**CSC591: Data Intensive Computing**

**Dec 5, 2017**

# Benchmark and Comparison: HPCC Thor vs. Apache Spark

**Team #16**
**Pratik Singh(psingh22@ncsu.edu)**
**Shalki Shrivastava (spshriva@ncsu.edu)**

**Abstract**
HPCC Thor and Apache Spark are both big data analytics engine. Here we evaluate and compare their performance for batch processing over big data, over metrics like scalability, latency, and CPU utilization. We also make a note of the general trends in performance and analyse which engine is better suited for which kind of workload.

**Introduction**
There is lot of big data available today from many different sources - it is so large and complex that it is very difficult, if not impossible, to process it using the conventional data management technologies. The problems of volume, velocity, and variety in the big data pervasive today has lead to many different innovative technologies to enable its processing and extract meaningful information. Hadoop mapreduce is a very popular open source framework used for batch processing of huge amounts of data in scalable distributed manner using commodity hardware. But the main issue with the hadoop framework is the rigidity of is map reduce paradigm, which is not amenable for a lot of applications which do not naturally map to the sequence of map and reduce tasks well. Apache spark was developed at amplabs at UC Berkley in 2009. It provides a comprehensive and unified big data processing framework for big data processing requirements with a variety of datasets that are diverse in nature as well as source.

High performance computing cluster (HPCC) developed by Lexis Nexis risk solutions is also a open source platform for processing and analyzing large volumes of data. It has been in production since late 2000, and akin hadoop, it also utilizes commodity linux machines for distributed and scalable execution of data intensive applications. It has two different component for specialized processing of batch and streaming kinds of workloads, handled by thor and roxie respectively. Thor is a data refinery which takes inn massive volumes of raw data and performs jobs like data cleaning, ETL, index creation, queries, and large scale analytics etc. Thor is quite similar to the google and hadoop map reduce platforms in its functionality, but is supposed to provide much higher performance in equivalent hardware configurations.

In this paper, we are evaluating the performance of hpcc thor and apache spark for batch processing jobs on big data. We are evaluating the performance of the two engines for parameters like scalability, latency and cpu utilization for sql workload on datasets ranging from 2 GB to 32 GB in size. We have also compared the performance of thor and spark in context of these factors for different configurations, and studied the workloads and configurations for which each engine performs better.

**Related Work**

HPCC Thor has been released fairly recently, and there isn't much work that has been done in the space of benchmarking Thor. But we would like to refer [4] which was released when we were already working on this project. Our work differs from them in terms of configuration used and the sql workloads on which experiments were run.

**Goals**

We are evaluating the performance of HPCC Thor and Apache Spark for three parameters:

- Scalability: the variance in performance of the systems as a function of the dataset size and the resources available. Both dataset size and resources are independent variables here, and we are studying the performance of the system with the increase in the dataset size even as the resources available remain constant. We are also studying the performance with the change in the resources available even as the dataset size remains constant. But our focus for the purposes of this study is on strong scaling - maintaining a constant range of dataset size and increasing the resources. To evaluate the performance viz a viz weak scaling, larger datasets would have been required, and our monetary constraints prevented us from satisfiably studying that aspect in this experiment.

- Latency: we studied the latency of both the engines - the time gap between the submission of the job to getting the results. This is an important parameter for performance evaluation, although for batch processing jobs, it is not as critical as it ifor interactive jobs.

- Cpu utilization: the capacity to which the processing power of the cpu is being used as a function of time. Better cpu utilization indicates a better ability to harness the compute power available to obtain optimal results. An engine with less than satisfactory cpu utilization essentially wastes the compute power at its disposal, and thus invariably has higher latency than an engine which doesnt. Thus, cpu utilization and latency are inversely related, and lower cpu utilization always leads to a higher latency in the system.

**Experiment design**

The design and the setup of the experiment is one of the most critical phases of the entire benchmarking process. It is very important to pick each experiment being conducted very carefully, and calibrate the settings to ensure that only variable is varied at a time, so that the variance in the result can be reliably studied as a function of that variable. At this point, it is important to ensure that other variables remain constant, otherwise it can lead to noisy results, and it would be difficult to point  out exactly what factors are responsible for the results observed.

**Workload**

We decided to use the sql workload for the our experiments on thor and spark. The sql queries can be divided into three broad categories: filter, aggregate, and sort. For filtr queries, we used queries with two different selectivity measures - low and high. The query with high selectivity returns only around 2 rows as the result, whereas the query with low selectivity returns thousands of rows as the result. The aggregate queries were finding the minimum, maximum, average, count, and group by ver the dataset. While the sorting query sorts the randomized dataset given as input.

We did not do insert/update queries as part of the of sql workload. Both spark and thor work by creating immutable objects from the dataset over which further analysis is carried out. Inserting or updating a row would essentially result in creating the immutable object from scratch again. As both the engines are being evaluated for batch processing jobs, it is a fair assumption to make that the data they once receive does not change. Hence, in such a scenario, insert/update queries would be pointless.

**Datasets**

The datasets used were of sizes 2 GB, 4 GB, 8 GB, 16 GB, and 32 GB. we also used two types of datasets: integers and strings. Most systems handle the integer and string data differently, hence to test the performance at that granularity, we created separate integer and string datasets. Also, both integers and strings can be considered representative of the real life numeric and string datasets. Thus, the total number of datasets on which experiments were run are ten. We generated the dataset using a data generator, in which integers and strings were randomly generated. We chose to generate the values randomly, as that helped us in preserving the broad characteristics of the dataset as we increased its size. The code to generate the data can be found in our github repository at Rambharose/src/DataGenerator/Thor/DataGen.ecl.

**Cluster setup**

We used Amazon AWS services to setup the cluster to run the experiments. We set up a cluster with one node using AWS Educate, whereas for cluster with 3 and 5 nodes respectively, we used the AWS EMR service. All the clusters used the m1.large EC2 instance available in US-West Oregon region. Each instance had 2 cores, 8 GB RAM, and 850 GB EBS storage. The choice of the RAM for the machines used stemmed from the consideration that data analytics engines like Thor and Spark are faster when the data fits completely in-memory. But a lot of real world data rarely fits completely into the memory, and has to be invariably spilt onto the disk.

We wanted to evaluate the performance of the engines in both these scenarios. Hence, by picking 8 GB RAM, the assumption was that datasets of sizes 2 GB and 4 GB will easily fit into main memory, while for 8 GB, 16 GB, and 32 GB, some of the data will need to be stored on disk as well. We also used EBS storage here, because that was the only kind of disk storage available with this combination of cores and RAM, but another interesting choice could have been the use of SSD storage, as SSD disks are faster.

**Softwares**

We used the latest version of all softwares available at the time - HPCC Thor v6.4 and Apache Spark v2.2. We used Scala for Spark programming as Spark is written in Scala, and writing the application inn Scala provides an added level of optimization. We tuned Spark to use maximum number of cores and RAM possible by explicitly setting values for --num-executors, --executor-cores, and --executor memory parameters. The Thor programming was done in ECL which is written in C++. to monitor CPU utilization, we used a linux utility called dstat which provides the percentage time CPU was used for each second.

**Observations:**

The following trends were observed

(1) Variance in latency with respect to the data size: The latency for all the operations increases proportionately with the increment of datasize. We observed that as we double the size of the dataset the latency of the operations approximately doubles. This trend can be seen all the plots(a,b,c,d…,k,l). In all these figures, the Y-axis represents the time(in seconds) and the X-axis represents the different type of SQL workloads.

(2) Integer Dataset: In Thor, the count distinct workload and sort workload are the most expensive workloads and they roughly take the same time to complete. The reason might be due to the fact that count distinct operation might be using sorting as a first phase and then it uses a linear run to compute the count of distinct records. The aggregate operations like average, minimum, maximum do take roughly the same amount of time as all of them are O(n) operations. The aggregate by key operation takes little bit more time than other aggregate operations as it has to do extra work to compute the count of the record with each distinct keys.
In Spark, unlike Thor, count distinct is not as expensive as the sort operation. It takes just a little bit more time than the count operation. Spark might be using hashing techniques to count the distinct number of records. Infact, Spark takes about 1.5 time less time to perform count distinct operation than what Thor needs. The aggregate operations like average, minimum, maximum do take roughly the same amount of time as all of them are O(n) operations. The aggregate by key operation takes little bit more time than other aggregate operations as it has to do extra work to compute the count of the record with each distinct keys.
All of these trends are observed in all the clusters, with the differences in the latency of different operations becoming more pronounced and evident with the increase in the

dataset sizes.

(3) String Dataset: In Thor, the sort workload is the most expensive workloads. The count distinct operation for string does not the same times as of sorting as observed in the integer dataset. The reason might be due to the fact that count distinct operation in Thor has been implemented differently for numeric and non-numeric datasets. The aggregate operations like average, minimum, maximum do take roughly the same amount of time as all of them are O(n) operations. The aggregate by key operation takes little bit more time than other aggregate operations as it has to do extra work to compute the count of the record with each distinct keys.
In Spark, the same trend was observed, with the only noticeable difference being that the Spark takes more time (about 2-5 times) than Thor for each operation. This difference is visible more clearly in the larger datasets (16 and 32GB). The plots m, n, o and p show these trends. All of these trends are observed in all the clusters, with the differences in the latency of different operations becoming more pronounced and evident with the increase in the dataset sizes.

(4) Scalability factor: For Thor the scalability factor(latency of the original system/latency of the new system) was close to its ideal value(resources in the original system/resources in the new system). For the same dataset size across different size of the cluster, the latency varies directly with the number of nodes in the cluster. For example, the Thor cluster with size 3 performed approximately 3 times better than the cluster with size 1. The same trend was observed across all the other Thor clusters.
For Spark the scalability factor was smaller than the ideal scalability factor expected. For example, the Spark cluster with size 3 performed approximately 1.7 times(only) better than the cluster with size 1. In Spark, the overhead cost of cluster operations might be very large as compared to the same in Thor, hence the ideal scalability factor is not prevalent in Spark clusters.

**CPU Utilization**

We observed that Thor uses 58% CPU for sorting workloads, and 48% for AggregateMin and AggregateMax queries. For rest of the workloads, Thor uses approximately 55% of CPU. These figures are similar for both integer and string type datasets. But in Spark noticeable difference is observed for filter and sort queries. For integer datasets of all sizes, the CPU utilization stands at 96%, while for string datasets, the CPU utilization was at 96% for smaller datasets which can potentially fit into the RAM (<8GB), but for larger datasets, it drops to 50%. Similarly, for sort query, for larger string type datasets (>=16 GB), CPU utilization drops to 37%. For all other workloads, Spark averages a CPU utilization of 48 to 50%.

CPU Utilization for Filter Query in Spark

| | Integer | String |
|---|---|---|
| <8 GB | 96% | 96% |
| >=8 GB | 96% | 50% |

CPU Utilization for Sort Query in Spark

| | Integer | String |
|---|---|---|
| <16 GB | 96% | 90% |
| >=16 GB | 96% | 37% |

By looking at the log files obtained from dstat, we also observed that Spark has lower CPU utilization and higher idle time for string datasets as compared to integer datasets of the same size.

**Challenges Faced and Effort Required**

Here, we would like to make a not of the challenges faced in this study, and the amount of effort required to successfully complete it, as a guidance for future such endeavors.

The entire experiment took about 50 USD to use the AWS instances and clusters for the experiment. The experiments ran for a total of 60 hours. The individual breakdown of each experiment can be found in the appendix. AWS Educate account can be used to run single node cluster for free, but it does not allow the creation of EMR or even manual creation of cluster. We did not get the free credits in the AWS account, due to which we had to use our private account to run the experiments on EMR cluster.

