Sort FAQ (29 March 2018)

Common

Are there any requirements that are considered 'common knowledge' that may surprise novices ?
There is a "reasonable OS" requirement, since we are trying to test the IO subsystem. If you think the OS should do it, then it should. We are not OS guys, we are OS users. So, it should do what you would expect the OS to do if you were a DB guy or a graphics guy or a word processing guy.

Can we write our own file system? Can we define partitions, striping, etc. as we see fit?
This is a test of the OS file and IO system and the underlying hardware. If the OS or its extensions support partitioning among disks or anything else, cool!!!

Can we fix bugs in the OS or do we have to use available patches or service packs?
Yes, you can fix bugs, just make sure that the IO library still works as advertised. You are welcome to use patches as well

Can we use assembler?
Yes, but it is a bit surprising if you need to. This is a test of the OS IO system not of the compilers.

Can we use raw devices for temporary space?
No. This is a test of the file system, OS and hardware.

Does a sorting technique or method used by a previous winner automatically qualify for the current year's contest?
No. We sometimes change or clarify the rules from year to year. Previous winning algorithms or programs do not necessarily qualify for this year's contest. If in doubt, ask us.

Does all required sort functionality need to be implemented at the time of the entry?
Yes, claims that the sort program can be "easily modified" will not be accepted.

Input Data

How can I generate the sort input data?
Every entrant should use the same data generator *gensort*, or their own data generator based on the gensort code. This insures that all entrants use the same input records.
Usage: gensort [-a] #Records FileName
(Each record is 100 bytes long.)
JouleSort entrants are required to generate *ascii* records by specifying the "-a" argument to gensort. Ascii records have 95 possible values for each of the 10 key bytes - allowing for 95^{10} (59,873,693,923,837,890,625) possible key values.
GraySort, MinuteSort and CloudSort applicants must use the default binary records. Binary records have 256 possible values for each key byte - allowing for 2^{80} (1,208,925,819,614,629,174,706,176) possible key values.
Use the "-c" flag to have gensort calculate the checksum of the records it generates. You will need to insure that the checksum of the sort input matches the checksum of the sort output.

Can multiple input files be used?
For JouleSort, only one input file can be used.
For GraySort, MinuteSort and CloudSort categories multiple input files can be used. A concatenation of the input files must be the same as the single file that would be generated by gensort with same number of records. Input files that meet this requirement are simple to generate with gensort. See the gensort documentation for more information.

Can the data be compressed?
No. The data cannot be compressed at all; neither the input, temp, nor output files. Also, the data cannot be compressed when sent over the network. To keep in spirit of the original benchmark rules, we want 100 MB of data to generate 100 MB of IO (whether disk or network).

Can the input file be read from the file system's buffer cache in main memory?
We want the entire input file to be read from disk. If you are not using direct (unbuffered) IO, before you run the sort program you must first insure that the input file is not in the file system buffer cache (e.g. by reading a large, unrelated file as large as your system's main memory).

Can we control the placement of the file on the drive surface?
Yes, you can control placement (e.g. outer bands on the disk). You can place the file any way you like. I particular, you will probably want to defragment the disk and also place the file on the outer bands on the disk.

Sorting Process

Since the last 80 bytes of each 100-byte record generated by *gensort* can be derived from the first 20 bytes, can we sort the first 20 bytes and compute the rest to form the result file?
No, you must move all the bytes, since this is an IO benchmark. You must sort all of the data -- that's what the contest is all about. Internally, in RAM memory, you are allowed to use pointers to the data and then write out the data in order of the pointers.

In what order should the sorted file be?
For binary records (GraySort or MinuteSort), the 10-byte keys should be ordered as arrays of unsigned bytes. The memcmp() library routine can be used for this purpose.
For ascii records, there are three popular orders:

a. The ASCII binary order strncmp( ) or memcmp( ) : !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~ [To get this order, simply use strcmp( ) or memcmp( ).]
b. The case-sensitive order used by NTSort (the SORT command in Windows NT/2000/XP): '- !"#$%&()*,./:;?@[\]^_`{|}~+<=>0123456789AabBcCdDEeFfGghHIijJkKlLmMnNoOPpQqrRsStTuUvVwWxXYyZz
c. The strnicmp( ) order (NTSort default locale with "C", by using *sort /l "C" source /o dest*): !"#$%&'()*+,-./0123456789:;<=>?@[\]^_`AaBbCcdDeEFfGgHhiIjJKkLlMmnNoOpPqQrRSstTUuVvwWXxYyZz{|}~ [To get this order, use stricmp( ) or strnicmp( ).]

And then there are the various Unicode and code page orders. For the competition you can use any order you want in particular the binary order (a or c) is typically fastest. In the standard C library this is strncmp( ) or strnicmp( ). The validator uses memcmp for binary records and strncmp() for ascii.

