CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

Pradyot Mayank(A20405826)

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

Index

Contents **Page Number**

Abstract	3
Introduction	3
Literature Review	5
Background/Motivation	6
Approach	8
Design/Implementation	10
Evaluation	13
Conclusion/Future Scope	15
Acknowledgement	16
References	17

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

Abstract:

- **Parallel K-Means:** Clustering is one of the most popular methods for data analysis, which is prevalent in many disciplines such as image segmentation, bioinformatics, pattern recognition and statistics etc. The most popular and simplest clustering algorithm is K-means because of its easy implementation, simplicity, efficiency and empirical success. However, the real-world applications produce huge volumes of data, thus, how to efficiently handle of these data in an important mining task has been a challenging and significant issue. In addition, MPI (Message Passing Interface) as a programming model of message passing presents high performances, scalability and portability. Motivated by this, a parallel K-means clustering algorithm with MPI, called MKmeans, is proposed in this paper. The algorithm enables applying the clustering algorithm effectively in the parallel environment. Experimental study demonstrates that MKmeans is relatively stable and portable, and it performs with low overhead of time on large volumes of data sets.
- **External Sorting:** Sorting is one of the most common things we do in programming. We are given a bunch of numbers and we want to arrange them according to some rule. Let's say we want to arrange them in ascending order. To sort these numbers, people tend to use a sorting algorithm that takes place entirely within the memory of a computer. The memory we are talking about is the RAM. Here, we take all the numbers and store them in the memory so that we can sort them. This is possible only when the amount of data is small enough to be stored in the memory. What if we have a hundred trillion numbers to be sorted? It's too big to be stored in the computer's memory.

Introduction:

- **K-Means:** k-means is an unsupervised learning algorithm that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. The main idea is to define k centers, one for each cluster. These centers should be placed in a way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by:

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} \left(\left\| x_i - v_j \right\| \right)^2$$

where,
'||xi - vj||' is the Euclidean distance between xi and vj.
'ci' is the number of data points in ith cluster.
'c' is the number of cluster centers.

Algorithmic steps for k-means clustering

Let X = {x1, x2, x3,……..,xn} be the set of data points and V = {v1,v2,…….,vc} be the set of centers.

1) Randomly select 'c' cluster centers.
2) Calculate the distance between each data point and cluster centers.
3) Assign the data point to the cluster center whose distance from the cluster center is minimum of all the cluster centers.
4) Recalculate the new cluster center using:

$$v_i = (1 / c_i) \sum_{j=1}^{c_i} x_i$$

where, 'ci' represents the number of data points in ith cluster.

5) 5) Recalculate the distance between each data point and new obtained cluster centers.
6) 6) If no data point was reassigned then stop, otherwise repeat from step 3).

Below are the diagrams to show that how clusters are formed using the data points:

Figure 1: Shows the clusters of random data points

Figure 2: Show 3 clusters are formed using those data points.

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

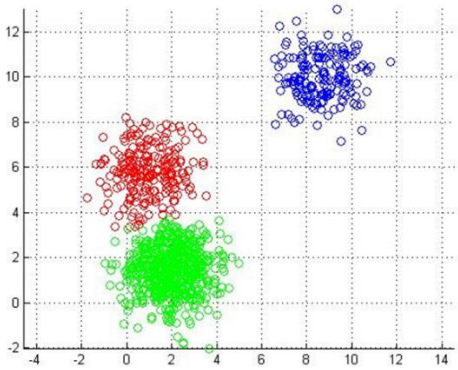


Figure 1

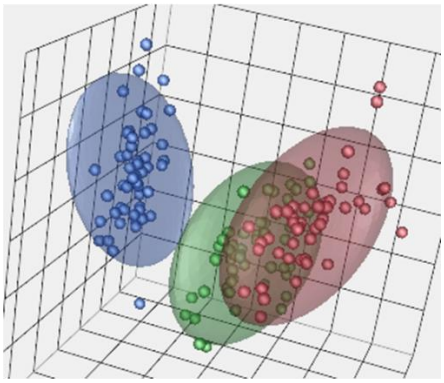


Figure 2

- **External Sorting:** External sorting is a class of sorting algorithms that can handle massive amounts of data. External sorting is required when the data being sorted do not fit into the main memory of a computing device (usually RAM) and instead they must reside in the slower external memory, usually a hard disk drive or SSD. Thus, external sorting algorithms are external memory algorithms and thus applicable in the external memory model of computation.