The cluster creation is also a very slow process, and it takes almost an hour just to launch the cluster. It also shows the error “core 2: limit exceeds”, but starts successfully after an hour. The error thrown, alongwith a huge amount of time to actually launch the cluster leads one to believe that the cluster creation process has failed, but that is not necessarily the case. Also, there is an internal limit on the size of the cluster one can create from the account, and a manual request has to be made to the AWS Customer Care to increase the limit. They take more than 24 hours to actually increase the cap on the cluster size, and the increase is by just 1 most times. So a request for the cap on cluster size to be increased should be made well in advance.

**Conclusion**

To conclude, we have done performance evaluation of HPCC Thor and Apache Spark for SQL workload on datasets ranging from 2 GB to 32 GB in size. We evaluated the performance for scalability, latency, and CPU utilization, and compared their performance against each other. We observed that Spark outperforms Thor by 0.5x for Aggregate Count queries. But for all other workloads, Thor outperforms Spark by 2-3x in case of Integer datasets, and by 4x in case of String datasets. In terms of scalability, both Thor and Spark scale equally well as a function of cluster size, and dataset size. For CPU utilization, Thor demonstrates more consistent performance at 55%, whereas Spark gives a really good CPU utilization for Filter and Sort queries (96%), but for other kinds of workloads it has a CPU utilization of approx 50%.

For future work, a further study can be carried out to decipher the reasons behind the performance differences observed between Thor and Spark. Potential directions to explore can be the role of JVM, memory management, and instruction count have on slowing down Spark. Spark Scala is JVM based, whereas ECL is written in C++ which is native, and the aded overhead of using JVM can slow things down, and also result in bloated code due to considerable increase in the number of instructions. ALso, JVM does implicit memory management, while C++ does explicit memory management which can have a role to play here as well.

Other interesting angles to explore in this study is to evaluate the performance of both the engines for even larger sized datasets (~ 2 TB), larger clusters, and more RAM per node. We have also not covered join queries as part of the SQL workload here, but that is a very practically important kind of workload to benchmark the performance on.

The entire code and logs for this project can be found at https://github.ncsu.edu/CSC591-DIC/RamBharose

**Acknowledgements**

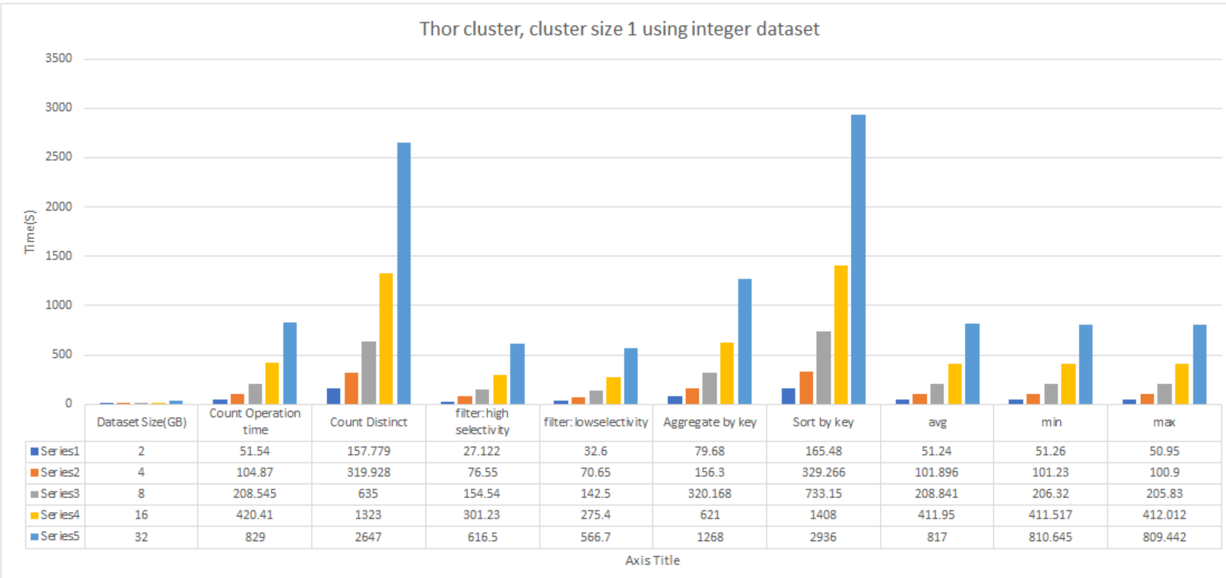
We would like to acknowledge Dr. Vincent Freeh for his guidance throughout the project. We would also like to thank Anshuman Goel for letting us avail his AWS credits to run the experiments.

**References**

[1] https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html
[2] http://aws.amazon.com/
[3] http://cdn.hpccsystems.com/whitepapers/wp_introduction_HPCC.pdf
[4] http://cdn.hpccsystems.com/whitepapers/hpccsystems_thor_spark.pdf
[5] http://cdn.hpccsystems.com/whitepapers/wp_ecl_overview.pdf
[6] https://0x0fff.com/spark-architecture/
[7] https://www.networkworld.com/article/3033888/linux/getting-system-insights-with-dstat.html

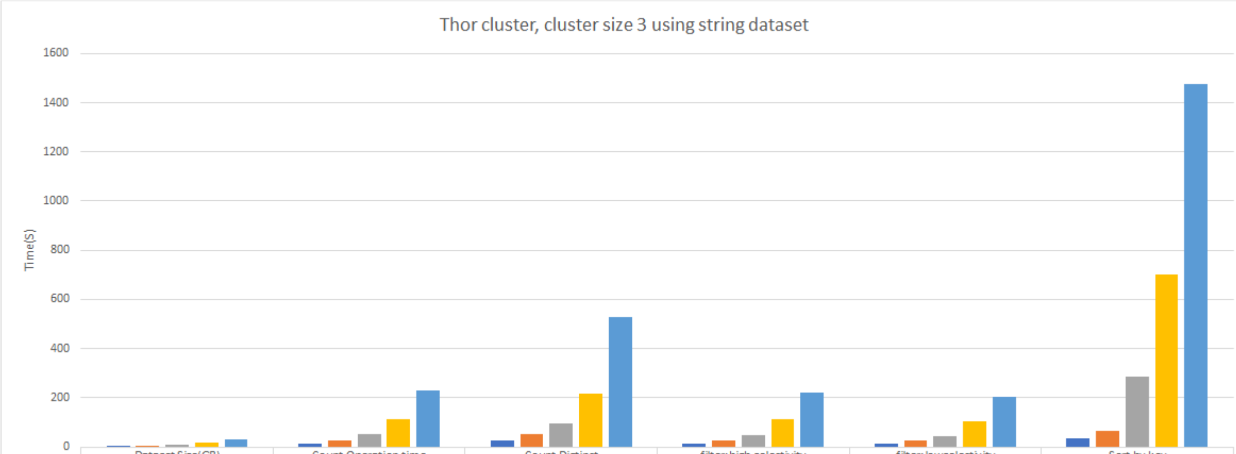
## Appendix A

## Graphs

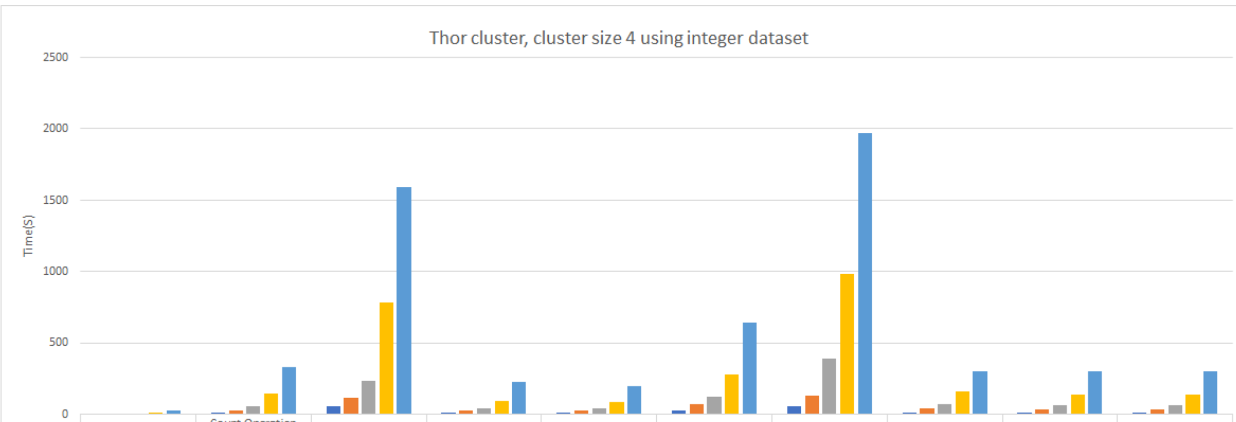



| | Dataset Size(GB) | Count Operation time | Count Distinct | filter:high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|---|
| Series1 | 2 | 51.54 | 157.779 | 27.122 | 32.6 | 79.68 | 165.48 | 51.24 | 51.26 | 50.95 |
| Series2 | 4 | 104.87 | 319.928 | 76.55 | 70.65 | 156.3 | 329.266 | 101.896 | 101.23 | 100.9 |
| Series3 | 8 | 208.545 | 635 | 154.54 | 142.5 | 320.168 | 733.15 | 208.841 | 206.32 | 205.83 |
| Series4 | 16 | 420.41 | 1323 | 301.23 | 275.4 | 621 | 1408 | 411.95 | 411.517 | 412.012 |
| Series5 | 32 | 829 | 2647 | 616.5 | 566.7 | 1268 | 2936 | 817 | 810.645 | 809.442 |

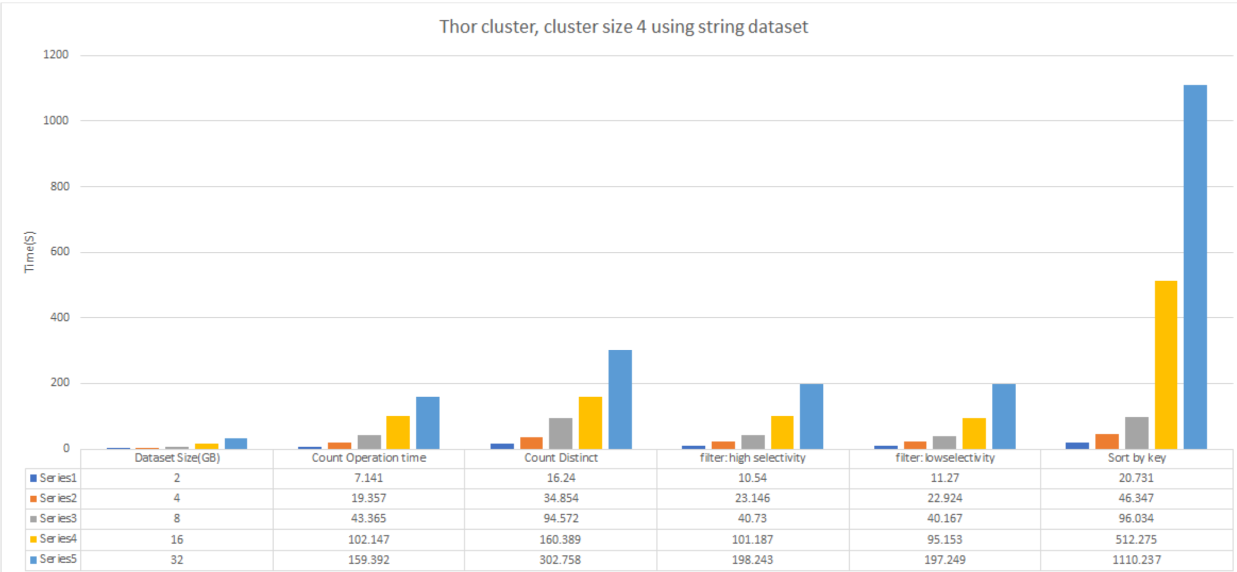

| | Dataset Size(GB) | Count Operation time | Count Distinct | filter:high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|---|
| Series1 | 2 | 27.264 | 40.45 | 23.45 | 25.63 | 41.22 |
| Series2 | 4 | 55.12 | 78.356 | 46.154 | 51.031 | 127.93 |
| Series3 | 8 | 110.35 | 173.68 | 93.59 | 99.11 | 322.67 |
| Series4 | 16 | 222.823 | 348.71 | 201.687 | 205.683 | 670 |
| Series5 | 32 | 442.98 | 700 | 403.72 | 410.601 | 1359 |