Output Data

What is the required output format?
For JouleSort, the output must be a single file. Output file is a sorted permutation of the input file (same checksum). It should be a sequential file, as seen by fopen( ) and a linear seek or whatever other file manipulation verbs are native to your operating system.
For GraySort, MinuteSort, and CloudSort categories multiple output files can be created. In this case a concatenation of the output files must be the same as one ordered output file.

Can the output file already exist when the sort program is started?
No, it cannot. The output file must be created by the sort program.

Can the output file be written to the file system's buffer cache?
Yes, this is what happens if you are using normal (non-direct or unbuffered) writes. In this case your sort program should flush the output file to disk before exiting. The entire output file must be written to disk during the elapsed time measured for the sort.

Should the output file be on sequential sectors on the surface of a disk?
It needn't be. What we need is a sequential file that can be opened by the file system.

How can I deal with the temporary files? Must I delete them? What about the format on striped disk systems?
File space must be zeroed or the OS must have some other strategy to prevent unprivileged users from seeing your files after you delete them. This is a standard OS security issue. There are ways to avoid zeroing. Indeed this was a big performance improvement in some operating systems.

Can we leave the sorted file in two disks? I mean, each disk will have only 1/2 of the result, and only if I simply concatenate them together, it will be the final result.
You can have partitioned output files for the GraySort, MinuteSort, and CloudSort categories.
For JouleSort, input and output has to look like a single file (this is a file system test). So if you want to distribute the data file on different disks, you need use disk striping or RAID.

Can the sort output destroy (reuse) the sort input file?
For the Daytona sorts, destroying or overwriting the input file is never allowed.
For the Indy sorts, the input file can be overwritten or destroyed if there is not sufficient disk space to hold both the input and output files. However the sort must always create a new output file and allocate space for it. It can delete the input file after reading it and reclaim that space. That is at the end of phase 1 (when the runs have been generated) the program could delete the input file and then create the output file and merge the runs into that new file.

How can we verify the correctness of the output file(s)?
The program *valsort*, included with the *gensort* tar file, validates the order of the records in the output file using memcmp ( ). If you sorted the records using a different sorting order, you should modify *valsort* to use the appropriate key comparison.
The validation program also computes a checksum of the records and counts the number of duplicate keys. You should verify that the checksum of the output file(s) matches the checksum of the input file(s). You should report the checksum and number of duplicate keys as part of your contest entry.

GraySort

What is the GraySort Benchmark?
The goal of the benchmark is to measure the performance of very large sorts. It is named after Jim Gray, the author of the original sort benchmark and the sponsor of them until 2007.

Why is the metric *TBs sorted per minute*?
Back in 1994 when the MinuteSort benchmark was first conceived, Jim's goal for the benchmark was to see a terabyte sorted in a minute. The metric is a tribute to that vision.

Why is there a minimum size requirement of 100 TB? Why not just fix the size at 100 TB and measure the elapsed time?
We want to make sure the sort data set does not fit in main memory, so we require a minimum input size. Entries with larger input sizes are welcome. We plan to increase the minimum size requirement over time as system capacities increase. Using a speed metric of TBs sorted per minute will allow us to track the progress of benchmark winners over time. Even as the minimum changes, the name of this benchmark will remain GraySort.

JouleSort

Which meter should I use to measure power ?
There are a number of meters that can measure power of all kinds of systems. The SPEC committee lists some here. In the original JouleSort paper, we used the BrandElectronics Model 20-1850/CI. We have two requirements: (1) The meter should be fairly accurate, within +/- 1%. (2) One should be able to synchronize the power meter with the test run through a computer interface, e.g. serial port, USB, ethernet, etc.

How should I take and report the energy measurement ?
You need to synchronize the timing measurements with the power measurements, as mentioned above. Power should be sampled at least once a second; the samples should be equally spaced. Timing must be done externally as specified below. Energy is the product of average power (during the run) and time. Take five consecutive runs, and report the average energy and its standard deviation.

What are the rules for sorting ?
The same as for all the other sorts.

CloudSort

See the original CloudSort definition here.

Can I store the input and output on virtual machine disks, if there is sufficient redudnancy, e.g. HDFS uses 3x replication ?
No. In the spirit of earlier sort benchmarks, we want to model a scenario where users would expect their input and output data to persist across software faults. Unfortunately, on rare, low-level (kernel) software faults, VM disks are reset. These faults have occurred in production, and can be correlated, forcing vendors to reset entire clusters. On the other hand, contestants are free to use VM disks for intermediate runs.

To prorate a monthly storage cost, should I use the number of days in the month that I ran the test?
No. Always use 30 days per month to prorate a monthly storage cost.

