Appendix B: experimental data

Table of Contents

Legend

blue = relevant value (neutral)

green = lowest runtime
red = highest runtime
green / **red** = + lowest / highest runtime of full table

can be combined with all font colours:

zzz = (additional) dependency (speedup/slowdown columns)

yyy = + “better” value than previous table
yyy = + “worse” value than previous table
yyy = + same value as previous table

B.1 Test setup

test setup:

- `nvcc`/`nvprof` 10.1.168, driver 430.26
- Titan Xp (Pascal GP102, CC 6.1; 30 SM, 128 CUDA cores/SM → 3840 CUDA cores;
  96KiB shared memory per SM, 64Ki 32-bit registers per SM;
  max. warps per SM: 64 → 64 warps * 32 threads/warp = max. 2048 CUDA threads;
  max. occupancy only if 64Ki 32b registers / 2048 max. threads = 32 registers per thread
- `-arch sm_30`, L1 disabled for global loads (`-Xptxas -dlcm=cg`, default setting)
- reducing overhead `nvprof`: `--concurrent-kernels off --profile-api-trace none`

defaults (not for real-world data):

-

uncompressed:	number of entries: 2^{26} → table size: 256MiB;
compressed:	number of entries: 2^{25} → table size: 128MiB

- bucket size: 32

default input file: `04-08000000-0205-2-04002399-08088848.txt`:
8,000,000 vectors, four elements each; 4,002,399 unique vectors (→ duplication: 2.00);
excellent compression ratio: 0.51 (→ 8,088,848 elements in table)
→ size of elements in table:
uncompressed: ca. 61.0MiB (→ fill rate: 0.24), compressed: ca. 30.9MiB (→ fill rate: 0.24)

all:

- (y.yy) in tables means that y.yy is the expected value, assuming corresponding variables are independent
- all experiments ceteris paribus (as much as possible)
- all runtimes/number of rehashes: average of three runs + one warm-up round

'optimal execution configuration' tables:

- all block dimensions achieve the maximal occupancy that is possible for register usage
- all grid dimensions are multiple of number of SMs, *i.e.*, multiple of 30
- table size is adapted to each input (file) to achieve same default fill rate (see next)

'different input' tables:

- table size is adapted to each input (file) to achieve same default fill rate (unless otherwise stated)
- uncompressed: this requires taking into account the "lost" entries for vector length 3 (depends on bucket size)
- compressed: this requires taking into account the "lost" entries for vector length 1

'real-world data' table:

- optimal settings determined from previous experiments
- + variations: bucket size/block dimension/grid dimension one step up and one step down (if possible), in the case of lower runtime another step up and down, resp. + combinations; `compressed_lr_64` also tried without `_lp8` and `_s2`

B.2 Hash functions/replication-free `findOrPut()`

using old (smaller) random input files:
`04-1000000-100-2-0500225-1020450.txt` and `01-4000000-000-2-1998487-3996974.txt`

GPUexplore 2.0 (using larger prime constant (Mersenne prime 2^{31} - 1), allowing more buckets):

`uncompressed_fixed`:	32 registers → max. warps (per SM) 64 (→ occ. 1.00)
`uncompressed_fixed` (replication-free):	32 registers** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64`:	24 registers*** → max. warps (p. SM) 64 (→ occ. 1.00)

** + 8 bytes stack frame (`findOrPut()`)
*** + 24 bytes stack frame (`treeRec()`)

Pair-multiply-shift:

`uncompressed_fixed` (+ replication-free):	32 registers* → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64`:	24 registers*** → max. warps (p. SM) 64 (→ occ. 1.00)

* + 16 bytes stack frame (`findOrPut()`), 8 bytes stack frame (`hash()`)
*** + 24 bytes stack frame (`treeRec()`)

Pair-multiply-shift (parallel):

`uncompressed_fixed`:	32 registers* → max. warps (per SM) 64 (→ occ. 1.00)
`uncompressed_fixed` (replication-free):	32 registers** → max. warps (per SM) 64 (→ occ. 1.00)

* + 8 bytes stack frame (`findOrPut()`, `hash()`)
** + 8 bytes stack frame (`findOrPut()`)

[U-H/R] *Different hash functions/replication-free* `findOrPut()` *(uncompressed)*

using optimal execution configurations:
block dimension: 256
grid dimension:
60 (low f.r. 0.24 + bucket sizes 1/2; `1394.1` *+ bucket size 4 +*
(replication-free: Pair-multiply-shift (parallel); replication: Pair-multiply-shift (+ parallel));
120 (`1394.1` *+ bucket size 16 (replication);* `1394.1` *+ bucket size 8 (replication-free);*
`1394.1` *+ bucket size 4 (replication-free; except Pair-multiply-shift (parallel));*
`wafer_stepper.1` *+ bucket size 8 (+ replication: Pair-multiply-shift (+ parallel)));*
240

table parameters	**number of rehashes (runtime (ms))**		
	GPUexplore 2.0	**Pair-multiply-shift**	**Pair-multiply-shift (parallel)**
*default (vector length: 4) – **replication 1.00***			
> default table size/fill rate (0.24) <			
default bucket size (32)	0.0M (1.42)	0.0M (1.24)	0.0M (1.13)
bucket size: 4	0.1M (0.72)	0.1M (0.72)	0.1M (0.72)
optimal bucket size: 8	0.0M (0.70)	0.0M (0.70)	0.0M (0.69)
> high fill rate: 0.79 <			
default bucket size (32)	0.1M (1.50)	0.1M (1.30)	0.1M (1.19)
bucket size: 4	0.6M (0.98)	0.6M (1.02)	0.6M (0.95)
optimal bucket size: 8	0.3M (0.90)	0.3M (0.92)	0.3M (0.83)
replication-free*			
> default table size/fill rate (0.24) <			
default bucket size (32)	(1.40)	(1.23)	(1.10)
bucket size: 4	(0.70)	(0.71)	(0.70)
optimal bucket size: 8	(0.69)	(0.70)	(0.69)
> high fill rate: 0.79 <			
default bucket size (32)	(1.48)	(1.29)	(1.16)
bucket size: 4	(0.95)	(0.99)	(0.92)
optimal bucket size: 8	(0.88)	(0.89)	(0.83)

* no effect on number of rehashes

table continues on next page…

*lower vector length: 1 – **replication 1.00***			
> default table size/fill rate (0.24) <			
default bucket size (32)	0 (4.83)	0 (3.45)	0 (3.78)
bucket size: 1 (optimal)	0.3M (2.63)	0.5M (2.66)	0.4M (2.66)
> high fill rate: 0.79 <			
default bucket size (32)	0.0M (4.87)	0.0M (3.33)	0.0M (3.71)
bucket size: 1	3M (2.82)	3M (2.83)	2M (2.79)
optimal bucket size: 4	0.3M (2.34)	0.3M (2.29)	0.3M (2.29)
replication-free*			
> default table size/fill rate (0.24) <			
default bucket size (32)	(4.71)	(3.43)	(3.66)
bucket size: 1 (optimal)	(2.56)	(2.60)	(2.62)
> high fill rate: 0.79 <			
default bucket size (32)	(4.70)	(3.31)	(3.56)
bucket size: 1	(2.86)	(2.85)	(2.85)
optimal bucket size: 4	(2.34)	(2.33)	(2.35)

* no effect on number of rehashes

table continues on next page…

table size: 6GiB
amount of replication after ‘number of rehashes’

real-world data: `1394.1` *(vector length: 3) –* ***replication 1.18 – 1.50***			
default bucket size (32)	0.0M / 1.18 (31)	0 / 1.20 (30)	0 / 1.19 (30)
bucket size: 4	37M / 1.41 (45)	6M / 1.50 (32)	6M / 1.49 (32)
optimal bucket size: 16	2M / 1.30 (29)	0.0M / 1.42 (28)	0.0M / 1.39 (27)
replication-free			
default bucket size (32)	0.0M (31)	0 (28)	0 (29)
bucket size: 4	23M (37)	0.3M (28)	0.3M (25)
optimal bucket size: 16/8/4	0.7M (29)	0.0M (26)	

real-world data: `wafer_stepper.1` *(vector length: 8) –* ***replication 1.04 – 1.27***			
default bucket size (32)	0.1M / 1.04 (26)	0.0M / 1.05 (28)	0.0M / 1.07 (19)
bucket size: 8	5M / 1.11 (19)	1M / 1.23 (16)	1M / 1.27 (17)
optimal bucket size: 16/8/8	1M / 1.08 (17)		
replication-free			
default bucket size (32)	0.0M (26)	0 (25)	0 (19)
bucket size: 8	3M (17)	0.2M (15)	0.2M (16)
optimal bucket size: 16/8/8	0.6M (16)		

[Clr64-H] Different hash functions (compressed, less recursion, 64-bit)

using optimal execution configurations:
block dimension: 256
grid dimension:
30 (low fill rate 0.24/6GiB table + bucket sizes 2/4);
60 (high fill rate 0.79 + bucket sizes 2/4/8);
120 (bucket size 32)

table parameters	**number of rehashes (runtime (ms))**	
	GPUexplore 2.0	**Pair-multiply-shift**
default (vector length: 4)		
> default table size/fill rate (0.24) <		
default bucket size (32)	0 (3.38)	0 (3.17)
bucket size: 2 (optimal)	0.1M (1.24)	0.1M (1.22)
> high fill rate: 0.79 <		
default bucket size (32)	0.0M (3.36)	0.0M (3.13)
bucket size: 2	0.7M (1.51)	0.7M (1.46)
optimal bucket size: 4	0.3M (1.42)	0.3M (1.37)
lower vector length: 1		
> default table size/fill rate (0.24) <		
default bucket size (32)	0 (5.82)	0 (5.50)
bucket size: 2	0.4M (4.24)	0.3M (4.07)
optimal bucket size: 4	0.0M (3.91)	0.0M (3.88)
> high fill rate: 0.79 <		
default bucket size (32)	0.0M (5.77)	0.1M (5.52)
bucket size: 2	2M (6.48)	2M (6.46)
optimal bucket size: 8	0.4M (4.37)	0.3M (4.30)

table continues on next page…

table size: 6GiB

real-world data: `1394.1` *(vector length: 3)*		
default bucket size (32)	0.0M (56)	0 (57)
bucket size: 2	34M (44)	0.3M (37)
optimal bucket size: 4/2	13M (40)	
real-world data: `wafer_stepper.1` *(vector length: 8)*		
default bucket size (32)	0 (114)	0 (105)
bucket size: 2 (optimal)	13M (40)	0.1M (34)

B.3 Random-data experiments

see Appendix A for more details about (the generation of) the parameterised random data

B.3.1 Uncompressed (`uncompressed_fixed` (+ `_lp8`/`_lp32`/`_lp32x`))

`uncompressed_fixed`:	32 registers* → max. warps (per SM) 64 (→ occupancy 1.00)
`uncompressed_fixed_lp8, _lp32x`:	32 registers → max. warps (per SM) 64 (→ occupancy 1.00)
`uncompressed_fixed_lp32`:	38 registers → max. warps (per SM) 48 (→ occupancy 0.75)

* + 8 bytes stack frame, 8 bytes spill stores, 8 bytes spill loads (`findOrPut()`)

[U-E] *Optimal execution configuration*

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	8.33	*1*
grid dimension: 270 → total threads: 69,120	12.90	0.65
grid dimension: 210 → total threads: 53,760	9.62	0.87
grid dimension: 180 → total threads: 46,080	10.19	0.82
grid dimension: 150 → total threads: 38,400	11.10	0.75
grid dimension: 120 → total threads: 30,720	12.72	0.65
grid dimension: 90 → total threads: 23,040	15.68	0.53
grid dimension: 60 → total threads: 15,360	21.90	0.38
grid dimension: 30 → total threads: 7,680	39.28	0.21
grid dimension: 360 → total threads: 92,160	10.09	0.83
grid dimension: 480 → total threads: 122,880	8.43	0.99
grid dimension: 960 → total threads: 245,760	8.36	1.00
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	8.33	1.00
grid dimension: 240 → total threads: 30,720	11.60	0.72
grid dimension: 120 → total threads: 15,360	19.59	0.43
grid dimension: 60 → total threads: 7,680	39.66	0.21
grid dimension: 30 → total threads: 3,840	72.70	0.11
grid dimension: 960 → total threads: 122,880	8.15	1.02
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	8.49	0.98
grid dimension: 60 → total threads: 30,720	11.24	0.74
grid dimension: 30 → total threads: 15,360	18.94	0.44
grid dimension: 240 → total threads: 122,880	8.38	0.99

same optimal execution configuration (and (almost) same (relative) *behaviour*):

`_lp8`, `_lp32`, `_lp32x`

different input (vector length (1, 2, 3, 8), duplication (1.12 → 7,110,687 unique vectors), number of (unique) vectors (16,000,000 vectors → 8,000,672 unique))

different table parameters (table sizes/fill rates (512MiB/fill rate 0.12, 128MiB/fill rate 0.48, 76.8MiB/fill rate 0.80), bucket sizes (16, 8, **4**))

[U-I] Different input (vector length, duplication, no. of (unique) vectors)

block dimension: 256; grid dimension: 240

input	runtime (ms)	speedup vs. default of vector lgth.	speedup vs. vector length 4
default (vector length: 4)			
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique)	9.31	*1*	*1*
less duplication: 1.13 → 7,110,687 unique vectors	10.41	0.89	*1*
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	4.60	2.02	*1*
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	18.94	0.49	*1*
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	5.20	1.79 (1.81)	*1*
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	20.65	0.45 (0.44)	*1*
lower vector length: 1			
default (same total number of elements): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	32.50	*1*	0.29
less duplication: 1.12 → 28,445,271 unique vectors	36.51	0.89	0.29
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	7.72	4.21	[1.21]
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	15.74	2.07	0.29
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	9.04	3.60 (3.75)	**[1.15]**
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	18.17	1.79 (1.84)	0.29
lower vector length: 2			
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	17.46	*1*	0.53
less duplication: 1.12 → 14,223,349 unique vectors	19.83	0.88	0.52
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	8.62	2.02	0.53 [1.08]
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	9.91	1.76 (1.78)	0.52 [1.05]

table continues on next page…

lower vector length: 3 ***[with 32/30 compensation for "lost" entries]***			
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique	13.37	*1*	0.70
less duplication: 1.13 → 9,480,500 unique vectors	15.14	0.88	0.69
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	9.98	1.34	[0.93]
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	11.34	1.18 (1.18)	[0.92]
lower vector length: 3 ***[no compensation for "lost" entries]***			
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique	13.37	*1*	0.70
less duplication: 1.13 → 9,480,500 unique vectors	15.15	0.88	0.69
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	9.98	1.34	[0.93]
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	11.35	1.18 (1.18)	[0.92]
higher vector length: 8			
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique	5.05	*1*	1.84
less duplication: 1.13 → 3,554,816 unique vectors	5.60	0.90	1.86
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	10.17	0.50	1.86 [0.92]
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	11.28	0.45 (0.45)	1.83 [0.92]

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication)

`_lp8`, `_lp32x`: 0-3% slower (average: 2% slower)
`_lp32`: 3-7% slower (average: 5% slower)

[U-T] Different table parameters (table sizes/fill rates, bucket sizes)

block dimension: 256; grid dimension: 240

table parameters	**runtime (ms)**	**speedup vs. bucket size 32**	**speedup vs. low fill rate 0.24**
default: low fill rate (table size: 256MiB → fill rate 0.24)			
default (bucket size: 32)	9.32	*1*	*1*
bucket size: 16	7.18	1.30	*1*
bucket size: 8	6.44	1.45	*1*
bucket size: 4	6.52	1.43	*1*
very low fill rate: 2x table size: 512MiB → 0.5x fill rate: 0.12			
default (bucket size: 32)	9.31	*1*	1.00
bucket size: 16	7.19	1.29	1.00
bucket size: 8	6.36	1.46	1.01
bucket size: 4	**6.30**	1.48	**1.04**
medium fill rate: 0.5x table size: 128MiB → 2x fill rate: 0.48			
default (bucket size: 32)	9.35	*1*	1.00
bucket size: 16	7.30	1.28	**0.98**
bucket size: 8	6.79	1.38	**0.95**
bucket size: 4	7.14	1.31	**0.91**
high fill rate: 0.3x table size: 76.8MiB → 3.33x fill rate: 0.80			
default (bucket size: 32)	**9.86**	*1*	0.95
bucket size: 16	8.11	1.22	**0.88**
bucket size: 8	7.97	1.24	**0.81**
bucket size: 4	9.03	1.09	**0.72**