  External sorting algorithms generally fall into two types, distribution sorting, which resembles quicksort, and external merge sort, which resembles merge sort. The latter typically uses a hybrid sort-merge strategy. In the sorting phase, chunks of data small enough to fit in main memory are read, sorted, and written out to a temporary file. In the merge phase, the sorted sub files are combined into a single larger file.

Literature Review:

- **Parallel K-Means:** K-mean is the most popular partitioning method of clustering. Mac Queen in 1967, firstly proposed this technique, though the idea goes back to Hugo Steinhaus in 1957. The standard algorithm was first proposed by Stuart Lloyd in 1957 as a technique for pulse-code modulation. Sometimes it is referred as Lloyd-Forgybecause In 1965, E.W.Forgy published essentially the same method [8].
- **External Sorting**: Sorting is a filed which has been a field of interest for all of the researcher who has been somehow connected with the faster execution of the database and all sorted arrays.

CS597-Reading and Special Problems

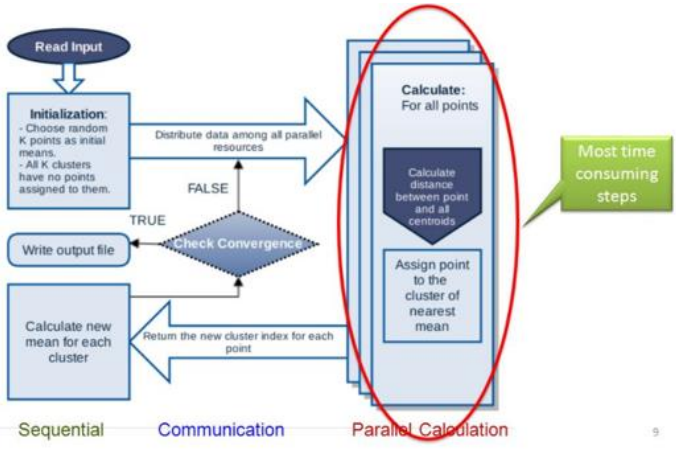
Parallel K-Means and External Sorting Implementation

A lot of researcher have their own view about sorting. Let's get to know first that what they think about sorting and then we will be listing the problems which are now currently in sorting and then we would be providing a possible solution for this let's see how it goes.
Torsten Grust says that DBMS does not execute a query as a large monolithic block but rather provides a number of specialized routine.

Background/ Motivation:

- **Parallel K-Means:** In **[1]** authors present a clustering using a coarse-grained parallel genetic algorithm to obtain an optimal minimum squared-error partitions and use a distributed algorithm to improve the total execution time. Their algorithm is based on a SIMD model. In **[2]** authors present an optimal adaptive K-means algorithm with dynamic adjustment of learning rate to induce a near-optimal clustering solution in a situation where the pattern ensemble is not available. In **[3]** authors define the notion of a well-separated pair decomposition of points in d-dimensional space and develop efficient sequential and parallel algorithms for computing such a decomposition. The authors then present a decomposition of multidimensional point sets and its applications to k-Nearest-Neighbors and n-body potential fields. Their parallel algorithm is based on a CREW PRAM model. These are some papers related to parallel clustering and K-means algorithms that are known to us.

Parallel Feature Extraction



CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

- **External Sorting:** External sorting **[4]** is a type of sorting algorithms that handles large-scale data which does not fit into the memory. As illustrated in Fig. 2, the traditional external sorting consists of two phases: Partial sorting and Merge. During the Partial sorting phase, the input data is divided into chunks whose size is smaller than the available memory. The data of each chunk is sorted using an in-memory sorting algorithm and the sorted data of the chunk is written to the storage (Step 1). In the Merge phase, the partially sorted chunks are read from the storage and merged to produce the final sorted data (Step 2). The final sorted data is written into the storage (Step 3), and usually sent to the other task later for subsequent data processing (Step 4).