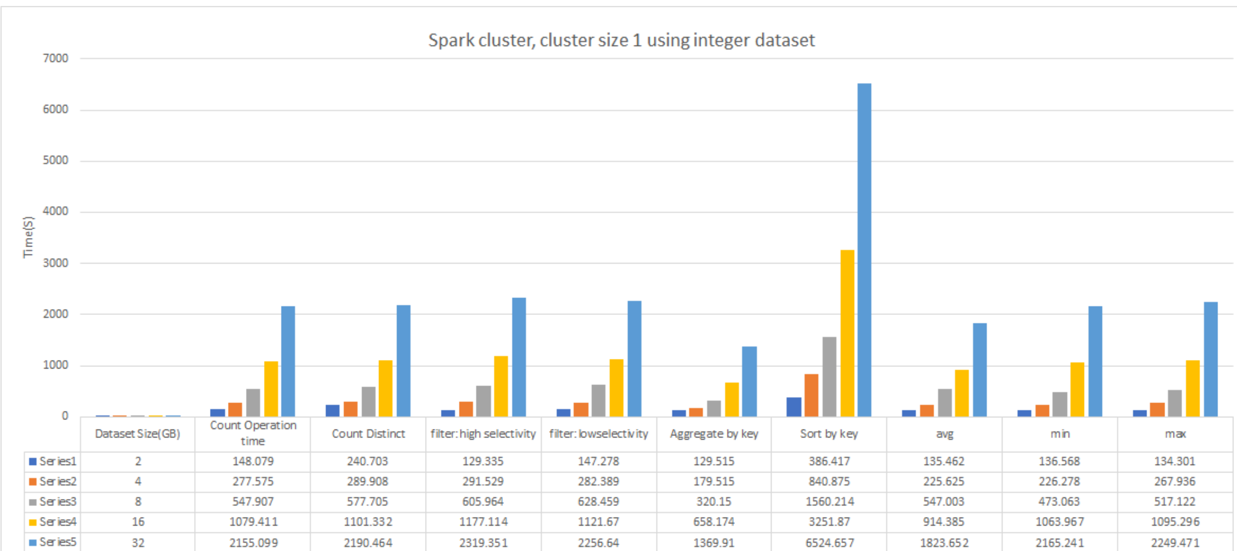

| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|---|
| Series1 | 2 | 13.26 | 25.517 | 13.11 | 12.52 | 33.65 |
| Series2 | 4 | 24.888 | 53.439 | 26.151 | 25.941 | 66.36 |
| Series3 | 8 | 51.588 | 97.232 | 48.512 | 43.175 | 285.484 |
| Series4 | 16 | 113.9 | 217.566 | 112.657 | 103.974 | 700 |
| Series5 | 32 | 229.623 | 526.33 | 218.791 | 205.163 | 1475.31 |




| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|---|
| Series1 | 2 | 13.33 | 56.145 | 12.93 | 12.56 | 29.743 | 59.56 | 12.96 | 14.994 | 13.841 |
| Series2 | 4 | 28.95 | 115.21 | 28.654 | 26.91 | 70.134 | 134.085 | 40.61 | 34.934 | 34.646 |
| Series3 | 8 | 57.83 | 239.32 | 46.856 | 46.38 | 125.584 | 390.64 | 71.56 | 65.513 | 66.953 |
| Series4 | 16 | 148.21 | 782 | 94.587 | 90.28 | 283.118 | 984.713 | 163.89 | 139.156 | 137.566 |
| Series5 | 32 | 332.31 | 1590 | 227.039 | 196.39 | 646 | 1974 | 301.265 | 302.888 | 306.672 |

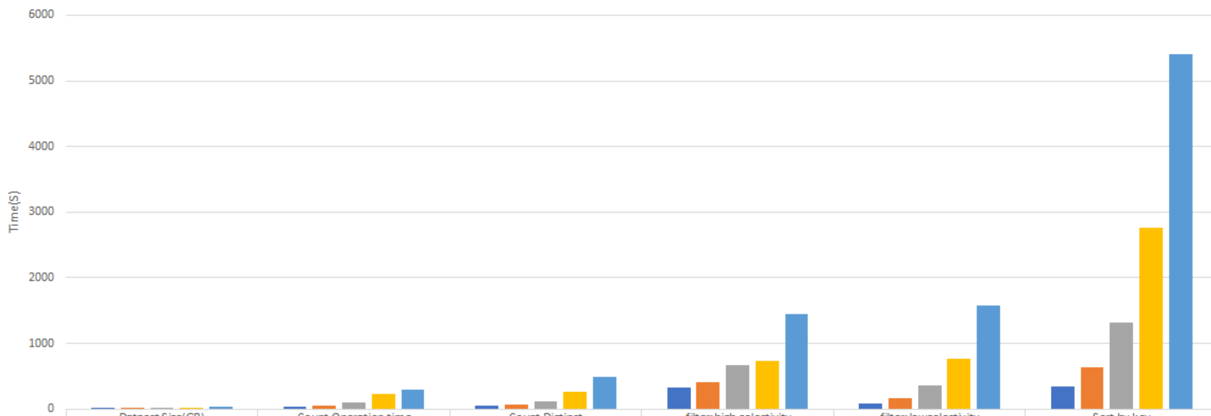



| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|---|
| Series1 | 2 | 7.141 | 16.24 | 10.54 | 11.27 | 20.731 |
| Series2 | 4 | 19.357 | 34.854 | 23.146 | 22.924 | 46.347 |
| Series3 | 8 | 43.365 | 94.572 | 40.73 | 40.167 | 96.034 |
| Series4 | 16 | 102.147 | 160.389 | 101.187 | 95.153 | 512.275 |
| Series5 | 32 | 159.392 | 302.758 | 198.243 | 197.249 | 1110.237 |



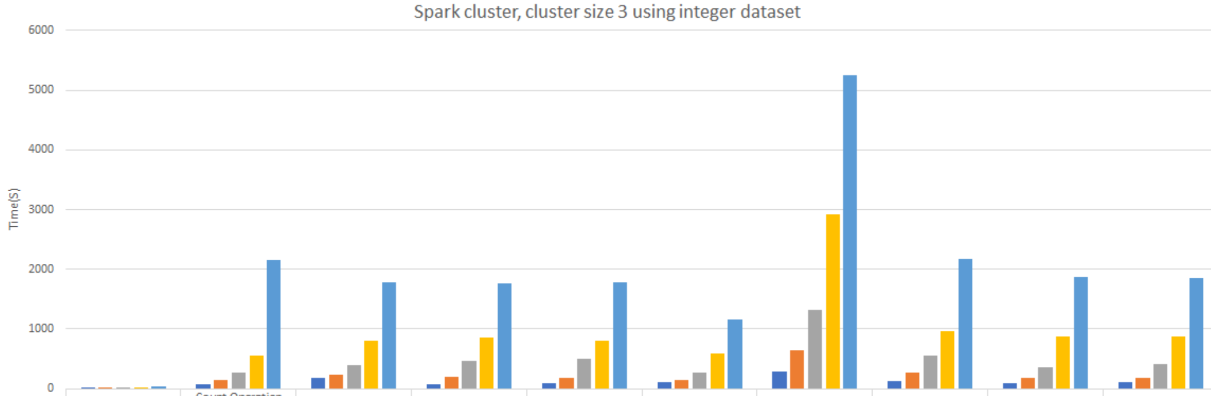

| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|---|
| Series1 | 2 | 148.079 | 240.703 | 129.335 | 147.278 | 129.515 | 386.417 | 135.462 | 136.568 | 134.301 |
| Series2 | 4 | 277.575 | 289.908 | 291.529 | 282.389 | 179.515 | 840.875 | 225.625 | 226.278 | 267.936 |
| Series3 | 8 | 547.907 | 577.705 | 605.964 | 628.459 | 320.15 | 1560.214 | 547.003 | 473.063 | 517.122 |
| Series4 | 16 | 1079.411 | 1101.332 | 1177.114 | 1121.67 | 658.174 | 3251.87 | 914.385 | 1063.967 | 1095.296 |
| Series5 | 32 | 2155.099 | 2190.464 | 2319.351 | 2256.64 | 1369.91 | 6524.657 | 1823.652 | 2165.241 | 2249.471 |

Spark cluster, cluster size 1 using string dataset

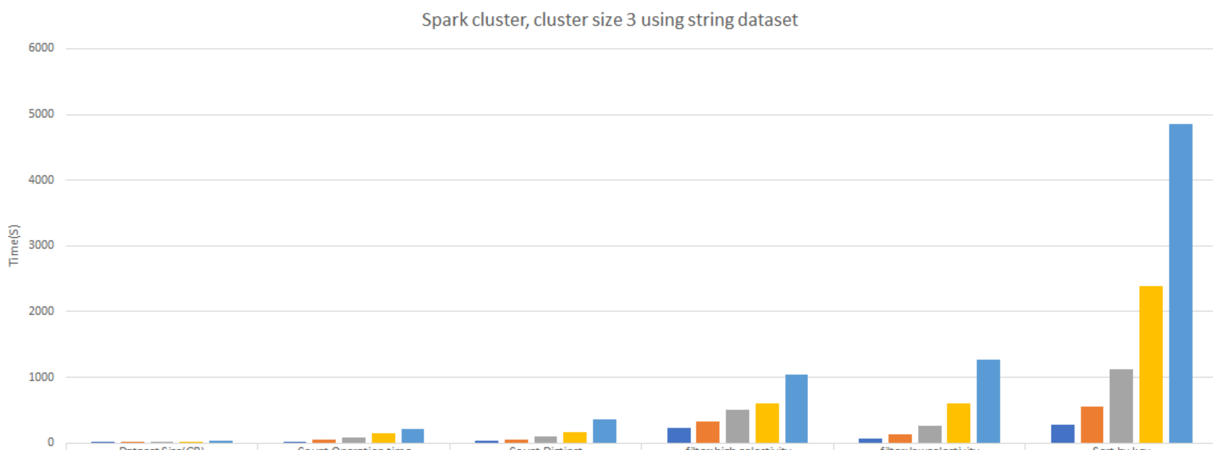



| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|---|
| Series1 | 2 | 31.764 | 46.83 | 320.423 | 89.654 | 350.121 |
| Series2 | 4 | 57.431 | 60.99 | 400.027 | 170.364 | 640.457 |
| Series3 | 8 | 106.818 | 113.548 | 667.195 | 361.983 | 1320.867 |
| Series4 | 16 | 222.12 | 263.684 | 735.274 | 756.318 | 2760.877 |
| Series5 | 32 | 299.081 | 488.939 | 1449.611 | 1568.99 | 5400.645 |

