Sorting huge Number of Integers from hard disk

Given 100 GB integer Data on Hard Disk with RAM amounting to 2 GB, how to sort the integers with minimal disk operation. Here fetching one number from disk is considered as one disk operation(though in reality a block of data can be fetched).

We can use additional space on the disk for temporary storage and no need to consider the operations of cleaning up temporary spaces used.

algorithm

asked Oct 25 '10 at 7:16

Shamim Hafiz
8,769 14 57 115

is this a kind of Homework? and put some code that you have tried ? – FosterZ Oct 25 '10 at 7:21

possible duplicate of How to sort 100GB worth of strings. – Gabe Oct 25 '10 at 7:36

1 See also stackoverflow.com/questions/134158/… and stackoverflow.com/questions/3961245/… – Gabe Oct 25 '10 at 7:51

@FosterZ: No this isn't homework :) – Shamim Hafiz Oct 25 '10 at 8:02

2 How big are the integers? – Gabe Oct 25 '10 at 8:11

show **1** more comment

6 Answers

As other people have noted, you can use an O(n) counting sort. However there are some additional problems you need to consider. We'll assume you're storing 32-bit integers, so 100GB ~ 27e9 ints.

If all the integers are the same, then it will occur ~27e9 times, which is larger than a 32 bit int. Therefore your counters will have to be 64-bit integers.

With 2GB of RAM, you can only store ~125e6 counters in RAM at any one time. If we can't make any assumptions about the distribution of integers, we would either have to:

- individually increment the counters on the HD, or
- ignore all integers we encounter that are not in the counter array we currently have stored in RAM.

I think the latter option is better. Since we need ~4e9 64-bit counters and can only store 2GB, we would need to run through the entire array ~16 times. The first option is clearly no good if we consider encountering a sequence of integers such as 0,1<<31,0. These counters would not be stored in RAM at the same time, and thus at least 2 HD writes are required.

Because of this, I think for the specific size of your problem (100GB), an N-way merge sort would be better, as this would only require reading the entire array log_2(100) ~ 8 times.

However, if the interviewer immediately changed the question to "10TB array, still 2GB of RAM", then the counting sort would easily win.

edited Oct 26 '10 at 19:59

answered Oct 26 '10 at 14:21

Dijkstra
1,078 2 9 28

What happens if the array is more than 16 Exabyte? – Sarp Kaya Apr 3 '14 at 6:05

add a comment

Since the data being sorted is integer type (4 bytes) and the amount of data is 100 GB (where GB is 2^30), you'd have 26,843,545,600 integers to sort. Since you have 4,294,967,296 possible integer values, you can represent this data as an array of longs which serve as counters, which would consume about 34 GB of disk space. Read through the 100 GB of data once, incrementing the individual counters for each possible integer value (300 GB total disk access to read the value, read the counter, write the incremented counter), then read the counters in order, writing out the number of values that you read of each value (134 GB total disk access).

This would sort the data using a total of 434 GB of disk access. If you use RAM to store part of the range of integer value counters, you could technically lower the amt of disk access even more.

edited Oct 25 '10 at 16:59

answered Oct 25 '10 at 16:54
Mark Synowiec
3,139 10 15

This looks to be a nice answer. Read all elements and count them and write the results back. I guess counting sort is the way to go if we want to reduce disk access. – Shamim Hafiz Oct 25 '10 at 17:36

There are 2^32 32-bit integers, and 8 bytes in a long, so it would take *exactly* 32GB (where GB is 2^30) to store all the counters. However, each counter requires only 35 bits to store up to 26,843,545,600, so you need 2^32*35/8 bytes, or under 18GB to hold the counters. Furthermore, you can use your 2GB of RAM to cache frequently used counters, reducing your disk usage even more. – Gabe Oct 25 '10 at 19:11

@Gabe: Yes, keeping some values in memory would also improve the performance. Another possibility could be to actually keep the counters in memory until we reach a point where we won't be able to accommodate any more. In that case, we flush these and update the counters on the disk. – Shamim Hafiz Oct 26 '10 at 7:20

I'm afraid that in this case the disk access has to be measured as number of accesses and not as traffic (although the traffic could be also huge). HDDs has horrible seek times, and SSDs aren't used largely (and they have poor writing speed). – ruslik Oct 26 '10 at 7:43

2 I think this would be terribly slow if you have to read/write to the HD for every single integer. As Gabe said, caching would improve results, but if you've only got 2GB of RAM you would only halve the number of read/writes to HD (~5e10). I think that having a 2GB array of int64 counters in RAM (since we could potentially have the same number 26e9 times) and sweeping through the HD 100 times, ignoring those integers that are not within the array bounds, would be faster. Of course, if you know something about the distribution of integers you could improve further. – Dijkstra Oct 26 '10 at 10:38

show **5** more comments

I think that a for fast algorithm another 100GB of free HDD space are precondition.

