Table I: Average Run Time of $DPPSort_{qsort}$ and $DPPSort_{STL}$ at $N = 200\times10^6$ (Uint64 Random data)

Algorithms	Run Time (sec)					
qsort	26.06					
STLSort	16.06					
	c=*N*/2	*c*=*N*/4	*c*=*N*/8	*c*=*N*/16	*c*=*N*/32	*c*=*N*/64
$DPPSort_{qsort}$	4.49	4.26	4.14	4.04	**3.97**	3.98
$DPPSort_{STL}$	2.98	**2.87**	**2.87**	2.93	3.02	3.05

Table II: Average Run Time (Uint64 data) of each distribution at $c = N/8$ of $DPPSort_{qsort}$, $DPPSort_{STL}$, *STLSort* and *qsort*

Algorithms	Distributions	Run Time (sec)		
		$N=50\times10^6$	$N=100\times10^6$	$N=200\times10^6$
$DPPSort_{STL}$	Random	0.66	1.37	2.87
	Reversed	**0.55**	**1.14**	2.50
	Nearly Sorted	0.56	1.15	**2.44**
	Few Unique	**0.55**	**1.14**	2.45
$DPPSort_{qsort}$	Random	0.91	1.93	4.14
	Reversed	**0.52**	**1.13**	**2.41**
	Nearly Sorted	0.73	1.54	3.30
	Few Unique	0.69	1.51	3.17
STLSort	Random	3.84	7.76	16.06
	Reversed	**0.56**	**1.15**	**2.40**
	Nearly Sorted	3.09	6.49	13.50
	Few Unique	1.54	3.13	6.45
qsort	Random	6.03	12.53	26.06
	Reversed	**1.72**	**3.57**	**7.46**
	Nearly Sorted	3.51	7.28	15.10
	Few Unique	3.97	8.15	16.76

Table III: Average Speedup of $DPPSort_{qsort}$ and $DPPSort_{STL}$ versus c (Uint32 Random)

Algorithms	$N(10^6)$	c					
		$N/2$	$N/4$	$N/8$	$N/16$	$N/32$	$N/64$
$DPPSort_{qsort}$	50	6.49	7.01	7.65	8.04	8.34	**8.59**
	100	6.35	7.05	7.54	7.92	8.14	**8.38**
	200	6.22	6.86	7.58	7.90	8.05	**8.26**
$DPPSort_{STL}$	50	5.22	5.78	6.10	6.33	**6.36**	6.33
	100	5.37	6.00	6.22	6.36	**6.47**	6.41
	200	5.46	6.12	6.25	6.33	**6.47**	6.33

Table IV: Average Speedup of $DPPSort_{qsort}$ and $DPPSort_{STL}$ versus c (Uint64 Random)

Algorithms	$N(10^6)$	c					
		$N/2$	$N/4$	$N/8$	$N/16$	$N/32$	$N/64$
$DPPSort_{qsort}$	50	6.02	6.41	6.64	6.76	**6.82**	6.78
	100	5.81	6.25	6.48	6.55	6.66	**6.69**
	200	5.80	6.12	6.29	6.44	**6.56**	6.55
$DPPSort_{STL}$	50	5.65	5.59	5.84	**5.88**	5.54	5.45
	100	5.49	**5.69**	**5.69**	5.61	5.50	5.33
	200	5.39	**5.60**	**5.60**	5.47	5.32	5.26

Table V: Comparison of Speedup per core and thread of Parallel Sorting Algorithms (DF: Deque-Free)

Algorithms	Speedup	core	thread	Speedup/core	Speedup/thread
DPPSort$_{qsort}$	6.82	4	8	**1.71**	**0.85**
DPPSort$_{STL}$	5.88	4	8	1.47	0.74
DF IntroSort [8]	8.1	16	16	0.51	0.51
MultiSort [9]	13.6	32	32	0.43	0.43
psort [11, 12]	11.00	24	24	0.46	0.46
Introqsort [13]	1.47	2	2	0.74	0.74
PPMQSort [16]	12.29	8	16	1.54	0.77
HDPSort [17]	2.49	4	8	0.62	0.31

