# C SUPPLEMENTARY

## C.1 Performance on Pareto data

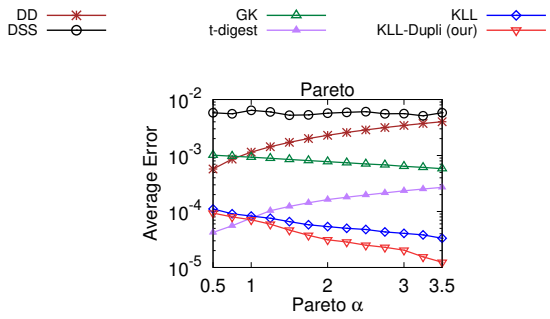


**Figure 16: Comparison by varying Pareto data**

We conduct an experiment on synthetic data following a typical heavy-tailed distribution, Pareto distribution. The data have a scale parameter of $x_m = 1$, a relative precision of 1.0005 for duplicate data, and a shape parameter $\alpha$ varied from 0.5 to 3.5. As shown in the figure 16, the benefits of our proposal increase as $\alpha$ increases and the tail becomes less heavy.

## C.2 Performance on applications

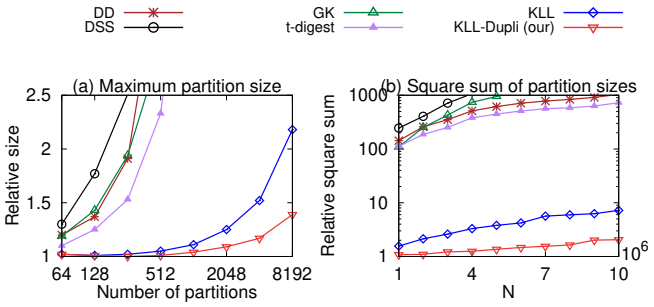


**Figure 17: Performance of approximate quantiles in range partitioning data for (a) parallel computing and (b) histogram sort**

We conducted experiments on the application of range partitioning data to illustrate the benefits of the proposed KLL-Dupli. As in [m1], Quantiles are applied to partition data by value ranges where ideal equi-depth histograms result in equal-sized partitions while approximate quantiles result in skewed partitions. Results of approximate quantiles for (a) parallel computing and (b) histogram sort are shown in the figure 17.

In the application of parallel computing like MapReduce [m2] tasks, the overall runtime of reducers is dominated by the largest partition. As shown in the figure (a), when partitioning 2E7 data in Price dataset with a 256KB sketch, our KLL-Dupli can result in the maximum partition which is 64% the size of the vanilla KLL. In this case, our proposal benefits the slowest task in parallel computing.

In the application of a typical histogram sort algorithm Flash-Sort[m3], data are range-partitioned and partitions are sorted with insertion sort. The performance can be evaluated by the square sum of partition sizes, i.e., the total number of required comparisons. The figure (b) shows the performance in Price dataset with a 256KB sketch and $0.1N$ partitions by varying data size $N$. In this case, square sum of partition sizes of our KLL-Dupli can be below 30% of that of baselines and benefits the histogram sort.

[m1] Yannis E. Ioannidis. The History of Histograms (abridged). VLDB 2003

[m2] Jeffrey Dean and Sanjay Ghemawat. 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 1 (2008)

[m3] Karl-Dietrich Neubert. 1998. The Flashsort1 Algorithm. Dr. Dobb's Journal (02 1998), 123–125

## C.3 Performance on non-duplicate data

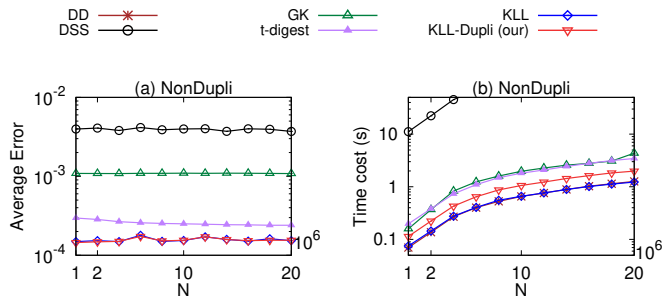


**Figure 18: Comparison by varying queried data size of non-duplicate data**

We conduct experiments on synthetic non-duplicate data by varying the queried data size as shown in Figure 18, where the memory limit is 64KB. The synthetic non-duplicate dataset consists of i.i.d. samples drawn from a distribution similar to the $\mathcal{D}_{hard}$ in [9]. The distribution is $\mathcal{X} \sim (-1)^b \cdot 10^{(2 \cdot R^8 - 1) \cdot E_{max}}$ and the detailed meanings of parameters can be found in [9]. In this dataset, DDSketch has unbounded error as the distribution is not one-sided long-tailed. Compared with KLL, KLL-Dupli has the same estimation error but takes 59% more time. The result of the same estimation error does not violate our analysis in Proposition 4.3, while the result of 59% more time cost is slightly better than that (67%) in datasets with heavy duplicates in Section 6.3.2.

## C.4 New baseline ReqSketch

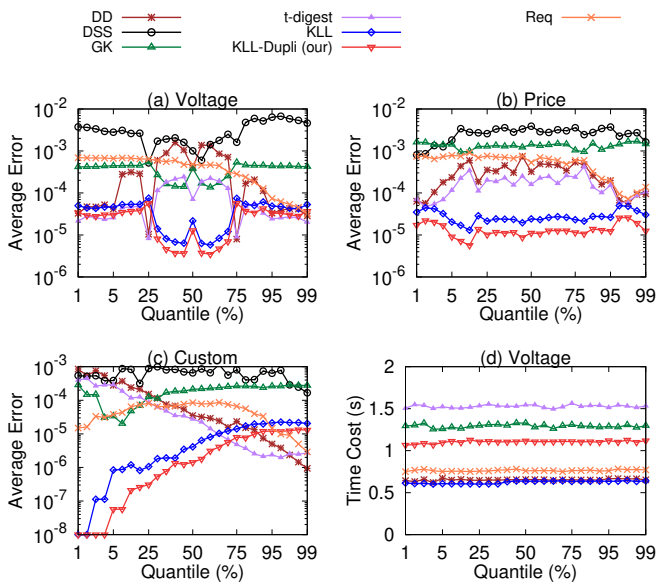


**Figure 19: Comparison by varying queried quantile, where the ReqSketch is added**

We compare add ReqSketch as a baseline in the experiment varying queried quantile, since ReqSketch is designed to be more accurate on extreme quantiles. In brief it performs well only for quite extreme quantiles like 99%.