Implementation of a solution Cloud Computing with MapReduce model

Implementation of a solution Cloud Computing with MapReduce model

Chalabi Baya
Higher National School of Computer Science ,Algeria

E-mail: b_chalabi@esi.dz

Abstract. In recent years, large scale computer systems have emerged to meet the demands of high storage, supercomputing, and applications using very large data sets. The emergence of Cloud Computing offers the potentiel for analysis and processing of large data sets.
Mapreduce is the most popular programming model which is used to support the development of such applications. It was initially designed by Google for building large datacenters on a large scale, to provide Web search services with rapid response and high availability.
In this paper we will test the clustering algorithm K-means Clustering in a Cloud Computing. This algorithm is implemented on MapReduce. It has been chosen for its characteristics that are representative of many iterative data analysis algorithms. Then, we modify the framework CloudSim to simulate the MapReduce execution of K-means Clustering on different Cloud Computing, depending on their size and characteristics of target platforms.
The experiment show that the implementation of K-means Clustering gives good results especially for large data set and the Cloud infrastructure has an influence on these results.

1. Introduction

In recent years, we are witnessing the development of Applications that deal with very large data sets (about Peta bytes) and distributed throughout the world. [9]. These applications require infrastructure that offer particularly wide storage capacity and processing capabilities very important[9]. In addition to these fundamental characteristics, the communication aspect is also important insofar as these applications require of high speed networks and secure. To respond to Such requirements, which are out of reach for people individual, a new class of architectures has seen the day. This category includes grid architectures [14] and Cloud architectures [3]. The expansion of these architectures is done both in the field of industry and in the research science. On the industrial side, companies such as Google [1] have introduced large data centers on a global scale to provide web services that are both stable and highly available, ensuring fast response times for users. In the academic field, many scientific research projects have been launched to provide solutions to manage large data sets in a very broad scale [4]. Research conducted within the framework of these differents projects have led to the emergence a new concept in computing, namely the Cloud Computing [3]. The interest in this new concept led many companies specializing in computer to devote substantial budgets for make it available to users. Among the products that have been placed on the market, we can cite Google App Engine [1]

provides an overview of the MapReduce technique and a brief introduction to Hadoop. In Section 4, we presentCloudSim simulator. Section 5 is allocated for description of the data mining algorithm that requires important data set which is k-means clustering. While Section 6 presents the implementation of K-means Clustering in the Cloud Computing by using Hadoop.

2. Related work

Currently k-means Clustering can be implemented in cloud frameworks for iterative algorithms . Zhao et al [23] adapt this algorithm for Hadoop MapReduce. The work focuses on implementing the K-mans clustering with the read-only convergence heuristic in the Hadoop MapReduce pattern . However, as Hadoop does not support iterative processing by design, this implementation has little choice regarding data affinity even with the read- only convergence heuristic . Twister [15] and HaLoop [25] add modifications to MapReduce for supporting iterative computation. In ower work we implement k-means with Hadoop, but with different infrastractures that used by Zhao et al[23]; also in the experiments we give the memory size used and network consumption by the benchmarks which are not done in others works. Based on these experiments, we have changed the framework CloudSim for supporting the simulation of MapReduce execution to see the effect of the availability of virtual machine (VM)on the execution time.

3. MapReduce model

In this section, we present a brief introduction of MapReduce. MapReduce is a partitioning mechanism of tasks for distributed execution on a large number of servers [16]. It is primarily intended to tasks such as batch. Its principle is summarized as follows: it is to break a task into smaller tasks, or more precisely cut a task involving very large volumes of data on identical tasks with subsets of the data. Tasks (and their data) are then dispatched to different servers, then results are retrieved and consolidated. The upstream phase, decomposition of tasks is called Map part, while the downstream phase, the consolidation of the results is called the party Reduce. The principle is relatively simple but the contribution of MapReduce is to well conceptualize view to normalizing the operations of stewardship so that same framework can support a variety of partitionable tasks, allowing developers to focus on the treatments themselves, while The framework supports the logistics of distribution.
Operations "Map" and "Reduce" are general and have a simple interface. Each one receives a sequence records (records within the meaning of the model), and each usually produces output record. A record consists of a key and a value. The input records presented to the mapper by its appellant, does not guarantee or ordering relation for the execution of Map tasks. The work of mapper is to create (or not) a number of records in response to each input record. Records are presented to reducer by its caller and are placed in a cell according to their key, so that all records with a same key are presented as a package to the reducer. The reducer then examines packets sequentially using an iterator.

