; revised ; date of current version 10 December, 2024.

Supplementary Material: Fast Distributed Selection with Graphics Processing Units

JEFFREY D. BLANCHARD*, RUIZHE FU†, AND TRISTAN KNOTH‡

[1]Mathematics Department, Grinnell College, Grinnell, IA 50112 USA
[2]Computer Science Department, Grinnell College, Grinnell, IA 50112 USA and
Fu Foundation School of Engineering and Applied Science, Columbia University, New York, NY 10027 USA
[3]Persimmons, Inc., San Jose, CA, 95131 USA

CORRESPONDING AUTHOR: Jeffrey D. Blanchard (email: jeff@math.grinnell.edu).

ABSTRACT Supplementary Material for J.D. Blanchard, R. Fu, T. Knoth, *Fast Distributed Selection with Graphics Processing Units*, 2024, submitted.

INDEX TERMS Selection, Distributed Selection, Multiselection, Order Statistics, Parallel Selection, Graphics Processing Units, GPGPU, Distributed Computing.

Supplement to II. Distributed Multiselection

The algorithms DISTRIBUTEDBMS and DIBMS are relatively straightforward in principle: send control parameters and counts between the nodes and host while assigning data on the nodes to buckets until we identify the order statistics. The execution of these tasks requires meticulous bookkeeping with many parameters and counts. To aid a reader of the main paper [1], we reproduce the main article's Table 1 as Table S1 describing the size variables. Furthermore, Table S2 lists the variable names used in the description of the algorithm. For clarity, the data type (float, double, int), maximum size of the data structure (memory requirement), and location of the variable are given. In practice, the maximum size is the size of allocation so that required data can be pre-allocated at the start; these allocations are reused from iteration to iteration. The location is where the variable is created or lives, and we have indicated the primary communication required for use in the distributed setting. When a location is listed as *Each Node Only*, this means each code has its own distinct version of this variable and the contents of this variable are never transmitted. Finally, a basic description of the variable is given to aid the reader in keeping track of the relationships between these variables and the algorithm.

S	distributed data set
n	size (total number of elements) of data set
m	number of order statistics
q	number of nodes (including host node 0)
n_j	size of data on node $j \in \{0, \dots, q-1\}$
B	total number of buckets used in algorithms
P	number of pivot intervals
b_i	number of buckets given to pivot interval $i \in \{0, \dots, P-1\}$

TABLE S1. Reference list of variables.

Supplement to II.B. Distributed Iterative Bucket Multiselect

Each iteration of DIBMS retains all data assigned to one of the active buckets. When an active bucket has only one element, the order statistic is found and removed from the problem. Figure S1 gives a schematic of the iteration when there are $B = 8$ buckets. In reality, the data are not physically moved to a bucket; this figure demonstrates the concept of creating new buckets from one iteration to the next including reallocating the number of buckets per pivot interval, b_i.

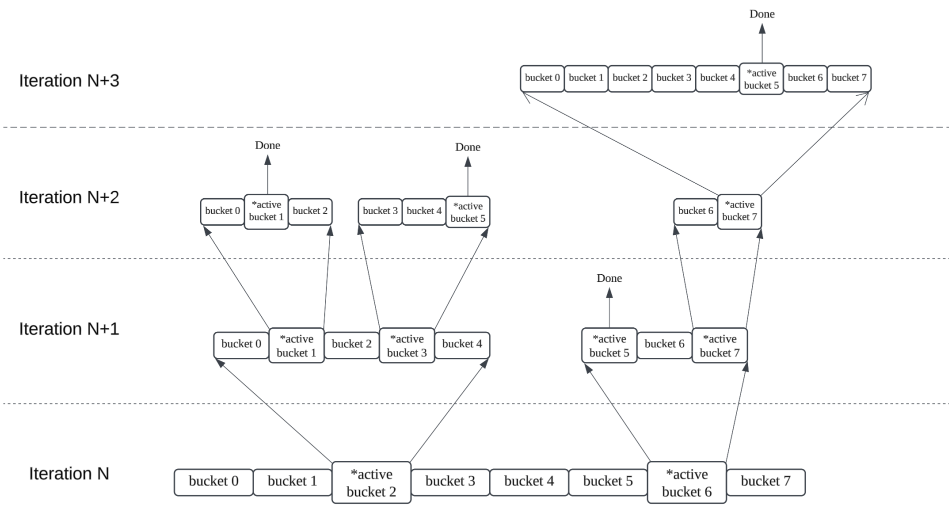


FIGURE S1. A schematic of how buckets are removed and created in DIBMS using the information from the previous iteration when searching for 4 order statistics using 8 buckets.