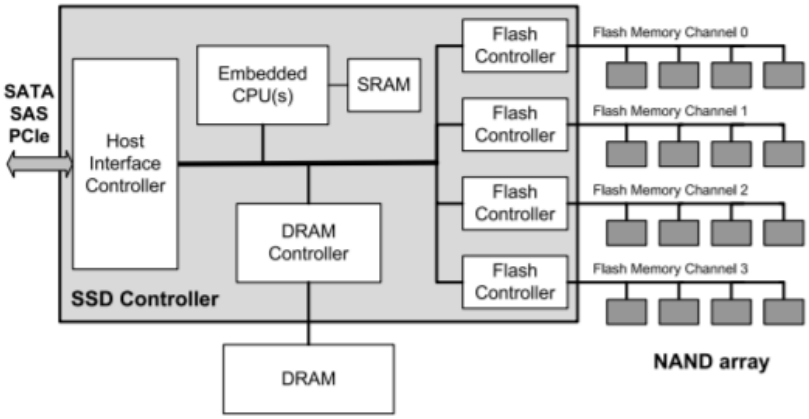


Fig. 1. The architecture of SSD.

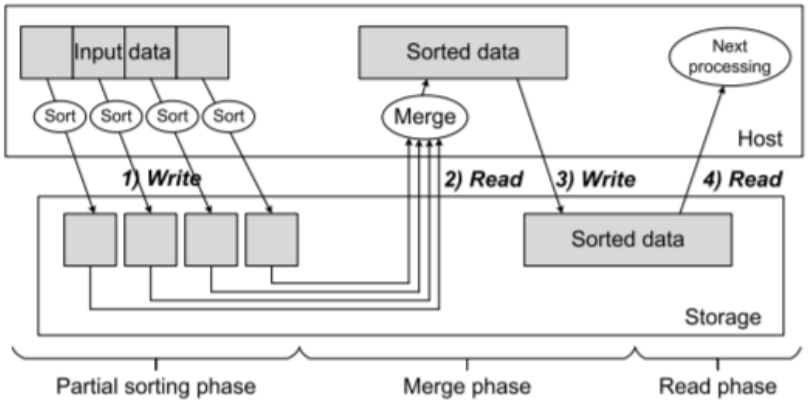


Fig. 2. The process of the traditional external sorting.

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

External sorting is a core function to execute MapReduce applications in Hadoop. In the map task, the split, which is the input data of a map task, can be larger than the available memory size. In this case, the intermediate results produced by applying the map function to the input data are sorted and then stored in the local disk as spill files. This procedure corresponds to the Partial sorting phase in the external sorting. After processing all the input data of the map task, the spill files are merged to pass the sorted data to the reduce tasks. This step corresponds to the Merge phase in the external sorting. The reduce task also executes external sorting to group the records by key in the reduce function. However, for the reduce task, only the Merge phase of the external sorting is required since the intermediate data is already sorted by the map task.

Since sorting is an essential ingredient of most external memory algorithms, considerable work has been invested in finding I/O-optimal parallel disk sorting algorithms (e. g., [5]) that approach the lower bound of 2N/DB(1 + dlogM/B N/Me) I/O operations for sorting on a machine with D disks. The challenge is to avoid getting a base M/DB for the logarithm that spoils performance for very large systems. An important motivation for this paper is the observation that a large D only makes sense in a system with many processors. Although [7], [6], [1] develop sophisticated asymptotically optimal parallel algorithms, these algorithms imply considerable constant factors of overhead with respect to both I/Os and communication compared to the best randomized sequential algorithms [5], [6]. We only note in passing that there is also considerable work on parallel disk sorting with shared-memory parallel processors, e. g., [10]. This is an easier problem since communication overhead is less of an issue.

Approach:

- **Parallel-K-Means:** In this project I implemented a parallel C++ algorithm using MPI (Message Passing Interface) classes. The algorithm to have been implemented is a simple machine learning algorithm to divide given data points into groups according to a distance measure. In this project I used 2D Cartesian points and Euclidean distances.

  Method:
  1. Assign the data point to appropriate cluster center.
  2. Compute the Intra cluster Maximum distance from all clusters.
  3. Calculate the Maximum value from among these distances.
  4. Find the Inter Cluster Minimum distance from all clusters.
  5. Calculate the Minimum value from among these distances.
  6. Apply the Dunn's Equation to get the optimum number of clusters.