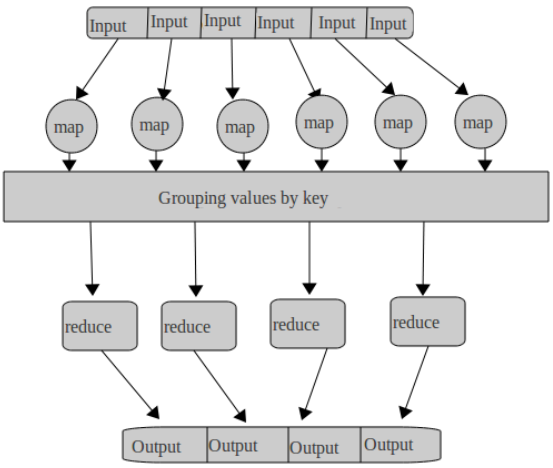

Figure 1. Operating principle of MapReduce

.

3.1. Hadoop

The implementation of MapReduce by Apache Hadoop is coupled with the distributed file system (HDFS). It is similar to the architecture of Google MapReduce runtime in which accesses data to HDFS [2], all mape local disks of the compute nodes in a file system single hierarchy, thus allowing the replication of calculation data between all nodes. HDFS also replicates data on multiple nodes so that all failures nodes containing a portion of data does not affect the calculations that use these data. Hadoop order the MapReduce computation tasks according to data locality, thereby improving the overall strip width. the outputs mapped tasks are first stored in local disks to perform later access to operation reduce[24] over HTTP connections. Although this approach simplifies the mechanism tasks processing, it generates a significant communication overhead for intermediate data transfers, especially for applications that frequently produce small intermediate results. The main mean used by hadoop to achieve fault tolerance is to complete execution (or rerun) the assigned tasks in the cluster [5]. Another special feature that it is useful in Hadoop mentioning is the speculative execution model. In a parallel system, there is the problem of the presence of slower nodes that slow the calculation of the entire cluster. In Hadoop, since the tasks are executed isolated from each other, then the same input may be treated several times in parallel by different nodes.

4. CloudSim

CloudSim is a framework that provides a simulation generalized and extensible allowing modeling, simulation and experimentation of new infrastructure of Cloud Computing and application services involved. It covers most of the activities that take place within a Data Center in detail. It allows [21]:

- Simulate the definition of physical data centers in terms of physical machines consisting of processors, storage devices, memory and internal bandwidth.
- Simulate the virtual machine specification, their creation and destruction. Management of virtual machine, allocation physical hardware resource for the operation of virtual machines based on differents policies.

5. Proposed implementation

The K-means method [11] is known for its degradation when all the data growing in terms of number of objects and dimensions [17, 18]. To address this requirement, we implement this algorithm with MapReduce. In this section, we describe the clustering algorithm k-means clustering with MapReduce model.

(i) The set of input data (file of data points) is partitioned into N portions. Each part is sent to a Maper.

(ii) In the Map function, the distance (Euclidean distance) between each point and each cluster center is calculated and each point is labeled with the index of the center cluster where the distance is the smallest. The outputs of the function Map is the key-value pair:
- Key: cluster ID.
- Value: the coordinates of the point (recording).

(iii) The function "Map" produced a large amount of data. So, we use a function Combine to reduce the size before sending it to the function Reduce. Combine function calculates the average coordinates for each cluster ID, and the number of records. It outputs a pair key-value for each cluster:
- The Key: Custer ID.
- Value: the number of records and the values average coordinates.

(iv) All data points in the same cluster are sent to one Reducer. In the Reduce function, new coordinates of the centers of clusters are easily calculated (same treatment as the Combine function except that the Combine function is executed at the Mapers nodes while Reduce is executed in Reducers nodes). The output of the reducer is:
- Key: cluster ID.
- Value: new coordinates of the center of cluster.

(v) The new coordinates of the centers of clusters are compared to the original. To do this, we need to save the centers of clusters of the previous iteration. If the difference does not exceed a threshold predefined, then the program ends, and we found clusters. Otherwise, set the centers of clusters newly generated and we repeat steps 2 to 5.

