Energy-Efficient Fast Sorting 2011

Andreas Beckmann[*], Ulrich Meyer; Goethe University Frankfurt
Peter Sanders, Johannes Singler[†]; Karlsruhe Institute of Technology

In this submission to the 2011 deadline of the Sort Benchmark, we polish some of our former results, breaking the current records in two categories, namely Indy Gray and 1 TB Daytona Joule. By measuring the consumed energy, the Indy Gray run also gives the first-ever submitted result for 100 TB Indy Joule category.

1 History

Authors of this report participated in the Sort Benchmark contests in 2009 and 2010. In 2009, our DEMSort program took the lead in the then-new Indy Gray category [RSSK09, RSS10], sorting 100 TB on a cluster with about 200 nodes. A tie was declared with Yahoo, whose Hadoop-based program achieved about the same result in the Daytona class, but with 17 times the hardware effort. Former results in the then-expired Indy Terabyte category were outperformed by a factor of 3–4, the same for the Indy Minute results, beating Yahoo's new Daytona Minute result by almost a factor of 2.

In 2010, our group focused on energy efficiency [BMSS10b, BMSS10a]. With specifically selected hardware, namely a machine featuring an Atom processor and four solid state disks, we improved the records in the different Indy and Gray size categories by factors of up to 5. For the 10 GB inputs, we were slightly beaten by FAWNsort [VTA+10] though. However, the better result of FAWNsort is mostly due to the large 12 GiB of RAM, which fits the whole input, so this approach does not scale much further. Due to disk space restrictions, we had not been able to submit a valid result for the 1 TB Daytona Joule category. To fill in this gap, we have run this category on a standard server machine, and report the results here.

In the meantime, in 2010, TritonSort [RMM+10] took over the lead in the Indy Minute and Indy Gray categories, although beating DEMSort by just a few percent, which was considered a tie in the Gray case by the committee. Just as DEMSort, TritonSort exploits the given hardware very well, it uses about the same number of disks. However, the described algorithm is quite basic, and seems to work well only for uniformly distributed input, but not in general. In particular, it is unclear how the distribution step works. Sampling or something similar is not mentioned, so the splitters are probably hard-coded for the Sort Benchmark input. Worst-case inputs would probably crash the program due to bad load-balance and out-of-space problems. In contrast to that, DEMSort gives worst-case guarantees.

In order to be competitive using a machine with little computational power, we had tuned our algorithms already for our 2010 JouleSort submissions. These improvements included using one file per sorted run instead of one file per block, using radix sort for the internal sorting, reading/writing many blocks at once where possible (range I/O), and fixing a bug that prevented full overlap of I/O with computation and communication. By utilizing the improved version, we hoped for just a small improvement, in order to regain the lead. However, none of the improvements has helped for Minute Sort, where there is only one pass and one sorted run, and sorting does not have to be in-place. So we had to resort to the 100 TB case, running the improved DEMSort program on the same machine as in 2009.

[*]Supported in part by MADALGO – Center for Massive Data Algorithmics, a Center of the Danish National Research Foundation, and by DFG grant ME 3250/1-2.
[†]Partially supported by DFG grant SA 933/3-2.

Details on the algorithm and implementation contained by DEMSort can be found in [RSS09, RSS10] (initial papers), [RSSK09] (implementation details, Sort Benchmark adaption details), [BMSS10a] (recent improvements to DEMSort).

2 Energy Measurement

For measuring the energy consumption, we again used the ZES ZIMMER LMG95 precision power meter, which has an accuracy of less than ±0.1% for the applicable measurement range. This device was controlled via the serial port by the machine running the sort. The power measurement was started just before the sorting program and stopped right after, sampling power in 1-second intervals. We calculated the energy by multiplying the average power by the time reported by `/usr/bin/time` for the running time of the actual sort program. The temperature was about 23 °C throughout the tests, the CPU fan was running. All measured power values for our machine denote overall power consumption, i. e. they include losses within the DC power supply.

3 Results for 1 TB Daytona JouleSort

Sadly, we were unable to further improve our 1 TB Indy Joule results from 2010. While we stated in the 2010 report that TRIM support would improve the situation, we were unsuccessful to benefit from it in practice. TRIM support had already arrived for Linux kernel version 2.6.32, but was not supported for software RAID via the device mapper until version 2.6.36. Both XFS and EXT4 were TRIM-enabled at the time of testing, but we had similar problems with both of them. While the long-run performance of the SSDs stayed much more stable in fact, the repeated reorganizations thwarted the effective bandwidths during the test runs, even if done only every 20 GiB. However, we still think that this is not a matter of principle, but will sooner or later be solved by renewed operating software and firmware.