Variable Name	Data Type	Max Size	Location	Description
S	T	n	Distributed	The complete data distributed across all nodes.
`vec`	T	n_i	Each Node Only	The local vector holding the local elements of the data set.
`leftPivots` `rightPivots`	T	P	Created on Host Sent to All Nodes	The left and right endpoints of the pivot intervals. The number of pivot intervals, P, is 16 in Phase 1. During Phase 2, P is the number of unique active buckets from the previous phase or iteration.
`slopes`	T	P	Created on Host Sent to all Nodes	The vector of slopes defining a linear function whose integer part partitions the corresponding pivot interval into b_i buckets.
b_i	`int`	P	Created on Host Sent to all Nodes	A vector indicating the number of buckets assigned to each pivot interval.
`preCount`	`int`	P	Each Node Only	An inclusive sum of the number of buckets assigned to each pivot interval.
`buckets`	`int`	n_i	Each Node Only	A vector recording the bucket to which the corresponding data value in `vec` is assigned.
`CounterArray`	`int`	$B \times$ `numBlocks`	Each Node Sends 1 Column to Host	An array with position i, j indicating the number of elements in `vec` that the GPU block j assigned to bucket i.
`DesiredOrderStats`	`int`	m	Host Only	The list of order statistics which we are still searching for.
`UniqueActiveBuckets`	`int`	m	Created on Host Sent to all Nodes	The unique list of buckets that contain one of the `DesiredOrderStats`.
`UniqueActiveCounts`	`int`	m	Created on Host Sent to all Nodes	The number of entries in the full data set, S, that were assigned to the corresponding active bucket in `UniqueActiveBuckets`.
`ReducedVector`	T	n_i	Each Node Only	Elements from `vec` which were assigned to one of `UniqueActiveBuckets`.
`UpdatedOrderStats`	`int`	m	Host Only	The desired order statistics of `ReducedVector` that are the same values as the `DesiredOrderStats` from the last data set.

TABLE S2. Descriptions of variables. The data type T refers to primitive, numeric data types in CUDA-C, e.g. float, double, or (unsigned) integer. For a specific problem, T is fixed.

Supplement to III. Analysis

As stated in the main article [1], both DISTRIBUTEDBMS and DIBMS are expected to have massive communication savings over SORT&CHOOSE for any reasonable number of order statistics. Of course, if we asked for the same number of order statistics as we have buckets, there is a great likelihood that no problem reduction could occur. We have focused on 11, 101, 501, and 1001 order statistics as representative of the deciles, percentiles, $\frac{1}{5}$th-percentiles, and $\frac{1}{10}$th-percentiles. These numbers of order statistics provide incredibly accurate approximations of density functions for your data set.

Tables S3-S6 show the ratios of the expected communications cost for DIBMS to SORT&CHOOSE and DISTRIBUTEDBMS. The blue numbers indicate that DIBMS will require fewer communications than the algorithm it is compared to; red numbers indicate DIBMS will require more communication. The tables focus on a single number of order statistics, m, while looking at a range of data sizes and number of nodes. The tables capture the fact that the analysis counted every communication to each node. In network topologies where communication can happen simultaneously to all nodes, DIBMS could have an even greater advantage.

In Table S3 we see that the smaller number of order statistics indicates that **DISTRIBUTEDBMS** requires fewer communications. If transferring the data assigned to the 11 active buckets (expected to be less than 1% of the total data) is acceptable, **DISTRIBUTEDBMS** will be very effective. We see that as the data sizes grow, even a small number of order statistics will eventually be too expensive.