6. Experements with Hadoop and results

To implement K-means clustering with MapReduce we use Hadoop and programming language Java is chosen, we experience this algorithm in a Cloud Clustering which constitutes five machines where the hardware specifications of the master node (jobtracker) are:

- Memory: 1GB.
- processor :2.8 GHz.

For the four virtual machines, we have used virtualbox and the client nodes (tasktracker) hardware specifications are:

- Memory: 512 MB.
- processor : 1.4 GHz.

The OS is Ubuntu-10.10.
The benchmarks used to make the experiments of K-means Clustering with Hadoop are generated aleatory and the number of clusters K is equal to 9. Also 9 centers of initial clusters are randomly generated.
The Size of benchmarcks in MB are respectively :5.70, 28.50, 56.90, 170.70, 284.50. Ganglia [12] is used for extracting the metrics. We will retrieve metrics values for an iteration execution of K-means clustering. The following figures show the results of experiments (execution time, memory size consumed, consumption network):

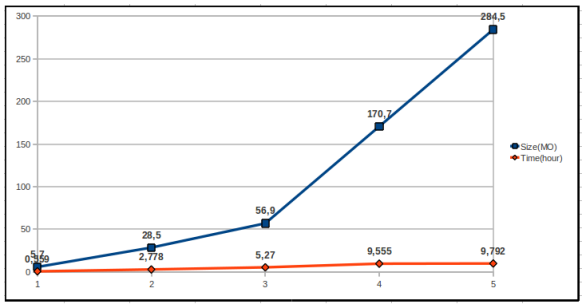

Figure 2. Execution time of five benchmarks used for K-means clustering in the Cloud Clustering .

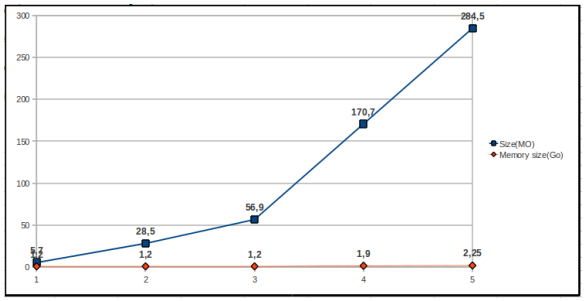

Figure 3. Memory size used by the five benchmarks used to K-means Clustering in the Cloud Clustering.

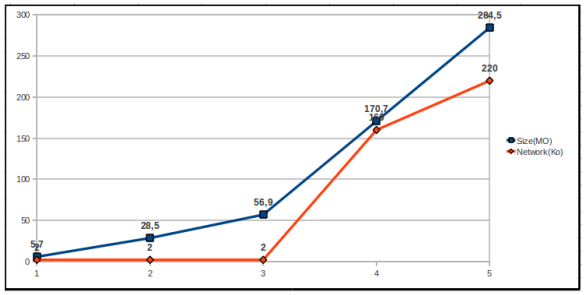

Figure 4. Network consumption by the five benchmarks used to K-means Clustering in the Cloud Clustering.

6.1. Discussion

Based on these results, we note that for the three first experiments, the execution time increases almost with the same value as the size of the data handled because these three experiments are executed at the same node which is the master node (one task) as the block size used by HDFS file is 64MB and data used are all (5.7,28.5, 56.9) less than 64MB. For the last two experiments, the execution time is almost the same despite that the size of the data manipulated in the last experiment and almost double that of the fourth experiment, because to the fourth iteration

the number of tasks Map is 3 and the three tasks are executed in parallel then the task Reduce (we choose to use a single Reducer in all experiments). For the last experiment the number of tasks Map is 5 and since we have a sufficient number of nodes, all tasks are processed in parallel. Parallel execution shows that we just near about the same execution time for recent experiments (the time difference is due to data locality) for the fourth experimentation, all tasks using local data (3 replies) but for the last experiment it is not all tasks that access their data locally.
The second finding is that, compared to the execution time of the third experiment with the execution time of the fourth and fifth experiment, we notice that there is a big difference (although the size of data processed in the first experiment is 56.9 MB and the data size of each task Map addressed in the fourth and fifth experiment is 64 MB. These tasks are performed in parallel. And since the task of the third experiment is performed on the master node and the tasks of the fourth and fifth experiment are treated at the client nodes which have different physical characteristics of the the master node. This implies that the material characteristics have an influence on the execution time, is to say more than the memory size is large and the speed processor is fast, the execution time is good.

