Comparison and Simulation of External-Sort on a huge dataset

Hitesh Kumar Dasika,Graduate Student
School of Informatics and Computing

Jayendra Khandare, Graduate Student
School of Informatics and Computing

December 6, 2016

1 Introduction

It goes without saying that sorting plays an important task in most of the computer mechanisms. Most importantly, sorting algorithms can be used to solve other problems like counting duplicates, deciding rankings, finding medians in a dataset and many other problems. Commercially talking sorting can be used for tasks like event- driven simulations, searching for some specific information, numerical computations. Various sorting algorithms are in place to handle sorting tasks for a limited size of datasets.

Outline When the size of dataset exceeds the RAM capacity of a specific machine, external sort algorithms should be used. Example: If you want to sort a dataset of 500 GB with a machine which has 12 GB of RAM (Random Access Memory), you cant do that using basic sorting algorithms like insertion sort, selection sort. Because sometimes you need to access all the data for a specific comparison in some algorithms. You will be required to use an altogether different approach. You will be required to make modifications in the general programming approach as well as the sorting algorithm you want to use.

When a data set is too large to fit in internal memory, it is stored in external memory (EM) on one or more magnetic disks. EM algorithms explicitly control data placement and transfer, thus it is important for algorithm designers to have a simple but reasonably accurate model of the memory systems characteristics.Because I/O is done in units of blocks, algorithms can run considerably faster when the pattern of memory accesses exhibit locality of reference as opposed to a uniformly random distribution. However, even if an application can structure its pattern of memory accesses and exploit locality, there is still a substantial access gap between internal and external memory performance. In fact the access gap is growing, since the latency and bandwidth of memory chips are improving more quickly than those of disks. Use of parallel processors (or multi cores) further widens the gap.

2 Diving In

Now a days if there is huge amount of data to be stored,it is done using RAID (Redundant Array of Independent disks)approach. Hence data is divided according to raid level and stored in different disks

To talk about external sorting, let us understand the factors that can affect the performance of an efficient external sorting algorithm.One is locality of reference and the other is parallel disk access. Let us take an example of a single I/O operation.If there are D disk hard drives,each disk can give B blocks of data items and if the size of Internal Memory is M, then number of blocks of data that can fit in M is B/M. Hence it is not possible to sort data from D*B blocks in a single stretch. This requires us to find ways to sort such a huge amount of data.The sorting power also depends on number of CPUs used and is it possible to perform parallel processing using those CPUs.

Most external sorts as said above bank on locality of reference i.e. the tendency to refer nearby items instead of looking for data items which are scattered elsewhere. To achieve this,a fast internal sorting algorithm which has good locality of reference can be used.If there is a decent amount of virtual memory present, then it becomes easy to perform such kind of internal sort,else it will be difficult to accommodate huge data into that virtual memory and work on it.

Coming back to locality of reference,to explain why locality of reference will help in faster sorting, let us take the example of Quicksort. Quicksort, Mergesort and Heapsort have O(n*log(n)) as average case time complexity. Having that mind, it becomes difficult to choose one out of those. Generally we talk about other factors like implementation details, cache performance etc and then choose. But Quicksort is proven to have better locality of reference than the above said algorithms.One of the minor reasons why Quicksort has better is the innermost loop of operations are simpler.On top of that, in partitioning step then elements are rearranged around the pivot element i.e. all the elements in that same contiguous memory allocation which is an array of elements is sorted.Coming to Mergesort, as it maintains an auxillary buffer in merge phase it has to pay for the extra memory access which is bit scattered. Hence quicksort is preferred when a fast internal algorithm is required to sort pretty sizable data.

Most external sort algorithms which need to sort huge amounts of data rely in merge sort.They typically break a large data file into number of shorter, sorted runs.These can be typically done by taking some chunk of data which can fit into the RAM fo the system, sort that data using Quicksort and write it back to the disk.After all the data has been sorted chunk by chunk, then a merge algorithm is applied to merge the small chunks into the big file of data. The simplest form is the two-way external sorting algorithm.If two-way external sorting algorithm is understood then it becomes easy to understand other external sorting algorithms.