Note that in Tables S3-S6 there are very few occasions where we expect **DIBMS** to require more communication. Those occasions occur when the total data set is small and the number of nodes large. For 101 order statistics, **DIBMS** requires the fewest communications for even 8 nodes once $n = 2^{25}$. With 501 order statistics, that required data size drops to 2^{24} and falls to 2^{22} when we seek 1001 order statistics.

Supplement to IV. Performance Comparisons

In the main article, the empirical performance comparisons presented focused on data sets S drawn from the uniform distribution. In this supplementary material, we reproduce some of those results for ease of comparison to additional observations when the data in S is drawn from the normal and half normal distributions. The half normal distribution is obtained by taking the absolute value of a set of data drawn from the normal distribution. The overarching observation is that the algorithms' use of the kernel density estimator via small sampling in the first phase followed by redistribution of buckets in the second phase reduce the impact of the distribution from which the data is drawn. While problems with data drawn from the normal and half normal distributions typically take longer to solve, the run time variations across data distributions are not substantial: the run time from solving a problem created with normally distributed data is always within 30% of the run time of a problem from the uniform distribution.

We begin with a supplement to [1, Table 4] in Table S7. In this table we expand the range of order statistics and data size and show run time performance ratios for **DIBMS** to **SORT&CHOOSE** and to **DISTRIBUTEDBMS**. The run times displayed can be useful when interpreting the figures that follow.

In Figs. S2 and S3 we provide the data analogous to [1, Fig. 1], but for the normal and half normal distributions. We see in these two figures a nearly identical relationship between the algorithms with slightly longer run times on the normal and half normal distributions.

In Fig. S4, we examine the performance of **DIBMS** when selecting 101, 501, and 1001 order statistics from single precision data. We reproduce the results for uniform data from [1, Fig. 2] in (a) and include the same representation of performance when the data are from the (b) normal and (c) half normal distributions. Similarly, Fig. S5 includes a reproduction of [1, Fig. 3] in (a) with data drawn from the (b) normal and (c) half normal distributions. Due to limits on shared memory capacity, the double precision data was only tested to 501 order statistics. Data sets drawn from the half normal distribution were only tested in single precision. From Figs. S4 and S5 we observe that while there can be up to a thirty percent increase in run time, the overall relationships between the algorithms is consistent.

Finally, Fig. S6 produces [1, Fig. 4] but for double precision data. The double precision data requires more memory and therefore the results are only for 2^{25} elements per node. As expected, we see the same relationships between the performances on differing numbers of nodes. Both [1, Fig. 4] and Fig. S6 indicate that increasing the number of nodes increases the run time despite the ability to broadcast a message to all nodes simultaneously.

REFERENCES

[1] J. Blanchard, R. Fu, and T. Knoth, “Fast distributed selection with graphics processing units,” *Submitted*, 2024.

TABLE S3. Expected Communication Savings. The ratios of expected communications for DIBMS to SORT&CHOOSE and to DISTRIBUTEDBMS for 11 order statistics.

# OS	Data Size	# Nodes	Expected Communications Ratio	
m	$\log_2(n)$	q	$\frac{\text{DIBMS}}{\text{SORT\&CHOOSE}}$	$\frac{\text{DIBMS}}{\text{DISTRIBUTEDBMS}}$
11	22	2	**0.0110**	**3.8011**
		4	**0.0324**	**4.5243**
		8	**0.0753**	**4.9485**
	23	2	**0.0055**	**3.0836**
		4	**0.0162**	**3.9666**
		8	**0.0376**	**4.5937**
	24	2	**0.0027**	**2.2386**
		4	**0.0081**	**3.1820**
		8	**0.0188**	**4.0175**
	25	2	**0.0014**	**1.4461**
		4	**0.0041**	**2.2801**
		8	**0.0094**	**3.2119**
	26	2	**0.0007**	**0.8466**
		4	**0.0020**	**1.4551**
		8	**0.0047**	**2.2924**
	27	2	**0.0003**	**0.4629**
		4	**0.0010**	**0.8442**
		8	**0.0024**	**1.4578**
	28	2	**0.0002**	**0.2428**
		4	**0.0005**	**0.4589**
		8	**0.0012**	**0.8436**

TABLE S4. Expected Communication Savings. The ratios of expected communications for DIBMS to SORT&CHOOSE and to DISTRIBUTEDBMS for 101 order statistics.