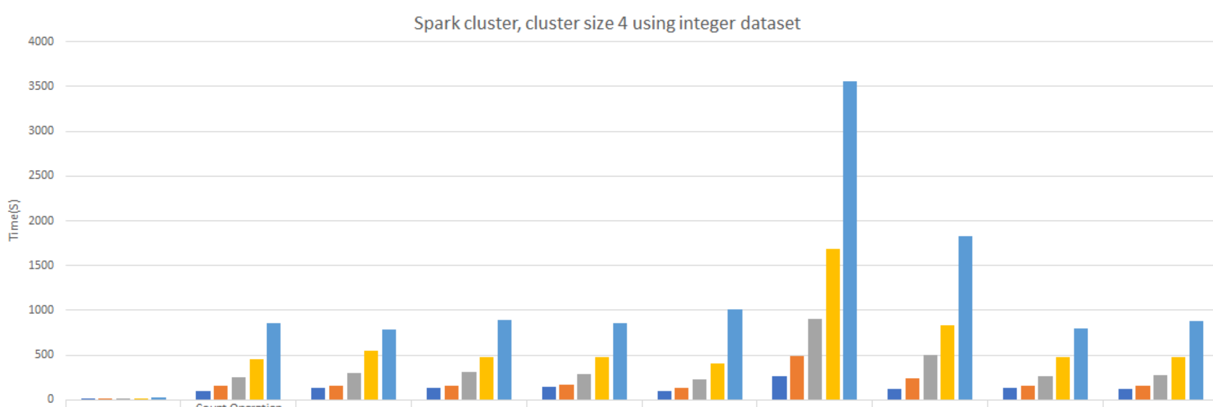
Spark cluster, cluster size 3 using integer dataset




| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|---|
| Series1 | 2 | 78.15 | 180.968 | 79.185 | 98.654 | 104.575 | 296.324 | 135.654 | 96.674 | 104.51 |
| Series2 | 4 | 147.211 | 229.645 | 201.954 | 184.545 | 145.35 | 645.521 | 268.194 | 185.357 | 187.334 |
| Series3 | 8 | 267.157 | 388.454 | 458.122 | 501.98 | 262.28 | 1328.419 | 547.003 | 357.653 | 406.357 |
| Series4 | 16 | 559.326 | 797.358 | 858.363 | 802.77 | 584.254 | 2915.947 | 965.264 | 877.368 | 868.938 |
| Series5 | 32 | 2155.099 | 1781.157 | 1764.554 | 1786.357 | 1154.03 | 5257.157 | 2166.386 | 1867.687 | 1844.45 |

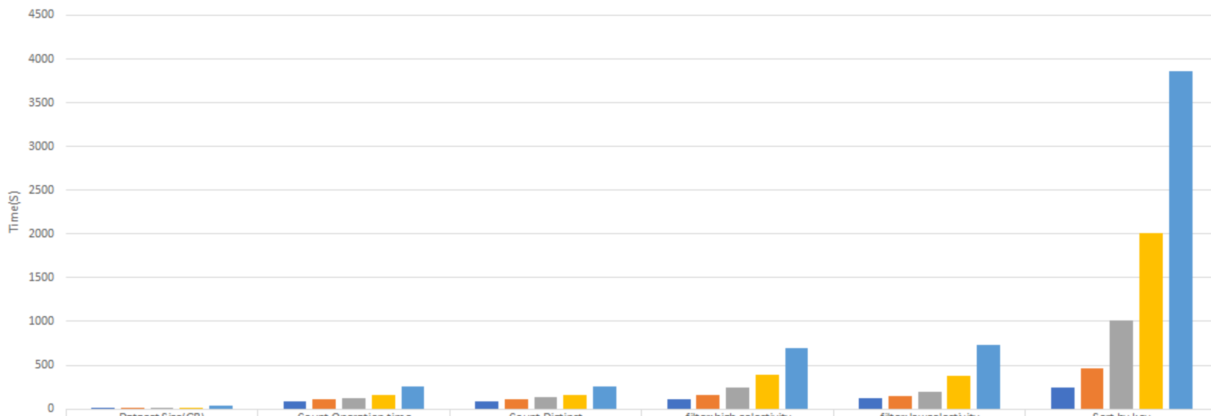



| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|---|
| Series1 | 2 | 25.157 | 36.387 | 220.543 | 68.544 | 270.15 |
| Series2 | 4 | 43.852 | 52.154 | 328.848 | 135.271 | 550.152 |
| Series3 | 8 | 78.957 | 90.675 | 508.584 | 261.217 | 1125.199 |
| Series4 | 16 | 148.856 | 167.76 | 605.814 | 598.884 | 2380.978 |
| Series5 | 32 | 220.367 | 351.864 | 1041.608 | 1268.89 | 4860.122 |



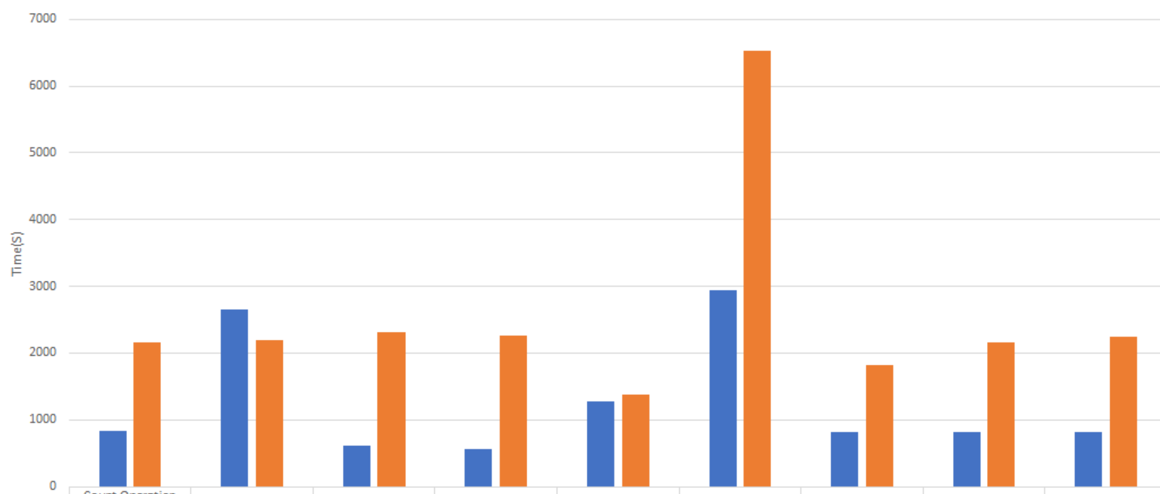

| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|---|
| Series1 | 2 | 98.254 | 130.947 | 131.848 | 139.878 | 95.351 | 265.123 | 120.245 | 130.465 | 119.342 |
| Series2 | 4 | 158.767 | 161.325 | 161.551 | 162.894 | 130.57 | 490.613 | 242.352 | 150.641 | 160.39 |
| Series3 | 8 | 249.861 | 294.32 | 310.159 | 291.455 | 225.22 | 901.37 | 498.958 | 267.844 | 270.75 |
| Series4 | 16 | 450.682 | 551.941 | 480.756 | 479.122 | 402.688 | 1683.587 | 835.665 | 475.36 | 481.623 |
| Series5 | 32 | 860.327 | 780.246 | 885.944 | 860.784 | 1006.033 | 3556.14 | 1825.358 | 791.33 | 885.367 |

Spark cluster, cluster size 4 using string dataset

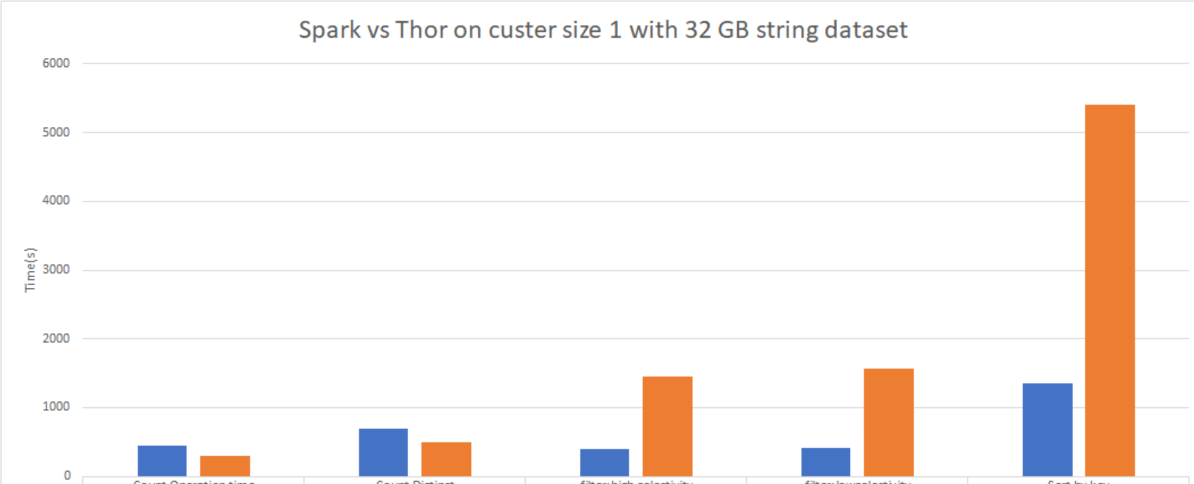



| | Dataset Size(GB) | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|---|
| Series1 | 2 | 82.645 | 80.761 | 110.248 | 118.26 | 242.36 |
| Series2 | 4 | 106.298 | 110.38 | 155.899 | 152.664 | 465.682 |
| Series3 | 8 | 126.58 | 140.847 | 241.127 | 201.844 | 1005.874 |
| Series4 | 16 | 164.251 | 165.369 | 390.545 | 380.154 | 2011.874 |
| Series5 | 32 | 254.946 | 251.793 | 695.427 | 725.757 | 3857.125 |

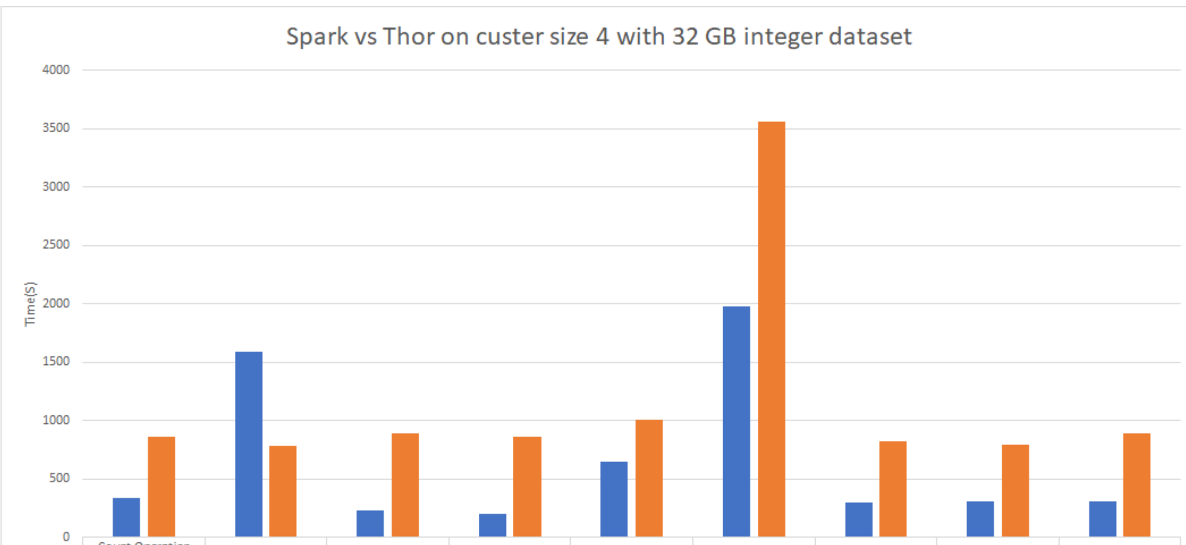
Spark vs Thor on custer size 1 with 32 GB integer dataset



