Tencent Sort

Jie Jiang*, Lixiong Zheng*, Junfeng Pu*, Xiong Cheng*, Chongqing Zhao*, Mark R Nutter**, Jeremy D Schaub**
*Tencent Corporation, China
**Technical Support

***Abstract*—Following the GraySort rules of sortbenchmark.org, this paper reports results of 100 TB in less than 100 sec (60.7 TB/min) for the "Indy" GraySort, 44.8 TB/min for the "Daytona" Graysort , 55.3 TB for the "Indy" MinuteSort, and 36.9 TB for the "Daytona" Minutesort benchmarks using a cluster of 512 OpenPOWER servers optimized for hyperscale data centers.**

***Index Terms*— sorting, distributed algorithms**

I. INTRODUCTION

The introduction of high-bandwidth NVMe solid-state storage devices, 100Gb Mellanox networking, combined with 160 hardware threads across two 10-core OpenPOWER (POWER8) CPUs, has allowed for a substantial step forward in cluster-level performance and an order of magnitude improvement in per-node sort performance. Disproportionate growth in network and storage bandwidth puts a new degree of pressure on the CPU, and while sort alone still utilizes less than 10%/70% (average/peak) of the CPU, managing the combination of sort, network and storage now demands a high-performance CPU to achieve maximum performance.

The sort application consists of three major phases: 1) reading from storage and partitioning the data into non-overlapping ranges according to the sort keys, 2) distributing the ranges to the destination nodes (shuffle), and 3) at the destination nodes combining equivalent ranges from all the nodes into a sorted output file. In the case of the Daytona sort there are a number of enhancements: 1) Input and output files are duplicated across the cluster. 2) The application can handle inputs larger than the combined memory in the cluster. 3) The application can handle arbitrarily skewed input data sets. 4) The application can handle a variety of key types and key and record lengths.

While tuned for the benchmark, the sort application is able to handle a variety of key and record lengths and a variety of sort orders. For all the results reported in this paper, the lexicographic ordering was used, as per the benchmark guidance. Tencent Sort supports a variable number of nodes. The application supports the sorting of skewed datasets as well as datasets that do not fit in the aggregate cluster memory. In order to support a variety of networking protocols, including those that do not guarantee delivery, network retry is handled within the application in a modular fashion. Input and/or output data can be recovered and the application restarted without data loss in the case of node failure.

II. SYSTEM CONFIGURATION

The system used for these benchmarks is a 512-node OpenPOWER cluster with a 100GbE Mellanox data network. Node attributes are summarized in *Table 1* (hardware) and *Table 2* (software).

CPU	**2x OpenPOWER 10-core POWER8 2.926 (2.06-3.49)GHz**
Memory	**16x 32GB 2RX4 (2133MHz) DDR4 RDIMM**
Data Storage	**4x Huawei ES3600P V3, 1.2TB NVMe SSD (RAID0)**
Program Storage	**240 GB SATA 3.0 SSD**
Network	**100Gb Mellanox ConnectX4-EN**

Table 1: Node Hardware Configuration (SuperMicro OpenPOWER SSP-6028UP-ENR4T)

OS	**Ubuntu 16.04 (Linux kernel 4.4)**
File System	**xfs**
Temporary Files	**tmpfs**
Application	**Python, C/OpenMP**

Table 2: Node Software Configuration

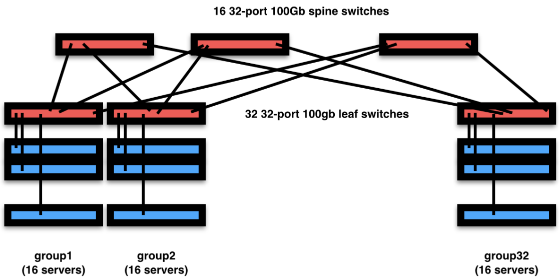


Figure 1: Full-bandwidth leaf-spine 100GbE Network. 512 OpenPOWER servers(blue), 48 Mellanox Spectrum SN2700 switches(red), Mellanox 100Gb LinkX optical cables between switches.