As stated before, we were unable to submit a valid result in 2010 for the 1 TB *Daytona* Joule category using our Atom machine. This was because the input file must not be overwritten following the Daytona regulations, but we had only 1023 GB of space physically available, where at least 2000 GB would have been needed. Apart from that, the *Nsort* program, which we used for the other Daytona Joule entries, needs even three times the space of the input. This constellation left the 1 TB Daytona Joule category with a very outdated result, achieved on a quite standard server in 2007.

Thus, we just ran Nsort on an up-to-date server machine, expecting some improvement by the performance/watt ratio, which generally gets a bit better over time. Our server features two sockets, each containing a quad-core Intel Xeon X5550 processor (codenamed Nehalem) running at 2.67 GHz. It is equipped with 48 GiB of RAM, and includes 8 1 TB SATA disks (Seagate, 7200 rpm), which were configured as a software RAID 0, and formatted using XFS.

In average over five runs, we sorted 1 TB in 6486 s, using 1897 kJ per run, which is equivalent to 5273 records per Joule. This result improves over the former 2007 CoolSort record by 54%.

DEMSort was in fact even a bit better (faster) than Nsort, but does not qualify for the Daytona class. For 100 GB, we achieved similar results, for 10 GB, the efficiency doubles as the input completely fits into the RAM. With more than 10 000 records sorted per Joule, this result approaches the first records for 10 GB, given in 2007 by using a special low-power computer.

4 Results for 100 TB Indy JouleSort and Indy GraySort

The 100 TB JouleSort category was introduced shortly before the 2010 deadline thanks to our suggestions in the preliminary 2010 submission. Its virtue is the fact that 100 TB can hardly be handled on a single machine nowadays. Thus, it enforces the usage of a compute cluster, which is likely to scale to even more data, giving sensible values for huge data sets.

The testing machine was again the 200-node Intel Xeon Cluster as used in 2009. This time, it ran Suse Linux Enterprise 11 with kernel 2.6.27. Each node consists of two Quad-Core Intel Xeon X5355 processors clocked at 2 667 MHz with 16 GiB main memory and 2 × 4 MiB Cache, and has attached 4 250 GB Seagate Barracuda 7200.10 disks. So in total, the machine has 3 TiB of internal and exactly

Class	**Daytona**	**Indy**
Size (records)	**10^{10}**	**10^{12}**
Machine	Nehalem	IC1
Program	Nsort	DEMSort
Data Volume	1 TB	100 TB
Exact Number of Records	10 000 000 000	1 000 029 388 800
Checksum	12a06cd06eeb64b16	746b3136e818e3396f
Time	6486±16 s	9835±21 s
Energy	**1897 ±4.7 kJ**	**694±1.5 MJ**
Average Power	292.4±0.15 W	70.5±0.01 kW
Records per Joule	**5273±13**	**1 441±3**
Typical Bandwidth Input Reads	309 MiB/s	286 MiB/s per node
Typical Bandwidth Temp Writes	309 MiB/s	264 MiB/s per node
Typical Bandwidth Temp Reads	316 MiB/s	202 MiB/s per node
Typical Bandwidth Output Writes	315 MiB/s	181 MiB/s per node

Table 1: Subsummation of our new results, the right one being valid in two categories at the same time.

200 TB of disk space, but only 117 TiB were usable to us. The typical maximum I/O rate of the disks is about 80 MiB/s for reading as well as for writing (outermost tracks). Most of the disks are of type ST3250820AS, some of them having been exchanged by different models of at least the same speed. All disks of a node were configured as a software RAID 0, and formatted using XFS. For the contest runs, we selected the 195 nodes with the fastest disks available to us. The nodes are connected by an 288-port InfiniBand 4xDDR switch. The theoretical point to point bandwidth between two nodes is more than 1 300 MiB/s, which decreases to an average of 500 MiB/s when all nodes are communicating. We used GCC 4.4.4 as the compiler, and the MPI-implementation MVAPICH 1.1. The details of the algorithms and the implementation can be found in [RSSK09] and [BMSS10b, BMSS11].