# OS	Data Size	# Nodes	Expected Communications Ratio	
m	$\log_2(n)$	q	$\frac{\text{DIBMS}}{\text{SORT\&CHOOSE}}$	$\frac{\text{DIBMS}}{\text{DISTRIBUTEDBMS}}$
101	22	2	**0.0143**	**1.6948**
		4	**0.0421**	**2.7134**
		8	**0.0979**	**3.8955**
	23	2	**0.0071**	**0.9786**
		4	**0.0211**	**1.7007**
		8	**0.0489**	**2.7253**
	24	2	**0.0036**	**0.5303**
		4	**0.0105**	**0.9739**
		8	**0.0245**	**1.7025**
	25	2	**0.0018**	**0.2768**
		4	**0.0053**	**0.5251**
		8	**0.0122**	**0.9725**
	26	2	**0.0011**	**0.1713**
		4	**0.0032**	**0.3313**
		8	**0.0074**	**0.6353**
	27	2	**0.0005**	**0.0866**
		4	**0.0016**	**0.1691**
		8	**0.0037**	**0.3303**
	28	2	**0.0003**	**0.0435**
		4	**0.0008**	**0.0854**
		8	**0.0019**	**0.1685**

TABLE S5. Expected Communication Savings. The ratios of expected communications for DIBMS to SORT&CHOOSE and to DISTRIBUTEDBMS for 501 order statistics.

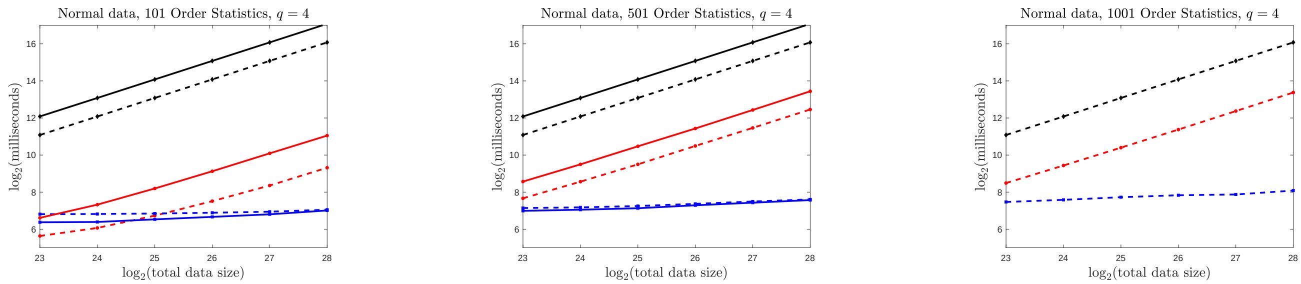
# OS	Data Size	# Nodes	Expected Communications Ratio	
m	$\log_2(n)$	q	$\frac{\text{DIBMS}}{\text{SORT\&CHOOSE}}$	$\frac{\text{DIBMS}}{\text{DISTRIBUTEDBMS}}$
501	22	2	**0.0228**	**0.6892**
		4	**0.0664**	**1.2587**
		8	**0.1536**	**2.2196**
	23	2	**0.0114**	**0.3579**
		4	**0.0332**	**0.6731**
		8	**0.0768**	**1.2516**
	24	2	**0.0057**	**0.1824**
		4	**0.0166**	**0.3487**
		8	**0.0384**	**0.6685**
	25	2	**0.0033**	**0.1058**
		4	**0.0095**	**0.2041**
		8	**0.0221**	**0.3978**
	26	2	**0.0016**	**0.0532**
		4	**0.0048**	**0.1030**
		8	**0.0110**	**0.2025**
	27	2	**0.0008**	**0.0267**
		4	**0.0024**	**0.0517**
		8	**0.0055**	**0.1022**
	28	2	**0.0004**	**0.0133**
		4	**0.0012**	**0.0259**
		8	**0.0028**	**0.0513**

TABLE S6. Expected Communication Savings. The ratios of expected communications for DIBMS to SORT&CHOOSE and to DISTRIBUTEDBMS for 1001 order statistics.