7. Experimenting with cloudsim and results

To see the effect of the physical characteristics on the execution time, we simulate the performance of K-means clustering on different Clouds characteristics. For Data-center model used (IaaS), we used Hosts with the following characteristics:

- Ram :3000 Mo.
- Bw = 1000.
- Storage = 1000000 MO.

For processors, they are modeled in terms of MIPS (millions of instructions per second). We used a Pes with a value of Mips égalle 1000. In all experiments the number of virtual machines is 20, we have used virtual machines VirtualBox to take the same type as that used in the experiments with Hadoop. We used different models:

Table 1. Size memory and processor MIPS used for each expérimantation.

	Memory Size (MO)	**Processor Mips**
1	512	140
2	1024	280
3	1536	420
4	2048	560

The data sizes processed in GB are: 0.312, 0.937, 1.937, 2.812.
For each data size, we will allocate experiments with four different characteristics of virtual machine (RAM, CPU).

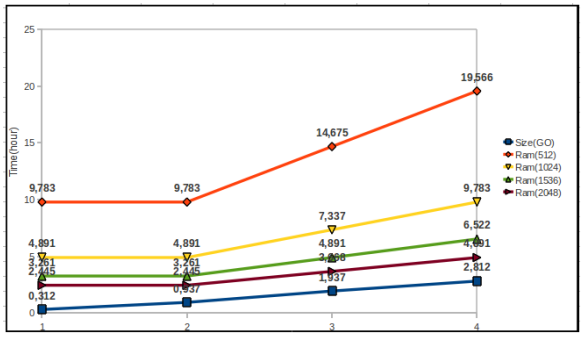

Figure 5. Variation of the execution time under different configurations (memory, CPU).

7.1. Discussion
Figure 5 shows the execution time is inversely proportional with the capacity of the RAM used by the nodes and processing capacity. Calculation of the gain between the four experiments, ascending Mips and Ram is as follows:

Table 2. varying the gain of execution time as a function of the size of the ram and the speed of processor.

Taille(GO)	**Gain of time between different experiments**		
	Between(1 & 2)	**Between(2& 3)**	**Between(3& 4)**
0.31	2.05	1.499	1.480
0.94	2.05	1.499	1.480
1.94	2.00	1.500	1.333
2.81	2.00	1.500	1.333

For the first and the second experiment, the size of data is triple that used in the first. By cons, we note that the execution time remains the same, that the size of each task Map is 64Mo and the number of tasks for each size is 5 and 15, respectively, and each task is modeled by one cloudlet. The cloudlets are executed in parallel before pass their results to cloudlets Reducer that beginning their treatment after the cloudlets that model Map tasks are completed.
For the last two, the execution time is different since the number of virtual machines needed to perform their tasks (Map) is not available (the number Task (cloudlets) is respectively 30 and 45) for the fourth and fifth experiment, and the number of available virtual machines is 20, and since each task (cloudlet) must be executed by a VM. This implies that Not all tasks are run in parallel.
The table 2 shows that gain degree is different, it does not increase by the same amount even though the added power is the same.

8. Conclusion
The popularity of K-means algorithm is due to its procedure simple and rapid convergence to a suitable solution but K-means is known for its degradation when the data set grows. MapReduce model is more appropriate for this type of algorithms that require large power storage and computation. We are implementing K-means with MapReduce (Hadoop) which is fault tolerant and given the opportunity to run speculative. The results of our experiments showed that cloud clustering is most effective when the data set is very large. Plus size data large, implying a fairly

large number of tasks Map, the more consumption of resources (memory, network) increases. Based on simulations with CloudSim we can conclude that the number of available nodes and power storage and computing have an influence on the execution time. If the number of nodes required for the tasks Map is available, then the gain of the execution time is proportional to the power of nodes used and this gain degree is different, it does not increase by the same amount even though the added power is the same.