However, energy measurement is very difficult for a cluster. In our case, the cluster is not owned by us exclusively, so the time and the modalities of exclusive access are very limited. In the following, we describe the utmost possible effort, given the circumstances and respecting the interests of the other users of the cluster. We attached the LMG95 to a multiway plug, which served two randomly selected compute nodes, effectively averaging the electrical power consumed by them. To extrapolate to the whole cluster, we multiplied this value by 97.5. We were not able to additionally measure the power consumption of the connecting switch. Thus, we upper-bounded its energy usage by the maximum wattage stated on its A/C adapters (2×700 W), and even rounded up to 2000 W, which after all does alter the result much.

We ran two executions of the whole test, and averaged over both. In average, sorting 100 TB took 9835 s, which is equivalent to 610 GB/min. This constitutes a new all-time record for the Indy/Daytona Gray category (plain performance), improving by about 8% over our previous results, and by almost 5% over the current record-holder TritonSort. This little improvements shows that there was in fact not much potential left, we suppose any further improvement to be hard bounded by maybe 5% better.

The sorting consumed an estimated 694 MJ of energy, i. e. 352 W per node (without the share on the switch), and 1441 sorted records per Joule. Not surprisingly, this is much worse than all JouleSort results ever reported. There is a multitude of reasons for this:

- The used cluster is quite dated, it was delivered in 2008, but commissioned already in 2007.
- It is optimized for scientific high-performance computing workloads, which typically load all computational units, and usually are not thwarted by disk I/O or the similar.
- The compute nodes are not well balanced with respect to energy consumption. They are too powerful computewise compared to the relatively scarce disk bandwidth and capacity.
- Power saving measures could not be enabled due to stability problems. They are not desperately needed because of the generally high load on the cluster.

- The purchase was made without energy efficiency in as a criterion.

Nevertheless, the result is very interesting. Compared to our best result with the same software (1 TB Indy JouleSort), the difference amounts to a factor of 12. Compared to the best Daytona result (FAWNsort for 10 GB, 44 900 recs/J), there is even a factor of 31 in difference. Of course, the results for the smaller inputs will not scale linearly to 100 TB, but more than a factor of 2 should not be needed as overhead. This shows that there is a lot of room for improvement in data-intensive computing, despite the fact that our software is almost optimally exploiting the hardware. We suppose that in practice much less efficient software is used.

5 Conclusions

The achieved results are subsumed in Table 1, including standard deviations and checksums. The checksum may differs compared to earlier submission due to a new version of the generator program.

Overall, we have improved our Indy GraySort result a little, filled in a gap by running 1 TB Daytona JouleSort on a commodity server, and provided the first result for 100 TB JouleSort.

We thank the people from the Steinbuch Center for Computing (SCC) at KIT for their continuing support. The DEMSort source code is available on request from the authors.

References

[BMSS10a] Andreas Beckmann, Ulrich Meyer, Peter Sanders, and Johannes Singler. EcoSort, May 2010. http://sortbenchmark.org/ecosort_2010_May_15.pdf.

[BMSS10b] Andreas Beckmann, Ulrich Meyer, Peter Sanders, and Johannes Singler. Energy-efficient sorting using solid state disks. In *1st International Green Computing Conference*, August 2010.

[BMSS11] Andreas Beckmann, Ulrich Meyer, Peter Sanders, and Johannes Singler. Energy-efficient sorting using solid state disks. *Sustainable Computing: Informatics and Systems*, 1(2):151 – 163, 2011. Special Issue on Selected Papers from the 2010 International Green Computing Conference.

[RMM+10] Alex Rasmussen, Radhika Niranjan Mysore, Harsha V. Madhyastha, Michael Conley, George Porter, Amin Vahdat, and Alexander Pucher. TritonSort, May 15, 2010. Sort Benchmark final 2010 submission.

[RSS09] Mirko Rahn, Peter Sanders, and Johannes Singler. Scalable distributed-memory external sorting. *CoRR*, abs/0910.2582, 2009.

[RSS10] Mirko Rahn, Peter Sanders, and Johannes Singler. Scalable distributed-memory external sorting. In *26th IEEE International Conference on Data Engineering (ICDE)*, pages 685–688, 2010.

[RSSK09] Mirko Rahn, Peter Sanders, Johannes Singler, and Tim Kieritz. DEMSort – distributed external memory sort, 2009. http://sortbenchmark.org/demsort.pdf.

[VTA+10] Vijay Vasudevan, Lawrence Tan, David Andersen, Michael Kaminsky, Michael A. Kozuch, and Padmanabhan Pillai. FAWNSort: Energy-efficient sorting of 10 GB, May 15, 2010. Sort Benchmark final 2010 submission.