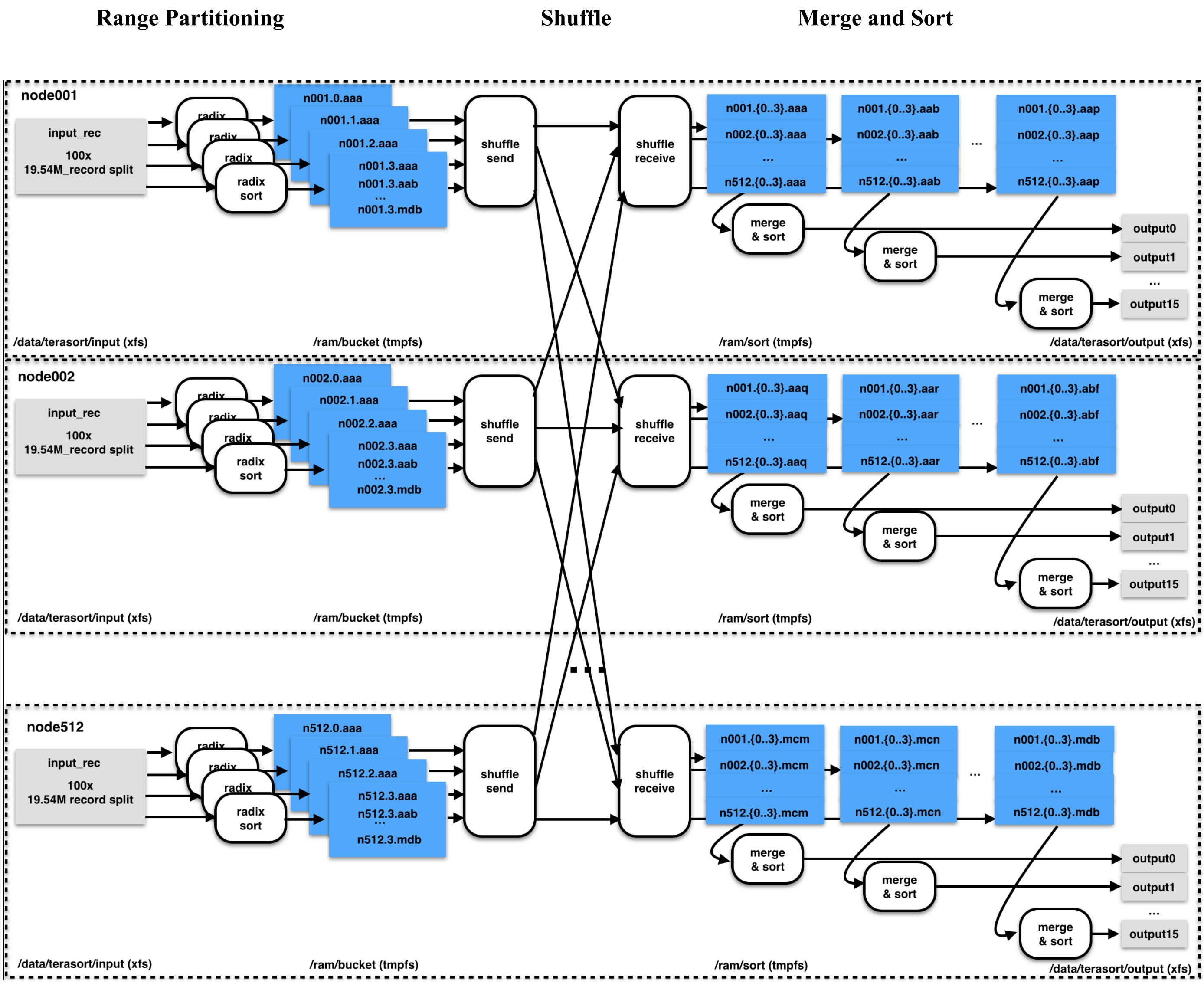


Figure 2: Indy Sort Software Architecture. Parameters shown are for the 100TB Indy GraySort benchmark.

III. INDY SORT

Indy sort consists of the following major phases shown in Figure 2:

A.Input Generation (not shown in Figure 2)
B.Range Partitioning
C.Network Shuffle
D.Merge & Sort
E.Output Validation (not shown in Figure 2)

Stages B-D comprise the timed Indy GraySort and MinuteSort application. At the top level these stages are controlled by the host node which issues commands to all other nodes and collects their responses to determine successful completion of the complete application. For the Indy sorts the range partitioning is completed before the other stages start. The shuffle and merge and sort phase are partially overlapped, with some of the sort and merge tasks starting before the shuffle completes. In our benchmark runs four of the merge and sort tasks typically operated concurrently.