Table VI: Perf results of $DPPSort_{qsort}$ and $DPPSort_{STL}$ at $N = 200 \times 10^6$ (Uint32) and $c = N/8$

Distributions	Algorithms	Run Time (sec)	cache misses	branch load misses
Random	$DPPSort_{STL}$	2.35	3.81×10^8	2.43×10^9
	$DPPSort_{qsort}$	3.64	5.61×10^8	2.52×10^9
	STLSort	14.68	2.26×10^8	2.50×10^9
	qsort	27.57	4.43×10^8	2.75×10^9
Reversed	$DPPSort_{STL}$	1.99	4.85×10^8	4.99×10^8
	$DPPSort_{qsort}$	1.53	4.77×10^8	3.18×10^7
	STLSort	2.18	1.65×10^8	2.35×10^7
	qsort	6.61	3.35×10^8	1.02×10^7
Nearly Sorted	$DPPSort_{STL}$	1.81	3.56×10^8	1.63×10^9
	$DPPSort_{qsort}$	2.65	5.47×10^8	1.24×10^9
	STLSort	12.14	2.24×10^8	1.90×10^9
	qsort	17.17	3.17×10^8	1.20×10^9
Few Unique	$DPPSort_{STL}$	1.47	3.72×10^8	7.13×10^8
	$DPPSort_{qsort}$	2.68	5.24×10^8	9.88×10^8
	STLSort	5.58	2.05×10^8	7.01×10^8
	qsort	17.69	4.35×10^8	9.88×10^8

Table VII: Perf results of $DPPSort_{qsort}$ and $DPPSort_{STL}$ at $N = 200 \times 10^6$ (Uint64) and $c = N/8$

Distributions	Algorithms	Run Time (sec)	cache misses	branch load misses
Random	$DPPSort_{STL}$	2.87	8.04×10^8	2.28×10^9
	$DPPSort_{qsort}$	4.14	1.20×10^9	2.49×10^9
	STLSort	16.06	4.65×10^8	2.25×10^9
	qsort	26.06	9.58×10^8	2.73×10^9
Reversed	$DPPSort_{STL}$	2.50	1.02×10^9	4.71×10^8
	$DPPSort_{qsort}$	2.41	9.84×10^8	3.81×10^7
	STLSort	2.39	3.04×10^8	1.87×10^7
	qsort	7.46	7.27×10^8	1.03×10^7
Nearly Sorted	$DPPSort_{STL}$	2.44	7.92×10^8	1.50×10^9
	$DPPSort_{qsort}$	3.30	1.17×10^9	1.24×10^9
	STLSort	13.50	4.74×10^8	1.81×10^9
	qsort	15.10	6.85×10^8	1.24×10^9
Few Unique	$DPPSort_{STL}$	2.45	7.53×10^8	6.75×10^8
	$DPPSort_{qsort}$	3.17	1.08×10^9	8.94×10^8
	STLSort	6.45	4.58×10^8	6.69×10^8
	qsort	16.76	9.36×10^8	1.00×10^9

Table VIII: Run time and standard deviation (STD) of our proposed vs *Hoare's partitioning algorithm* at $N = 200 \times 10^6$ (Uint32)

c	Proposed (sec)	*Hoare's* (sec)	Proposed STD	*Hoare's* STD
N/2	3.15	**3.14**	0.1662	**0.1367**
N/4	2.57	**2.51**	0.0997	**0.0869**
N/8	**2.24**	2.28	0.0970	**0.0770**
N/16	**2.14**	**2.14**	**0.0591**	0.0658
N/32	**2.14**	**2.14**	**0.0687**	0.0891
N/64	**2.10**	2.15	**0.0649**	0.0796