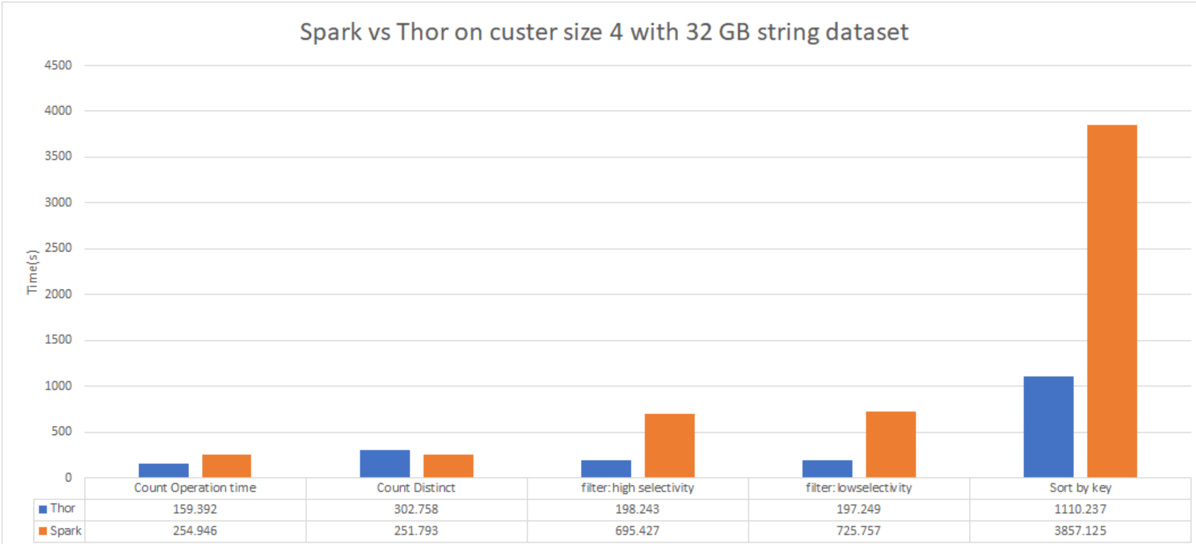

| | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|
| Thor | 829 | 2647 | 616.5 | 566.7 | 1268 | 2936 | 817 | 810.645 | 809.442 |
| Spark | 2155.099 | 2190.464 | 2319.351 | 2256.64 | 1369.91 | 6524.657 | 1823.652 | 2165.241 | 2249.471 |




| | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Sort by key |
|---|---|---|---|---|---|
| Thor | 442.98 | 700 | 403.72 | 410.601 | 1359 |
| Spark | 299.081 | 488.939 | 1449.611 | 1568.99 | 5400.645 |




| | Count Operation time | Count Distinct | filter: high selectivity | filter: lowselectivity | Aggregate by key | Sort by key | avg | min | max |
|---|---|---|---|---|---|---|---|---|---|
| Thor | 332.31 | 1590 | 227.039 | 196.39 | 646 | 1974 | 301.265 | 302.888 | 306.672 |
| Spark | 860.327 | 780.246 | 885.944 | 860.784 | 1006.033 | 3556.14 | 825.358 | 791.33 | 885.367 |


Spark vs Thor on custer size 4 with 32 GB string dataset
Time(s)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Count Operation time
Count Distinct
filter: high selectivity
filter: lowselectivity
Sort by key
Thor
159.392
302.758
198.243
197.249
1110.237
Spark
254.946
251.793
695.427
725.757
3857.125

## Appendix B

### SQL Queries

The details of the queries in each of the workloads are as follows:
Filter queries:
FilterLowSelectivity: select * from wutable where _c0=\"19107\"
FilterHighSelectivity: select * from wutable where _c0=\"53800\"

Aggregate queries:
AggregateCountDistinct: select count(distinct _c0) from wutable
AggregateCountTotal: select count(_c0) from wutable
AggregateMin: select min(_c0) from wutable
AggregateMax: select max(_c0) from wutable
AggregateAvg: select avg(_c0) from wutable
AggregateGroupby: select count(_c1) from wutable group by _c0

Sort query:
SortRandom: select * from wutable order by _c0

### Sample Scripts

**Sample Scala script for Aggregate Groupby job:**

```
import org.apache.spark.{SparkContext, SparkConf}
import org.apache.spark.sql.types.StructType
import org.apache.spark.sql.types.{IntegerType, StringType}

object AggregateGroupby {
  def main(args: Array[String]) {
        val conf = new SparkConf().setAppName("AggregateGroupby Application")

        val sc = new SparkContext(conf)
        val sqlContext = new org.apache.spark.sql.SQLContext(sc)
        val fileName = args(0)
        val datatype = args(1) //1 for int, 0 for string

        def time[R](block: => R): R = {
        val t0 = System.nanoTime()
        val result = block  // call-by-name
        val t1 = System.nanoTime()
        println("Elapsed time: " + (t1 - t0) + "ns")
        result
        }

        val customSchema_int = new StructType().add("_c0", IntegerType).add("_c1", IntegerType)

        val customSchema_string = new StructType().add("_c0", IntegerType).add("_c1", StringType)
```

```
            if(datatype==1)
            {
            val             wudf            =            sqlContext.read.format("com.databricks.spark.csv").option("header",
"false").schema(customSchema_int).load(fileName)
            wudf.registerTempTable("wutable")
            }
            else
            {
            val             wudf            =            sqlContext.read.format("com.databricks.spark.csv").option("header",
"false").schema(customSchema_string).load(fileName)
            wudf.registerTempTable("wutable")

            }

            ///////aggregate queries

            //group by
            val groupby_int = time{sqlContext.sql("select count(_c1) from wutable group by _c0")}
            val c = groupby_int.count()
            println(s"Query Finished - there are $c lines here.")

  //StoppingSpark context
  sc.stop()
 }
}
```

## ECL script for Aggregate Groupby:

```
rs := {integer key, integer fill};

#WORKUNIT('name', 'aggr_2');
dataset_name2 := 'IntegerDataSet_2GB';
outdata2 := DATASET(dataset_name2, rs, THOR);
outdata22 := TABLE(outdata2, {key, SUM(GROUP, fill)}, key, FEW);
OUTPUT(COUNT(NOFOLD(outdata22)));

#WORKUNIT('name', 'aggr_4');
dataset_name4 := 'IntegerDataSet_4GB';
outdata4 := DATASET(dataset_name4, rs, THOR);
outdata44 := TABLE(outdata4, {key, SUM(GROUP, fill)}, key, FEW);
OUTPUT(COUNT(NOFOLD(outdata44)));

#WORKUNIT('name', 'aggr_8');
dataset_name8 := 'IntegerDataSet_8GB';
outdata8 := DATASET(dataset_name8, rs, THOR);
outdata88 := TABLE(outdata8, {key, SUM(GROUP, fill)}, key, FEW);
OUTPUT(COUNT(NOFOLD(outdata88)));

#WORKUNIT('name', 'aggr_16');
dataset_name16 := 'IntegerDataSet_16GB';
outdata16 := DATASET(dataset_name16, rs, THOR);
outdata1616 := TABLE(outdata16, {key, SUM(GROUP, fill)}, key, FEW);
OUTPUT(COUNT(NOFOLD(outdata1616)));
```

```
#WORKUNIT('name', 'aggr_32');
dataset_name32 := 'IntegerDataSet_32GB';
outdata32 := DATASET(dataset_name32, rs, THOR);
outdata3232 := TABLE(outdata32, {key, SUM(GROUP, fill)}, key, FEW);
OUTPUT(COUNT(NOFOLD(outdata3232)));
```