[U-T (`_lp8`)] Different table parameters (table sizes/fill rates, bucket sizes)

block dimension: 256; grid dimension: 240

table parameters	**runtime (ms)**	**speedup vs. no `_lp8` [U-T]**
default: low fill rate (table size: 256MiB → fill rate 0.24)		
default (bucket size: 32)	9.52	0.98
bucket size: 16	7.21	**1.00**
bucket size: 8	6.45	**1.00**
bucket size: 4	**6.20**	**1.05**
very low fill rate: 2x table size: 512MiB → 0.5x fill rate: 0.12		
default (bucket size: 32)	9.54	0.98
bucket size: 16	7.22	1.00
bucket size: 8	6.36	1.00
bucket size: 4	**6.11**	**1.03**
medium fill rate: 0.5x table size: 128MiB → 2x fill rate: 0.48		
default (bucket size: 32)	9.56	0.98
bucket size: 16	7.36	0.99
bucket size: 8	6.79	1.00
bucket size: 4	6.48	**1.10**
high fill rate: 0.3x table size: 76.8MiB → 3.33x fill rate: 0.80		
default (bucket size: 32)	**10.12**	0.97
bucket size: 16	8.20	0.99
bucket size: 8	8.03	0.99
bucket size: 4	7.69	**1.17**

`_lp32`: 0-7% slower (average: 2% slower)
`_lp32x`: 0-11% slower (average: 1% slower)

[U-I-hfr] Different input (vector length, duplication, no. of (unique) vectors) [high fill rate 0.80]

block dimension: 256; grid dimension: 240

input	**runtime (ms)**	**speedup vs. low fill rate 0.24 [U-I]**
default (vector length: 4, duplication 2.00)	4.88 – 19.98	0.94 – 0.95
less duplication: 1.12	5.63 – 22.57	**0.91 – 0.92**
lower vector length: 1	7.59 – 36.58	**1.00 – 1.02**
lower vector length: 2	8.63 – 20.06	**0.99 – 1.00**
lower vector length: 3 [with 32/30 compensation for "lost" entries]	10.45 – 16.05	**0.94 – 0.96**
lower vector length: 3 [no compensation for "lost" entries]	10.64 – 16.54	0.92 – 0.94
higher vector length: 8	5.70 – 13.08	**0.86 – 0.90**

`_lp8`, `_lp32x`: 0-3% slower (average: 2% slower)
`_lp32`: 3-7% slower (average: 5% slower)

[U-I-s8] Different input (vector length, duplication, no. of (unique) vectors)
[bucket size 8]

block dimension: 256; grid dimension: 240

input	runtime (ms)	speedup vs. bucket size 32 [U-I]
default (vector length: 4, duplication 2.00)	3.18 – 13.00	1.44 – 1.46
less duplication: 1.12	4.19 – 16.91	**1.22 – 1.24**
lower vector length: 1	5.79 – 32.29	**1.13 – 1.33**
lower vector length: 2	6.14 – 16.46	**1.20 – 1.40**
lower vector length: 3 [with **4/3** compensation for “lost” entries]	8.48 – 11.34	**1.34 – 1.54**
lower vector length: 3 [no compensation for “lost” entries]	8.57 – 11.45	1.32 – 1.51
higher vector length: 8	3.48 – 8.91	1.25 – 1.46

`_lp8`: no effect
`_lp32`, `_lp32x`: 0-2% faster (average: 1% faster)

[U-I-s4] Different input (vector length, duplication, no. of (unique) vectors)
[bucket size 4]

block dimension: 256; grid dimension: 240

3% slower – 4% faster (average: 1% slower)

`_lp8`: 0-7% faster (average: 3% faster); no effect vector length 1

[U-I-s4 (`_lp8`)] Different input (vector length, duplication, no. of (unique) vectors)
[bucket size 4]

block dimension: 256; grid dimension: 240

input	**runtime (ms)**	**speedup vs. bucket size 8 [U-I-s8]**
default (vector length: 4, duplication 2.00)	3.05 – 12.43	1.04 – 1.05
less duplication: 1.12	4.12 – 16.59	**1.02**
lower vector length: 1	5.74 – 32.65	**0.98 – 1.03**
lower vector length: 2	5.94 – 16.30	1.01 – 1.04
lower vector length: 3 [with **4/3** compensation for “lost” entries]	6.22 – 11.15	1.01 – 1.05
lower vector length: 3 [no compensation for “lost” entries]	6.32 – 11.28	1.01 – 1.05

`_lp32`: no effect
`_lp32x`: 0-2% slower (average: 1% slower)

[U-I-hfr-s8] Different input (vector length, duplication, no. of (unique) vectors)
[high fill rate 0.80 + bucket size 8]

block dimension: 256; grid dimension: 240

input	**speedup vs. low fill rate 0.24 [U-I-s8]**	**speedup vs. bucket size 32 [U-I-hfr]**
default (vector length: 4, duplication 2.00)	0.81	1.23 – 1.25
less duplication: 1.12	**0.84**	**1.12**
lower vector length: 1	**0.99 – 1.05**	1.12 – 1.37
lower vector length: 2	**0.92 – 0.95**	**1.14 – 1.34**
lower vector length: 3 [with **4/3** compensation for "lost" entries]	**0.78 – 0.81**	1.15 – 1.25
higher vector length: 8	**0.68 – 0.72**	**1.04 – 1.13**

`_lp8`: 0-3% slower (average: 1% slower)
`_lp32`, `_lp32x`: 0-10% slower (average: 4% slower)

[U-I-hfr-s4] Different input (vector length, duplication, no. of (unique) vectors) [high fill rate 0.80 + bucket size 4]

block dimension: 256; grid dimension: 240

input	speedup vs. low fill rate 0.24 [U-I-s4]	speedup vs. bucket size 8 [U-I-hfr-s8]
default (vector length: 4, duplication 2.00)	0.72	0.88
less duplication: 1.12	**0.75**	0.88
lower vector length: 1	**0.96 – 1.02**	**0.96 – 0.99**
lower vector length: 2	**0.86 – 0.90**	**0.92 – 0.96**
lower vector length: 3 [with **4/3** compensation for "lost" entries]	**0.70 – 0.74**	**0.89 – 0.90**

_lp8: 1-19% faster (average: 11% faster)

[U-I-hfr-s4 (_lp8)] Different input (vector length, duplication, no. of (unique) vectors) [high fill rate 0.80 + bucket size 4]

block dimension: 256; grid dimension: 240

input	speedup vs. low fill rate 0.24 [U-I-s4 (_lp8)]	speedup vs. bucket size 8 [U-I-hfr-s8]
default (vector length: 4, duplication 2.00)	0.81	1.04
less duplication: 1.12	**0.83 – 0.84**	**1.01 – 1.02**
lower vector length: 1	**0.99 – 1.06**	**0.97 – 1.03**
lower vector length: 2	**0.93 – 0.95**	1.00 – 1.08
lower vector length: 3 [with **4/3** compensation for "lost" entries]	**0.78 – 0.83**	**1.04 – 1.06**

_lp32: 0-7% slower (average: 3% slower)
_lp32: 0-10% slower (average: 5% slower)

[U-I-fn] Different input (vector length, duplication, no. of (unique) vectors)
[fixed table size]

block dimension: 256; grid dimension: 240

same as [U-I] except for two h.f.r. cases (in blue) → speedup vs. fixed f.r. 0.24 [U-I]: 0.89 / 0.83

input	**runtime (ms)**	**speedup vs. default of vector lgth.**	**speedup vs. vector length 4**
default (vector length: 4)			
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique)	9.31	*1*	*1*
less duplication: 1.13 → 7,110,687 unique vectors	10.41	0.89	*1*
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	4.60	2.02	*1*
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	18.94	0.49	*1*
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	5.20	1.79 (1.81)	*1*
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	23.18	**0.40 (0.44)**	*1*
lower vector length: 1			
default (same total number of elements): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	32.50	*1*	0.29
less duplication: 1.12 → 28,445,271 unique vectors	36.51	0.89	0.29
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	7.72	4.21	[1.21]
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	15.74	2.07	0.29
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	9.04	3.60 (3.75)	**[1.15]**
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	18.17	1.79 (1.84)	0.29
lower vector length: 2			
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	17.46	*1*	0.53
less duplication: 1.12 → 14,223,349 unique vectors	19.83	0.88	0.52
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	8.62	2.02	0.53 [1.08]
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	9.91	1.76 (1.78)	0.52 [1.05]

table continues on next page…

lower vector length: 3 [no compensation for "lost" entries]			
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique	13.37	*1*	0.70
less duplication: 1.13 → 9,480,500 unique vectors	15.15	0.88	0.69
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	9.98	1.34	[0.93]
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	11.35	1.18 (1.18)	[0.92]
higher vector length: 8			
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique	5.05	*1*	1.84
less duplication: 1.13 → 3,554,816 unique vectors	5.60	0.90	1.86
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	10.17	0.50	1.86 [0.92]
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	13.58	0.37 (0.45)	**1.71 [0.77]**

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication)

`_lp8`, `_lp32x`: 1-3% slower (average: 2% slower)
`_lp32`: 3-6% slower (average: 5% slower)

[U-I-fn-os] Different input (vector length, duplication, no. of (unique) vectors)
[fixed table size + optimal bucket size]

block dimension: 256; grid dimension: 240
bucket size: 16 (vector length 8); 4 + `_lp8`

input	runtime (ms)	speedup vs. bucket size 32 [U-I-fn]	speedup vs. default vector lgth.	speedup vs. vector length 4
default (vector length: 4)				
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique)	6.19	1.50	*1*	*1*
less duplication: 1.13 → 7,110,687 unique	8.53	1.22	0.73	*1*
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	3.03	1.52	2.04	*1*
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	13.03	1.45	0.47	*1*
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	4.11	1.26	1.50 (1.48)	*1*
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	20.81	1.11	0.30 (0.34)	*1*
lower vector length: 1				
default (same total number of elements): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	23.93	1.36	*1*	0.26
less duplication: 1.12 → 28,445,271 unique	32.43	1.12	0.74	0.26
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	5.93	1.31	4.03	[1.04]
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	12.15	1.30	1.97	0.25
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.07	1.12	2.96 (2.98)	[1.06]
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	16.17	1.12	1.48 (1.45)	0.25

table continues on next page…

lower vector length: 2				
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	12.01	1.45	*1*	0.51
less duplication: 1.12 → 14,223,349 unique	16.67	1.19	0.72	0.51
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	5.97	1.45	2.01	0.51 [1.04]
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	8.10	1.22	1.48 (1.45)	0.51 [1.05]
lower vector length: 3 [no compensation for "lost" entries]				
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique	8.44	1.58	*1*	0.73
less duplication: 1.13 → 9,480,500 unique	11.90	1.27	0.71	0.72
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	6.24	1.60	1.35	[0.99]
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	8.58	1.32	0.98 (0.96)	[0.99]
higher vector length: 8				
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique	3.69	1.37	*1*	1.68
less duplication: 1.13 → 3,554,816 unique	4.71	1.21	0.78	1.81
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	7.90	1.31	0.47	1.65 [0.78]
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	12.23	1.11	0.30 (0.37)	1.70 [0.70]

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication)

B.3.2 Compressed with recursion (`compressed_` and `compressed_lr*`)

`compressed_`:	32 registers* → max. warps (per SM) 64 (→ occupancy 1.00)
`compressed_lr`:	26 registers** → max. warps (per SM) 64 (→ occupancy 1.00)
`compressed_lr_64`:	26 registers*** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64_lp8`:	26 registers*** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64_lp8_s2`:	22 registers**** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64_lp32`:	26 registers*** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64_lp32_s2`:	22 registers***** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64_lp32x`:	26 registers*** → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_lr_64_lp32x_s2`:	22 registers**** → max. warps (per SM) 64 (→ occ. 1.00)

* 40 bytes stack frame, 28 bytes spill stores, 28 bytes spill loads (`treeRec()`);
8 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads (`insertTable()`, `treeFindOrPut()`)

** 40 bytes stack frame, 28 bytes spill stores, 28 bytes spill loads (`treeRec()`)

*** 24 bytes stack frame, 24 bytes spill stores, 24 bytes spill loads (`treeRec()`)
**** 24 bytes stack frame, 20 bytes spill stores, 20 bytes spill loads (`treeRec()`);
16 bytes stack frame, 16 bytes spill stores, 16 bytes spill loads (`findOrPut()`)
***** 24 bytes stack frame, 20 bytes spill stores, 20 bytes spill loads (`treeRec()`)

[C-E] Optimal execution configuration

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	147.15	*1*
grid dimension: 270 → total threads: 69,120	153.35	0.96
grid dimension: 210 → total threads: 53,760	161.07	0.91
grid dimension: 180 → total threads: 46,080	182.06	0.81
grid dimension: 150 → total threads: 38,400	162.62	0.90
grid dimension: 120 → total threads: 30,720	145.47	1.01
grid dimension: 90 → total threads: 23,040	113.61	1.30
grid dimension: 60 → total threads: 15,360	126.01	1.17
grid dimension: 30 → total threads: 7,680	198.63	0.74
grid dimension: 360 → total threads: 92,160	160.04	0.92
grid dimension: 480 → total threads: 122,880	165.77	0.89
grid dimension: 960 → total threads: 245,760	161.12	0.91
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	147.50	1.00
grid dimension: 240 → total threads: 30,720	145.43	1.01
grid dimension: 120 → total threads: 15,360	127.73	1.15
grid dimension: 60 → total threads: 7,680	194.63	0.76
grid dimension: 30 → total threads: 3,840	343.29	0.43
grid dimension: 960 → total threads: 122,880	163.02	0.90
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	142.67	1.03
grid dimension: 60 → total threads: 30,720	146.69	1.00
grid dimension: 30 → total threads: 15,360	126.77	1.16
grid dimension: 240 → total threads: 122,880	143.97	1.02

same optimal execution configuration (and (almost) same *behaviour*):
  different input (vector length (3, 8), duplication (1.12 → 7,110,687 unique vectors),
    number of (unique) vectors (16,000,000 vectors → 8,000,672 unique),
    compression ratio (0.73 → 11,675,190 elements elements in table))
  different table parameters (table sizes/fill rates (256MiB/fill rate 0.12, 64MiB/fill rate 0.48,
    38.4MiB/fill rate 0.80), bucket sizes (16))

bucket sizes 8, 4 and 2: block dimension 256; grid dimension: **60**

[C-E-vl1] Optimal execution configuration
[vector length 1 (compensated)]

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	69.84	*1*
grid dimension: 270 → total threads: 69,120	65.84	1.06
grid dimension: 210 → total threads: 53,760	76.51	0.91
grid dimension: 180 → total threads: 46,080	85.72	0.81
grid dimension: 150 → total threads: 38,400	66.18	1.06
grid dimension: 120 → total threads: 30,720	69.85	1.00
grid dimension: 90 → total threads: 23,040	83.59	0.84
grid dimension: 60 → total threads: 15,360	106.25	0.66
grid dimension: 30 → total threads: 7,680	177.44	0.39
grid dimension: 360 → total threads: 92,160	70.19	1.00
grid dimension: 480 → total threads: 122,880	70.36	0.99
grid dimension: 960 → total threads: 245,760	64.80	1.08
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	70.10	1.00
grid dimension: 240 → total threads: 30,720	69.86	1.00
grid dimension: 120 → total threads: 15,360	106.23	0.66
grid dimension: 60 → total threads: 7,680	178.12	0.39
grid dimension: 30 → total threads: 3,840	316.70	0.22
grid dimension: 960 → total threads: 122,880	70.44	0.99
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	67.74	1.03
grid dimension: 60 → total threads: 30,720	68.42	1.02
grid dimension: 30 → total threads: 15,360	105.41	0.66
grid dimension: 240 → total threads: 122,880	68.55	1.02

bucket size 16: block dimension 256; grid dimension: **150**
bucket size 8: block dimension 256; grid dimension: **90**
bucket size 4: block dimension 256; grid dimension: **60**
bucket size 2: block dimension 256; grid dimension: **30**

[C-E-vl2] *Optimal execution configuration*
[vector length 2]