What do you mean by billing granularity? What is the maximum allowed?
Cloud pricing is often quoted in terms of cost per unit resource per unit time for resources like compute and storage. For example, S3 costs $0.023 per GB per month for the first 50TB. Or, an m4.xlarge EC2 instance costs $0.192 per instance per hour. How pricing is quoted, whether it is per minute, per hour, per day, or per month, is different from its billing granularity.

Billing granularity is the increment in which a cloud resource is charged. For example, EC2 instances are priced by hour but billed per second. So, if an instance is run for 465 seconds, the customer is charged for only that use, not a full hour.

The requirement is that the *maximum* billing granularity for compute-based resources should be 1-day, but can be lower, e.g. per hour, per minute, or per second. Regardless of the billing granularity, when calculating the cost of the run, we will prorate it to the second.

Indy vs. Daytona

What is an Indy sort?
Much like the Indy classification of racing cars, an Indy sort can be custom made for the sort benchmark. It can assume and take advantage of

- the fixed key and record sizes
- the random, dense and uniformly distributed input keys - both for sorting records and for partitioning records into multiple output files

What is a Daytona sort?
Daytona sorts should

- be capable of sorting other record and key types besides 100-byte records with 10-byte random keys
- not significantly degrade in performance when sorting other key and record types
- not be overly dependent on the uniform and random distribution of key values in the sort input. If the sort data is to be divided into multiple partitions (for instance in a cluster sort), the sort should not rely on any predetermined partition boundaries. The partition boundaries must be determined either by sampling the input data or during the sort. If sampling is used, it should be done evenly across all input partitions. That is, the same number of records should be sampled from each input partition. The report should explain how the sampling was done and state its elapsed time.
  Similarly, non-uniform key distributions should not significantly increase network traffic in a cluster sort.
- not overwrite or destroy their input file
- be able the run continuously for one hour without a system failure. This requirement is specific to GraySort and MinuteSort benchmarks.
- use the same degree of replication for the input and output files. This is particularly important for the GraySort. For example, if the input resides on a clustered file system and the input files are replicated 3-fold, then one can take advantage of the replication to improve performance. But, the output files must also be replicated 3-fold.
- be able to sort the alternate, skewed-keys input data set in an elapsed time of no more than twice the elapsed time of the benchmark entry. Benchmark results are still based on the elapsed sort times using the normal (non-skewed) input data generated by the gensort program. The gensort "-s" command line option should be used to generate the alternate, skewed-keys input data set. The elapsed time for sorting the skewed input must be included in the entry report.
- be capable of reliably sorting data sets that are larger the collective main memories of the system. For cluster based sort systems that subdivide the input data into partitions each of which is then sorted by a single node, one of the following must be true:
  - the single-node sort can gracefully handle the situation where the partition size does not fit in the main memory its node; or
  - the partition sizes are guaranteed to fit in the main memory of a node. A high probability of this based on sampling a subset of the input keys is not a guarantee.
- for GraySort and MinuteSort, have persistent storage for both input and output data. With multi-node systems, the input and output data must be replicated across nodes.

Measuring the Sort

Can the sort program measure its own elapsed time?
No. We want to make sure all sort program setup and shutdown activity is included in the elapsed time. Use an external program or shell command such as the Unix *time* command.

How should we handle variances in the elapsed time from one run of the sort program to another?
You should report the median time of all your attempts with a particular input size. For MinuteSort, you should make at least 15 consecutive attempts and the median time should be 60.00 seconds or less.

How should we report the size of a sort data set?
In units of:

- gigabytes (10^9 bytes)
- terabytes (10^{12} bytes)

Since when is a gigabyte 10^9 bytes?
Clearly, a gigabyte of main memory is 2^{30} bytes. However, according to disk industry standards, a gigabyte of disk memory is 10^9 bytes. The sort benchmarks are disk-to-disk. Also, with the 100-byte records used in the sort benchmarks, it is much easier to measure in powers of 10. We used to accept

entries using either standard but it became tedious to compare different records (now did they use decimal or binary gigabytes?).
To simplify things, we use powers of 10. When the Unix *ls* or the Windows *dir* commands display file sizes in hexadecimal, we might reconsider.

Report Submission

What should be submitted?
A report describing the work and the system. You should also tell us, separate from the report, the checksum of the input and output file(s), and the number of duplicate keys in the output.

Whom should our results be submitted to?
Submit report to chris dot nyberg at ordinal dot com

Awards

What will we get from joining the competition?
You will have a chance to set a new world record, which evaluates the rapid progress has been making on hardware and software. If you win, your results will be listed on the Sort Benchmark homepage, and your report will be a good reference to the researchers world-wide who are interested in this field. If so, you'll be proud of your accomplishment, since it is rare in one's life to be the best-in-the-world in something, and to be recognized for it.

Are there any medals for the winners?
Unfortunately, this is no longer the case.