Two-Way External Sorting Before proceeding further, some points need to be clarified to understand the below concept.Data will be huge and hence at a time only some pages of data are accommodated in virtual memory.Keeping that mind,lets see two way external sorting.

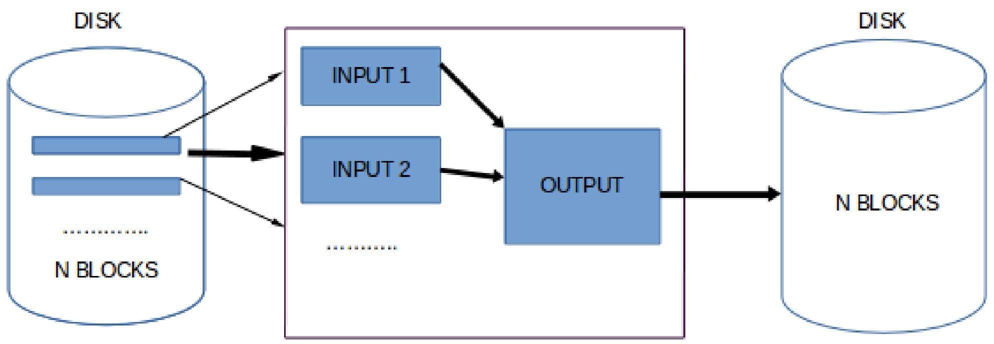

Figure 1: Two-Way Merge Sort Pictorial Representation

It involves two phases of operations.Figure 1 describes it
Phase-1: Read a page of data at a time and sort that data and write it back to the memory. In this process we use one buffer page of memory.This process is called runs or 1-page runs.
Phase-2: Data is read many times i.e. in many phases to perform merges. Every time a new pass is made, number of pages in the runs are increased.

3 Implementation

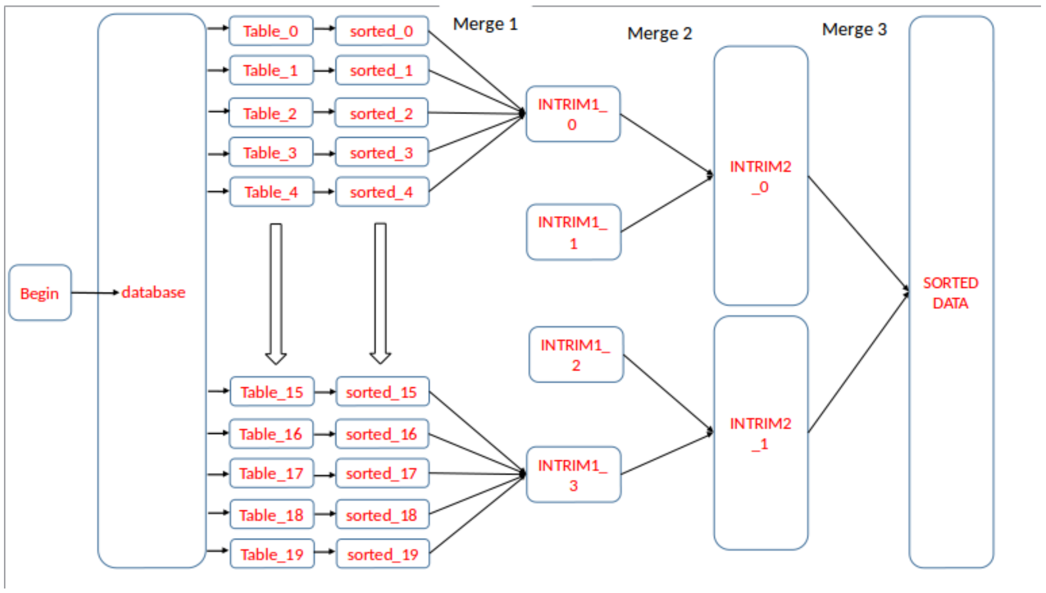

Figure 2: Five-Way Merge Sort Schematic Implementation Details

We tried to implement the external sorting mechanism by considering a 5-way merge sort technique in this project. There are five steps involved in this whole process
Let the steps involved be named as follows
CREATE
POPULATE
SORT
MERGE
In **CREATE** phase,a database was created using postgres and data which