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	112.79	*1*
grid dimension: 270 → total threads: 69,120	107.84	1.05
grid dimension: 210 → total threads: 53,760	121.95	0.92
grid dimension: 180 → total threads: 46,080	134.19	0.84
grid dimension: 150 → total threads: 38,400	119.32	0.95
grid dimension: 120 → total threads: 30,720	102.31	1.10
grid dimension: 90 → total threads: 23,040	91.04	1.24
grid dimension: 60 → total threads: 15,360	105.75	1.07
grid dimension: 30 → total threads: 7,680	182.95	0.62
grid dimension: 360 → total threads: 92,160	119.22	0.95
grid dimension: 480 → total threads: 122,880	118.22	0.95
grid dimension: 960 → total threads: 245,760	116.52	0.97
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	112.93	1.00
grid dimension: 240 → total threads: 30,720	102.26	1.10
grid dimension: 120 → total threads: 15,360	104.52	1.08
grid dimension: 60 → total threads: 7,680	183.22	0.62
grid dimension: 30 → total threads: 3,840	328.92	0.34
grid dimension: 960 → total threads: 122,880	118.19	0.95
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	98.60	1.14
grid dimension: 60 → total threads: 30,720	101.94	1.11
grid dimension: 30 → total threads: 15,360	106.07	1.06
grid dimension: 240 → total threads: 122,880	98.81	1.14

bucket sizes 16, 8 and 4: block dimension 256; grid dimension: **60**
bucket size 2: block dimension 256; grid dimension: **30**

[C-I] Different input (vector length, duplication, no. of (unique) vectors, compression ratio)

block dimension: 256
grid dimension: 960 (vector length 1); 90

input	**runtime (ms)**	**speedup vs. default of vector length**	**speedup vs. vector length 4**	**slow-down vs. unc. [U-I]**
default (vector length: 4)				
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	113.66	*1*	*1*	12.20
less duplication: 1.13 → 7,110,687 unique vectors	115.88	0.98	*1*	**11.14**
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	57.11	1.99	*1*	**12.41**
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	221.71	0.51	*1*	**11.70**
worse compression ratio: 0.73 → 11,675,190 elements in table	115.92	0.98	*1*	**12.45**
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	57.66	1.97 (1.95)	*1*	11.09
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	225.88	0.50 (0.50)	*1*	10.94
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	118.33	0.96 (0.96)	*1*	11.37
0.5x number of (unique) vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	58.09	1.96 (1.95)	*1*	12.63
2x number of (unique) vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	225.65	0.50 (0.50)	*1*	11.91
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vectors, 0.73 → 10,391,626 elements in table	59.10	1.92 (1.91)	*1*	11.37
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vectors, 0.73 → 41,474,428 elements in table	227.69	0.50 (0.49)	*1*	11.03

table continues on next page…

lower vector length: 1 ***[with 2/1 compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	55.42	*1*	2.05	**1.71**
less duplication: 1.12 → 28,445,271 unique vectors	56.88	0.97	2.04	1.56
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	13.16	4.21	[8.64]	1.71
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	28.33	1.96	2.02	1.80
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	15.19	3.65 (4.10)	**[7.63]**	1.68
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	35.11	1.58 (1.91)	**1.64**	**1.93**
lower vector length: 1 ***[no compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	63.25	*1*	1.80	1.95
less duplication: 1.12 → 28,445,271 unique vectors	61.41	1.03	1.89	1.68
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	14.01	4.51	[8.11]	1.82
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	28.23	2.24	2.02	1.79
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	15.12	4.18 (4.65)	[7.66]	1.67
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	30.27	2.09 (2.31)	1.91	1.67
*lower vector length: 2**				
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	91.03	*1*	1.25	**5.21**
less duplication: 1.12 → 14,223,349 unique vectors	95.39	0.95	**1.21**	4.81
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	45.39	2.01	1.26 [2.50]	5.26
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	47.59	1.91 (1.91)	1.21 **[2.43]**	4.80

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

*lower vector length: 3**				
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	116.70	*1*	0.99	**8.73**
less duplication: 1.13 → 9,480,500 unique vectors	117.38	0.99	1.01	7.75
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	87.10	1.34	[1.33]	8.73
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	87.42	1.33 (1.33)	[1.35]	7.70
higher vector length: 8				
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	130.20	*1*	0.87	**25.77**
less duplication: 1.13 → 3,554,816 unique vectors	128.10	**1.02**	0.90	22.88
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	257.66	0.51	0.86 [0.44]	25.34
worse compression ratio: 0.73 → 11,637,432 elements in table	132.30	0.98	0.88	26.19
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	258.38	0.50 (0.51)	0.87 [0.45]	22.90
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	134.93	0.96 (1.00)	0.88	24.10
2x number of (unique) vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	261.23	0.50 (0.50)	0.86 [0.44]	25.69
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vectors, 0.73 → 41,347,914 elements in table	264.07	0.49 (0.51)	0.86 [0.45]	23.41

* slowdown vs. uncompressed: comparison to no compensation for "lost" entries

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

[C-T] Different table parameters (table sizes/fill rates, bucket sizes)

block dimension: 256
grid dimension: 60 (bucket sizes 8, 4 and 2); 90

table parameters	**runtime (ms)**	**speedup vs. bucket size 32**	**speedup vs. low fill rate 0.24**	**slowdown vs. uncomp. [U-T] *(same fill rate)***	**slowdown vs. uncomp. [U-T] *(same table size)***
default: low fill rate (table size: 128MiB → fill rate 0.24)					
default (bucket size: 32)	113.52	*1*	*1*	12.18	12.14
bucket size: 16	64.16	1.77	*1*	**8.94**	8.78
bucket size: 8	38.63	2.94	*1*	**6.00**	5.69
bucket size: 4	23.09	4.92	*1*	**3.54**	3.23
bucket size: 2	20.59	5.51	*1*	**3.16**	2.88
very low fill rate: 2x table size: 256MiB → 0.5x fill rate: 0.12					
default (bucket size: 32)	114.11	*1*	0.99	12.26	**12.25**
bucket size: 16	63.85	1.79	1.00	8.88	**8.89**
bucket size: 8	38.58	2.96	1.00	6.07	**5.99**
bucket size: 4	22.87	4.99	**1.01**	3.63	**3.51**
bucket size: 2	**20.33**	5.61	**1.01**	3.23	**3.12**
very, very low fill rate: 4x table size: 512MiB → 0.25x fill rate: 0.06					
default (bucket size: 32)	114.15	*1*	0.99	-	12.26
bucket size: 16	64.15	1.78	1.00	-	8.92
bucket size: 8	38.60	2.96	1.00	-	6.07
bucket size: 4	22.85	4.99	**1.01**	-	3.63
bucket size: 2	**20.21**	5.65	**1.02**	-	3.21
medium fill rate: 0.5x table size: 64MiB → 2x fill rate: 0.48					
default (bucket size: 32)	112.98	*1*	1.00	12.08	-
bucket size: 16	63.99	1.77	1.00	**8.76**	-
bucket size: 8	39.07	2.89	0.99	**5.76**	-
bucket size: 4	23.87	4.73	**0.97**	**3.34**	-
bucket size: 2	21.41	5.28	**0.96**	**3.00**	-

table continues on next page…

high fill rate: 0.3x table size: 38.4MiB → 3.33x fill rate: 0.80					
default (bucket size: 32)	**114.30**	*1*	0.99	**11.59**	-
bucket size: 16	63.89	1.79	1.00	**7.87**	-
bucket size: 8	41.09	2.78	**0.94**	**5.16**	-
bucket size: 4	26.45	4.32	**0.87**	**2.93**	-
bucket size: 2	23.71	4.82	**0.87**	**2.62**	-
medium fill rate: 0.6x table size: 76.8MiB → 1.67x fill rate: 0.40					
default (bucket size: 32)	**114.44**	*1*	0.99	-	**11.61**
bucket size: 16	64.06	1.79	1.00	-	**7.89**
bucket size: 8	38.84	2.95	0.99	-	**4.87**
bucket size: 4	23.54	4.86	**0.98**	-	**2.61**
bucket size: 2	21.04	5.44	**0.98**	-	**2.33**

bucket size 2 compressed is compared to bucket size 4 uncompressed

[C-I-hfr] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [high fill rate 0.80]

block dimension: 256
grid dimension: 960 (vector length 1); 90

input	speedup vs. low fill rate 0.24 [C-I]	slowdown vs. uncomp. [U-I-hfr]
default (vector length: 4, duplication 2.00)	1.00 – 1.01	11.07 – 11.87
less duplication: 1.12	0.99 – 1.01	**9.88 – 10.53**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**0.81 – 1.00**	**1.91 – 2.13**
lower vector length: 2	0.96 – 1.00	**4.74 – 5.45**
lower vector length: 3	0.99 – 1.01	**7.05 – 8.21***
higher vector length: 8	0.98 – 1.01	**19.73 – 23.39**

* comparison to no compensation for “lost” entries

[C-I-s8] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 8]

block dimension: 256
grid dimension: 90 (vector length 1); 60

input	**speedup vs. bucket size 32 [C-I]**	**slowdown vs. uncompr. [U-I-s8]**
default (vector length: 4, duplication 2.00)	2.83 – 3.01	5.67 – 6.31
less duplication: 1.12	2.75 – 3.06	**4.36 – 4.94**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**1.54 – 2.02**	**1.09 – 1.47**
lower vector length: 1 [no compensation for "lost" entries]	1.64 – 1.81	1.09 – 1.46
lower vector length: 2	**2.66 – 2.74**	**2.12 – 2.77**
lower vector length: 3	2.93 – 2.99	**3.46 – 4.43***
higher vector length: 8	**3.07 – 3.29**	**8.95 – 12.18**

* comparison to no compensation for "lost" entries

[C-I-s4] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 4]

block dimension: 256; grid dimension: 60

input	**speedup vs. bucket size 8 [C-I-s8]**	**slowdown vs. uncompr. (bucket size 8) [U-I-s8]**
default (vector length: 4, duplication 2.00)	1.55 – 1.68	3.58 – 4.04
less duplication: 1.12	**1.51 – 1.67**	**2.78 – 3.26**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**0.98 – 1.02**	**1.06 – 1.45**
lower vector length: 1 [no compensation for "lost" entries]	0.97 – 1.01	1.11 – 1.47
lower vector length: 2	**1.47 – 1.51**	**1.40 – 1.85**
lower vector length: 3	**1.47 – 1.49**	**2.36 – 2.97***
higher vector length: 8	**1.56 – 1.79**	**5.18 – 7.29**

* comparison to no compensation for "lost" entries

[C-I-s2] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 2]

block dimension: 256
grid dimension: 30 (vector lengths 1 and 2); 60

input	**speedup vs. bucket size 4 [C-I-s4]**	**slowdown vs. uncompr. (bucket size 8) [U-I-s8]**
default (vector length: 4, duplication 2.00)	1.09 – 1.12	3.19 – 3.69
less duplication: 1.12	**1.08 – 1.12**	**2.49 – 3.00**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**0.93 – 1.07**	**1.12 – 1.55**
lower vector length: 1 [no compensation for "lost" entries]	0.80 – 1.00	1.13 – 1.78
lower vector length: 2	**0.98 – 1.17**	**1.33 – 1.70**
lower vector length: 3	1.11 – 1.12	**2.11 – 2.65***
higher vector length: 8	**1.14 – 1.38**	**3.98 – 5.90**

* comparison to no compensation for "lost" entries

[C-I-hfr-s8] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [high fill rate 0.80 + bucket size 8]

block dimension: 256
grid dimension: 90 (vector length 1); 60

input	**speedup vs. l.f.r. 0.24 [C-I-s8]**	**speedup vs. bucket size 32 [C-I-hfr]**	**slowdown vs. uncompr. [U-I-hfr-s8]**
default (vector length: 4, duplication 2.00)	0.89 – 0.94	2.53 – 2.78	5.11 – 5.73
less duplication: 1.12	**0.86 – 0.92**	**2.47 – 2.72**	**4.23 – 4.72**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	0.83 – 0.95	**1.67 – 1.83**	**1.25 – 1.64**
lower vector length: 2	**0.79 – 0.88**	**2.15 – 2.41**	**2.27 – 3.27**
lower vector length: 3	0.88 – 0.91	2.60 – 2.73	-
higher vector length: 8	**0.89 – 0.95**	**2.76 – 3.08**	**7.14 – 9.18**

[C-I-hfr-s4] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [high fill rate 0.80 + bucket size 4]

block dimension: 256; grid dimension: 60

input	**speedup vs. l.f.r. 0.24 [C-I-s4]**	**speedup vs. bucket size 8 [C-I-hfr-s8]**	**slowdown vs. uncompr. (bucket size 8) [U-I-hfr-s8]**
default (vector length: 4, duplication 2.00)	0.80 – 0.87	1.35 – 1.56	3.31 – 4.10
less duplication: 1.12	**0.78 – 0.85**	**1.30 – 1.53**	**2.76 – 3.49**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	0.73 – 0.88	**0.76 – 0.94**	**1.36 – 2.17**
lower vector length: 2	**0.72 – 0.85**	**1.21 – 1.57**	**1.67 – 2.08**
lower vector length: 3	0.83 – 0.87	1.39 – 1.42	-
higher vector length: 8	0.79 – 0.84	**1.45 – 1.57**	**4.56 – 6.06**

[C-I-hfr-s2] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [high fill rate 0.80 + bucket size 2]

block dimension: 256
grid dimension: 30 (vector lengths 1 and 2); 60

input	speedup vs. l.f.r. 0.24 [C-I-s2]	speedup vs. bucket size 4 [C-I-hfr-s4]	slowdown vs. uncompr. (bucket size 8) [U-I-hfr-s8]
default (vector length: 4, duplication 2.00)	0.77 – 0.88	1.06 – 1.14	2.95 – 3.84
less duplication: 1.12	**0.75 – 0.84**	**1.03 – 1.09**	**2.53 – 3.35**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**0.46 – 0.58**	**0.58 – 0.84**	**2.12 – 2.72**
lower vector length: 2	**0.53 – 0.63**	**0.69 – 0.87**	**2.13 – 3.00**
lower vector length: 3	0.76 – 0.80	**1.01 – 1.04**	-
higher vector length: 8	0.75 – 0.82	**1.10 – 1.27**	**3.80 – 5.12**

[C-I-fn] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [fixed table size]

table size: 256MiB
block dimension: 256
grid dimension: 960 (vector length 1); 90

same as [C-I] except for vector length 1 (no compensation for "lost" entries) → speedup vs. fixed f.r. 0.24 [C-I]: 0.79 – 0.98; slowdown vs. unc. different for less duplication, 2x number of vectors

input	**runtime (ms)**	**speedup vs. default of vector length**	**speedup vs. vector length 4**	**slow-down vs. unc. [U-I-fn]**
default (vector length: 4)				
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	113.66	*1*	*1*	12.20
less duplication: 1.13 → 7,110,687 unique vectors	115.88	0.98	*1*	**11.14**
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	57.11	1.99	*1*	**12.41**
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	221.71	0.51	*1*	**11.70**
worse compression ratio: 0.73 → 11,675,190 elements in table	115.92	0.98	*1*	**12.45**
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	57.66	1.97 (1.95)	*1*	11.09
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	225.88	0.50 (0.50)	*1*	**9.70**
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	118.33	0.96 (0.96)	*1*	11.37
0.5x number of (unique) vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	58.09	1.96 (1.95)	*1*	12.63
2x number of (unique) vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	225.65	0.50 (0.50)	*1*	11.91
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vectors, 0.73 → 10,391,626 elements in table	59.10	1.92 (1.91)	*1*	11.37
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vectors, 0.73 → 41,474,428 elements in table	227.69	0.50 (0.49)	*1*	**9.83**

table continues on next page…

lower vector length: 1 ***[no compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	64.30	*1*	1.77	**1.98**
less duplication: 1.12 → 28,445,271 unique vectors	70.43	0.91	**1.59**	**1.93**
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	16.23	3.96	[7.01]	2.09
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	32.39	1.99	1.76	2.05
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	17.36	3.70 (3.62)	**[6.46]**	1.92
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	38.52	1.67 (1.81)	**1.49**	**2.12**
*lower vector length: 2**				
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	91.03	*1*	1.25	**5.21**
less duplication: 1.12 → 14,223,349 unique vectors	95.39	0.95	**1.21**	4.81
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	45.39	2.01	1.26 [2.50]	5.26
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	47.59	1.91 (1.91)	1.21 **[2.43]**	4.80