# OS	Data Size	# Nodes	Expected Communications Ratio	
m	$\log_2(n)$	q	$\frac{\text{DIBMS}}{\text{SORT\&CHOOSE}}$	$\frac{\text{DIBMS}}{\text{DISTRIBUTEDBMS}}$
1001	22	2	**0.0336**	**0.5266**
		4	**0.0964**	**0.9721**
		8	**0.2222**	**1.7880**
	23	2	**0.0168**	**0.2689**
		4	**0.0482**	**0.5053**
		8	**0.1111**	**0.9611**
	24	2	**0.0084**	**0.1359**
		4	**0.0241**	**0.2578**
		8	**0.0555**	**0.4993**
	25	2	**0.0047**	**0.0762**
		4	**0.0134**	**0.1452**
		8	**0.0310**	**0.2839**
	26	2	**0.0023**	**0.0382**
		4	**0.0067**	**0.0730**
		8	**0.0155**	**0.1434**
	27	2	**0.0012**	**0.0191**
		4	**0.0034**	**0.0366**
		8	**0.0077**	**0.0721**
	28	2	**0.0006**	**0.0106**
		4	**0.0019**	**0.0202**
		8	**0.0043**	**0.0399**

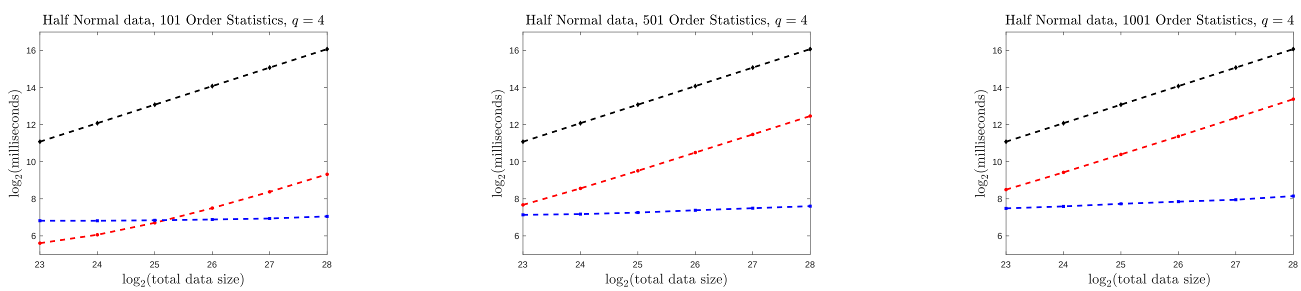
TABLE S7. selecting percentiles, $1/5$-percentiles, and $1/10$-percentiles: mean timings (ms) and performance ratios for DIBMS to SORT&CHOOSE and DISTRIBUTEDBMS.