A.Input Generation

Input data is generated on each node in the cluster prior to running the benchmark, using *gensort* with appropriate data offsets for each node. The input dataset (*gensort* output) for each node is split to multiple segments that are stored on the local xfs file system mounted on NVMe. To best balance NVMe read bandwidth, CPU throughput, and main memory size, a segment size of 19.54M records is selected.

For the 100TB GraySort benchmarks on a 512-node cluster, each node has 100 19.54M-record segments (195.4GB), for total of 51200 segments spread equally across the cluster (a bit more than 100TB total). For the MinuteSort benchmarks the number of segments is reduced and *gensort* offsets are adjusted accordingly.

B.Range Partitioning

During the first stage, each node reads the input dataset and partitions the data into a set of non-overlapping data ranges which are written to a memory-based tmpfs file system. A flexible partitioner is used that is a customized parallel (C/OpenMP) version of radix sort optimized to leverage the large number of threads and large caches of the POWER processor architecture. Output data ranges can be fully or partially sorted, depending on the number of radix sort iterations. For the benchmark configurations this stage only partitions.

For the 100TB benchmark, the first stage creates 8K non-overlapping ranges on each node (labeled "aaa" though "mdb"), such that in the second stage each of 512 nodes can concurrently process 16 such ranges (16x512=8K). In order to leverage the available NVMe bandwidth, 4 concurrent copies of the range partitioner are run on each node. Each node achieves a sustained read bandwidth of about 10GB/s in this stage (about 5TB/s in aggregate).

C.Network Shuffle

In the shuffle stage, the output files from the partitioner are communicated to the destination nodes such that each node collects an approximately equal number of ranges. In our 100TB run each range consists of 2K files (512 nodes x 4 partitioners) and if 512 nodes are available each node processes 16 such ranges (more if there are fewer nodes).

Successful file transfers are acknowledged and retry is handled within the shuffle routine. While the current shuffle is sockets-based, this approach allows for a variety of network protocols, including those that do not guarantee delivery, as is the case when ethernet packet retry is turned off.

So as not to overload the network, the number of simultaneous file transfers is controlled by a tuning parameter.

Maintaining a large number of network connections can lead to excessive context switching To limit context switching, a single thread in the shuffle handles multiple connections.

Mellanox ConnectX-4 100GbE NIC optimizations include enabling Large Send Offload (LSO), Large Receive Offload (LRO), and 64KB socket buffers to leverage LSO and LRO, using large packets (MTU 9000), and managing interrupt NUMA affinity. When the shuffle stage is run in isolation, per-node sustained throughput is close to 10GB/s.

D.Merge And Sort

The final stage integrates data ranges from multiple input sources and produces a final order across all keys within that range using a second sort routine, which is a customized parallel (C/OpenMP) version of a merge and sort optimized to best leverage cache and thread attributes of the POWER processor. As this operation has no dependencies across the ranges, multiple instances can operate in parallel, each producing their own output files. For the 100TB benchmarks, sixteen merge sorters per node are used, each producing one output file. Note that because each POWER8 processor core supports 8 (SMT) hardware threads, 160 threads are available to the application, and each sorter uses multiple threads for input, sort, and output processing. The output "odirect" flag is used, and an "fsync" is performed before a node signals completion.

E.Output Validation

To validate the sort outputs, *valsort* is run (unmodified) first on each local output directory in each node, and then globally on the collected outputs of the local runs. The *valsort* output (checksums and duplicate key counts) are recorded.

The settings indicated in the diagram were found to be optimal for the 100TB GraySort, and were modified only modestly for the other benchmarks.

IV. DAYTONA SORT

The Daytona sort has the same basic architecture as the Indy sort, but with some modifications to most stages and with two additional stages added:

A.Input Generation (not shown in Figure 2)
B.Range Partitioning
C.Skew check and repartitioning (added)
D.Network Shuffle
E.Merge & Sort
F.Output replication (2nd write added)
G.Output Validation (not shown in Figure 2)

Each stage is discussed below.

A.Input Generation

Input data is the same as that of of Indy sort, and every input file is copied to a designated backup node to enable recovery in the case of node failure.