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

*lower vector length: 3**				
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	116.70	*1*	0.99	**8.73**
less duplication: 1.13 → 9,480,500 unique vectors	117.38	0.99	1.01	7.75
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	87.10	1.34	[1.33]	8.73
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	87.42	1.33 (1.33)	[1.35]	7.70
higher vector length: 8				
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	130.20	*1*	0.87	**25.77**
less duplication: 1.13 → 3,554,816 unique vectors	128.10	**1.02**	0.90	22.88
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	257.66	0.51	0.86 [0.44]	25.34
worse compression ratio: 0.73 → 11,637,432 elements in table	132.30	0.98	0.88	26.19
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	258.38	0.50 (0.51)	0.87 [0.45]	19.07
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	134.93	0.96 (1.00)	0.88	24.10
2x number of (unique) vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	261.23	0.50 (0.50)	0.86 [0.44]	25.69
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vectors, 0.73 → 41,347,914 elements in table	264.07	0.49 (0.51)	0.86 [0.45]	19.37

* slowdown vs. uncompressed: comparison to no compensation for "lost" entries

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

[C-I-fn-os] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [fixed table size + optimal bucket size]

table size: 256MiB
block dimension: 256
grid dimension: 90 (vector length 1); 30 (vector length 2); 60
bucket size: 8 (vector length 1); 2

input	runtime (ms)	speedup vs. b. size 32 [C-I-fn]	speedup vs. default vector length	speedup vs. vector length 4	slow-down vs. unc. [U-I-fn-os]
default (vector length: 4)					
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	20.33	5.59	*1*	*1*	3.29
less duplication: 1.13 → 7,110,687 unique	21.28	5.27	0.96	*1*	2.50
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	10.03	5.68	2.03	*1*	3.31
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	42.39	5.35	0.48	*1*	3.25
worse compression ratio: 0.73 → 11,675,190 elements in table	23.31	5.00	0.87	*1*	3.77
less duplication, 0.5x number of vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	10.17	5.66	2.00 (1.94)	*1*	2.47
less duplication, 2x number of vectors: 1.13 → 14,220,385 unique, 16,000,000 vecs.	45.47	4.95	0.45 (0.46)	*1*	2.18
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	25.68	4.61	0.79 (0.83)	*1*	3.01
0.5x number of vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	11.41	5.10	1.78 (1.77)	*1*	3.77
2x number of vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	48.49	4.63	0.42 (0.42)	*1*	3.72
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vecs., 0.73 → 10,391,626 elements in table	12.33	4.79	1.65 (1.69)	*1*	3.00
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vecs. 0.73 → 41,474,428 elements in table	57.73	3.95	0.35 (0.40)	*1*	2.77

table continues on next page…

lower vector length: 1 ***[no compensation for "lost" entries]****					
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	35.02	1.84	*1*	0.58	1.46
less duplication: 1.12 → 28,445,271 unique	42.37	1.66	0.83	0.50	1.31
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	8.60	1.89	4.07	[2.36]	1.45
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	17.26	1.88	2.03	0.58	1.42
less duplication, 0.25x number of vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.71	1.99	4.02 (3.37)	[2.44]	1.08
less duplication, 0.5x number of vectors: 1.12 → 14,224,527 unique, 16,000,000 vecs.	17.73	2.17	1.98 (1.68)	0.57	1.10
*lower vector length: 2**					
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	19.68	4.47	*1*	1.03	1.64
less duplication: 1.12 → 14,223,349 unique	24.48	3.68	0.80	0.87	1.47
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	9.37	4.85	2.10	1.07 [2.17]	1.57
less duplication, 0.5x number of vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	10.65	4.47	1.85 (1.69)	0.95 [2.00]	1.32

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

*lower vector length: 3**					
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	24.31	4.80	*1*	0.96	2.88
less duplication: 1.13 → 9,480,500 unique	25.75	4.56	0.94	1.00	2.16
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	18.05	4.84	1.35	[1.29]	2.89
less duplication, lower number of vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	18.90	4.67	1.29 (1.27)	[1.36]	2.20
higher vector length: 8					
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	15.47	8.27	*1*	1.31	4.20
less duplication: 1.13 → 3,554,816 unique	17.71	7.32	0.87	1.20	3.76
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	32.49	7.89	0.48	1.30 [0.63]	4.11
worse compression ratio: 0.73 → 11,637,432 elements in table	19.62	6.80	0.79	1.19	5.32
less duplication, 2x number of vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	38.39	6.75	0.40 (0.42)	1.18 [0.55]	3.14
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	23.72	5.60	0.65 (0.69)	1.08	5.04
2x number of vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	41.91	6.26	0.37 (0.38)	1.16 [0.56]	5.31
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vecs., 0.73 → 41,347,914 elements in table	53.08	4.96	0.29 (0.33)	1.09 [0.48]	4.34

* slowdown vs. uncompressed: comparison to no compensation for “lost” entries

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

[Clr-E] *Optimal execution configuration*

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	83.40	*1*
grid dimension: 270 → total threads: 69,120	77.59	1.07
grid dimension: 210 → total threads: 53,760	89.09	0.94
grid dimension: 180 → total threads: 46,080	99.85	0.84
grid dimension: 150 → total threads: 38,400	85.94	0.97
grid dimension: 120 → total threads: 30,720	72.14	1.16
grid dimension: 90 → total threads: 23,040	76.97	1.08
grid dimension: 60 → total threads: 15,360	89.63	0.93
grid dimension: 30 → total threads: 7,680	147.83	0.56
grid dimension: 360 → total threads: 92,160	87.36	0.95
grid dimension: 480 → total threads: 122,880	88.19	0.95
grid dimension: 960 → total threads: 245,760	84.61	0.99
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	82.50	1.01
grid dimension: 240 → total threads: 30,720	72.52	1.15
grid dimension: 120 → total threads: 15,360	92.27	0.90
grid dimension: 60 → total threads: 7,680	147.80	0.56
grid dimension: 30 → total threads: 3,840	273.93	0.30
grid dimension: 960 → total threads: 122,880	88.76	0.94
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	69.82	1.19
grid dimension: 60 → total threads: 30,720	72.33	1.15
grid dimension: 30 → total threads: 15,360	90.38	0.92
grid dimension: 240 → total threads: 122,880	69.88	1.19

same optimal execution configuration (and (almost) same *behaviour*):
  different input (vector length (3, 8), duplication (1.12 → 7,110,687 unique vectors),
    number of (unique) vectors (16,000,000 vectors → 8,000,672 unique),
    compression ratio (0.73 → 11,675,190 elements elements in table))
  different table parameters (table sizes/fill rates (256MiB/fill rate 0.12, 64MiB/fill rate 0.48,
    38.4MiB/fill rate 0.80), bucket sizes (16))

bucket size 8: block dimension 256; grid dimension: **90**
bucket sizes 4 and 2: block dimension 256; grid dimension: **60**

[Clr-E-vl1/2] Optimal execution configuration
[vector length 1 (compensated)/2]

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	98.46	*1*
grid dimension: 270 → total threads: 69,120	92.58	1.06
grid dimension: 210 → total threads: 53,760	108.99	0.90
grid dimension: 180 → total threads: 46,080	123.20	0.80
grid dimension: 150 → total threads: 38,400	90.78	1.08
grid dimension: 120 → total threads: 30,720	96.80	1.02
grid dimension: 90 → total threads: 23,040	107.38	0.92
grid dimension: 60 → total threads: 15,360	135.70	0.73
grid dimension: 30 → total threads: 7,680	236.43	0.42
grid dimension: 360 → total threads: 92,160	98.45	1.00
grid dimension: 480 → total threads: 122,880	99.45	0.99
grid dimension: 960 → total threads: 245,760	91.67	1.07
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	98.42	1.00
grid dimension: 240 → total threads: 30,720	97.63	1.01
grid dimension: 120 → total threads: 15,360	135.84	0.72
grid dimension: 60 → total threads: 7,680	238.40	0.41
grid dimension: 30 → total threads: 3,840	433.92	0.23
grid dimension: 960 → total threads: 122,880	99.93	0.99
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	95.13	1.03
grid dimension: 60 → total threads: 30,720	97.55	1.01
grid dimension: 30 → total threads: 15,360	136.15	0.72
grid dimension: 240 → total threads: 122,880	96.24	1.02

bucket size 16: block dimension 256; grid dimension: **120**
bucket size 8: block dimension 256; grid dimension: **90**
bucket size 4: block dimension 256; grid dimension: **60**
bucket size 2: block dimension 256; grid dimension: **30**

[Clr-I] Different input (vector length, duplication, no. of (unique) vectors, compression ratio)

block dimension: 256
grid dimension: 150 (vector lengths 1 and 2); 120

input	**runtime (ms)**	**speedup vs. default of vector length**	**speedup vs. vector length 4**	**speedup vs. m.r. [C-I]**
default (vector length: 4)				
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	72.64	*1*	*1*	1.56
less duplication: 1.13 → 7,110,687 unique vectors	73.55	0.99	*1*	1.58
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	36.25	2.00	*1*	1.58
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	145.06	0.50	*1*	1.53
worse compression ratio: 0.73 → 11,675,190 elements in table	74.98	0.97	*1*	1.55
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	36.48	1.99 (1.98)	*1*	1.58
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	143.32	0.51 (0.49)	*1*	1.58
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	76.66	0.95 (0.96)	*1*	1.54
0.5x number of (unique) vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	37.30	1.95 (1.94)	*1*	1.56
2x number of (unique) vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	149.23	0.49 (0.49)	*1*	1.51
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vectors, 0.73 → 10,391,626 elements in table	38.14	1.90 (1.92)	*1*	1.55
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vectors, 0.73 → 41,474,428 elements in table	151.74	0.48 (0.48)	*1*	1.50

table continues on next page…

lower vector length: 1 ***[with 2/1 compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	90.86	*1*	0.80	**0.61**
less duplication: 1.12 → 28,445,271 unique vectors	96.08	0.95	**0.77**	**0.59**
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	22.23	4.09	[3.27]	**0.59**
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	44.89	2.02	0,81	0.63
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	23.67	3.84 (3.86)	**[3.11]**	**0.64**
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	47.73	1.90 (1.91)	0.76	**0.74**
lower vector length: 1 ***[no compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	90.46	*1*	0.80	0.70
less duplication: 1.12 → 28,445,271 unique vectors	94.30	0.96	0.78	0.65
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	22.19	4.08	[3.27]	0.63
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	44.79	2.02	0.81	0.63
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	23.64	3.83 (3.91)	[3.11]	0.64
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	47.73	1.90 (1.94)	0.76	0.63
*lower vector length: 2**				
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	47.89	*1*	1.52	**1.90**
less duplication: 1.12 → 14,223,349 unique vectors	50.47	0.95	**1.46**	1.89
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	23.72	2.02	1.53 [3.06]	1.91
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	25.04	1.91 (1.92)	1.46 **[2.94]**	1.90

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

lower vector length: 3				
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	79.85	*1*	0.94	**1.46**
less duplication: 1.13 → 9,480,500 unique vectors	80.35	0.99	0.95	1.46
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	59.76	1.34	[1.25]	1.46
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	60.35	1.32 (1.33)	[1.27]	1.45
higher vector length: 8				
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	97.27	*1*	0.75	**1.34**
less duplication: 1.13 → 3,554,816 unique vectors	97.94	0.99	0.75	1.31
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	194.73	0.50	0.74 [0.37]	1.32
worse compression ratio: 0.73 → 11,637,432 elements in table	100.30	0.97	0.75	1.32
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	196.03	0.50 (0.50)	0.73 [0.38]	1.32
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	102.26	0.95 (0.96)	0.75	1.32
2x number of (unique) vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	200.79	0.48 (0.48)	0.74 [0.37]	1.30
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vectors, 0.73 → 41,347,914 elements in table	204.47	0.48 (0.48)	0.74 [0.37]	1.29

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

[Clr-T] Different table parameters (table sizes/fill rates, bucket sizes)

block dimension: 256
grid dimension: 60 (bucket sizes 4 and 2); 90 (bucket size 8); 120

table parameters	**runtime (ms)**	**speedup vs. bucket size 32**	**speedup vs. low fill rate 0.24**	**speedup vs. m.r. [C-T]**
default: low fill rate (table size: 128MiB → fill rate 0.24)				
default (bucket size: 32)	**72.41**	*1*	*1*	1.57
bucket size: 16	43.52	1.66	*1*	**1.47**
bucket size: 8	26.35	2.75	*1*	**1.47**
bucket size: 4	18.48	3.92	*1*	**1.25**
bucket size: 2	18.13	3.99	*1*	**1.14**
very low fill rate: 2x table size: 256MiB → 0.5x fill rate: 0.12				
default (bucket size: 32)	**72.41**	*1*	1.00	1.58
bucket size: 16	43.49	1.66	1.00	1.47
bucket size: 8	26.36	2.75	1.00	1.46
bucket size: 4	18.26	3.96	**1.01**	1.25
bucket size: 2	**17.79**	4.07	**1.02**	1.14
medium fill rate: 0.5x table size: 64MiB → 2x fill rate: 0.48				
default (bucket size: 32)	72.32	*1*	1.00	1.56
bucket size: 16	43.43	1.67	1.00	1.47
bucket size: 8	26.48	2.73	1.00	1.48
bucket size: 4	19.43	3.72	**0.95**	**1.23**
bucket size: 2	18.90	3.83	**0.96**	**1.13**
high fill rate: 0.3x table size: 38.4MiB → 3.33x fill rate: 0.80				
default (bucket size: 32)	72.05	*1*	1.01	1.59
bucket size: 16	43.44	1.66	1.00	1.47
bucket size: 8	27.47	2.62	**0.96**	1.50
bucket size: 4	22.38	3.22	**0.83**	**1.18**
bucket size: 2	21.42	3.36	**0.85**	**1.11**

[Clr-I-hfr] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [high fill rate 0.80]

block dimension: 256
grid dimension: 150 (vector lengths 1 and 2); 120

no significant difference to [Clr-I]

[Clr-I-s8] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 8]

block dimension: 256; grid dimension: 90

input	**speedup vs. bucket size 32 [Clr-I]**	**speedup vs. m.r. [C-I-s8]**
default (vector length: 4, compr. r. 0.51)	2.63 – 2.77	1.35 – 1.47
worse compression ratio: 0.73	**2.50 – 2.61**	**1.37 – 1.40**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**1.85 – 2.16**	**0.71 – 0.80**
lower vector length: 1 [no compensation for "lost" entries]	2.04 – 2.16	0.78 – 0.80
lower vector length: 2	**2.13 – 2.20**	**1.51 – 1.53**
lower vector length: 3 [compression ratio: 0.73]	**2.65 – 2.71**	**1.31 – 1.32**
higher vector length: 8	**2.89 – 3.56**	**1.22 – 1.45**

[Clr-I-s4] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 4]

block dimension: 256; grid dimension: 60

input	**speedup vs. bucket size 8 [Clr-I-s8]**	**speedup vs. m.r. [C-I-s4]**
default (vector length: 4, compr. r. 0.51)	1.39 – 1.43	1.23 – 1.26
worse compression ratio: 0.73	**1.33 – 1.35**	**1.20 – 1.21**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**1.13 – 1.31**	**0.88 – 0.94**
lower vector length: 1 [no compensation for "lost" entries]	0.98 – 1.31	0.80 – 0.94
lower vector length: 2	1.19 – 1.20	**1.21 – 1.24**
lower vector length: 3 [compression ratio: 0.73]	**1.39 – 1.40**	1.24
higher vector length: 8	**1.48 – 1.56**	1.16 – 1.27

[Clr-I-s2] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 2]

block dimension: 256
grid dimension: 30 (vector lengths 1 and 2); 60

input	**speedup vs. bucket size 4 [Clr-I-s4]**	**speedup vs. m.r. [C-I-s2]**
default (vector length: 4, compr. r. 0.51)	1.02 – 1.03	1.13 – 1.15
worse compression ratio: 0.73	**1.02**	**1.12 – 1.13**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	0.96 – 1.15	**0.87 – 1.10**
lower vector length: 1 [no compensation for "lost" entries]	0.88 – 1.06	0.79 – 1.12
lower vector length: 2	**0.85 – 1.08**	**0.89 – 1.33**
lower vector length: 3 [compression ratio: 0.73]	**0.98**	**1.08 – 1.09**
higher vector length: 8	**1.07 – 1.27**	**1.09 – 1.14**