- **External Sorting:** In this project I implemented the external sorting in C++ using the MinHeap approach. In this approach all the inputs form the file are constructed in a MinHeap array and then sorted according in ascending order. While sorting the program generates the temporary files or chunks to store the sorted data in parts. Temporary file is then merged into a single output file and all the temporary files are deleted then.

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

One example of external sorting is the external merge sort algorithm, which sorts chunks that each fit in RAM, then merges the sorted chunks together. We first divide the file into runs such that the size of a run is small enough to fit into main memory. Then sort each run in main memory using merge sort sorting algorithm. Finally merge the resulting runs together into successively bigger runs, until the file is sorted.

Method:

Let's consider a specific example and see how we can use external sorting to sort it. One of the popular techniques of external sorting is the external merge sort algorithm. In this procedure, we break down the data into smaller chunks that can fit into the RAM. We then merge the sorted chunks together. Let's say we have only 2 GB of RAM and we want to sort 8 GB of data. We have 8 GB of data in main memory and we want to sort it using just 2 GB of RAM. We use the following procedure:

1. Read the first 2 GB of data from the main memory into the RAM.
2. Sort the data in the RAM using quick sort.
3. Write the sorted data back to the main memory.
4. Repeat steps 1-3 until all the 2 GB chunks are sorted. Now, we have 4 chunks (8GB / 2GB = 4 chunks) and we need to merge them into one single sorted output file.
5. Read the first 0.4 GB of each sorted chunk and put it into RAM. Since we have 4 chunks, 1.6 GB (4 x 0.4 GB = 1.6 GB) is occupied in the RAM. We need the remaining 0.4 GB in the RAM as an output buffer.
6. We now need to perform a 4-way merge and store the result in the output buffer. We just keep adding data into the output buffer and whenever the output buffer fills, we will write it to the final sorted file. Once the data is written out, we will purge the output buffer.
7. As we keep merging and writing into the output buffer, our input chunks (4 chunks, 0.4 GB each) in the RAM can get empty. If that happens, we need to fill it with the next 0.4 GB of its associated 2 GB sorted chunk from the main memory.
8. We need to keep doing this until no more data from the chunks is available. This is an important step in this external sorting procedure and this is the step that makes external merge sort work "externally". The good thing about this merge algorithm is that it makes only one pass. sequentially through each of the chunks. As we can see here, each chunk does not have to be loaded completely. We just load them sequentially as and when needed.

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

Design and Implementation:

- **Parallel-K-Means Implementation:**
  Programming Interface – Input & Output.
  Both compilation & run commands for Linux terminal environment are those;
  which can be compiled as;
  mpicc <Filename>.c -o <Executable File>

  And the resulting executable file can be run as;

  mpiexec -n <num_of_processors>. /<name_for_the_executable_file> <Number of Cluster><input_file> <output_file>

  Where the input file is a text file with the format:
  NUM_CLUSTERS
  NUM_DATA_POINTS
  POINT_1_X, POINT_1_Y