B.Range Partitioning

Daytona sort requires handling input data sets that do not fit in node memory. If the aggregate input size exceeds the size that can be processed, the partitioning stage operates on a part of the input data set that fits in memory and write the output files for each set to (xfs) storage. For a 1 PetaByte sort, for example, eight input file sets of 250GB per node could be processed in sequence. For very large input sets the number of ranges in the first stage must be increased to ensure a single globally aggregated range fits in node memory in the sort and merge stage. For the purposes of the reported benchmarks, the output of the first stage is written to local tmpfs.

Also for Daytona the application must be able to handle a variety of key and record sizes and must be able to support a variety of sort orders. To achieve this while maintaining good performance the (local) partitioner uses an internal data representation consisting of a key and pointer to a data record in memory. The partitioner (and similarly the stage 2 sort and merge) reconstitutes the 100B records before they are output.

C.Skew Check and Optional Data Repartitioning

At the end of the first stage, before the shuffle, the sizes of each global range are conservatively approximated. If the size of any global range exceeds the smallest range size by 2x or more, then the data is re-partitioned. If re-partitioning is required then each of the partitioned input data sets (i.e. the output of the range partitioner) is aggregated/divided into a reasonable number of input files (in our 100TB runs 8 input files of about 25GB for each multiple of about 200GB in the input) and each of these new input files are sorted.

If the data was small enough to fit in memory then this process does not require storage access. If the input data size exceeds the available memory then the output of the range partitioner resides in local storage and this stage must read and write from local storage as well.

Depending on the number of desired non-overlapping ranges (8K in our case), largest record keys are determined for four times that number (32K in our example) of nearly equal-sized ranges for each of the sorted files that are the input for this stage of the repartitioner. These keys plus for each such key a data field indicating their location (node, file, and range id) are collected globally. For the 100TB sort with 200GB input data sets and 8 sorted files per node, this results in 512(nodes) x 8(sorted files per node) x 32K(ranges) = 128M key/location pairs. These key/location records are then sorted by key, maintaining key ordering within each file, i.e. ensuring that keys originating from the same sorted file do not go out of order. A linear scan of the globally collected largest range keys while maintaining a list of the last key/location pairs for each file is performed, outputting this list for every n_th key in the globally collected list. In our example n=16K and 8K such lists (of 4K key/location pairs each: one pair for each sorted file in each node) are output. Identification of which range to split in each file is required to ensure an even distribution even in the case of keys that are repeated often. The results of this calculation are communicated back to all of the nodes, and within each node each locally sorted file is divided into a new set of ranges (8K in our example) based on the list of global split keys and for each global split key the range (of the 32K) to split. Note that the range to split is the one following the last range that was fully included, indicated by the corresponding last key for that file prior to the split key in the globally sorted list. A binary search locates the first element within the range to be split equal to or larger than the global split key. Note also that because the same range may be split multiple times a new range may be empty.

For datasets that do not exceed the size of the available memory this stage also completes in memory .

This procedure is guaranteed to result in global range partitions where each global partition is at least 3/4th the size of the average partition and at most 5/4th the average size.

D.Network Shuffle

In the case where the input data does not fit into main memory, the network shuffle reads from xfs, and merge and sort are alternated, but is otherwise identical to Indy sort.

E.Merge And Sort

The merge and sort stage writes its output to tmpfs instead of xfs, but is otherwise the same as for the Indy sort.

F.Output Replication

For Daytona output is written both to local xfs and to the designated backup node from which data can be recovered in the case of node failure. The local copy is written by "dd" using "oflag=direct", the output replica is written by the shuffle server using OS cached write. A background periodic sync and drop cache mitigates the cost of the final synchronization.

G.Output Validation

Output Validation for Daytona is the same as for Indy, but with the added requirement that Daytona sort needs to be able to run continuously for an hour without system failure. This capability is documented in Table 7 with 30 successive runs of more than 2 minutes each..

V. BENCHMARK RESULTS

Benchmark	*Input Size*	*Iterations*	*Median Time*	*Result*
100TB Indy GraySort	**100.0448 TB**	**10**	**98.845 sec**	**60.7283 TB/min**
Indy MinuteSort	**55.296 TB**	**15**	**59.910 sec**	**55.296 TB**
100TB Daytona GraySort	**100.0448 TB**	**30**	**134.100 sec**	**44.7628 TB/min**
100TB Daytona (skewed)	**100.0448 TB**	**8**	**257.960 sec**	**23.2698 TB/min**
Daytona MinuteSort	**36.864 TB**	**15**	**57.140 sec**	**36.864 TB**
Daytona MinuteSort (skewed)	**36.864 TB**	**15**	**108.590 sec**	**20.369 TB/min**

Table 3: Benchmark Results

For each of the reported results, time was measured (using the linux *time* command) on the host node that initiates the computation. Care was taken to ensure no results of prior runs remain in the caches, and that outputs were fully written to secondary storage before the data nodes indicate their completion to the host.