### Appendix C: Observation Table

| Cluster Type | #Nodes | Total Cluster RAM(GB) | Dataset Size(GB) | Data Type | Opeartion | Time(s) | CPU % |
|---|---|---|---|---|---|---|---|
| THOR | 1 | 8 | 2 | Integer | Total Count | 51.54 | 51 |
| THOR | 1 | 8 | 4 | Integer | Total Count | 104.87 | 51 |
| THOR | 1 | 8 | 8 | Integer | Total Count | 208.545 | 51 |
| THOR | 1 | 8 | 16 | Integer | Total Count | 420.41 | 51 |
| THOR | 1 | 8 | 32 | Integer | Total Count | 829 | 51 |
| THOR | 1 | 8 | 2 | Integer | Distinct Count | 157.779 | 56 |
| THOR | 1 | 8 | 4 | Integer | Distinct Count | 319.928 | 56 |
| THOR | 1 | 8 | 8 | Integer | Distinct Count | 635 | 56 |
| THOR | 1 | 8 | 16 | Integer | Distinct Count | 1323 | 56 |
| THOR | 1 | 8 | 32 | Integer | Distinct Count | 2647 | 56 |
| THOR | 1 | 8 | 2 | Integer | Filter: High Selectivity | 27.122 | 52 |
| THOR | 1 | 8 | 4 | Integer | Filter: High Selectivity | 76.55 | 52 |
| THOR | 1 | 8 | 8 | Integer | Filter: High Selectivity | 154.54 | 52 |
| THOR | 1 | 8 | 16 | Integer | Filter: High Selectivity | 301.23 | 52 |
| THOR | 1 | 8 | 32 | Integer | Filter: High Selectivity | 616.5 | 52 |
| THOR | 1 | 8 | 2 | Integer | Filter: Low Selectivity | 32.6 | 52 |
| THOR | 1 | 8 | 4 | Integer | Filter: Low Selectivity | 70.65 | 52 |
| THOR | 1 | 8 | 8 | Integer | Filter: Low Selectivity | 142.5 | 52 |
| THOR | 1 | 8 | 16 | Integer | Filter: Low Selectivity | 275.4 | 52 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 1 | 8 | 32 | Integer | Filter: Low Selectivity | 566.7 | 52 |
| THOR | 1 | 8 | 2 | Integer | Aggregate by key | 79.68 | 58 |
| THOR | 1 | 8 | 4 | Integer | Aggregate by key | 156.3 | 58 |
| THOR | 1 | 8 | 8 | Integer | Aggregate by key | 320.168 | 58 |
| THOR | 1 | 8 | 16 | Integer | Aggregate by key | 621 | 58 |
| THOR | 1 | 8 | 32 | Integer | Aggregate by key | 1268 | 58 |
| THOR | 1 | 8 | 2 | Integer | Sort by key | 165.48 | 59 |
| THOR | 1 | 8 | 4 | Integer | Sort by key | 329.266 | 59 |
| THOR | 1 | 8 | 8 | Integer | Sort by key | 733.15 | 59 |
| THOR | 1 | 8 | 16 | Integer | Sort by key | 1408 | 59 |
| THOR | 1 | 8 | 32 | Integer | Sort by key | 2936 | 59 |
| THOR | 1 | 8 | 2 | Integer | Average value of Key | 51.24 | 55 |
| THOR | 1 | 8 | 4 | Integer | Average value of Key | 101.896 | 55 |
| THOR | 1 | 8 | 8 | Integer | Average value of Key | 208.841 | 55 |
| THOR | 1 | 8 | 16 | Integer | Average value of Key | 411.95 | 55 |
| THOR | 1 | 8 | 32 | Integer | Average value of Key | 817 | 55 |
| THOR | 1 | 8 | 2 | Integer | Min key | 51.26 | 52 |
| THOR | 1 | 8 | 4 | Integer | Min key | 101.23 | 52 |
| THOR | 1 | 8 | 8 | Integer | Min key | 206.32 | 52 |
| THOR | 1 | 8 | 16 | Integer | Min key | 411.517 | 52 |
| THOR | 1 | 8 | 32 | Integer | Min key | 810.645 | 52 |
| THOR | 1 | 8 | 2 | Integer | Max Key | 50.95 | 51 |
| THOR | 1 | 8 | 4 | Integer | Max Key | 100.9 | 51 |
| THOR | 1 | 8 | 8 | Integer | Max Key | 205.83 | 51 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 1 | 8 | 16 | Integer | Max Key | 412.012 | 51 |
| THOR | 1 | 8 | 32 | Integer | Max Key | 809.442 | 51 |
| THOR | 3 | 24 | 2 | Integer | Total Count | 23.951 | 51 |
| THOR | 3 | 24 | 4 | Integer | Total Count | 52.166 | 51 |
| THOR | 3 | 24 | 8 | Integer | Total Count | 102.823 | 51 |
| THOR | 3 | 24 | 16 | Integer | Total Count | 171.6 | 51 |
| THOR | 3 | 24 | 32 | Integer | Total Count | 445.21 | 51 |
| THOR | 3 | 24 | 2 | Integer | Distinct Count | 65.49 | 56 |
| THOR | 3 | 24 | 4 | Integer | Distinct Count | 144.3 | 56 |
| THOR | 3 | 24 | 8 | Integer | Distinct Count | 425 | 56 |
| THOR | 3 | 24 | 16 | Integer | Distinct Count | 880 | 56 |
| THOR | 3 | 24 | 32 | Integer | Distinct Count | 1821.23 | 56 |
| THOR | 3 | 24 | 2 | Integer | Filter: High Selectivity | 16.23 | 52 |
| THOR | 3 | 24 | 4 | Integer | Filter: High Selectivity | 31.21 | 52 |
| THOR | 3 | 24 | 8 | Integer | Filter: High Selectivity | 81.26 | 52 |
| THOR | 3 | 24 | 16 | Integer | Filter: High Selectivity | 150.64 | 52 |
| THOR | 3 | 24 | 32 | Integer | Filter: High Selectivity | 340.51 | 52 |
| THOR | 3 | 24 | 2 | Integer | Filter: Low Selectivity | 14.939 | 52 |
| THOR | 3 | 24 | 4 | Integer | Filter: Low Selectivity | 27.668 | 52 |
| THOR | 3 | 24 | 8 | Integer | Filter: Low Selectivity | 75.112 | 52 |
| THOR | 3 | 24 | 16 | Integer | Filter: Low Selectivity | 140.26 | 52 |
| THOR | 3 | 24 | 32 | Integer | Filter: Low Selectivity | 311.12 | 52 |
| THOR | 3 | 24 | 2 | Integer | Aggregate by key | 40.91 | 58 |
| THOR | 3 | 24 | 4 | Integer | Aggregate | 80.25 | 58 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | by key | | |
| THOR | 3 | 24 | 8 | Integer | Aggregate by key | 192.3 | 58 |
| THOR | 3 | 24 | 16 | Integer | Aggregate by key | 412.31 | 58 |
| THOR | 3 | 24 | 32 | Integer | Aggregate by key | 868.23 | 58 |
| THOR | 3 | 24 | 2 | Integer | Sort by key | 82.31 | 59 |
| THOR | 3 | 24 | 4 | Integer | Sort by key | 240.12 | 59 |
| THOR | 3 | 24 | 8 | Integer | Sort by key | 522.11 | 59 |
| THOR | 3 | 24 | 16 | Integer | Sort by key | 1174.15 | 59 |
| THOR | 3 | 24 | 32 | Integer | Sort by key | 2454 | 59 |
| THOR | 3 | 24 | 2 | Integer | Average value of Key | 17.795 | 55 |
| THOR | 3 | 24 | 4 | Integer | Average value of Key | 43.737 | 55 |
| THOR | 3 | 24 | 8 | Integer | Average value of Key | 74.816 | 55 |
| THOR | 3 | 24 | 16 | Integer | Average value of Key | 174.55 | 55 |
| THOR | 3 | 24 | 32 | Integer | Average value of Key | 366.65 | 55 |
| THOR | 3 | 24 | 2 | Integer | Min key | 20.31 | 52 |
| THOR | 3 | 24 | 4 | Integer | Min key | 44.93 | 52 |
| THOR | 3 | 24 | 8 | Integer | Min key | 81.21 | 52 |
| THOR | 3 | 24 | 16 | Integer | Min key | 178.54 | 52 |
| THOR | 3 | 24 | 32 | Integer | Min key | 371.26 | 52 |
| THOR | 3 | 24 | 2 | Integer | Max Key | 20.54 | 51 |
| THOR | 3 | 24 | 4 | Integer | Max Key | 43.52 | 51 |
| THOR | 3 | 24 | 8 | Integer | Max Key | 78.65 | 51 |
| THOR | 3 | 24 | 16 | Integer | Max Key | 169.81 | 51 |
| THOR | 3 | 24 | 32 | Integer | Max Key | 384.21 | 51 |
| THOR | 4 | 32 | 2 | Integer | Total Count | 13.33 | 51 |
| THOR | 4 | 32 | 4 | Integer | Total Count | 28.95 | 51 |
| THOR | 4 | 32 | 8 | Integer | Total Count | 57.83 | 51 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 4 | 32 | 16 | Integer | Total Count | 148.21 | 51 |
| THOR | 4 | 32 | 32 | Integer | Total Count | 332.31 | 51 |
| THOR | 4 | 32 | 2 | Integer | Distinct Count | 56.145 | 56 |
| THOR | 4 | 32 | 4 | Integer | Distinct Count | 115.21 | 56 |
| THOR | 4 | 32 | 8 | Integer | Distinct Count | 239.32 | 56 |
| THOR | 4 | 32 | 16 | Integer | Distinct Count | 782 | 56 |
| THOR | 4 | 32 | 32 | Integer | Distinct Count | 1590 | 56 |
| THOR | 4 | 32 | 2 | Integer | Filter: High Selectivity | 12.93 | 52 |
| THOR | 4 | 32 | 4 | Integer | Filter: High Selectivity | 28.654 | 52 |
| THOR | 4 | 32 | 8 | Integer | Filter: High Selectivity | 46.856 | 52 |
| THOR | 4 | 32 | 16 | Integer | Filter: High Selectivity | 94.587 | 52 |
| THOR | 4 | 32 | 32 | Integer | Filter: High Selectivity | 227.039 | 52 |
| THOR | 4 | 32 | 2 | Integer | Filter: Low Selectivity | 12.56 | 52 |
| THOR | 4 | 32 | 4 | Integer | Filter: Low Selectivity | 26.91 | 52 |
| THOR | 4 | 32 | 8 | Integer | Filter: Low Selectivity | 46.38 | 52 |
| THOR | 4 | 32 | 16 | Integer | Filter: Low Selectivity | 90.28 | 52 |
| THOR | 4 | 32 | 32 | Integer | Filter: Low Selectivity | 196.39 | 52 |
| THOR | 4 | 32 | 2 | Integer | Aggregate by key | 29.743 | 58 |
| THOR | 4 | 32 | 4 | Integer | Aggregate by key | 70.134 | 58 |
| THOR | 4 | 32 | 8 | Integer | Aggregate by key | 125.584 | 58 |
| THOR | 4 | 32 | 16 | Integer | Aggregate by key | 283.118 | 58 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 4 | 32 | 32 | Integer | Aggregate by key | 646 | 58 |
| THOR | 4 | 32 | 2 | Integer | Sort by key | 59.56 | 59 |
| THOR | 4 | 32 | 4 | Integer | Sort by key | 134.085 | 59 |
| THOR | 4 | 32 | 8 | Integer | Sort by key | 390.64 | 59 |
| THOR | 4 | 32 | 16 | Integer | Sort by key | 984.713 | 59 |
| THOR | 4 | 32 | 32 | Integer | Sort by key | 1974 | 59 |
| THOR | 4 | 32 | 2 | Integer | Average value of Key | 12.96 | 55 |
| THOR | 4 | 32 | 4 | Integer | Average value of Key | 40.61 | 55 |
| THOR | 4 | 32 | 8 | Integer | Average value of Key | 71.56 | 55 |
| THOR | 4 | 32 | 16 | Integer | Average value of Key | 163.89 | 55 |
| THOR | 4 | 32 | 32 | Integer | Average value of Key | 301.265 | 55 |
| THOR | 4 | 32 | 2 | Integer | Min key | 14.994 | 52 |
| THOR | 4 | 32 | 4 | Integer | Min key | 34.934 | 52 |
| THOR | 4 | 32 | 8 | Integer | Min key | 65.513 | 52 |
| THOR | 4 | 32 | 16 | Integer | Min key | 139.156 | 52 |
| THOR | 4 | 32 | 32 | Integer | Min key | 302.888 | 52 |
| THOR | 4 | 32 | 2 | Integer | Max Key | 13.841 | 51 |
| THOR | 4 | 32 | 4 | Integer | Max Key | 34.646 | 51 |
| THOR | 4 | 32 | 8 | Integer | Max Key | 66.953 | 51 |
| THOR | 4 | 32 | 16 | Integer | Max Key | 137.566 | 51 |
| THOR | 4 | 32 | 32 | Integer | Max Key | 306.672 | 51 |
| SPARK | 1 | 8 | 2 | Integer | Total Count | 148.079 | |
| SPARK | 1 | 8 | 4 | Integer | Total Count | 277.575 | |
| SPARK | 1 | 8 | 8 | Integer | Total Count | 547.907 | |
| SPARK | 1 | 8 | 16 | Integer | Total Count | 1079.411 | |
| SPARK | 1 | 8 | 32 | Integer | Total Count | 2155.099 | |
| SPARK | 1 | 8 | 2 | Integer | Distinct Count | 240.703 | |
| SPARK | 1 | 8 | 4 | Integer | Distinct | 289.908 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Count | | |