  POINT_2_X, POINT_2_Y
  ...
  ...
  …
  POINT_2_N, POINT_N_Y

  And the output file is another text file with the format:
  NUM_CLUSTERS

  NUM_DATA_POINTS

  CENTROID_1_X, CENTROID_1_Y

  CENTROID_2_X, CENTROID_2_Y
  ...
  ...
  …
  CENTROID_k_X, CENTROID_k_Y

  POINT_1_X, POINT_1_Y, CLUSTER_VALUE

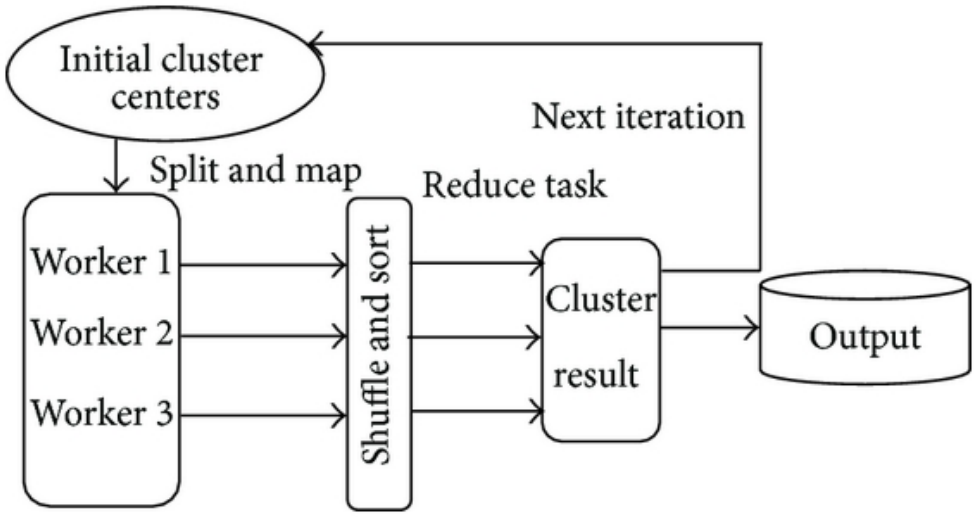
  POINT_2_X, POINT_2_Y, CLUSTER_VALUE

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

...
...
….
POINT_2_N, POINT_N_Y, CLUSTER_VALUE

Parallel K-Means Simulation



- **External Sorting Implementation:**
  Inputs:
  input_file: Name of input file. input.txt

  output_file: Name of output file, output.txt

  run_size: Size of a run (can fit in RAM)

  num_ways: Number of runs to be merged

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

Output:

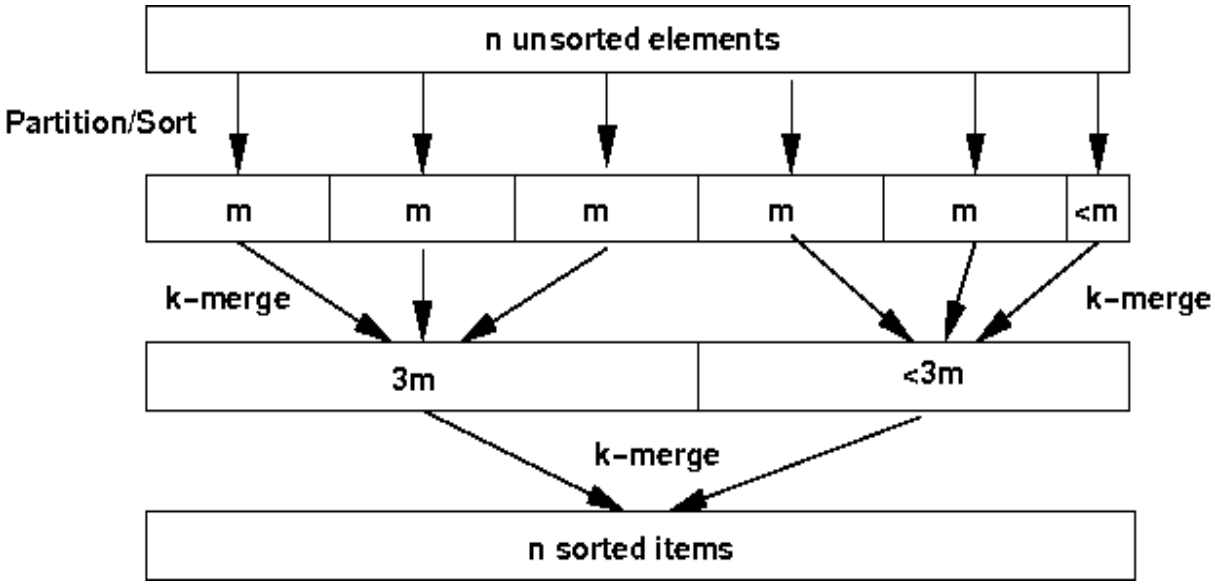
1. Read input_file such that at most 'run_size' elements are read at a time. Do following for every run read in an array.
   a) Sort the run using Merge Sort.
   b) Store the sorted run in a temporary file, say 'i' for i'th run.