Approximate Duration (sec)	*Range Partition*	*Skew Check& (opt.)Repartition*	*Network Shuffle*	*Sort Merge*	*TOTAL*
100TB Indy GraySort	**23**	**N/A**	**26***	**67***	**99**
Indy MinuteSort	**14**	**N/A**	**20***	**40***	**60**
100TB Daytona GraySort	**23**	**3**	**23**	**85**	**134**
100TB Daytona (skewed)	**23**	**104**	**23**	**108**	**258**
Daytona MinuteSort	**10**	**3**	**10**	**34**	**57**
Daytona MinuteSort(skewed)	**10**	**46**	**10**	**43**	**109**

*Table 4: Approximate timing of the individual Stages. * indicates overlapping stages.*

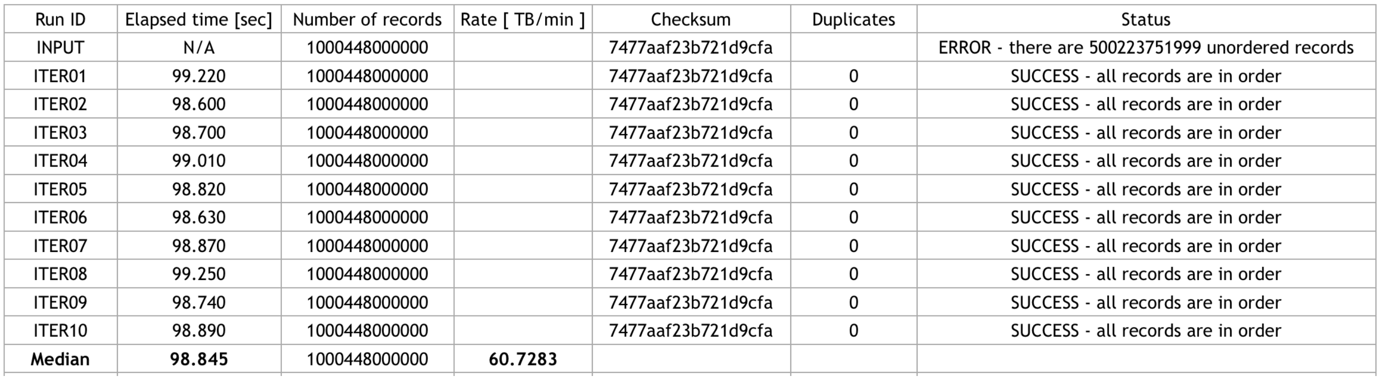
Table 4 summarizes the typical time spent in each of the stages of the computation. Note that for the Indy sorts the network shuffle and merge and sort stages partially overlap. This lengthens the shuffle and sort and merge stages somewhat. Also note that, as one would expect, even after redistribution node-to-node variability in runtime is larger for the skewed datasets. CPU utilization is typically less than 10% for stages other than sort and merge, where it ranges from 20-30% but peaks at about 70% when sort and networking are overlapped.

A summary of all the runtimes of all the consecutive iterations for each benchmark is attached in Tables 5-10.

Acknowledgement

The authors thank Chris Nyberg for his extensive feedback resulting in many improvements to the presentation.

512 Node, 100.0448 TB Indy GraySort (98.845 sec, 60.7283 TB/min)



Run ID	Elapsed time [sec]	Number of records	Rate [TB/min]	Checksum	Duplicates	Status
INPUT	N/A	1000448000000		7477aaf23b721d9cfa		ERROR - there are 500223751999 unordered records
ITER01	99.220	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER02	98.600	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER03	98.700	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER04	99.010	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER05	98.820	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER06	98.630	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER07	98.870	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER08	99.250	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER09	98.740	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER10	98.890	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
Median	**98.845**	1000448000000	**60.7283**			