[Clr-I-hfr-s8] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [high fill rate 0.80 + bucket size 8]

block dimension: 256; grid dimension: 90

input	speedup vs. l.f.r. 0.24 [Clr-I-s8]	speedup vs. bucket size 32 [Clr-I-hfr]	speedup vs. m.r. [C-I-hfr-s8]
default (vector length: 4, compr. r. 0.51)	0.93 – 0.97	2.51 – 2.64	1.46 – 1.50
worse compression ratio: 0.73	**0.91 – 0.94**	**2.24 – 2.43**	**1.34 – 1.47**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	0.82 – 0.97	**1.51 – 2.01**	**0.62 – 0.86**
lower vector length: 2	**0.78 – 0.94**	1.73 – 1.99	**1.51 – 1.75**
lower vector length: 3 [compression ratio: 0.73]	**0.93 – 0.96**	**2.47 – 2.59**	1.38 – 1.39
higher vector length: 8	0.94 – 0.97	**2.82 – 3.38**	**1.31 – 1.47**

[Clr-I-hfr-s4] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [high fill rate 0.80 + bucket size 4]

block dimension: 256; grid dimension: 60

input	speedup vs. l.f.r. 0.24 [Clr-I-s4]	speedup vs. bucket size 8 [Clr-I-hfr-s8]	speedup vs. m.r. [C-I-hfr-s4]
default (vector length: 4, compr. r. 0.51)	0.80 – 0.84	1.20 – 1.23	1.17 – 1.18
worse compression ratio: 0.73	**0.74 – 0.75**	**1.08 – 1.10**	**1.13 – 1.14**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	0.78 – 0.85	0.98 – 1.35	**0.91 – 1.11**
lower vector length: 2	**0.76 – 0.82**	**0.96 – 1.17**	**1.11 – 1.23**
lower vector length: 3 [compression ratio: 0.73]	**0.78 – 0.82**	**1.16 – 1.19**	**1.16 – 1.17**
higher vector length: 8	0.75 – 0.81	**1.23 – 1.27**	1.11 – 1.18

[Clr-I-hfr-s2] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [high fill rate 0.80 + bucket size 2]

block dimension: 256
grid dimension: 30 (vector lengths 1 and 2); 60

input	**speedup vs. l.f.r. 0.24 [Clr-I-s2]**	**speedup vs. bucket size 4 [Clr-I-hfr-s4]**	**speedup vs. m.r. [C-I-hfr-s2]**
default (vector length: 4, compr. r. 0.51)	0.80 – 0.86	1.01 – 1.07	1.08 – 1.11
worse compression ratio: 0.73	**0.71 – 0.75**	**0.98 – 1.02**	**1.07 – 1.09**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**0.47 – 0.64**	**0.65 – 0.74**	**0.86 – 1.08**
lower vector length: 2	**0.48 – 0.71**	**0.68 – 0.79**	**1.02 – 1.18**
lower vector length: 3 [compression ratio: 0.73]	**0.76 – 0.81**	**0.95 – 0.97**	**1.09 – 1.10**
higher vector length: 8	0.72 – 0.80	**1.06 – 1.18**	1.07 – 1.12

[Clr-I-fn] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [fixed table size]

table size: 256MiB
block dimension: 256
grid dimension: 150 (vector lengths 1 and 2); 120

no significant difference to [Clr-I], except speedup vs. m.r. for vector length 1:
now 0.71 – 0.81, was 0.59 – 0.74

[Clr-I-fn-os] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [fixed table size + optimal bucket size]

table size: 256MiB
block dimension: 256; grid dimension: 60
bucket size: 4 (vector lengths ≤ 3); 2

input	**runtime (ms)**	**speedup vs. b. size 32 [Clr-I-fn]**	**speedup vs. default vector length**	**speedup vs. vector length 4**	**speed-up vs. m.r. [C-I-fn-os]**
default (vector length: 4)					
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	17.75	4.09	*1*	*1*	1.15
less duplication: 1.13 → 7,110,687 unique	18.68	3.94	0.95	*1*	1.14
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	8.73	4.15	2.03	*1*	1.15
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	37.19	3.87	0.48	*1*	1.14
worse compression ratio: 0.73 → 11,675,190 elements in table	20.62	3.64	0.86	*1*	1.13
less duplication, 0.5x number of vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	8.92	4.10	1.99 (1.93)	*1*	1.14
less duplication, 2x number of vectors: 1.13 → 14,220,385 unique, 16,000,000 vecs.	40.29	3.57	0.44 (0.45)	*1*	1.13
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	22.85	3.36	0.78 (0.82)	*1*	1.12
0.5x number of vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	10.07	3.71	1.76 (1.75)	*1*	1.13
2x number of vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	43.29	3.41	0.41 (0.41)	*1*	1.12
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vecs., 0.73 → 10,391,626 elements in table	10.93	3.49	1.62 (1.66)	*1*	1.13
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vecs. 0.73 → 41,474,428 elements in table	52.97	2.84	0.34 (0.39)	*1*	1.09

table continues on next page…

lower vector length: 1 ***[no compensation for "lost" entries]****					
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	37.61	2.41	*1*	0.47	0.93
less duplication: 1.12 → 28,445,271 unique	53.86	1.80	0.70	0.35	0.79
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	9.24	2.41	4.07	[1.92]	0.93
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	18.61	2.41	2.02	0.47	0.93
less duplication, 0.25x number of vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	9.43	2.51	3.99 (2.84)	[1.98]	0.92
less duplication, 0.5x number of vectors: 1.12 → 14,224,527 unique, 16,000,000 vecs.	19.02	2.51	1.98 (1.41)	0.47	0.93
*lower vector length: 2**					
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	18.76	2.55	*1*	0.95	1.05
less duplication: 1.12 → 14,223,349 unique	20.22	2.49	0.93	0.92	1.21
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	9.31	2.55	2.01	0.94 [1.91]	1.01
less duplication, 0.5x number of vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	9.47	2.64	1.98 (1.87)	0.94 [1.97]	1.13

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

lower vector length: 3					
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	21.03	3.79	*1*	0.98	1.16
less duplication: 1.13 → 9,480,500 unique	22.06	3.65	0.95	1.04	1.17
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	15.68	3.81	1.34	[1.32]	1.15
less duplication, lower number of vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	16.38	3.68	1.28 (1.28)	[1.39]	1.15
higher vector length: 8					
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	13.65	7.15	*1*	1.30	1.13
less duplication: 1.13 → 3,554,816 unique	15.80	6.21	0.86	1.18	1.12
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	28.59	6.87	0.48	1.30 [0.62]	1.14
worse compression ratio: 0.73 → 11,637,432 elements in table	17.89	5.61	0.76	1.15	1.10
less duplication, 2x number of vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	34.22	5.74	0.40 (0.41)	1.18 [0.55]	1.12
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	21.79	4.68	0.63 (0.66)	1.05	1.09
2x number of vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	38.43	5.22	0.36 (0.36)	1.13 [0.54]	1.09
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vecs., 0.73 → 41,347,914 elements in table	49.07	4.17	0.28 (0.31)	1.08 [0.47]	1.08

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

[Clr64-E] *Optimal execution configuration*

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	36.48	*1*
grid dimension: 270 → total threads: 69,120	33.74	1.08
grid dimension: 210 → total threads: 53,760	40.83	0.89
grid dimension: 180 → total threads: 46,080	34.11	1.07
grid dimension: 150 → total threads: 38,400	30.27	1.21
grid dimension: 120 → total threads: 30,720	27.84	1.31
grid dimension: 90 → total threads: 23,040	31.20	1.17
grid dimension: 60 → total threads: 15,360	38.79	0.94
grid dimension: 30 → total threads: 7,680	68.40	0.53
grid dimension: 360 → total threads: 92,160	34.04	1.07
grid dimension: 480 → total threads: 122,880	32.55	1.12
grid dimension: 960 → total threads: 245,760	33.72	1.08
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	36.56	1.00
grid dimension: 240 → total threads: 30,720	27.83	1.31
grid dimension: 120 → total threads: 15,360	38.78	0.94
grid dimension: 60 → total threads: 7,680	67.42	0.54
grid dimension: 30 → total threads: 3,840	126.15	0.29
grid dimension: 960 → total threads: 122,880	32.58	1.12
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	36.64	1.00
grid dimension: 60 → total threads: 30,720	27.85	1.31
grid dimension: 30 → total threads: 15,360	38.78	0.94
grid dimension: 240 → total threads: 122,880	32.56	1.12

same optimal execution configuration (and (almost) same *behaviour*):
 `_lp8, _lp8_s2, _lp32, _lp32_s2, _lp32x, _lp32x_s2`
 different input (vector length (1, 2, 3, 8), duplication (1.12 → 7,110,687 unique vectors),
  number of (unique) vectors (16,000,000 vectors → 8,000,672 unique),
  compression ratio (0.73 → 11,675,190 elements elements in table))
 different table parameters (table sizes/fill rates (256MiB/fill rate 0.12, 64MiB/fill rate 0.48,
  38.4MiB/fill rate 0.80))

bucket size 16: block dimension 256; grid dimension: **90**
bucket size 8: block dimension 256; grid dimension: **60**
bucket sizes 4 and 2: block dimension 256; grid dimension: **30**

[Clr64-I] Different input (vector length, duplication, no. of (unique) vectors, compression ratio)

block dimension: 256; grid dimension: 120

input	**runtime (ms)**	**speedup vs. default of vector length**	**speedup vs. vector length 4**	**speedup vs. 32b [Clr-I]**
default (vector length: 4)				
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	27.85	*1*	*1*	2.61
less duplication: 1.13 → 7,110,687 unique vectors	28.68	0.97	*1*	**2.56**
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	13.85	2.01	*1*	2.62
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	56.27	0.49	*1*	2.58
worse compression ratio: 0.73 → 11,675,190 elements in table	29.75	0.94	*1*	**2.52**
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	14.16	1.97 (1.95)	*1*	2.58
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	57.38	0.49 (0.48)	*1*	2.50
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	31.35	0.89 (0.91)	*1*	2.45
0.5x number of (unique) vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	14.78	1.88 (1.88)	*1*	2.52
2x number of (unique) vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	59.16	0.47 (0.46)	*1*	2.52
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vectors, 0.73 → 10,391,626 elements in table	15.63	1.78 (1.83)	*1*	2.44
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vectors, 0.73 → 41,474,428 elements in table	62.63	0.44 (0.45)	*1*	2.42

table continues on next page…

lower vector length: 1 ***[with 2/1 compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	45.39	*1*	0.61	**2.00**
less duplication: 1.12 → 28,445,271 unique vectors	47.44	0.96	0.60	**2.03**
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	11.22	4.05	[2.48]	1.98
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	22.80	1.99	0.61	1.97
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	11.78	3.85 (3.87)	[2.43]	2.01
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	23.62	1.92 (1.90)	0.60	2.02
lower vector length: 1 ***[no compensation for "lost" entries]****				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	45.35	*1*	0.61	1.99
less duplication: 1.12 → 28,445,271 unique vectors	47.78	0.95	0.60	1.97
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	11.16	4.06	[2.50]	1.99
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	22.52	2.01	0.62	1.99
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	11.76	3.86 (3.86)	[2.44]	2.01
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	23.56	1.92 (1.91)	0.60	2.03
*lower vector length: 2**				
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	23.52	*1*	1.18	**2.04**
less duplication: 1.12 → 14,223,349 unique vectors	24.58	0.96	1.17	**2.05**
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	11.71	2.01	1.18 [2.38]	2.03
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	12.27	1.92 (1.92)	1.15 [2.34]	2.04

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

lower vector length: 3				
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	30.92	*1*	0.96	**2.58**
less duplication: 1.13 → 9,480,500 unique vectors	31.85	0.97	0.98	2.52
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	23.07	1.34	[1.29]	2.59
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	23.83	1.30 (1.30)	[1.32]	2.53
higher vector length: 8				
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	29.73	*1*	0.94	**3.27**
less duplication: 1.13 → 3,554,816 unique vectors	31.29	0.95	0.92	3.13
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	60.27	0.49	0.93 [0.46]	3.23
worse compression ratio: 0.73 → 11,637,432 elements in table	33.34	0.89	**0.89**	**3.01**
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	63.05	0.47 (0.47)	0.91 [0.45]	3.11
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	35.57	0.84 (0.85)	0.88	2.88
2x number of (unique) vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	65.88	0.45 (0.44)	0.90 [0.45]	3.05
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vectors, 0.73 → 41,347,914 elements in table	72.11	0.41 (0.42)	0.87 [0.43]	2.84

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

`_lp8`: 0-1% slower (average: 1% slower)
`_lp32`: 0-4% slower (average: 2% slower)
`_lp32x`: 0-3% slower (average: 1% slower)

[Clr64-T] Different table parameters (table sizes/fill rates, bucket sizes)

block dimension: 256
grid dimension: 30 (bucket sizes 4 and 2); 60 (bucket size 8); 90 (bucket size 16); 120

table parameters	**runtime (ms)**	**speedup vs. bucket size 32**	**speedup vs. low fill rate 0.24**	**speedup vs. 32b [Clr-T]**
default: low fill rate (table size: 128MiB → fill rate 0.24)				
default (bucket size: 32)	27.82	*1*	*1*	2.60
bucket size: 16	19.19	1.45	*1*	**2.27**
bucket size: 8	15.42	1.80	*1*	**1.71**
bucket size: 4	14.28	1.95	*1*	**1.29**
bucket size: 2	14.57	1.91	*1*	**1.24**
very low fill rate: 2x table size: 256MiB → 0.5x fill rate: 0.12				
default (bucket size: 32)	27.88	*1*	1.00	2.60
bucket size: 16	19.23	1.45	1.00	2.26
bucket size: 8	15.44	1.81	1.00	1.71
bucket size: 4	**13.95**	2.00	**1.02**	1.31
bucket size: 2	14.33	1.95	**1.02**	1.24
medium fill rate: 0.5x table size: 64MiB → 2x fill rate: 0.48				
default (bucket size: 32)	27.75	*1*	1.00	2.61
bucket size: 16	19.14	1.45	1.00	2.27
bucket size: 8	15.54	1.79	**0.99**	1.70
bucket size: 4	15.41	1.80	**0.93**	1.26
bucket size: 2	15.73	1.76	**0.93**	**1.20**
high fill rate: 0.3x table size: 38.4MiB → 3.33x fill rate: 0.80				
default (bucket size: 32)	27.84	*1*	1.00	2.59
bucket size: 16	19.59	1.42	**0.98**	**2.22**
bucket size: 8	16.77	1.66	**0.92**	**1.64**
bucket size: 4	18.42	1.51	**0.78**	**1.21**
bucket size: 2	19.38	1.44	**0.75**	**1.11**

[Clr64-T (_lp8)] Different table parameters (table sizes/fill rates, bucket sizes)

block dimension: 256
grid dimension: 30 (bucket sizes 4 and 2); 60 (bucket size 8); 90 (bucket size 16); 120

table parameters	runtime (ms)	speedup vs. no _lp8 [Clr64-T]
default: low fill rate (table size: 128MiB → fill rate 0.24)		
default (bucket size: 32)	28.03	0.99
bucket size: 16	19.23	1.00
bucket size: 8	15.45	1.00
bucket size: 4	14.07	**1.02**
bucket size: 2	14.11	**1.03**
very low fill rate: 2x table size: 256MiB → 0.5x fill rate: 0.12		
default (bucket size: 32)	28.07	0.99
bucket size: 16	19.33	0.99
bucket size: 8	15.39	1.00
bucket size: 4	**13.90**	**1.00**
bucket size: 2	**14.00**	**1.02**
medium fill rate: 0.5x table size: 64MiB → 2x fill rate: 0.48		
default (bucket size: 32)	27.94	0.99
bucket size: 16	19.19	1.00
bucket size: 8	15.56	1.00
bucket size: 4	15.01	**1.03**
bucket size: 2	14.39	**1.09**
high fill rate: 0.3x table size: 38.4MiB → 3.33x fill rate: 0.80		
default (bucket size: 32)	28.06	0.99
bucket size: 16	19.69	0.99
bucket size: 8	16.90	0.99
bucket size: 4	17.83	**1.03**
bucket size: 2	16.80	**1.15**

_lp32: 0-6% slower (average: 1% slower)
_lp32x: 0-9% slower (average: 1% slower)

_lp8_s2: 3%, 3%, 1% faster; 5% slower

[Clr64-I-hfr] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [high fill rate 0.80]

block dimension: 256; grid dimension: 120

no significant difference to [Clr64-I]