2. Merge the sorted files using the approach discussed in a Single Output file.

3. Delete the temp files from the hard disk as it was created due to the partition of the input size.

4. Stores the merged output in a single file on the disk.
   For Execution:

   gcc ExternalSort.cpp -o Externalsort
   ./Externalsrort Input.txt Output.txt



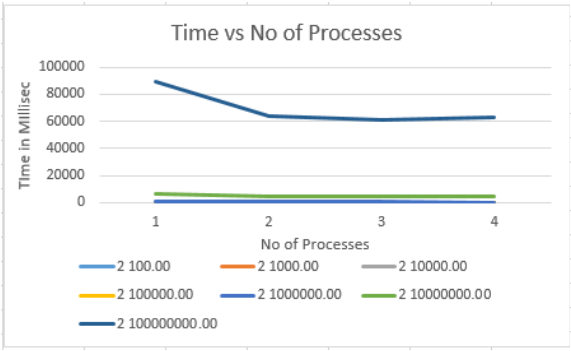
CS597-Reading and Special Problems

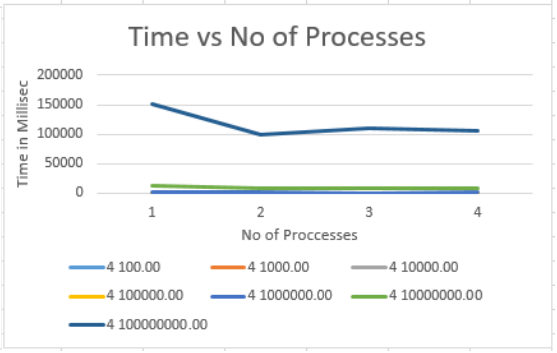
Parallel K-Means and External Sorting Implementation

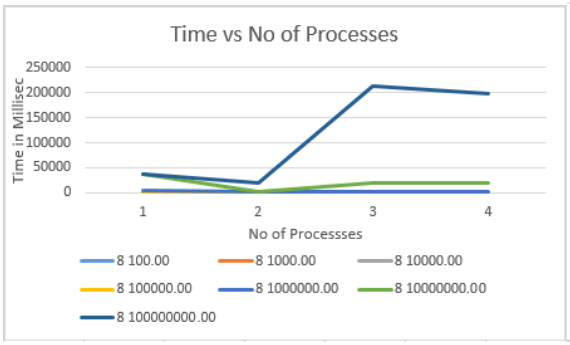
Evaluation:

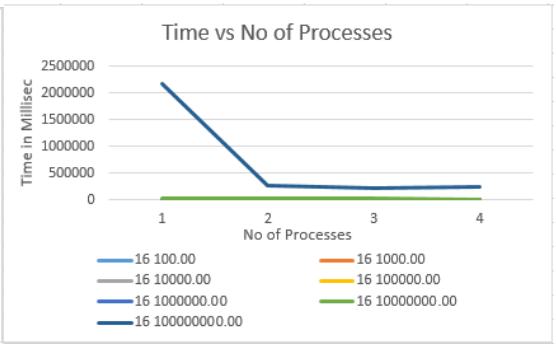
- **Parallel-K means:**

The below graphs show the comparison between the no of processes with number of data points with respect to time.







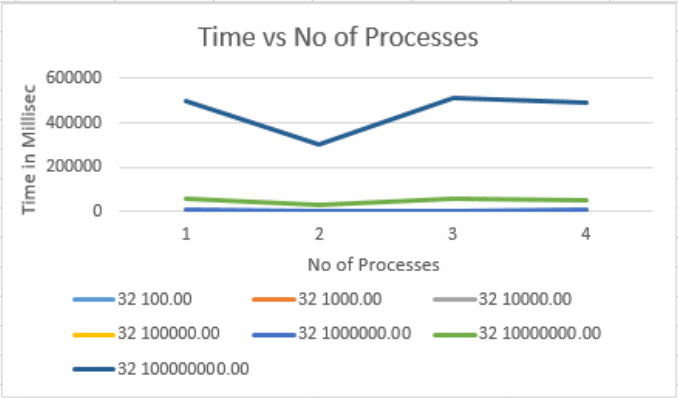


Data recoded for number of points with number of clusters with 4 processes:

With 4 processes time in Milli Secs					
No. of Points	**Clusters**				
	2	**4**	**8**	**16**	**32**
100	84	116	109	90	294
1000	82	116	68	117	362
10000	91	348	88	166	156
100000	135	191	284	412	717
1000000	426	1662	1422	2048	6388
10000000	4537	8893	19422	16456	50842
100000000	62744	106000	198494	239276	493807