Table 5: Successive Indy GraySort attempts

Run ID	Elapsed time [sec]	Number of records	Checksum	Duplicates	Status
INPUT	N/A	552960000000	405f7bcfe1bf60dbc4		ERROR - there are 276479831183 unordered records
ITER01	57.890	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER02	60.250	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER03	60.220	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER04	59.660	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER05	59.920	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER06	59.600	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER07	59.640	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER08	60.110	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER09	59.910	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER10	60.090	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER11	59.670	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER12	59.960	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER13	59.720	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER14	59.560	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
ITER15	60.600	552960000000	405f7bcfe1bf60dbc4	0	SUCCESS - all records are in order
Median	**59.910**	**552960000000**			

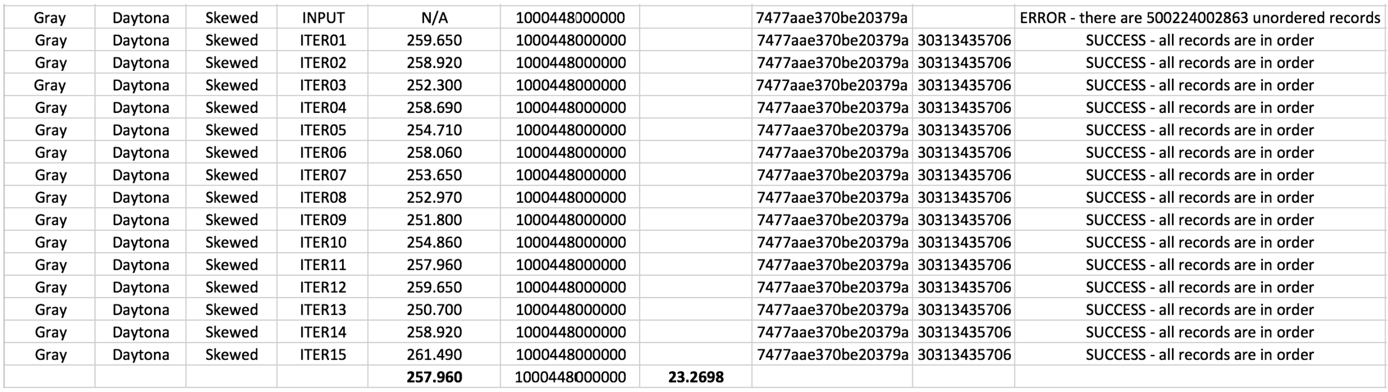
Table 6: Successive Indy MinuteSort attempts

512 Node, 100.0448TB Daytona GraySort
(median 134.100 sec, 44.7628 TB/min)

Run ID	Elapsed time [sec]	Number of records	Rate [TB/min]	Checksum	Duplicates	Status
INPUT	N/A	1000448000000		7477aaf23b721d9cfa		ERROR - there are 500223751999 unordered records
ITER01	133.500	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER02	134.210	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER03	134.960	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER04	133.960	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER05	133.500	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER06	134.220	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER07	134.320	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER08	134.100	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER09	133.910	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER10	134.510	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER11	133.500	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER12	134.520	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER13	134.450	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER14	134.010	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER15	134.480	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER16	133.960	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER17	134.330	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER18	134.100	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER19	134.040	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER20	134.150	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER21	134.210	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER22	134.300	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER23	134.110	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER24	133.920	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER25	134.050	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER26	133.980	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER27	134.240	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER28	134.060	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER29	133.880	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
ITER30	133.740	1000448000000		7477aaf23b721d9cfa	0	SUCCESS - all records are in order
Median	**134.100**	1000448000000	**44.7628**			

