніп, қазіргі заманғы технологиялық циклді - дамудан бастап, жаппай өндіруге дейін - жартылай өткізгіш өнімдерді жарықтандыру холдингтің ауқымды міндетін шешу деп түсінеді. Жаңа жартылай өткізгіш жарық көздерін пайдалану – жарықтандыру құрылғылары үшін маңызды энергия, үнемдеуге қол жеткізетін бірден бір жол.