CS597-Reading and Special Problems

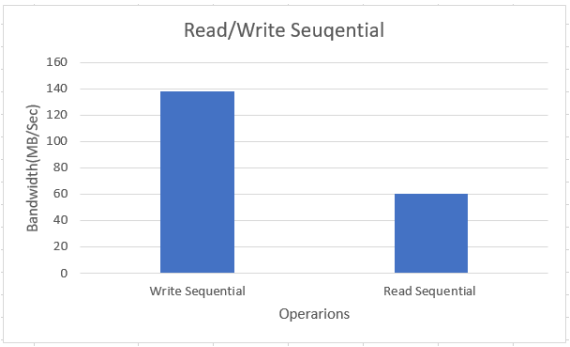
Parallel K-Means and External Sorting Implementation

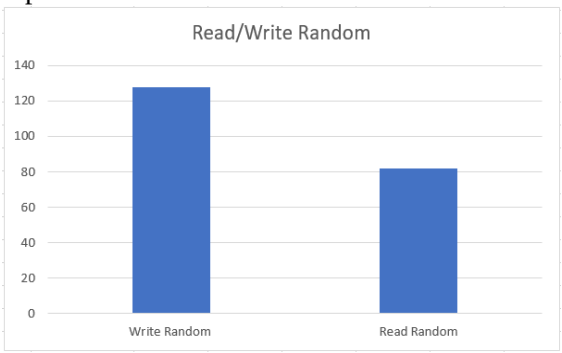


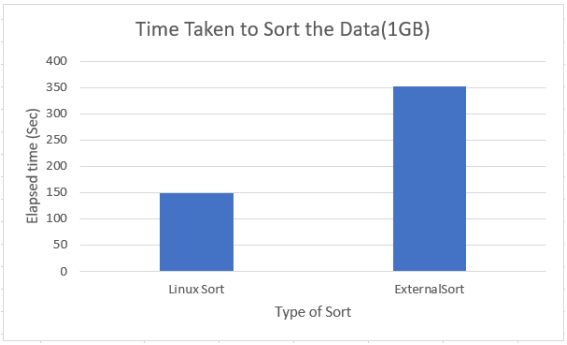
By looking at the above graphs and data we can say that as the number of process increases the the time taken by the program increases for clustering the data. The data is tested on the 4 virtual core processor machine and with the increase number of hardware threads with MPI the time taken to form the cluster decreases but a saturation is reached as the machine power is limited.

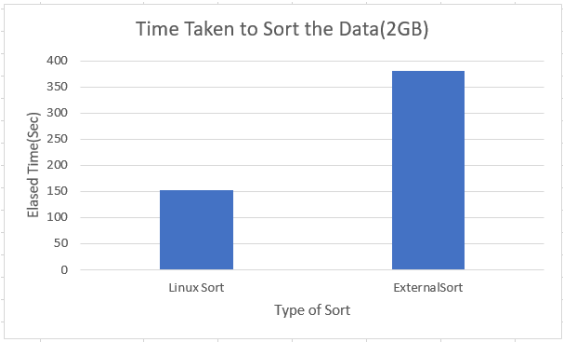
- **External Sorting:**

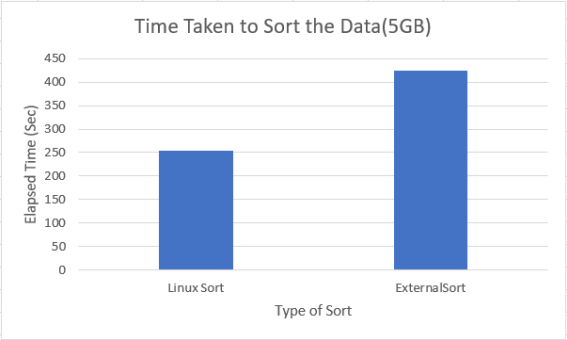
Below graphs represents the Read/Write analysis operations on different sizes of data.











By looking at the above graphs and comparing the Linux inbuilt and External sort we can surely say that Linux inbuilt sort does take lesser time than external sort algorithm which is implemented in the project. The external sorting algorithm is implemented using Min-Heap algorithm which takes longer time to sort the large amount of data.