Just use any sort on 2GB chunks and put them back. Now you have 50 sorted chunks in file, and you cand use the merge sort as suggested by Mihir on them. Write output buffer as it fills in the output file. You'll just have to fine-tune the input and output buffer sizes.

There are some solutions with counting. It cannot be used on such large range and maximum possible count. You could only store QWORD counters on disk, but this means many random accesses, that will certainly slower than working with larger buffers.

edited Oct 26 '10 at 7:38

answered Oct 25 '10 at 10:41
ruslik
8,712 15 29

The first answer that came to my mind is actually this one. But, now it seems that, counter based solutions are more convenient. – Shamim Hafiz Oct 26 '10 at 7:24

@Gunner but how? – ruslik Oct 26 '10 at 7:36

Look at the post by Mark Synowiec. – Shamim Hafiz Oct 26 '10 at 7:39

If you split the input into 50 equal size pieces, Merge Sort is a good answer. You get 400 GB total I/O - 100 GB to read the input, 100 GB to write the 50 files, 100 GB to read all 50 files back in, and 100 GB to produce the output. – Mark Ransom Oct 26 '10 at 22:25

@Gunner, since you never specified the size of the integers it seems premature to say a counting solution would be practical. Even if it is, the worst case performance will be dismal. – Mark Ransom Oct 26 '10 at 22:29

add a comment

To me the answer to this question depends cruically on the expected distribution of the numbers in the file.

There are 12.5 Billion ints in 100 Gigs of int data. There are also only ~4.3 Billion distinct ints.

Given a perfectly uniform distribution across all possible ints you'd expect each int to show up roughly 3 times give or take. This low level of duplication does not warrant changing from a standard sort routine (one that sorts chunks at a time and then merges the chunks together).

However, if we restrict the "file ints" to all be non-negative then we immediately expect each valid int to appear roughly 6 times. This is approaching a level of duplication that may warrant a change in sorting routine. So, I think you should ask the interviewer if anything more can be assumed about the distribution of ints on disk. After all, it would be odd to have 100GB worth of data and have no idea if it exhibits any predictable pattern.

answered Oct 25 '10 at 14:46

Ivan
473 3 14

This is actually interview question and the interviewer is probably interested in seeing how one approaches the problem. Therefore, we should not expect some kind of pattern in the data. On the issue anyway, I wonder if there is ever a need to sort such a huge amount of data in real life. – Shamim Hafiz Oct 25 '10 at 16:19

Yeah, I get that you wrote out the actual interview question. But you should ask in the interview if the numbers in the file come from one distribution or another. Because having that knowledge (or not) has dramatic implications -- you should show that you realize that. – Ivan Oct 25 '10 at 17:54

Good point, most interviewers actually expect the interviewee to ask few clarification questions. Those signify a lot on how the person thinks and deals with problems presented. – Shamim Hafiz Oct 26 '10 at 7:22

I thought about that, but counting is good only when range is restricted. It should be able to handle any input, and even size of DWORD isn't enough to store the max possible count. – ruslik Oct 26 '10 at 7:35

add a comment

Merge Sort is a popular approach when it comes to limited memory

answered Oct 25 '10 at 7:30

Mihir Mathuria
3,240 12 12

1 Actually no, merger sort's main disadvantage over quicksort and heapsort is requirement of additional memory, which roughly equals to twice as much as source data memory. – Shamim Hafiz Oct 26 '10 at 7:25

1 Gunner, actually, no :) you are probably thinking about main memory sorting algorithms, which is not the discussion here. – Dimitris Andreou Oct 26 '10 at 15:00

add a comment

100GB of integer data means you'll have a large number of duplicate data. I'd personally choose a (bucketsort/selection) / mergesort approach as my first instinct if I'm trying to minimize disk I/O.

First read a bit under 1 Gig of data into memory, mergesort that data in memory. Flush to disk. Repeat for each chunk of memory. Then you can walk each chunk of data and grab all the 0s, repeat for each integer. It's going to take a long time, but that's only 203GB Read and 200GB write worst case (theoretical).

answered Oct 25 '10 at 7:30

OmnipotentEntity
6,582 1 22 57

If you have 2GB of RAM, why only read 1GB at a time? – Gabe Oct 25 '10 at 7:34

Merge sort requires O(n) additional memory. – OmnipotentEntity Oct 25 '10 at 7:36

You're trying to minimize disk operations and have no limit on CPU operations. You can do a merge in O(1) additional space if you do your merge in O(n^2) CPU time. Personally, though, I would just read in 2GB and QuickSort it. – Gabe Oct 25 '10 at 7:46

That's true, and it does optimize the disk usage slightly in the selection sort phase. – OmnipotentEntity Oct 25 '10 at 7:50

1 I think the RAM based sorting doesn't matter much. We can use Heapsort with constant amount of additional memory use and no worst case scenario of quicksort. I choose heapsort, because it's inplace. – Shamim Hafiz Oct 25 '10 at 8:04

show **3** more comments

Not the answer you're looking for? Browse other questions tagged algorithm **or ask your own question.**