Table 7: Successive Daytona GraySort attempts

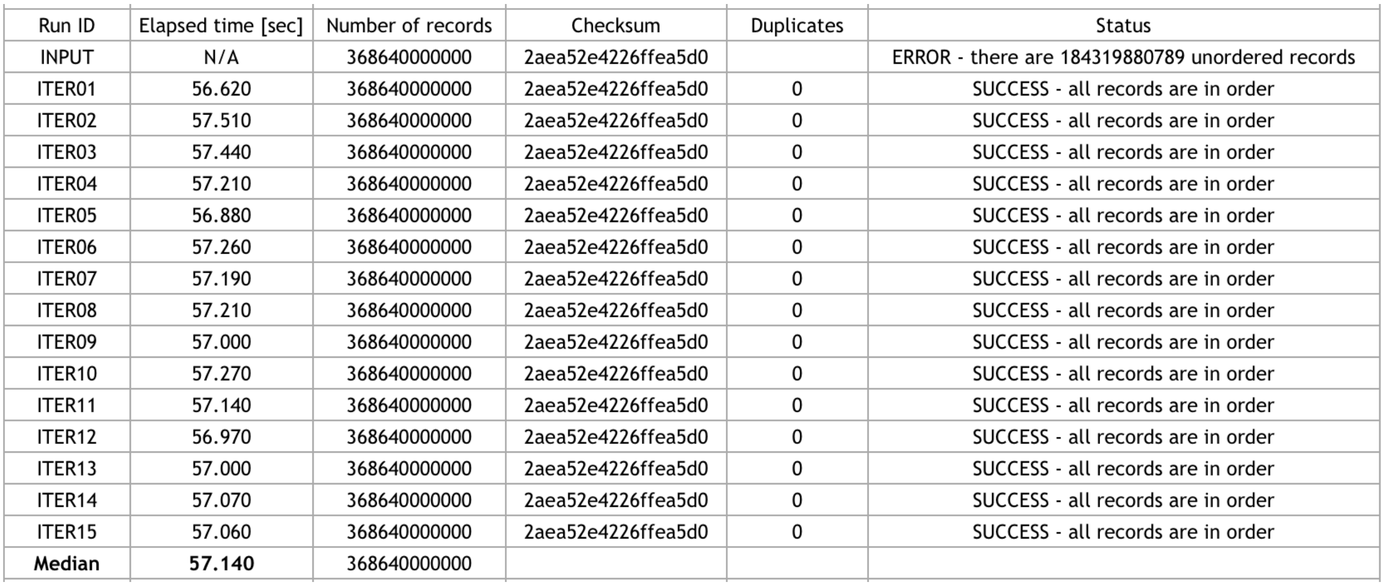
512 Node, 100.0448TB Daytona GraySort Skewed
(median 257.960 sec, 23.2698 TB/min)



Gray	Daytona	Skewed	INPUT	N/A	1000448000000		7477aae370be20379a		ERROR - there are 500224002863 unordered records
Gray	Daytona	Skewed	ITER01	259.650	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER02	258.920	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER03	252.300	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER04	258.690	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER05	254.710	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER06	258.060	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER07	253.650	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER08	252.970	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER09	251.800	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER10	254.860	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER11	257.960	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER12	259.650	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER13	250.700	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER14	258.920	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
Gray	Daytona	Skewed	ITER15	261.490	1000448000000		7477aae370be20379a	30313435706	SUCCESS - all records are in order
				257.960	1000448000000	**23.2698**			

Table 8: Daytona Graysort skewed dataset

512 Node, Daytona Minutesort
(median 57.14 sec, 36.864 TB/min)



Run ID	Elapsed time [sec]	Number of records	Checksum	Duplicates	Status
INPUT	N/A	368640000000	2aea52e4226ffea5d0		ERROR - there are 184319880789 unordered records
ITER01	56.620	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER02	57.510	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER03	57.440	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER04	57.210	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER05	56.880	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER06	57.260	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER07	57.190	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER08	57.210	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER09	57.000	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER10	57.270	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER11	57.140	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER12	56.970	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER13	57.000	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER14	57.070	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
ITER15	57.060	368640000000	2aea52e4226ffea5d0	0	SUCCESS - all records are in order
Median	**57.140**	368640000000			

Table 9: Successive Daytona MinuteSort attempts

512 Node, Daytona Minutesort Skewed Dataset
(median 108.590 sec, 20.369 TB)

Run ID	Elapsed time [sec]	Number of records	Checksum	Duplicates	Status
INPUT	N/A	368640000000	2aea524a7fc09fb4a9	0	ERROR - there are 184320085444 unordered records
ITER01	107.960	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER02	109.280	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER03	108.280	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER04	106.850	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER05	109.850	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER06	109.670	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER07	107.260	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER08	108.260	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER09	106.780	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER10	108.850	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER11	107.120	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER12	108.590	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER13	108.770	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER14	111.070	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
ITER15	108.600	368640000000	2aea524a7fc09fb4a9	4782987147	SUCCESS - all records are in order
Median	**108.590**	**368640000000**			

Table 10: Daytona MinuteSort skewed datasets