[Clr64-I-s8] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 8]

block dimension: 256; grid dimension: 60

input	**speedup vs. bucket size 32 [Clr64-I]**	**speedup vs. 32b [Clr-I-s8]**
default (vector length: 4, compr. r. 0.51)	1.74 – 1.83	1.65 – 1.73
worse compression ratio: 0.73	**1.60 – 1.64**	**1.55 – 1.59**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**1.28 – 1.33**	**1.23 – 1.38**
lower vector length: 1 [no compensation for “lost” entries]	1.28 – 1.33	1.23 – 1.42
lower vector length: 2	**1.34 – 1.38**	1.27 – 1.28
lower vector length: 3 [compression ratio: 0.73]	**1.68 – 1.69**	**1.60 – 1.62**
higher vector length: 8	**1.88 – 2.30**	**1.85 – 2.11**

`_lp8`, `_lp32`, `_lp32x`: no effect

[Clr64-I-s4] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 4]

block dimension: 256; grid dimension: 30

input	speedup vs. bucket size 8 [Clr64-I-s8]	speedup vs. 32b [Clr-I-s4]
default (vector length: 4, compr. r. 0.51)	1.04 – 1.08	1.23 – 1.30
worse compression ratio: 0.73	1.04 – 1.05	**1.22 – 1.23**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	1.05 – 1.07	**1.11 – 1.17**
lower vector length: 1 [no compensation for "lost" entries]	0.97 – 1.04	1.07 – 1.24
lower vector length: 2	1.04 – 1.07	1.11 – 1.15
lower vector length: 3 [compression ratio: 0.73]	**1.12**	**1.29**
higher vector length: 8	**1.05 – 1.15**	**1.41 – 1.45**

_lp8: 0-5% faster (average: 2% faster)

[Clr64-I-s4 (_lp8)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 4]

block dimension: 256; grid dimension: 30

input	speedup vs. bucket size 8 [Clr64-I-s8]	speedup vs. 32b [Clr-I-s4]
default (vector length: 4, compr. r. 0.51)	1.06 – 1.10	1.26 – 1.32
worse compression ratio: 0.73	1.07 – 1.09	**1.25 – 1.27**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	1.06 – 1.07	**1.12 – 1.17**
lower vector length: 1 [no compensation for "lost" entries]	1.02 – 1.06	1.13 – 1.29
lower vector length: 2	1.06 – 1.07	1.13 – 1.14
lower vector length: 3 [compression ratio: 0.73]	**1.13**	**1.31**
higher vector length: 8	**1.07 – 1.18**	**1.45 – 1.48**

_lp32: no effect
_lp32x: 0-3% slower (average: 1% slower)

[Clr64-I-s2] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 2]

block dimension: 256; grid dimension: 30

input	speedup vs. bucket size 4 [Clr64-I-s4]	speedup vs. 32b [Clr-I-s2]
default (vector length: 4, compr. r. 0.51)	0.98 – 1.01	1.21 – 1.25
worse compression ratio: 0.73	**0.96 – 0.97**	**1.15 – 1.16**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**0.91 – 1.00**	**0.93 – 1.17**
lower vector length: 1 [no compensation for “lost” entries]	0.87 – 0.94	0.95 – 1.37
lower vector length: 2	0.93 – 0.94	**0.97 – 1.24**
lower vector length: 3 [compression ratio: 0.73]	**0.93**	**1.23**
higher vector length: 8	**0.94 – 1.14**	**1.25 – 1.31**

`_lp8`: 0-17% faster (average: 7% faster)

[Clr64-I-s2 (`_lp8`)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [bucket size 2]

block dimension: 256; grid dimension: 30

input	speedup vs. bucket size 4 (`_lp8`) [Clr64-I-s4 (`_lp8`)]	speedup vs. 32b [Clr-I-s2]
default (vector length: 4, compr. r. 0.51)	1.00 – 1.03	1.26 – 1.31
worse compression ratio: 0.73	**1.01 – 1.03**	**1.26 – 1.27**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**1.00**	**1.02 – 1.16**
lower vector length: 1 [no compensation for “lost” entries]	0.92 – 1.01	1.11 – 1.59
lower vector length: 2	**0.98 – 0.99**	**1.04 – 1.33**
lower vector length: 3 [compression ratio: 0.73]	**0.96 – 0.97**	**1.29 – 1.30**
higher vector length: 8	**0.97 – 1.22**	**1.32 – 1.43**

`_lp32`: no effect; `_lp32x`: 0-7% slower (average: 2% slower)
`_lp8_s2`: 5% slower – 3% faster (average: 0%)

[Clr64-I-hfr-s8]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[high fill rate 0.80 + bucket size 8]

block dimension: 256; grid dimension: 60

input	speedup vs. l.f.r. 0.24 [Clr64-I-s8]	speedup vs. bucket size 32 [Clr64-I-hfr]	speedup vs. 32b [Clr-I-hfr-s8]
default (vector length: 4, compr. r. 0.51)	0.88 – 0.92	1.53 – 1.68	1.55 – 1.66
worse compression ratio: 0.73	**0.85 – 0.88**	**1.38 – 1.45**	**1.44 – 1.49**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**0.90 – 0.97**	**1.21 – 1.25**	**1.22 – 1.63**
lower vector length: 2	0.88 – 0.97	**1.24 – 1.30**	1.26 – 1.44
lower vector length: 3 [compression ratio: 0.73]	**0.89 – 0.94**	**1.51 – 1.58**	**1.53 – 1.57**
higher vector length: 8	**0.80 – 0.88**	**1.69 – 1.93**	**1.66 – 1.86**

`_lp8`: no effect
`_lp32`, `_lp32x`: 0-3% slower (average: 1% slower)

[Clr64-I-hfr-s4]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[high fill rate 0.80 + bucket size 4]

block dimension: 256; grid dimension: 30

input	speedup vs. l.f.r. 0.24 [Clr64-I-s4]	speedup vs. bucket size 8 [Clr64-I-hfr-s8]	speedup vs. 32b [Clr-I-hfr-s4]
default (vector length: 4, compr. r. 0.51)	0.75 – 0.80	0.90 – 0.92	1.19 – 1.23
worse compression ratio: 0.73	**0.68 – 0.70**	**0.82 – 0.85**	**1.12 – 1.14**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	**0.70 – 0.79**	**0.77 – 0.88**	**0.93 – 1.07**
lower vector length: 2	**0.66 – 0.78**	0.79 – 0.86	**1.05 – 1.12**
lower vector length: 3 [compression ratio: 0.73]	**0.71 – 0.73**	**0.88 – 0.90**	**1.16 – 1.18**
higher vector length: 8	**0.66 – 0.71**	**0.86 – 0.93**	**1.25 – 1.29**

_lp8: 0-18% faster (average: 5% faster)

[Clr64-I-hfr-s4 (_lp8)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[high fill rate 0.80 + bucket size 4]

block dimension: 256; grid dimension: 30

input	speedup vs. l.f.r. 0.24 [Clr64-I-s4 (_lp8)]	speedup vs. bucket size 8 [Clr64-I-hfr-s8]	speedup vs. 32b [Clr-I-hfr-s4]
default (vector length: 4, compr. r. 0.51)	0.77 – 0.81	0.94 – 0.96	1.23 – 1.27
worse compression ratio: 0.73	**0.70 – 0.73**	**0.89**	**1.17 – 1.22**
lower vector length: 1 [with 2/1 compensation for "lost" entries]	0.74 – 0.84	**0.87 – 0.93**	**1.08 – 1.13**
lower vector length: 2	0.75 – 0.77	**0.82 – 0.92**	1.03 – 1.20
lower vector length: 3 [compression ratio: 0.73]	**0.73 – 0.76**	**0.92 – 0.93**	1.21 – 1.22
higher vector length: 8	**0.66 – 0.72**	0.89 – 0.97	**1.29 – 1.35**

_lp32, _lp32x: 0-8% slower (average: 5% slower)

[Clr64-I-hfr-s2]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[high fill rate 0.80 + bucket size 2]

block dimension: 256; grid dimension: 30

input	speedup vs. l.f.r. 0.24 [Clr64-I-s2]	speedup vs. bucket size 8 [Clr64-I-hfr-s8]	speedup vs. 32b [Clr-I-hfr-s2]
default (vector length: 4, compr. r. 0.51)	0.72 – 0.76	0.86 – 0.87	1.09 – 1.13
worse compression ratio: 0.73	**0.65 – 0.67**	**0.76 – 0.78**	**1.02 – 1.06**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**0.62 – 0.76**	**0.68 – 0.78**	**1.24 – 1.34**
lower vector length: 2	0.66 – 0.73	0.73 – 0.77	**1.28 – 1.36**
lower vector length: 3 [compression ratio: 0.73]	**0.69 – 0.72**	**0.81**	**1.09 – 1.11**
higher vector length: 8	**0.63 – 0.73**	**0.88 – 0.92**	**1.11 – 1.15**

_lp8: 12-24% faster (average: 18% faster)

[Clr64-I-hfr-s2 (_lp8)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[high fill rate 0.80 + bucket size 2]

block dimension: 256; grid dimension: 30

input	speedup vs. l.f.r. 0.24 [Clr64-I-s2 (_lp8)]	speedup vs. bucket size 8 [Clr64-I-hfr-s8]	speedup vs. 32b [Clr-I-hfr-s2]
default (vector length: 4, compr. r. 0.51)	0.79 – 0.83	0.99 – 1.00	1.27 – 1.30
worse compression ratio: 0.73	**0.71 – 0.75**	**0.92 – 0.93**	**1.24 – 1.26**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	0.74 – 0.85	**0.87 – 0.95**	**1.49 – 1.66**
lower vector length: 2	**0.73 – 0.81**	**0.83 – 0.88**	**1.43 – 1.61**
lower vector length: 3 [compression ratio: 0.73]	**0.77 – 0.81**	**0.94 – 0.95**	**1.28 – 1.31**
higher vector length: 8	**0.66 - 0.79**	**1.01 – 1.04**	**1.26 – 1.32**

_lp32: 0-11% slower (average: 7% slower)
_lp32x: 2-17% slower (average: 11% slower)

_lp8_s2: 1-12% slower (average: 6% slower)

[Clr64-I-fn] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [fixed table size]

block dimension: 256; grid dimension: 120

no significant difference to [Clr64-I]

[Clr64-I-fn-os] Different input (vector length, duplication, no. of (unique) vectors, compression r.) [fixed table size + optimal bucket size]

table size: 256MiB
block dimension: 256
grid dimension: 60 (vectors lengths 1 and 2); 30
bucket size: 8 (vector lengths 1 and 2); 2

input	**runtime (ms)**	**speedup vs. b. size 32 [Clr64-I-fn]**	**speedup vs. default vector length**	**speedup vs. vector length 4**	**speed-up vs. 32b [Clr-I-fn-os]**
default (vector length: 4)					
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	14.01	1.99	*1*	*1*	1.27
less duplication: 1.13 → 7,110,687 unique	14.57	1.97	0.96	*1*	1.28
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	6.97	2.00	2.01	*1*	1.25
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	29.51	1.91	0.47	*1*	1.26
worse compression ratio: 0.73 → 11,675,190 elements in table	16.51	1.80	0.85	*1*	1.25
less duplication, 0.5x number of vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	6.96	2.04	2.01 (1.94)	*1*	1.28
less duplication, 2x number of vectors: 1.13 → 14,220,385 unique, 16,000,000 vecs.	31.15	1.80	0.45 (0.46)	*1*	1.29
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	17.97	1.74	0.78 (0.82)	*1*	1.27
0.5x number of vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	8.15	1.82	1.72 (1.71)	*1*	1.24
2x number of vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	33.79	1.76	0.41 (0.40)	*1*	1.28
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vecs., 0.73 → 10,391,626 elements in table	8.79	1.78	1.59 (1.64)	*1*	1.24
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vecs. 0.73 → 41,474,428 elements in table	41.69	1.51	0.34 (0.39)	*1*	1.27

table continues on next page…

lower vector length: 1 ***[no compensation for "lost" entries]****					
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	35.32	1.29	*1*	0.40	1.06
less duplication: 1.12 → 28,445,271 unique	41.03	1.18	0.86	0.36	1.31
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	8.75	1.29	4.04	[1.60]	1.06
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	17.63	1.28	2.00	0.40	1.06
less duplication, 0.25x number of vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.86	1.33	3.99 (3.48)	[1.64]	1.06
less duplication, 0.5x number of vectors: 1.12 → 14,224,527 unique, 16,000,000 vecs.	17.84	1.33	1.98 (1.72)	1.39	1.07
*lower vector length: 2**					
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	17.72	1.33	*1*	0.79	1.06
less duplication: 1.12 → 14,223,349 unique	18.27	1.35	0.97	0.80	1.11
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	8.78	1.34	2.02	0.79 [1.60]	1.06
less duplication, 0.5x number of vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	8.90	1.38	1.99 (1.96)	0.78 [1.64]	1.06

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

lower vector length: 3					
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	16.72	1.85	*1*	0.99	1.26
less duplication: 1.13 → 9,480,500 unique	17.33	1.83	0.96	1.04	1.27
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	12.47	1.86	1.34	[1.32]	1.26
less duplication, lower number of vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	12.90	1.85	1.30 (1.29)	[1.39]	1.27
higher vector length: 8					
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	9.68	3.08	*1*	1.45	1.41
less duplication: 1.13 → 3,554,816 unique	11.24	2.78	0.86	1.30	1.41
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	20.30	2.96	0.48	1.45 [0.69]	1.41
worse compression ratio: 0.73 → 11,637,432 elements in table	12.88	2.58	0.75	1.28	1.39
less duplication, 2x number of vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	24.71	2.55	0.39 (0.41)	1.26 [0.59]	1.38
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	15.96	2.24	0.61 (0.65)	1.13	1.37
2x number of vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	27.66	2.38	0.35 (0.36)	1.22 [0.60]	1.39
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vecs., 0.73 → 41,347,914 elements in table	36.51	1.99	0.27 (0.31)	1.14 [0.49]	1.34

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

B.3.3 Compressed without recursion (`compressed_nr_*`) [fixed bucket size 2]

`compressed_nr_1_ip_gm`:	32 registers → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_1_ip_sm`:	32 registers → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_1_np_gm`:	32 registers → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_1_np_sm`:	32 registers → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_1_np_s_sm`:	32 registers → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_l_ip_gm`:	32 registers* → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_l_ip_sm`:	32 registers* → max. warps (per SM) 64 (→ occ. 1.00)
`compressed_nr_l_np_gm`:	40 registers → max. warps (per SM) 48 (→ occ. 0.75)
`compressed_nr_l_np_lm`:	34 registers → max. warps (per SM) 48 (→ occ. 0.75)
`compressed_nr_l_np_sm`:	38 registers → max. warps (per SM) 48 (→ occ. 0.75)
`compressed_nr_ld_ip_gm`:	46 registers** → max. warps (per SM) 40 (→ occ. 0.63)
`compressed_nr_ld_ip_sm`:	40 registers → max. warps (per SM) 48 (→ occ. 0.75)
`compressed_nr_ld_np_gm`:	46 registers → max. warps (per SM) 40 (→ occ. 0.63)
`compressed_nr_ld_np_sm`:	40 registers* → max. warps (per SM) 48 (→ occ. 0.75)
`compressed_nr_ld_np_s_sm`:	40 registers → max. warps (per SM) 48 (→ occ. 0.75)

* + 8 bytes stack frame, 4 bytes spill stores, 4 bytes spill loads (`treeFindOrPut()`)
** + 8 bytes stack frame, 4 bytes spill stores, 4 bytes spill loads (`treeNoRec()`)

shared memory usage (dynamically allocated) does not restrict occupancy when:

- `*_l_*_sm`: vector length < 32
- `*_1_*_sm`: all cases (four bytes per CUDA thread → max. usage = 4 B/thread * 2048 max. threads = 8KiB shared memory ≤ 96KiB shared memory/SM)
- `*_ld_*_sm`: all cases (see above)