Data Type			Float					
Data Distribution			Uniform			Normal		
Data Size	# Nodes	# OS	Run time (ms)	Observed Ratios		Run time (ms)	Observed Ratios	
n	q	m	DIBMS	DIBMS / SORT&CHOOSE	DIBMS / DISTRIBUTEDBMS	DIBMS	DIBMS / SORT&CHOOSE	DIBMS / DISTRIBUTEDBMS
2^{23}	4	101	114.80	**0.0529**	**1.9820**	112.32	**0.0518**	**2.2605**
		501	142.84	**0.0659**	**0.8042**	141.22	**0.0651**	**0.6946**
		1001	183.51	**0.0846**	**0.5562**	177.24	**0.0817**	**0.4934**
2^{24}	4	101	97.09	**0.0224**	**1.1472**	112.95	**0.0261**	**1.6830**
		501	146.09	**0.0337**	**0.4483**	144.74	**0.0334**	**0.3824**
		1001	188.62	**0.0435**	**0.3008**	191.89	**0.0443**	**0.2769**
2^{25}	4	101	98.55	**0.0114**	**0.6984**	115.01	**0.0133**	**1.0826**
		501	148.04	**0.0171**	**0.2392**	152.33	**0.0176**	**0.2106**
		1001	195.15	**0.0225**	**0.1585**	211.71	**0.0244**	**0.1568**
2^{26}	4	101	101.82	**0.0059**	**0.4106**	118.44	**0.0068**	**0.6485**
		501	139.26	**0.0080**	**0.1143**	165.26	**0.0095**	**0.1148**
		1001	203.07	**0.0117**	**0.0835**	227.91	**0.0131**	**0.0858**
2^{27}	4	101	106.07	**0.0031**	**0.2256**	123.10	**0.0036**	**0.3753**
		501	140.99	**0.0041**	**0.0586**	180.43	**0.0052**	**0.0638**
		1001	208.34	**0.0060**	**0.0433**	234.49	**0.0068**	**0.0442**
2^{28}	4	101	116.59	**0.0017**	**0.1277**	132.87	**0.0019**	**0.2084**
		501	153.87	**0.0022**	**0.0324**	194.97	**0.0028**	**0.0346**
		1001	211.29	**0.0031**	**0.0220**	271.31	**0.0039**	**0.0256**
Data Type			Double					
Data Distribution			Uniform			Normal		
Data Size	# Nodes	# OS	Run time (ms)	Observed Ratios		Run time (ms)	Observed Ratios	
n	q	m	DIBMS	DIBMS / SORT&CHOOSE	DIBMS / DISTRIBUTEDBMS	DIBMS	DIBMS / SORT&CHOOSE	DIBMS / DISTRIBUTEDBMS
2^{23}	4	101	97.14	**0.0224**	**0.9979**	82.99	**0.0191**	**0.8475**
		501	137.15	**0.0316**	**0.4104**	127.20	**0.0294**	**0.3347**
2^{24}	4	101	83.74	**0.0097**	**0.5605**	83.88	**0.0097**	**0.5235**
		501	132.06	**0.0153**	**0.2090**	132.64	**0.0153**	**0.1835**
2^{25}	4	101	88.99	**0.0051**	**0.3367**	92.45	**0.0053**	**0.3150**
		501	136.42	**0.0079**	**0.1104**	140.19	**0.0081**	**0.0987**
2^{26}	4	101	98.61	**0.0028**	**0.1961**	101.48	**0.0029**	**0.1817**
		501	149.28	**0.0043**	**0.0613**	156.46	**0.0045**	**0.0566**
2^{27}	4	101	112.49	**0.0016**	**0.1168**	111.67	**0.0016**	**0.1023**
		501	168.54	**0.0024**	**0.0350**	172.56	**0.0025**	**0.0312**
2^{28}	4	101	126.98	**0.0009**	**0.0665**	128.90	**0.0009**	**0.0608**
		501	188.51	**0.0014**	**0.0194**	190.13	**0.0014**	**0.0171**



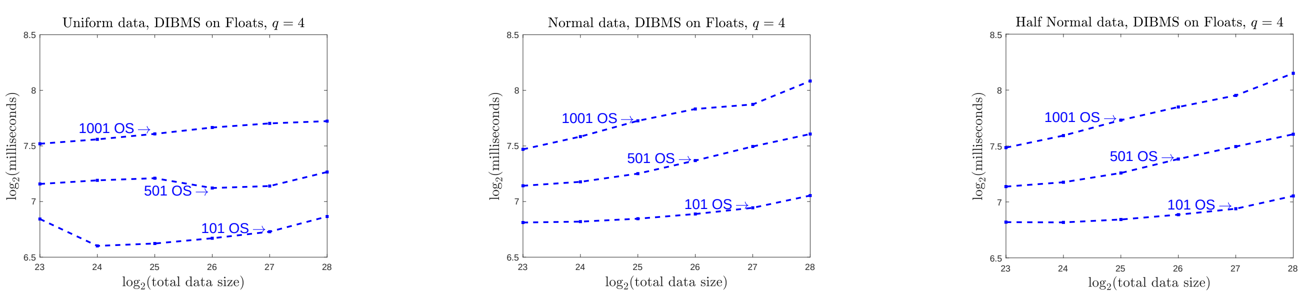
(a) (b) (c)

FIGURE S2. Time ($\log_2$(ms)) required for SORT&CHOOSE (black) DISTRIBUTEDBMS (red), DIBMS (blue) to find the (a) 101 percentiles, (b) 501 $\frac{1}{5}$th-percentiles, and (c) 1001 $\frac{1}{10}$th-percentiles for varying sizes of data sets, S, with the entries drawn from the normal distribution when the data is of type double (solid) or float (dashed).