| SPARK | 1 | 8 | 8 | Integer | Distinct Count | 577.705 | |
| SPARK | 1 | 8 | 16 | Integer | Distinct Count | 1101.332 | |
| SPARK | 1 | 8 | 32 | Integer | Distinct Count | 2190.464 | |
| SPARK | 1 | 8 | 2 | Integer | Filter: High Selectivity | 129.335 | |
| SPARK | 1 | 8 | 4 | Integer | Filter: High Selectivity | 291.529 | |
| SPARK | 1 | 8 | 8 | Integer | Filter: High Selectivity | 605.964 | |
| SPARK | 1 | 8 | 16 | Integer | Filter: High Selectivity | 1177.114 | |
| SPARK | 1 | 8 | 32 | Integer | Filter: High Selectivity | 2319.351 | |
| SPARK | 1 | 8 | 2 | Integer | Filter: Low Selectivity | 147.278 | |
| SPARK | 1 | 8 | 4 | Integer | Filter: Low Selectivity | 282.389 | |
| SPARK | 1 | 8 | 8 | Integer | Filter: Low Selectivity | 628.459 | |
| SPARK | 1 | 8 | 16 | Integer | Filter: Low Selectivity | 1121.67 | |
| SPARK | 1 | 8 | 32 | Integer | Filter: Low Selectivity | 2256.64 | |
| SPARK | 1 | 8 | 2 | Integer | Aggregate by key | 129.515 | |
| SPARK | 1 | 8 | 4 | Integer | Aggregate by key | 179.515 | |
| SPARK | 1 | 8 | 8 | Integer | Aggregate by key | 320.15 | |
| SPARK | 1 | 8 | 16 | Integer | Aggregate by key | 658.174 | |
| SPARK | 1 | 8 | 32 | Integer | Aggregate by key | 1369.91 | |
| SPARK | 1 | 8 | 2 | Integer | Sort by key | 386.417 | |
| SPARK | 1 | 8 | 4 | Integer | Sort by key | 840.875 | |
| SPARK | 1 | 8 | 8 | Integer | Sort by key | 1560.214 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SPARK | 1 | 8 | 16 | Integer | Sort by key | 3251.87 | |
| SPARK | 1 | 8 | 32 | Integer | Sort by key | 6524.657 | |
| SPARK | 1 | 8 | 2 | Integer | Average value of Key | 135.462 | |
| SPARK | 1 | 8 | 4 | Integer | Average value of Key | 225.625 | |
| SPARK | 1 | 8 | 8 | Integer | Average value of Key | 547.003 | |
| SPARK | 1 | 8 | 16 | Integer | Average value of Key | 914.385 | |
| SPARK | 1 | 8 | 32 | Integer | Average value of Key | 1823.652 | |
| SPARK | 1 | 8 | 2 | Integer | Min key | 136.568 | |
| SPARK | 1 | 8 | 4 | Integer | Min key | 226.278 | |
| SPARK | 1 | 8 | 8 | Integer | Min key | 473.063 | |
| SPARK | 1 | 8 | 16 | Integer | Min key | 1063.967 | |
| SPARK | 1 | 8 | 32 | Integer | Min key | 2165.241 | |
| SPARK | 1 | 8 | 2 | Integer | Max Key | 134.301 | |
| SPARK | 1 | 8 | 4 | Integer | Max Key | 267.936 | |
| SPARK | 1 | 8 | 8 | Integer | Max Key | 517.122 | |
| SPARK | 1 | 8 | 16 | Integer | Max Key | 1095.296 | |
| SPARK | 1 | 8 | 32 | Integer | Max Key | 2249.471 | |
| SPARK | 3 | 24 | 2 | Integer | Total Count | 78.15 | |
| SPARK | 3 | 24 | 4 | Integer | Total Count | 147.211 | |
| SPARK | 3 | 24 | 8 | Integer | Total Count | 267.157 | |
| SPARK | 3 | 24 | 16 | Integer | Total Count | 559.326 | |
| SPARK | 3 | 24 | 32 | Integer | Total Count | 2155.099 | |
| SPARK | 3 | 24 | 2 | Integer | Distinct Count | 180.968 | |
| SPARK | 3 | 24 | 4 | Integer | Distinct Count | 229.645 | |
| SPARK | 3 | 24 | 8 | Integer | Distinct Count | 388.454 | |
| SPARK | 3 | 24 | 16 | Integer | Distinct Count | 797.358 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SPARK | 3 | 24 | 32 | Integer | Distinct Count | 1781.157 | |
| SPARK | 3 | 24 | 2 | Integer | Filter: High Selectivity | 79.185 | |
| SPARK | 3 | 24 | 4 | Integer | Filter: High Selectivity | 201.954 | |
| SPARK | 3 | 24 | 8 | Integer | Filter: High Selectivity | 458.122 | |
| SPARK | 3 | 24 | 16 | Integer | Filter: High Selectivity | 858.363 | |
| SPARK | 3 | 24 | 32 | Integer | Filter: High Selectivity | 1764.554 | |
| SPARK | 3 | 24 | 2 | Integer | Filter: Low Selectivity | 98.654 | |
| SPARK | 3 | 24 | 4 | Integer | Filter: Low Selectivity | 184.545 | |
| SPARK | 3 | 24 | 8 | Integer | Filter: Low Selectivity | 501.98 | |
| SPARK | 3 | 24 | 16 | Integer | Filter: Low Selectivity | 802.77 | |
| SPARK | 3 | 24 | 32 | Integer | Filter: Low Selectivity | 1786.357 | |
| SPARK | 3 | 24 | 2 | Integer | Aggregate by key | 104.575 | |
| SPARK | 3 | 24 | 4 | Integer | Aggregate by key | 145.35 | |
| SPARK | 3 | 24 | 8 | Integer | Aggregate by key | 262.28 | |
| SPARK | 3 | 24 | 16 | Integer | Aggregate by key | 584.254 | |
| SPARK | 3 | 24 | 32 | Integer | Aggregate by key | 1154.03 | |
| SPARK | 3 | 24 | 2 | Integer | Sort by key | 296.324 | |
| SPARK | 3 | 24 | 4 | Integer | Sort by key | 645.521 | |
| SPARK | 3 | 24 | 8 | Integer | Sort by key | 1328.419 | |
| SPARK | 3 | 24 | 16 | Integer | Sort by key | 2915.947 | |
| SPARK | 3 | 24 | 32 | Integer | Sort by key | 5257.157 | |
| SPARK | 3 | 24 | 2 | Integer | Average value of Key | 135.654 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SPARK | 3 | 24 | 4 | Integer | Average value of Key | 268.194 | |
| SPARK | 3 | 24 | 8 | Integer | Average value of Key | 547.003 | |
| SPARK | 3 | 24 | 16 | Integer | Average value of Key | 965.264 | |
| SPARK | 3 | 24 | 32 | Integer | Average value of Key | 2166.386 | |
| SPARK | 3 | 24 | 2 | Integer | Min key | 96.674 | |
| SPARK | 3 | 24 | 4 | Integer | Min key | 185.357 | |
| SPARK | 3 | 24 | 8 | Integer | Min key | 357.653 | |
| SPARK | 3 | 24 | 16 | Integer | Min key | 877.368 | |
| SPARK | 3 | 24 | 32 | Integer | Min key | 1867.687 | |
| SPARK | 3 | 24 | 2 | Integer | Max Key | 104.51 | |
| SPARK | 3 | 24 | 4 | Integer | Max Key | 187.334 | |
| SPARK | 3 | 24 | 8 | Integer | Max Key | 406.357 | |
| SPARK | 3 | 24 | 16 | Integer | Max Key | 868.938 | |
| SPARK | 3 | 24 | 32 | Integer | Max Key | 1844.45 | |
| SPARK | 4 | 32 | 2 | Integer | Total Count | 98.254 | |
| SPARK | 4 | 32 | 4 | Integer | Total Count | 158.767 | |
| SPARK | 4 | 32 | 8 | Integer | Total Count | 249.861 | |
| SPARK | 4 | 32 | 16 | Integer | Total Count | 450.682 | |
| SPARK | 4 | 32 | 32 | Integer | Total Count | 860.327 | |
| SPARK | 4 | 32 | 2 | Integer | Distinct Count | 130.947 | |
| SPARK | 4 | 32 | 4 | Integer | Distinct Count | 161.325 | |
| SPARK | 4 | 32 | 8 | Integer | Distinct Count | 294.32 | |
| SPARK | 4 | 32 | 16 | Integer | Distinct Count | 551.941 | |
| SPARK | 4 | 32 | 32 | Integer | Distinct Count | 780.246 | |
| SPARK | 4 | 32 | 2 | Integer | Filter: High Selectivity | 131.848 | |
| SPARK | 4 | 32 | 4 | Integer | Filter: High | 161.551 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Selectivity | | |
| SPARK | 4 | 32 | 8 | Integer | Filter: High Selectivity | 310.159 | |
| SPARK | 4 | 32 | 16 | Integer | Filter: High Selectivity | 480.756 | |
| SPARK | 4 | 32 | 32 | Integer | Filter: High Selectivity | 885.944 | |
| SPARK | 4 | 32 | 2 | Integer | Filter: Low Selectivity | 139.878 | |
| SPARK | 4 | 32 | 4 | Integer | Filter: Low Selectivity | 162.894 | |
| SPARK | 4 | 32 | 8 | Integer | Filter: Low Selectivity | 291.455 | |
| SPARK | 4 | 32 | 16 | Integer | Filter: Low Selectivity | 479.122 | |
| SPARK | 4 | 32 | 32 | Integer | Filter: Low Selectivity | 860.784 | |
| SPARK | 4 | 32 | 2 | Integer | Aggregate by key | 95.351 | |
| SPARK | 4 | 32 | 4 | Integer | Aggregate by key | 130.57 | |
| SPARK | 4 | 32 | 8 | Integer | Aggregate by key | 225.22 | |
| SPARK | 4 | 32 | 16 | Integer | Aggregate by key | 402.688 | |
| SPARK | 4 | 32 | 32 | Integer | Aggregate by key | 1006.033 | |
| SPARK | 4 | 32 | 2 | Integer | Sort by key | 265.123 | |
| SPARK | 4 | 32 | 4 | Integer | Sort by key | 490.613 | |
| SPARK | 4 | 32 | 8 | Integer | Sort by key | 901.37 | |
| SPARK | 4 | 32 | 16 | Integer | Sort by key | 1683.587 | |
| SPARK | 4 | 32 | 32 | Integer | Sort by key | 3556.14 | |
| SPARK | 4 | 32 | 2 | Integer | Average value of Key | 120.245 | |
| SPARK | 4 | 32 | 4 | Integer | Average value of Key | 242.352 | |
| SPARK | 4 | 32 | 8 | Integer | Average value of | 498.958 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | Key | | |
| SPARK | 4 | 32 | 16 | Integer | Average value of Key | 835.665 | |
| SPARK | 4 | 32 | 32 | Integer | Average value of Key | 1825.358 | |
| SPARK | 4 | 32 | 2 | Integer | Min key | 130.465 | |
| SPARK | 4 | 32 | 4 | Integer | Min key | 150.641 | |
| SPARK | 4 | 32 | 8 | Integer | Min key | 267.844 | |
| SPARK | 4 | 32 | 16 | Integer | Min key | 475.36 | |
| SPARK | 4 | 32 | 32 | Integer | Min key | 791.33 | |
| SPARK | 4 | 32 | 2 | Integer | Max Key | 119.342 | |
| SPARK | 4 | 32 | 4 | Integer | Max Key | 160.39 | |
| SPARK | 4 | 32 | 8 | Integer | Max Key | 270.75 | |
| SPARK | 4 | 32 | 16 | Integer | Max Key | 481.623 | |
| SPARK | 4 | 32 | 32 | Integer | Max Key | 885.367 | |
| THOR | 1 | 8 | 2 | String | Total Count | 27.264 | 51 |
| THOR | 1 | 8 | 4 | String | Total Count | 55.12 | 51 |
| THOR | 1 | 8 | 8 | String | Total Count | 110.35 | 51 |
| THOR | 1 | 8 | 16 | String | Total Count | 222.823 | 51 |
| THOR | 1 | 8 | 32 | String | Total Count | 442.98 | 51 |
| THOR | 1 | 8 | 2 | String | Distinct Count | 40.45 | 56 |
| THOR | 1 | 8 | 4 | String | Distinct Count | 78.356 | 56 |
| THOR | 1 | 8 | 8 | String | Distinct Count | 173.68 | 56 |
| THOR | 1 | 8 | 16 | String | Distinct Count | 348.71 | 56 |
| THOR | 1 | 8 | 32 | String | Distinct Count | 700 | 56 |
| THOR | 1 | 8 | 2 | String | Filter: High Selectivity | 23.45 | 52 |
| THOR | 1 | 8 | 4 | String | Filter: High Selectivity | 46.154 | 52 |
| THOR | 1 | 8 | 8 | String | Filter: High Selectivity | 93.59 | 52 |
| THOR | 1 | 8 | 16 | String | Filter: High Selectivity | 201.687 | 52 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 1 | 8 | 32 | String | Filter: High Selectivity | 403.72 | 52 |