is randomly generated is inserted into the table.
In **POPULATE** phase, created database is populated with the random amount of data.
In **SORT** phase, data is divided into 20 chunks or 20 tables and each table is individually sorted. For this sorting, any efficient internal sort algorithm can be used. As Quicksort works best because of the concept of locality of reference as discussed above, it was chosen.
In **MERGE** phase, 5 sorted tables are taken at a time and are merged and generated tables are called INTRM1.In the next phase, generated 4 tables which are called INTRM2 are merged to form the sorted data

4 Analysis

The dataset which we considered here contained 10000000 integers. To sort this using internal sort is not feasible at all as the RAM will not be sufficient and such huge processing is not possible and also the stack frames for the program will not be supportive to carry that huge amounts of data. Hence the above mentioned sort algorithm is applied for this dataset.

Analyzing the time complexity of it, partitioning phase takes O(log(n)) time. As we are making 20 partitions here, it will be the same. Instead of performing a 20way merging, here 5-way merge is done for four times. To sort the data in each intermediate dataset, we can use Quicksort which has a time complexity of O(n*log(n)) in average case.Hence if there are n elements in the dataset, we are making n/20 blocks of datasets in the initial stage and as 5 way merge is used we need 5 partitions of data and hence each partition contains k/4 data elements.After the first pass we get 4 new partitions of data which we again merge by considering two datasets at a time. Hence finally, we have achieved what is required by merging the two intermediate datasets to one dataset which is the sorted dataset.

5 Tools used

To develop this project, we used python as the programming language to simulate the sorting techniques.
To store huge amounts of data, postgresql database was used and different

tables have been created in it for this purpose.
There are two flavors of this project. One is where data is stored in database and the other is where data is stored in file system.My teammate has implemented this using filesystem and as told above database was used here.

The difference which was found is that using database was a bit faster because we where able to index the data in the database and hence accessing the data was a bit faster. On the other hand in some cases, as database is another external application, we were not able to guarantee the performance of it when huge amounts of data has been read from it. This can be clearly observed in cases when partitioning data is happening as 20 different tables were created and read.

6 Results

Below is a table indicating the time taken at each phase of the external sort.SIZE indicates the number of elements in the dataset.
Due to different constraints, internal sorting algorithms cannot be applied to datasets with size greater than 1000000. Even if we apply these internal sort techniques to such datasets, the time taken will be very huge compared to what it takes for external sort.In the below mentioned tables, **time unit is seconds.**
Important Consideration:
These tests are preformed on a system with 12GB of RAM and Intel i5 architecture

SIZE	CREATE	POPULATE	SORT	MERGE
20	0.002005100	0.018048048	0.083221673	0.002004861
100	0.002006769	0.017045736	0.083221197	0.004011154
1000	0.011032342	0.034087181	0.148390769	0.027076244
10000	0.103252410	0.066176414	0.687828540	0.310827016
100000	1.096916675	0.473274707	5.889664173	1.870831489
1000000	7.230201959	3.349907875	52.89565730	24.95134806
10000000	74.765896558	35.807329416	688.74159669	269.60196828

References

[1] Jeffrey Scott Vitter.*Algorithms and Data Structures for External Memory*. Purdue University, West Lafayette.

[2] Ishwari Singh Rajput, Bhawnesh Kumar , Bhawnesh Kumar *Performance Comparison of Sequential Quick Sort and Parallel Quick Sort Algorithm.* J.P Institute of Engineering and Technology, Meerut U.P., India

[3] External Sorting in geeksforgeeks,
`http://www.geeksforgeeks.org/external-sorting/`

[4] Pictorial Representation of External Sorting,
`http://www.cs.cornell.edu/courses/cs432/2003fa/slides/ExternalSorting.pdf`

[5] Explanation on locality of reference
`http://stackoverflow.com/questions/4289024/why-is-quicksort-faster-in-average-t`

[6] Example on External Sort
`http://cis.stvincent.edu/html/tutorials/swd/extsort/extsort.html`