Conclusion and Future work:

- **Parallel-K means:** It was a huge experience gained as developing for parallel systems. Also blocking-nonblocking messaging, broadcasting etc. are good topics to dive in, that I understood while making this project. Parallelization is a hard but joyful process, in my opinion.

  The program fails to make partitions with unequal sizes. For example, 100 points cannot be partitioned to three processors with this source code (33+33+34). So main improvement can be done in this point. Also, information messages are printed to STDOUT, but in printing synchronizations problems are exist. And printing order is not fully established. This could be another good improvement to current development.
  At early stage of the development, I used structures to message passing to processors, but this heavy traffic of data makes the program execution unmanageable as a debugger. So, I reduced my message traffic as using predefined data types. Also, first I tried to use broadcasting system of MPI, namely MPI_BCast, but its unblocking behavior also made the program code untraceable in long run. In final source code, only MPI_Send & MPI_Recv are used, but if MPI_BCast can be implemented to this system, performance gain for sending the common data as a tree traversal can be achieved.

  Future work can focus on how to reduce the time complexity without compromising cluster quality and optimality. More experiments can be conducted with natural datasets with different features. use some more powerful parallel programming models like Intel's Cilkplus and OpenMP to obtain reduced execution time.

CS597-Reading and Special Problems

Parallel K-Means and External Sorting Implementation

External Sorting: The globally striped algorithm minimizes the required I/O's to the optimum. MinHeap Merge Sort is theoretically a bit less scalable but it has close to minimal communication overhead, and a more useful output format. For medium-sized machines or average case inputs, the I/O requirement remains closer to two passes than three passes. In this paper, we have evaluated the Self-Sorting SSD architecture which completely offloads sorting operations into SSDs by creating indexes at write time or on demand as required. The evaluation results show that the proposed scheme can completely eliminate write operations from the external sorting process, improving performance and the lifetime of SSDs.

External merge sort minimizes disk I/O cost Produces sorted runs of size B (# buffer pages). Later passes: merge runs.Number of runs merged at a time depends on B, and block size. Larger block size means less I/O cost per page. Larger block size means smaller # runs merged. In practice,of runs rarely more than 2 or 3.

Acknowledgement:

This work is supported by Dr.Xian-he Sun, Anthony Kougkas, PhD Student and Hariharan Devarajan(PhD Student).

References:

[1] N. K. Ratha, A. K. Jain, and M. J. Chung,"Clustering using a coarse-grained parallel Genetic Algorithm: A Preliminary Study", Proceedings of the 1995 Computer Architectures for Machine Perception, 1995, pp. 331-338.

[2] C. Chinrungrueng and C. H. Sequin, "Optimal Adaptive K-Means Algorithm with Dynamic Adjustment of Learning Rate", IEEE Transaction on Neural Networks, January 1995, pp. 157-169.

[3] P. B. Callahan and S. R. Kosaraju, "A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields", Proceedings of the 24th Annual ACM Symposium on Theory of Computing, 1992, pp. 546

[4] D. Knuth, The Art of Computer Programming, second ed., vol. 3, AddisonWesley, 1998

[5] D. A. Hutchinson, P. Sanders, and J. S. Vitter. Duality between prefetching and queued writing with parallel disks. SIAM Journal on Computing, 34(6):1443–1463, 2005.

[6] M. H. Nodine and J. S. Vitter. Deterministic distribution sort in shared and distributed memory multiprocessors. In 5th ACM Symposium on Parallel Algorithms and Architectures, pages 120–129, 1993.

[7] J. S. Vitter and E. A. M. Shriver. Algorithms for parallel memory, I/II. Algorithmica, 12(2/3):110–169, 1994.

[8] 2. Malwindersingh, Meenakshibansal ,î A Survey on Various K- Means algorithms for Clusteringî, IJCSNS International Journal of Computer Science and Network Security, VOL.15 No.6, June 2015

http://www.mcs.anl.gov/~thakur/papers/discs-15.pdf

http://www.mnkjournals.com/ijlrst_files/Download/Vol%205,%20Issue%204/3-4-13072016%20A%20REVIEW%20ON%20K-MEANS%20CLUSTERING.pdf