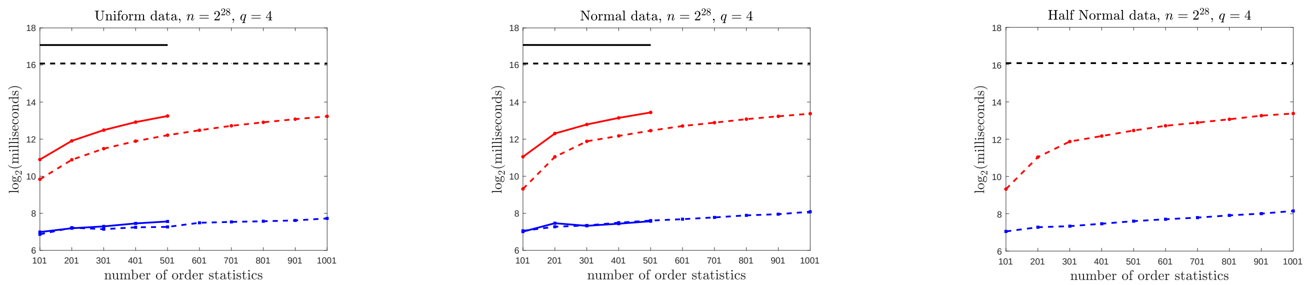
(a) (b) (c)

FIGURE S3. Time ($\log_2$(ms)) required for SORT&CHOOSE (black) DISTRIBUTEDBMS (red), DIBMS (blue) to find the (a) 101 percentiles, (b) 501 $\frac{1}{5}$th-percentiles, and (c) 1001 $\frac{1}{10}$th-percentiles for varying sizes of data sets, S, with the entries drawn from the half normal distribution when the data is of type float (dashed).



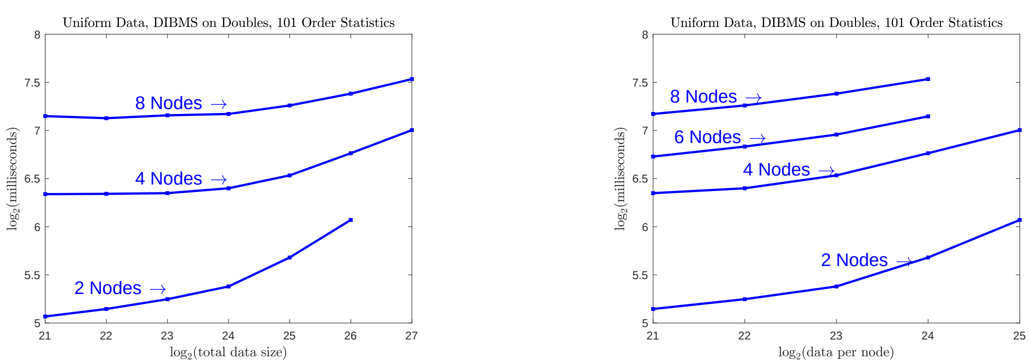
(a) (b) (c)

FIGURE S4. Time (ms) required for DIBMS (blue) to find the 101, 501, and 1001 uniformly spaced order statistics for varying length of vectors with the entries drawn from the (a) uniform, (b) normal, and (c) half normal vector distributions when the data is of type float (dashed).



(a) (b) (c)

FIGURE S5. Time ($\log_2$(ms) required for SORT&CHOOSE (black) DISTRIBUTEDBMS (red), DIBMS (blue) to find uniformly spaced order statistics for vectors of length of 2^{25} with the entries drawn from the (a) uniform, (b) normal, and (c) half normal vector distributions when the data is of type double (solid) or float (dashed).



(a) (b)

FIGURE S6. Time ($\log_2$(ms)) required for DIBMS (blue) to find 101 uniformly spaced order statistics for data sets distributed across 2, 4, 6, or 8 nodes. The data has elements drawn from the uniform distribution of type double (solid). (a) The horizontal axis is the total data size. (b) The horizontal axis is the size of data on each node.