[Cnr-E (1_*)] Optimal execution configuration

execution configuration	runtime (ms)	speedup
default (block dimension: 256)		
default (grid dimension: 240 → total threads: 61,440)	20.12	*1*
grid dimension: 270 → total threads: 69,120	19.73	1.02
grid dimension: 210 → total threads: 53,760	19.71	1.02
grid dimension: 180 → total threads: 46,080	19.15	1.05
grid dimension: 150 → total threads: 38,400	18.46	1.09
grid dimension: 120 → total threads: 30,720	17.49	1.15
grid dimension: 90 → total threads: 23,040	15.96	1.26
grid dimension: 60 → total threads: 15,360	14.37	1.40
grid dimension: 30 → total threads: 7,680	12.98	1.55
grid dimension: 360 → total threads: 92,160	19.35	1.04
grid dimension: 480 → total threads: 122,880	20.10	1.00
grid dimension: 960 → total threads: 245,760	20.04	1.00
lower block dimension: 128		
grid dimension: 480 → total threads: 61,440	20.10	1.00
grid dimension: 240 → total threads: 30,720	17.49	1.15
grid dimension: 120 → total threads: 15,360	14.36	1.40
grid dimension: 60 → total threads: 7,680	13.00	1.55
grid dimension: 30 → total threads: 3,840	13.04	1.54
grid dimension: 960 → total threads: 122,880	20.12	1.00
higher block dimension: 512		
grid dimension: 120 → total threads: 61,440	20.11	1.00
grid dimension: 60 → total threads: 30,720	17.51	1.15
grid dimension: 30 → total threads: 15,360	14.35	1.40
grid dimension: 240 → total threads: 122,880	20.10	1.00

same optimal execution configuration (and (almost) same *behaviour*):
different input (vector length (1, 2, 3, 8), duplication (1.12 → 7,110,687 unique vectors),
number of (unique) vectors (16,000,000 vectors → 8,000,672 unique),
compression ratio (0.73 → 11,675,190 elements elements in table))
different table parameters (table sizes/fill rates (256MiB/fill rate 0.12, 64MiB/fill rate 0.48,
16MiB/fill rate 0.80 (except vector length 8)))

table size 38.4MiB/fill rate 0.80 (vector length 8): block dimension: 256; grid dimension: **60**

[Cnr-E (1_*)] Optimal execution configuration

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 180 → total threads: 46,080)	16.47	*1*
grid dimension: 210 → total threads: 53,760	17.16	0.96
grid dimension: 150 → total threads: 38,400	15.69	1.05
grid dimension: 120 → total threads: 30,720	14.80	1.11
grid dimension: 90 → total threads: 23,040	13.96	1.18
grid dimension: 60 → total threads: 15,360	13.25	1.24
grid dimension: 30 → total threads: 7,680	12.70	1.30
grid dimension: 270 → total threads: 69,120	17.54	0.94
grid dimension: 360 → total threads: 92,160	17.08	0.96
grid dimension: 720 → total threads: 184,320	17.82	0.92
lower block dimension: 128		
grid dimension: 360 → total threads: 46,080	16.54	1.00
grid dimension: 180 → total threads: 23,040	13.94	1.18
grid dimension: 90 → total threads: 11,520	12.88	1.28
grid dimension: 60 → total threads: 7,680	12.72	1.30
grid dimension: 30 → total threads: 3,840	14.97	1.10
grid dimension: 720 → total threads: 92,160	17.21	0.96
higher block dimension: 512		
grid dimension: 90 → total threads: 46,080	16.48	1.00
grid dimension: 60 → total threads: 30,720	14.76	1.12
grid dimension: 30 → total threads: 15,360	13.22	1.25
grid dimension: 180 → total threads: 92,160	17.21	0.96

same optimal execution configuration (and (almost) same *behaviour*):
  different input (vector length (1, 2, 3), duplication (1.12 → 7,110,687 unique vectors),
    number of (unique) vectors (16,000,000 vectors → 8,000,672 unique),
    compression ratio (0.73 → 11,675,190 elements elements in table))
  different table parameters (table sizes/fill rates (256MiB/fill rate 0.12, 64MiB/fill rate 0.48,
    38.4MiB/fill rate 0.80 (except vector length 4)))

table size 38.4MiB/fill rate 0.80 (vector length 4): block dimension: 256; grid dimension: **60**

[Cnr-E-vl8 (`l_*/ld_*`)] Optimal execution configuration
[vector length 8]

execution configuration	runtime (ms)	speedup
default (block dimension: 256)		
default (grid dimension: 180 → total threads: 46,080)	9.26	*1*
grid dimension: 210 → total threads: 53,760	9.47	0.98
grid dimension: 150 → total threads: 38,400	9.09	1.02
grid dimension: 120 → total threads: 30,720	8.96	1.03
grid dimension: 90 → total threads: 23,040	8.85	1.05
grid dimension: 60 → total threads: 15,360	8.88	1.04
grid dimension: 30 → total threads: 7,680	10.34	0.90
grid dimension: 270 → total threads: 69,120	10.09	0.92
grid dimension: 360 → total threads: 92,160	9.56	0.97
grid dimension: 720 → total threads: 184,320	9.64	0.96
lower block dimension: 128		
grid dimension: 360 → total threads: 46,080	9.25	1.00
grid dimension: 180 → total threads: 23,040	8.86	1.05
grid dimension: 90 → total threads: 11,520	9.14	1.01
grid dimension: 60 → total threads: 7,680	10.32	0.90
grid dimension: 30 → total threads: 3,840	16.63	0.56
grid dimension: 720 → total threads: 92,160	9.59	0.97
higher block dimension: 512		
grid dimension: 90 → total threads: 46,080	9.25	1.00
grid dimension: 60 → total threads: 30,720	8.94	1.04
grid dimension: 30 → total threads: 15,360	8.86	1.04
grid dimension: 180 → total threads: 92,160	9.59	0.97

[Cnr-E (`ld_*`)] Optimal execution configuration

execution configuration	**runtime (ms)**	**speedup**
default (block dimension: 256)		
default (grid dimension: 180 → total threads: 46,080)	17.11	*1*
grid dimension: 210 → total threads: 53,760	17.86	0.96
grid dimension: 150 → total threads: 38,400	16.12	1.06
grid dimension: 120 → total threads: 30,720	15.09	1.13
grid dimension: 90 → total threads: 23,040	13.94	1.23
grid dimension: 60 → total threads: 15,360	13.10	1.31
grid dimension: 30 → total threads: 7,680	13.02	1.31
grid dimension: 270 → total threads: 69,120	18.14	0.94
grid dimension: 360 → total threads: 92,160	17.79	0.96
grid dimension: 720 → total threads: 184,320	18.40	0.93
lower block dimension: 128		
grid dimension: 360 → total threads: 46,080	17.04	1.00
grid dimension: 180 → total threads: 23,040	14.03	1.22
grid dimension: 90 → total threads: 11,520	12.87	1.33
grid dimension: 60 → total threads: 7,680	12.96	1.32
grid dimension: 30 → total threads: 3,840	17.63	0.97
grid dimension: 720 → total threads: 92,160	17.77	0.96
higher block dimension: 512		
grid dimension: 90 → total threads: 46,080	17.05	1.00
grid dimension: 60 → total threads: 30,720	14.90	1.15
grid dimension: 30 → total threads: 15,360	13.07	1.31
grid dimension: 180 → total threads: 92,160	17.65	0.97

same optimal execution configuration (and (almost) same *behaviour*):
- different input (duplication (1.12 → 7,110,687 unique vectors),
  - number of (unique) vectors (16,000,000 vectors → 8,000,672 unique),
  - compression ratio (0.73 → 11,675,190 elements elements in table))
- different table parameters (table sizes/fill rates (64MiB/fill rate 0.48)

table size 256MiB/fill rate 0.12: block dimension: 256; grid dimension: **30**
table size 38.4MiB/fill rate 0.80: block dimension: 256; grid dimension: **90**

[Cnr-E-vl1/2/3 (`1d_*`)] Optimal execution configuration
[vector length 1/2/3]

execution configuration	runtime (ms)	speedup
default (block dimension: 256)		
default (grid dimension: 180 → total threads: 46,080)	56.26	*1*
grid dimension: 210 → total threads: 53,760	58.65	0.96
grid dimension: 150 → total threads: 38,400	54.12	1.04
grid dimension: 120 → total threads: 30,720	48.17	1.17
grid dimension: 90 → total threads: 23,040	45.38	1.24
grid dimension: 60 → total threads: 15,360	36.47	1.54
grid dimension: 30 → total threads: 7,680	32.25	1.74
grid dimension: 270 → total threads: 69,120	67.06	0.84
grid dimension: 360 → total threads: 92,160	57.69	0.98
grid dimension: 720 → total threads: 184,320	59.19	0.95
lower block dimension: 128		
grid dimension: 360 → total threads: 46,080	57.41	0.98
grid dimension: 180 → total threads: 23,040	44.68	1.26
grid dimension: 90 → total threads: 11,520	33.72	1.67
grid dimension: 60 → total threads: 7,680	32.24	1.74
grid dimension: 30 → total threads: 3,840	32.00	1.76
grid dimension: 720 → total threads: 92,160	56.45	1.00
higher block dimension: 512		
grid dimension: 90 → total threads: 46,080	56.74	0.99
grid dimension: 60 → total threads: 30,720	53.16	1.06
grid dimension: 30 → total threads: 15,360	38.68	1.45
grid dimension: 180 → total threads: 92,160	59.39	0.95

[Cnr-I-s2 (`l_np_sm`)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)

block dimension: 256
grid dimension: 120 (vector length 8); 30

input	runtime (ms)	speedup vs. default of vector length	speedup vs. vector length 4	speedup vs. rec. [Clr64-I-s2 (`_lp8_s2`)]
default (vector length: 4)				
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	12.71	*1*	*1*	1.09
less duplication: 1.13 → 7,110,687 unique vectors	13.42	0.95	*1*	1.08
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	6.22	2.04	*1*	1.09
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	27.08	0.47	*1*	1.08
worse compression ratio: 0.73 → 11,675,190 elements in table	16.00	0.79	*1*	**1.06**
less duplication, 0.5x number of (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	6.33	2.01 **(1.94)**	*1*	1.09
less duplication, 2x number of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	28.05	0.45 (0.44)	*1*	**1.06**
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	17.03	0.75 (0.75)	*1*	1.06
0.5x number of (unique) vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	7.88	1.61 (1.62)	*1*	1.06
2x number of (unique) vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	32.18	0.39 (0.37)	*1*	1.06
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vectors, 0.73 → 10,391,626 elements in table	8.46	1.50 **(1.54)**	*1*	1.06
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vectors, 0.73 → 41,474,428 elements in table	34.13	0.37 (0.35)	*1*	1.07

table continues on next page…

lower vector length: 1 ***[with 2/1 compensation for "lost" entries]******				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	32.37	*1*	0.39	**1.02**
less duplication: 1.12 → 28,445,271 unique vectors	32.60	0.99	**0.41**	1.01
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	8.01	4.04	[1.59]	1.02
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	16.08	2.01	0.39	**1.08**
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.17	3.96 (4.01)	**[1.64]**	**1.04**
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	16.36	1.98 (2.00)	0.39	**1.02**
lower vector length: 1 ***[no compensation for "lost" entries]******				
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	32.65	*1*	0.39	1.02
less duplication: 1.12 → 28,445,271 unique vectors	33.28	0.98	0.40	1.08
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	8.52	3.83	[1.49]	0.98
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	16.16	2.02	0.38	1.04
less duplication, 0.25x number of (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.30	3.94 (3.76)	[1.62]	1.06
less duplication, 0.5x number of (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	16.64	1.96 (1.98)	0.38	1.11
*lower vector length: 2**				
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	16.23	*1*	0.78	**1.02**
less duplication: 1.12 → 14,223,349 unique vectors	16.26	1.00	**0.83**	**1.04**
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	8.01	2.02	0.78 [1.59]	1.02
less duplication, 0.5x number of (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	8.16	1.99 (2.02)	0.78 **[1.64]**	**1.08**

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

lower vector length: 3				
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	15.76	*1*	1.02	**1.06**
less duplication: 1.13 → 9,480,500 unique vectors	15.99	0.99	**1.06**	1.07
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	11.76	1.34	[1.36]	1.06
less duplication, lower number of (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	12.00	1.31 (1.32)	**[1.42]**	1.07
higher vector length: 8				
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	8.95	*1*	1.42	**1.10**
less duplication: 1.13 → 3,554,816 unique vectors	10.27	0.87	**1.31**	1.09
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	18.50	0.48	1.46 [0.69]	1.09
worse compression ratio: 0.73 → 11,637,432 elements in table	11.87	0.75	**1.35**	**1.07**
less duplication, 2x number of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	20.94	0.43 (0.42)	1.34 **[0.64]**	1.08
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	14.43	0.62 (0.66)	**1.18**	1.09
2x number of (unique) vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	25.24	0.35 (0.37)	**1.28** **[0.63]**	1.08
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vectors, 0.73 → 41,347,914 elements in table	30.78	0.29 (0.32)	1.11 **[0.55]**	1.07

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

`l_np_lm`, `l_ip_sm`, `l_ip_gm`, `ld_np_sm`, `ld_np_s_sm`, `ld_ip_sm`, `ld_np_gm`:
2% slower – 2% faster (average: 0%)
`l_np_gm`: 0-4% slower (average: 1% slower)
`1_np_sm`, `1_np_s_sm`, `1_ip_sm`: 2% slower – 5% faster (average: 1% faster)
`1_np_gm`, `1_ip_gm`: 2% slower – 6% faster (average: 0%)
`ld_ip_gm`: 4% slower – 2% faster (average: 1% slower)

[Cnr-T (`l_np_sm`)] Different table parameters (table sizes/fill rates)

block dimension: 256
grid dimension: 60 (high fill rate 0.80); 30

table parameters	**runtime (ms)**	**speedup vs. low fill rate 0.24**	**speedup vs. rec. [Clr64-T (`_lp8`)] *(bucket size 2)***
default: low fill rate (table size: 128MiB → fill rate 0.24)	12.67	*1*	1.11
very low fill rate: 2x table size: 256MiB → 0.5x fill rate: 0.12	**12.83**	0.99	**1.09**
medium fill rate: 0.5x table size: 64MiB → 2x fill rate: 0.48	13.14	0.96	**1.09**
high fill rate: 0.3x table size: 38.4MiB → 3.33x fill rate: 0.80	15.15	0.84	1.11

[Cnr-I-hfr-s2 (`1_np_sm`)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[high fill rate 0.80]

block dimension: 256
grid dimension: 120 (vector length 8); 60 (vector length 4); 30

input	speedup vs. low fill rate 0.24 [Cnr-I-s2 (`1_np_sm`)]	speedup vs. rec. [Clr64-I-hfr-s2 (`_1p8_s2`)]
default (vector length: 4, compr. r. 0.51)	0.79 – 0.87	1.12 – 1.21
worse compression ratio: 0.73	**0.73 – 0.79**	**1.17 – 1.24**
lower vector length: 1 [with 2/1 compensation for “lost” entries]	**0.76 – 0.85**	**1.07 – 1.11**
lower vector length: 2	**0.74 – 0.79**	1.05 – 1.13
lower vector length: 3	0.74 – 0.79	**1.12**
higher vector length: 8	0.77 – 0.85	**1.19 – 1.41**

[Cnr-I-fn-s2 (`1_np_sm`)]
Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [fixed table size]

table size: 256MiB
block dimension: 128 (vector length 4); 256
grid dimension: 120 (vector length 8); 90 (vector length 4); 30

input	**runtime (ms)**	**speedup vs. fixed l.f.r. .24 [Cnr-I-s2]**	**speedup vs. default vector length**	**speedup vs. vector length 4**	**speed-up vs. rec. [Clr64-I-fn-os]**
default (vector length: 4)					
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	13.04	0.97	*1*	*1*	1.07
less duplication: 1.13 → 7,110,687 unique	13.49	0.99	0.97	*1*	1.08
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	6.47	0.96	2.01	*1*	1.08
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 unique	27.25	0.99	0.48	*1*	1.08
worse compression ratio: 0.73 → 11,675,190 elements in table	15.77	1.02	0.83	*1*	1.05
less duplication, 0.5x number of vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	6.45	0.98	2.02 (1.95)	*1*	1.08
less duplication, 2x number of vectors: 1.13 → 14,220,385 unique, 16,000,000 vecs.	28.57	0.98	0.46 (0.46)	*1*	1.09
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	17.19	0.99	0.76 (0.80)	*1*	1.05
0.5x number of vectors, worse c. ratio: 4,000,000 vectors → 2,000,436 unique, 0.73 → 5,837,892 elements in table	7.73	1.02	1.69 (1.67)	*1*	1.05
2x number of vectors, worse c. ratio: 16,000,000 vectors → 8,004,559 unique, 0.73 → 23,286,302 elements in table	32.87	0.98	0.40 (0.40)	*1*	1.03
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vecs., 0.73 → 10,391,626 elements in table	8.32	1.02	1.57 (1.61)	*1*	1.06
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vecs. 0.73 → 41,474,428 elements in table	38.82	0.88	0.34 (0.38)	*1*	1.07

table continues on next page…

lower vector length: 1 ***[no compensation for “lost” entries]****					
default (total number of elements same): 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	33.38	0.98	*1*	0.39	1.06
less duplication: 1.12 → 28,445,271 unique	46.33	0.72	0.72	0.29	0.89
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	8.03	1.06	4.16	[1.62]	1.09
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	16.17	1.00	2.06	0.40	1.09
less duplication, 0.25x number of vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.33	1.00	4.01 (3.00)	[1.62]	1.06
less duplication, 0.5x number of vectors: 1.12 → 14,224,527 unique, 16,000,000 vecs.	16.37	1.02	2.04 (1.49)	0.39	1.09
*lower vector length: 2**					
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	16.39	0.99	*1*	0.80	1.08
less duplication: 1.12 → 14,223,349 unique	17.15	0.95	0.96	0.79	1.07
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	8.06	0.99	2.03	0.80 [1.62]	1.09
less duplication, 0.5x number of vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	8.11	1.01	2.02 (1.94)	0.80 [1.66]	1.10