References

[1] Google app engine. `http://appengine.google.com`, Juillet 2008.
[2] Apache. hdfs architecture. `http://hadoop.apache.org/common/docs/current/hdfs_design.html`, 2009.
[3] Amazon elastic compute cloud (amazon ec2). `http://aws.amazon.com/ec2/.`, 2010.
[4] Spring.net. `http://www.springframework.net.`, 2010.
[5] A. Rasin D. J. Abadi-D.J. DeWitt S. Madden An. Pavlo, E. Paulson and M. Stonebraker. A comparison of approaches to large-scale data analysis. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 165–178, 2009.
[6] B. Krishnamurthy C. Graham. Key differences between web 1.0 and web 2.0. *First Monday 13.6*, 2008.`http://www.uic.edu/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/2125/1972.`
[7] Y. An Lin-Y.Yuan Yu G. Bradski A. Ng K.Olukotun C-T. Chu, S. Kyun Kim. Map-reduce for machine learning on multicore. *NIPS*, pages 281–288, 2006.
[8] M. Mattess R. Buyya C. Vecchiola, X. Chu. *Aneka - Integration of Private and Public Clouds.* -13: 978-0470887998. Wiley Press, New York, USA, 2010.
[9] Raphael Y. De Camargo, Andrei Goldchleger, and Fabio Kon. *InteGrade: a tool for executing parallel applications on a Grid for opportunistic computing.* Fortaleza-CE, 2005.
[10] C. Grzegorczyk et al D. Nurmi, R. Wolski. Eucalyptus: A technical report on an elastic utility computing architecture linking your programs to useful systems. Technical report, University of California Santa Barbara Computer Science, Santa Barbara, 2008.
[11] V. Faber. Clustering and the continuous k-means algorithm. In *Los Alamos Science*, volume vol. 22, pages pp. 138–144, 1994.
[12] M.L. Massie F.D. Sacerdoti, M.J. Katz and D.E. Culler. Wide area cluster monitoring with ganglia. In Citeseer, editor, *In Proceedings of the IEEE Cluster 2003 Conference*, 2003.
[13] Y. Liu R. Buyya H. Sun, J. Huai. Rct: A distributed tree for supporting efficient range and multi-attribute queries in grid computing. In *Elsevier Science*, volume 24 of *0167-739X*, page 631–643. Amsterdam, Juillet 2008.
[14] C. Kesselman I. Foster. The grid: Blueprint for a new computing infrastructure. *Morgan Kaufmann*, page 677, 1998.
[15] B. Zhang T. Gunarathne S.-H. Bae J. Qiu J. Ekanayake, H. Li and G. Fox. "twister: a runtime for iterative mapreduce,". In Eds. ACM, editor, *in HPDC, S. Hariri and K. Keahey*, page pp. 810–818, 2010. [Online]. Available: http://dblp.uni-trier.
[16] C. Dyer J. Lin. Data-intensive text processing with mapreduce. *Morgan Claypool*, pages 102–105, Avril 2010.
[17] M. Joshi. Parallel k-means algorithm on distributed memory multiprocessors. Technical report, University of Minnesota, 2003.
[18] P. Sattayatham K. Kerdprasop, N. Kerdprasop. Weighted k-means for density-biased clustering, lecture notes in computer science. In *Lecture Notes in Computer Science*, pages 488–497. Data Warehousing and Knowledge Discovery (DaWaK), 2005.
[19] M. Kunze L. Wang and J. Tao. Performance evaluation of virtual machine based grid workflow system. *Concurrency and Computation: Practice and Experience*, 20(15):1759–1771, 2008.
[20] A.Beloglazov C. A. F. Rose-R. Buyya N. Rodrigo. Calheiros, R.Ranjan. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Wiley Press*, 41(1):23–50, January 2011.
[21] D. Karunamoorthy R. Buyya R. N. Calheiros, C. Vecchiola. Aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. *Future Generation Computer Systems*, 2011.
[22] M. Kunze V. B ge, Y. Kemp and G. Quast. Application of virtualisation techniques at a university grid center. *e-Science*, 2006.
[23] H. Ma W. Zhao and Q. He. "parallel k-means clustering based on mapreduce,". In *Cloud Computing, ser. Lecture Notes in Computer ScienceM. Jaatun, G. Zhao, and C. RongEds. Springer Berlin / Heidelberg, 2009.*
[24] T. White. *hadoop:the definitive guide.* 978-1-4493-8974-1. O'Reilly Book Evangelist, 2011.

[25] M. Balazinska Y. Bu, B. Howe and M. D. Ernst. "haloop: Efficient iterative data processing on large clusters,". In *36th International Conference on Very Large Data Bases*. Singapore, September 14–16, 2010.