| THOR | 1 | 8 | 2 | String | Filter: Low Selectivity | 25.63 | 52 |
| THOR | 1 | 8 | 4 | String | Filter: Low Selectivity | 51.031 | 52 |
| THOR | 1 | 8 | 8 | String | Filter: Low Selectivity | 99.11 | 52 |
| THOR | 1 | 8 | 16 | String | Filter: Low Selectivity | 205.683 | 52 |
| THOR | 1 | 8 | 32 | String | Filter: Low Selectivity | 410.601 | 52 |
| THOR | 1 | 8 | 2 | String | Sort by key | 41.22 | 58 |
| THOR | 1 | 8 | 4 | String | Sort by key | 127.93 | 58 |
| THOR | 1 | 8 | 8 | String | Sort by key | 322.67 | 58 |
| THOR | 1 | 8 | 16 | String | Sort by key | 670 | 58 |
| THOR | 1 | 8 | 32 | String | Sort by key | 1359 | 58 |
| THOR | 3 | 24 | 2 | String | Total Count | 13.26 | 59 |
| THOR | 3 | 24 | 4 | String | Total Count | 24.888 | 59 |
| THOR | 3 | 24 | 8 | String | Total Count | 51.588 | 59 |
| THOR | 3 | 24 | 16 | String | Total Count | 113.9 | 59 |
| THOR | 3 | 24 | 32 | String | Total Count | 229.623 | 59 |
| THOR | 3 | 24 | 2 | String | Distinct Count | 25.517 | 55 |
| THOR | 3 | 24 | 4 | String | Distinct Count | 53.439 | 55 |
| THOR | 3 | 24 | 8 | String | Distinct Count | 97.232 | 55 |
| THOR | 3 | 24 | 16 | String | Distinct Count | 217.566 | 55 |
| THOR | 3 | 24 | 32 | String | Distinct Count | 526.33 | 55 |
| THOR | 3 | 24 | 2 | String | Filter: High Selectivity | 13.11 | 52 |
| THOR | 3 | 24 | 4 | String | Filter: High Selectivity | 26.151 | 52 |
| THOR | 3 | 24 | 8 | String | Filter: High Selectivity | 48.512 | 52 |
| THOR | 3 | 24 | 16 | String | Filter: High Selectivity | 112.657 | 52 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 3 | 24 | 32 | String | Filter: High Selectivity | 218.791 | 52 |
| THOR | 3 | 24 | 2 | String | Filter: Low Selectivity | 12.52 | 51 |
| THOR | 3 | 24 | 4 | String | Filter: Low Selectivity | 25.941 | 51 |
| THOR | 3 | 24 | 8 | String | Filter: Low Selectivity | 43.175 | 51 |
| THOR | 3 | 24 | 16 | String | Filter: Low Selectivity | 103.974 | 51 |
| THOR | 3 | 24 | 32 | String | Filter: Low Selectivity | 205.163 | 51 |
| THOR | 3 | 24 | 2 | String | Sort by key | 33.65 | 51 |
| THOR | 3 | 24 | 4 | String | Sort by key | 66.36 | 51 |
| THOR | 3 | 24 | 8 | String | Sort by key | 285.484 | 51 |
| THOR | 3 | 24 | 16 | String | Sort by key | 700 | 51 |
| THOR | 3 | 24 | 32 | String | Sort by key | 1475.31 | 51 |
| THOR | 4 | 32 | 2 | String | Total Count | 7.141 | 51 |
| THOR | 4 | 32 | 4 | String | Total Count | 19.357 | 51 |
| THOR | 4 | 32 | 8 | String | Total Count | 43.365 | 51 |
| THOR | 4 | 32 | 16 | String | Total Count | 102.147 | 51 |
| THOR | 4 | 32 | 32 | String | Total Count | 159.392 | 51 |
| THOR | 4 | 32 | 2 | String | Distinct Count | 16.24 | 56 |
| THOR | 4 | 32 | 4 | String | Distinct Count | 34.854 | 56 |
| THOR | 4 | 32 | 8 | String | Distinct Count | 94.572 | 56 |
| THOR | 4 | 32 | 16 | String | Distinct Count | 160.389 | 56 |
| THOR | 4 | 32 | 32 | String | Distinct Count | 302.758 | 56 |
| THOR | 4 | 32 | 2 | String | Filter: High Selectivity | 10.54 | 52 |
| THOR | 4 | 32 | 4 | String | Filter: High Selectivity | 23.146 | 52 |
| THOR | 4 | 32 | 8 | String | Filter: High Selectivity | 40.73 | 52 |
| THOR | 4 | 32 | 16 | String | Filter: High Selectivity | 101.187 | 52 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| THOR | 4 | 32 | 32 | String | Filter: High Selectivity | 198.243 | 52 |
| THOR | 4 | 32 | 2 | String | Filter: Low Selectivity | 11.27 | 52 |
| THOR | 4 | 32 | 4 | String | Filter: Low Selectivity | 22.924 | 52 |
| THOR | 4 | 32 | 8 | String | Filter: Low Selectivity | 40.167 | 52 |
| THOR | 4 | 32 | 16 | String | Filter: Low Selectivity | 95.153 | 52 |
| THOR | 4 | 32 | 32 | String | Filter: Low Selectivity | 197.249 | 52 |
| THOR | 4 | 32 | 2 | String | Sort by key | 20.731 | 58 |
| THOR | 4 | 32 | 4 | String | Sort by key | 46.347 | 58 |
| THOR | 4 | 32 | 8 | String | Sort by key | 96.034 | 58 |
| THOR | 4 | 32 | 16 | String | Sort by key | 512.275 | 58 |
| THOR | 4 | 32 | 32 | String | Sort by key | 1110.237 | 58 |
| SPARK | 1 | 8 | 2 | String | Total Count | 31.764 | |
| SPARK | 1 | 8 | 4 | String | Total Count | 57.431 | |
| SPARK | 1 | 8 | 8 | String | Total Count | 106.818 | |
| SPARK | 1 | 8 | 16 | String | Total Count | 222.12 | |
| SPARK | 1 | 8 | 32 | String | Total Count | 299.081 | |
| SPARK | 1 | 8 | 2 | String | Distinct Count | 46.83 | |
| SPARK | 1 | 8 | 4 | String | Distinct Count | 60.99 | |
| SPARK | 1 | 8 | 8 | String | Distinct Count | 113.548 | |
| SPARK | 1 | 8 | 16 | String | Distinct Count | 263.684 | |
| SPARK | 1 | 8 | 32 | String | Distinct Count | 488.939 | |
| SPARK | 1 | 8 | 2 | String | Filter: High Selectivity | 320.423 | |
| SPARK | 1 | 8 | 4 | String | Filter: High Selectivity | 400.027 | |
| SPARK | 1 | 8 | 8 | String | Filter: High Selectivity | 667.195 | |
| SPARK | 1 | 8 | 16 | String | Filter: High Selectivity | 735.274 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SPARK | 1 | 8 | 32 | String | Filter: High Selectivity | 1449.611 | |
| SPARK | 1 | 8 | 2 | String | Filter: Low Selectivity | 89.654 | |
| SPARK | 1 | 8 | 4 | String | Filter: Low Selectivity | 170.364 | |
| SPARK | 1 | 8 | 8 | String | Filter: Low Selectivity | 361.983 | |
| SPARK | 1 | 8 | 16 | String | Filter: Low Selectivity | 756.318 | |
| SPARK | 1 | 8 | 32 | String | Filter: Low Selectivity | 1568.99 | |
| SPARK | 1 | 8 | 2 | String | Sort by key | 350.121 | |
| SPARK | 1 | 8 | 4 | String | Sort by key | 640.457 | |
| SPARK | 1 | 8 | 8 | String | Sort by key | 1320.867 | |
| SPARK | 1 | 8 | 16 | String | Sort by key | 2760.877 | |
| SPARK | 1 | 8 | 32 | String | Sort by key | 5400.645 | |
| SPARK | 3 | 24 | 2 | String | Total Count | 25.157 | |
| SPARK | 3 | 24 | 4 | String | Total Count | 43.852 | |
| SPARK | 3 | 24 | 8 | String | Total Count | 78.957 | |
| SPARK | 3 | 24 | 16 | String | Total Count | 148.856 | |
| SPARK | 3 | 24 | 32 | String | Total Count | 220.367 | |
| SPARK | 3 | 24 | 2 | String | Distinct Count | 36.387 | |
| SPARK | 3 | 24 | 4 | String | Distinct Count | 52.154 | |
| SPARK | 3 | 24 | 8 | String | Distinct Count | 90.675 | |
| SPARK | 3 | 24 | 16 | String | Distinct Count | 167.76 | |
| SPARK | 3 | 24 | 32 | String | Distinct Count | 351.864 | |
| SPARK | 3 | 24 | 2 | String | Filter: High Selectivity | 220.543 | |
| SPARK | 3 | 24 | 4 | String | Filter: High Selectivity | 328.848 | |
| SPARK | 3 | 24 | 8 | String | Filter: High Selectivity | 508.584 | |
| SPARK | 3 | 24 | 16 | String | Filter: High Selectivity | 605.814 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SPARK | 3 | 24 | 32 | String | Filter: High Selectivity | 1041.608 | |
| SPARK | 3 | 24 | 2 | String | Filter: Low Selectivity | 68.544 | |
| SPARK | 3 | 24 | 4 | String | Filter: Low Selectivity | 135.271 | |
| SPARK | 3 | 24 | 8 | String | Filter: Low Selectivity | 261.217 | |
| SPARK | 3 | 24 | 16 | String | Filter: Low Selectivity | 598.884 | |
| SPARK | 3 | 24 | 32 | String | Filter: Low Selectivity | 1268.89 | |
| SPARK | 3 | 24 | 2 | String | Sort by key | 270.15 | |
| SPARK | 3 | 24 | 4 | String | Sort by key | 550.152 | |
| SPARK | 3 | 24 | 8 | String | Sort by key | 1125.199 | |
| SPARK | 3 | 24 | 16 | String | Sort by key | 2380.978 | |
| SPARK | 3 | 24 | 32 | String | Sort by key | 4860.122 | |
| SPARK | 4 | 32 | 2 | String | Total Count | 82.645 | |
| SPARK | 4 | 32 | 4 | String | Total Count | 106.298 | |
| SPARK | 4 | 32 | 8 | String | Total Count | 126.58 | |
| SPARK | 4 | 32 | 16 | String | Total Count | 164.251 | |
| SPARK | 4 | 32 | 32 | String | Total Count | 254.946 | |
| SPARK | 4 | 32 | 2 | String | Distinct Count | 80.761 | |
| SPARK | 4 | 32 | 4 | String | Distinct Count | 110.38 | |
| SPARK | 4 | 32 | 8 | String | Distinct Count | 140.847 | |
| SPARK | 4 | 32 | 16 | String | Distinct Count | 165.369 | |
| SPARK | 4 | 32 | 32 | String | Distinct Count | 251.793 | |
| SPARK | 4 | 32 | 2 | String | Filter: High Selectivity | 110.248 | |
| SPARK | 4 | 32 | 4 | String | Filter: High Selectivity | 155.899 | |
| SPARK | 4 | 32 | 8 | String | Filter: High Selectivity | 241.127 | |
| SPARK | 4 | 32 | 16 | String | Filter: High Selectivity | 390.545 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| SPARK | 4 | 32 | 32 | String | Filter: High Selectivity | 695.427 | |
| SPARK | 4 | 32 | 2 | String | Filter: Low Selectivity | 118.26 | |
| SPARK | 4 | 32 | 4 | String | Filter: Low Selectivity | 152.664 | |
| SPARK | 4 | 32 | 8 | String | Filter: Low Selectivity | 201.844 | |
| SPARK | 4 | 32 | 16 | String | Filter: Low Selectivity | 380.154 | |
| SPARK | 4 | 32 | 32 | String | Filter: Low Selectivity | 725.757 | |
| SPARK | 4 | 32 | 2 | String | Sort by key | 242.36 | |
| SPARK | 4 | 32 | 4 | String | Sort by key | 465.682 | |
| SPARK | 4 | 32 | 8 | String | Sort by key | 1005.874 | |
| SPARK | 4 | 32 | 16 | String | Sort by key | 2011.874 | |
| SPARK | 4 | 32 | 32 | String | Sort by key | 3857.125 | |
| SPARK | 4 | 32 | 2 | String | Max Key | 85.144 | |
| SPARK | 4 | 32 | 4 | String | Max Key | 110.39 | |
| SPARK | 4 | 32 | 8 | String | Max Key | 135.57 | |
| SPARK | 4 | 32 | 16 | String | Max Key | 160.326 | |
| SPARK | 4 | 32 | 32 | String | Max Key | 250.94 | |
| SPARK | 4 | 32 | 2 | String | Min key | 84.872 | |
| SPARK | 4 | 32 | 4 | String | Min key | 95.612 | |
| SPARK | 4 | 32 | 8 | String | Min key | 120.35 | |
| SPARK | 4 | 32 | 16 | String | Min key | 140.484 | |
| SPARK | 4 | 32 | 32 | String | Min key | 180.367 | |