* speedup vs. vector length 4: comparison to compression ratio 0.51

table continues on next page…

lower vector length: 3					
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73 → 11,642,026 elements in table*	15.71	1.00	*1*	1.00	1.06
less duplication: 1.13 → 9,480,500 unique	16.15	0.99	0.97	1.06	1.07
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	11.71	1.00	1.34	[1.35]	1.06
less duplication, lower number of vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	11.98	1.00	1.31 (1.30)	[1.43]	1.08
higher vector length: 8					
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	8.86	1.01	*1*	1.47	1.09
less duplication: 1.13 → 3,554,816 unique	10.26	1.00	0.86	1.32	1.10
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	18.47	1.00	0.48	1.48 [0.71]	1.10
worse compression ratio: 0.73 → 11,637,432 elements in table	11.90	1.00	0.74	1.33	1.08
less duplication, 2x number of vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	21.73	0.96	0.41 (0.41)	1.31 [0.62]	1.14
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	14.48	1.00	0.61 (0.64)	1.19	1.10
2x number of vectors, worse c. ratio: 8,000,000 vectors → 4,000,463 unique, 0.73 → 23,227,868 elements in table	25.44	0.99	0.35 (0.36)	1.29 [0.62]	1.09
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vecs., 0.73 → 41,347,914 elements in table	32.85	0.94	0.27 (0.31)	1.18 [0.52]	1.11

[z.zz] is speedup vs. vector length 4, 8,000,000 vectors (+ same duplication and compr. ratio)

B.3.4 Summary of random-data experiments

table size: 256MiB (no compensation for “lost” entries)
using optimal execution configuration, optimal bucket size

uncompressed:
`uncompressed_fixed`
block dimension: 256; grid dimension: 240
bucket size: 4 + `_lp8` (vector lengths ≤ 4); smallest/second smallest possible (*i.e.*, 8, 16 or 32)

compressed:
`compressed_`
block dimension: 256
grid dimension: 90 (vector length 1); 30 (vector length 2); 60
bucket size: 8 (vector length 1); 2

compressed (lr):
`compressed_lr`
block dimension: 256; grid dimension: 60
bucket size: 4 (vector lengths ≤ 3); 2

compressed (lr + 64b):
`compressed_lr_64_lp8` (vector lengths 1 and 2 also possible without `_lp8`: no effect)
block dimension: 256
grid dimension: 60 (vectors lengths 1 and 2); 30
bucket size: 8 (vector lengths 1 and 2); 2

compressed (nr):
`compressed_nr_l_np_sm`
block dimension: 128 (vector length 4); 256
grid dimension: 30 (vector lengths ≤ 3); 90 (vector length 4); 120
(bucket size: 2)

[PRD] Different input (vector length, duplication, no. of (unique) vectors, compression ratio) [fixed table size + optimal bucket size]

input	runtime uncomp. (ms) [U-I-fn-os]	runtime compressed (ms)				slowdwn. best compr. vs. uncompr.
		base [C-I-fn-os]	less rec. [Clr-I-fn-os]	less rec. + 64b [Clr64-I-fn-os]	non-rec. [Cnr-I-fn-os]	
vector length: 1 (compr. ratio: 2)						
default: 32,000,000 vectors, duplication 2.00 → 15,998,859 unique	23.93	35.02	**37.61**	35.32	**33.38**	1.39
less duplication: 1.12 → 28,445,271 unique vectors	32.43	42.37	**53.86**	**41.03**	46.33	1.27
0.25x number of (unique) vectors: 8,000,000 vectors → 4,000,067 unique	5.93	8.60	**9.24**	8.75	**8.03**	1.35
0.5x number of (unique) vectors: 16,000,000 vectors → 7,999,225 unique	12.15	17.26	**18.61**	17.63	**16.17**	1.33
less duplication, 0.25x no. (unique) vectors: 1.12 → 7,112,491 unique, 8,000,000 vectors	8.07	8.71	**9.43**	8.86	**8.33**	1.03
less duplication, 0.5x no. (unique) vectors: 1.12 → 14,224,527 unique, 16,000,000 vectors	16.17	17.73	**19.02**	17.84	**16.37**	1.01
vector length: 2 (compr. ratio: 1)						
default: 16,000,000 vectors, duplication 2.00 → 8,001,261 unique	12.01	**19.68**	18.76	17.72	**16.39**	1.36
less duplication: 1.12 → 14,223,349 unique vectors	16.67	**24.48**	20.22	18.27	**17.15**	1.03
0.5x number of (unique) vectors: 8,000,000 vectors → 4,000,852 unique	5.97	**9.37**	**9.31**	8.78	**8.06**	1.35
less duplication, 0.5x no. (unique) vectors: 1.12 → 7,111,243 unique, 8,000,000 vectors	8.10	**10.65**	9.47	8.90	**8.11**	1.00

table continues on next page…

vector length: 3 (min. compr. ratio: 0.67)						
default: 10,666,666 vectors, duplication 2.00 → 5,331,050 unique, *c. ratio 0.73* *→ 11,642,026 elements in table*	8.44	**24.31**	21.03	16.72	**15.71**	1.86
less duplication: 1.13 → 9,480,500 unique vectors	11.90	**25.75**	22.06	17.33	**16.15**	1.36
lower number of (unique) vectors: 8,000,000 vectors → 4,001,692 unique	6.24	**18.05**	15.68	12.47	**11.71**	1.88
less duplication, lower no. (unique) vectors: 1.12 → 7,112,578 unique, 8,000,000 vectors	8.58	**18.90**	16.38	12.90	**11.98**	1.40

table continues on next page…

vector length: 4 (min. compr. ratio: 0.50)						
default (8,000,000 vectors, duplication 2.00 → 4,002,399 unique, c. ratio 0.51 → 8,088,848 elements in table)	6.19	**20.33**	17.75	14.01	**13.04**	2.11
less duplication: 1.13 → 7,110,687 unique	8.53	**21.28**	18.68	14.57	**13.49**	1.58
0.5x number of (unique) vectors: 4,000,000 vectors → 1,999,584 unique	3.03	**10.03**	8.73	6.97	**6.47**	2.14
2x number of (unique) vectors: 16,000,000 vectors → 8,000,672 uniq.	13.03	**42.39**	37.19	29.51	**27.25**	2.09
worse compression ratio: 0.73 → 11,675,190 elements in table	*(6.19)*	**23.31**	20.62	16.51	**15.77**	2.55
less duplication, 0.5x no. (unique) vectors: 1.13 → 3,555,969 unique, 4,000,000 vectors	4.11	**10.17**	8.92	6.96	**6.45**	1.57
less duplication, 2x no. of (unique) vectors: 1.13 → 14,220,385 unique, 16,000,000 vectors	20.81	**45.47**	40.29	31.15	**28.57**	1.37
less duplication, worse compression ratio: 1.12 → 7,111,327 unique, 0.73 → 20,756,906 elements in table	*(8.53)*	**25.68**	22.85	17.97	**17.19**	2.02
0.5x no. of (unique) vectors, worse compression ratio: 4,000,000 vectors → 2,000,436 uniq., 0.73 → 5,837,892 elements in table	*(3.03)*	**11.41**	10.07	8.15	**7.73**	2.55
2x no. of (unique) vectors, worse compression ratio: 16,000,000 vectors → 8,004,559 uniq., 0.73 → 23,286,302 elements in table	*(13.03)*	**48.49**	43.29	33.79	**32.87**	2.52
less duplication, 0.5x number of (unique) vectors, worse compression ratio: 1.12 → 3,554,818 unique, 4,000,000 vectors, 0.73 → 10,391,626 elements in tbl.	*(4.11)*	**12.33**	10.93	8.79	**8.32**	2.02
less duplication, 2x number of (unique) vectors, worse compression ratio: 1.13 → 14,221,639 unique, 16,000,000 vectors, 0.73 → 41,474,428 elements in tbl.	*(20.81)*	**57.73**	52.97	41.69	**38.82**	1.87

table continues on next page…

vector length: 8 (min. compr. ratio: 0.25)						
default: 4,000,000 vectors, duplication 2.00 → 2,001,551 unique, c. ratio 0.50 → 8,053,058 elements in table	3.69	**15.47**	13.65	9.68	**8.86**	2.40
less duplication: 1.13 → 3,554,816 unique vectors	4.71	**17.71**	15.80	11.24	**10.26**	2.18
2x number of (unique) vectors: 8,000,000 vectors → 4,000,799 unique	7.90	**32.49**	28.59	20.30	**18.47**	2.34
worse compression ratio: 0.73 → 11,637,432 elements in table	*(3.69)*	**19.62**	17.89	12.88	**11.90**	3.23
less duplication, 2x no. of (unique) vectors: 1.12 → 7,111,271 unique, 8,000,000 vectors	12.23	**38.39**	34.22	24.71	**21.73**	1.78
less duplication, worse compression ratio: 1.12 → 3,555,564 unique, 0.73 → 20,695,988 elements in table	*(4.71)*	**23.72**	21.79	15.96	**14.48**	3.07
2x no. of (unique) vectors, worse compression ratio: 8,000,000 vectors → 4,000,463 uniq., 0.73 → 23,227,868 elements in table	*(7.90)*	**41.91**	38.43	27.66	**25.44**	3.22
less duplication, 2x no. of (unique) vectors, worse compression ratio: 1.13 → 7,110,432 unique, 8,000,000 vectors, 0.73 → 41,347,914 elements in tbl.	*(12.23)*	**53.08**	49.07	36.51	**32.85**	2.69

(x.xx) are values copied from the experiments with default compression ratio (compression ratio has no effect for uncompressed table)

B.4 Real-world data experiments

table size: 3GiB (szymanski5); 4GiB (lamport8); 8GiB - 128 bytes

uncompressed:
uncompressed_fixed
block dimension: 256; grid dimension: 240
bucket size: 4 + _lp8 (vector lengths ≤ 4); smallest/second smallest possible (*i.e.*, 8, 16 or 32)

1394: grid dimension: 30
1394.1/wafer_stepper.1: grid dimension: 60
odp: grid dimension: 150; bucket size: 16 (+ _lp32)
odp.1: grid dimension: 180
transit: grid dimension: 210; bucket size: 16 (+ _lp32)
lamport8/szymanski5: bucket size: 16 (+ _lp32x)
acs.1: grid dimension: 120

compressed:
compressed_
block dimension: 256
grid dimension: 90 (vector length 1); 30 (vector length 2); 60
bucket size: 8 (vector length 1); 2

1394/odp/lamport8/szymanski5: grid dimension: 30
acs.1: grid dimension: 90

compressed (lr):
compressed_lr
block dimension: 256; grid dimension: 60
bucket size: 4 (vector lengths ≤ 3); 2

1394/1394.1: grid dimension: 30; bucket size: 2
odp/odp.1/transit: grid dimension: 30
wafer_stepper.1/acs.1/sieve: grid dimension: 90

compressed (lr + 64b):
compressed_lr_64_lp8 (vector lengths 1 and 2 also possible without _lp8: no effect)
block dimension: 256
grid dimension: 60 (vectors lengths 1 and 2); 30
bucket size: 8 (vector lengths 1 and 2); 2

1394.1/odp: block dimension: 128
asyn3/szymanski5: grid dimension: 60

compressed (nr):
compressed_nr_l_np_sm (bucket size: 2)
block dimension: 128 (vector length 4); 256
grid dimension: 30 (vector lengths ≤ 3); 90 (vector length 4); 120

1394/1394.1: block dimension: 128
odp/odp.1: block dimension: 128; grid dimension: 30
transit/lamport8/acs.1/asyn3: grid dimension: 30
wafer_stepper/szymanski5: grid dimension: 60

[RWD] Different input (vector length, duplication, no. of (unique) vectors, compression ratio)
[fixed table size + optimal bucket size]

input	runtime uncompr. (ms)	runtime compressed (ms)				slowdown best compr. vs. uncompr.
		base	less rec.	less rec. + 64b	non-rec.	
vector length: 3 (min. compr. ratio: 0.67)						
`1394` (mCRL2): 355,339 vectors, duplication 1.79 → 198,692 unique, compression ratio 0.84 input size: 4.1MiB uncompressed size: 2.3MiB compressed size: 1.9MiB	0.51	**0.89**	**0.88**	0.82	**0.80**	1.6
`1394.1` (mCRL2): 23,792,770 vectors, duplication 2.35 → 10,138,812 unique, compression ratio 0.75 input size: 272MiB uncompressed size: 116MiB compressed size: 87MiB	25.7	**42.4**	41.5	38.4	**37.4**	1.5
vector length: 5 (min. compr. ratio: 0.40)						
`odp` (CADP): 641,227 vectors, duplication 7.02 → 91,394 unique, compression ratio 0.44 input size: 12MiB uncompressed size: 1.7MiB compressed size: 0.8MiB / 0.7MiB	0.64	**1.68**	1.58	1.47	**1.41**	2.2
`odp.1` (CADP): 31,091,555 vectors, duplication 4.04 → 7,699,456 unique, compression ratio 0.40 input size: 593MiB uncompressed size: 147MiB compressed size: 59MiB	33.9	**69.2**	63.7	**57.0**	58.9	1.7
`transit` (CADP): 39,925,525 vectors, duplication 10.61 → 3,763,192 unique, compression ratio 0.57 input size: 762MiB uncompressed size: 72MiB compressed size: 41MiB	29.4	**107**	**106**	102	**97**	3.3

table continues on next page…

vector length: 6 (min. compr. ratio: 0.33)						
`lamport8` (BEEM): 269,192,486 vectors, duplication 4.30 → 62,669,317 unique, compression ratio **0.33 / 0.34*** input size: 6161MiB uncompressed size: 1434MiB compressed size: 479MiB / 485MiB	169	**654**	552	**375**	*388*	2.2
`szymanski5` (BEEM): 375,297,914 vectors, duplication 4.72 → 79,518,740 unique, compression ratio **0.33 / 0.34*** input size: 8590MiB uncompressed size: 1820MiB compressed size: 609MiB / 618MiB	269	**877**	727	349	***335***	1.2
vector length: 8 (min. compr. ratio: 0.25)						
`wafer_stepper.1` (mCRL2): 16,977,693 vectors, duplication 4.50 → 3,772,753 unique, compression ratio 0.27 input size: 518MiB uncompressed size: 115MiB compressed size: 31MiB	15.8	**48.0**	43.5	**36.2**	40.4	2.3
vector length: 9 (min. compr. ratio: 0.22)						
`acs.1` (mCRL2): 895,005 vectors, duplication 4.47 → 200,317 unique, compression ratio 0.24 input size: 31MiB uncompressed size: 7MiB compressed size: 1.7MiB	0.99	**3.23**	2.62	**1.95**	2.06	2.0
vector length: 10 (min. compr. ratio: 0.20)						
`asyn3` (CADP): 80,686,290 vectors, duplication 5.14 → 15,688,570 unique, compression ratio **0.20 / 0.21*** input size: 3078MiB uncompressed size: 598MiB compressed size: 123MiB / 125MiB	61.8	**321**	289	**235**	*266*	3.8

* compression ratio/size dependent on (non-)rec.
may differ from run to run (if vector length not power of 2)