Analysis of a Radix Sort Algorithm on NVIDIA GPUs

Reference

Nadathur Satish, Mark Harris, and Michael Garland, "Designing efficient sorting algorithms for manycore GPUs," in the Proceedings of the IEEE International Parallel & Distributed Processing Symposium 2009.

Symbols

Radix-Sort Specific

n	Number of elements to sort.
2^{20}	Number of elements used in examples here.
h	Number of bits in bin.
4	Meaning there are $2^4 = 16$ bins in example histogram.
k	Number of bits in sort key.
32	
e	Number of array elements per thread.
4	Based on efficient techniques to find prefix sum in a block.

Other Symbols

M Number of multiprocessors.

16 A medium sized CC 1.0 device, also a convenient power of 2.

G Number of blocks in the grid.

$4M = 64$ Choose so that there are four blocks per MP, which is the maximum number of active blocks.

B Number of threads in a block.

256

p Number of CUDA cores.

$$p = \begin{cases} 8\text{M} & \text{for compute capability } 1.0 \leq i \leq 1.3 \\ 32\text{M} & \text{for compute capability } 2.0 \\ 48\text{M} & \text{for compute capability } 2.1 \\ 192\text{M} & \text{for compute capability } 3.x \\ 128\text{M} & \text{for compute capability } 5.x, 6.1 \end{cases}$$

Derived Quantities:

$\frac{k}{h}$	Number of global iterations.
$\frac{32}{4} = 8$	
eB	Number of elements per tile.
$4 \times 256 = 1024$	
$\frac{n}{eB}$	Number of tiles in the array.
$2^{20}2^{-10} = 2^{10}$	

Data Analysis

Pass 1

$2n$	Read and write array elements.
2^{21}	
$\frac{2^h}{eB}n$	Write per-tile histograms ($\frac{n}{eB}$ tiles times 2^h bins per histogram).
$2^{10}2^4 = 2^{14}$	
$G2^h$	Write per-block histograms.
$64 \times 2^4 = 2^{10}$	

Pass 2

$2n$ 2^{21}	Read and write array elements.
$\frac{2^h}{eB}n$ $2^{10}2^4 = 2^{14}$	Read per-tile histograms.
$G^2 2^h$ histogram. $64^2 \times 2^4 = 2^{16}$	Read per-block histograms. Note that each block reads every per-block

Total data transfer for both passes and all global iterations

$$\left(4n + 2\frac{n}{eB}2^h + G^2 2^h + G2^h\right)\frac{k}{h}$$

$$\left(2^{22} + 2 \times 2^{10}2^4 + 64^2 2^4 + 64 \times 2^4\right)\frac{32}{4}$$

$$\left(2^{22} + 2^{15} + 2^{16} + 2^{10}\right)8$$

$$\left(2^{22} + 2^{11}2^h + 2^{12}2^h + 2^8 2^h\right)\frac{32}{h}$$

Choice of h to Minimize Data Transfer

Minimize expression of form $(a + 2^h)/h$.

If $a = 0$ then $h = 1/\ln 2 \approx 1.443$.

For a corresponding to tile sizes:

Tile Size	Bin Size To Minimize Data
1024,	h = 7.33
512,	h = 7.01
256,	h = 6.55
128,	h = 6.00

Computation Analysis

Pass 1

$c_s n h \overbrace{\lg(eB)}^{\text{Prefix}}$ Sort within block.

$c_s 2^{20} \times 4 \times 10$

$c_{c1} n + c_{c2} \frac{n}{eB} 2^h$ Compute histograms.

$c_{c1} 2^{20} + c_{c2} 2^{10} 2^4$

Pass 2

$c_{gh} G(G2^h + h2^h)$ Compute global histogram.

$c_{gh} 64(64 \times 2^4 + 2^6) \approx 2^{14}$

$c_{of} \frac{n}{eB} h 2^h$ Compute tile digit offsets.

$c_{of} 2^{10} 2^6$

$c_{sc} n$ Scatter.

$c_{sc} 2^{20}$

Total for both passes and all global iterations

$$\left[n\left(c_{\mathrm{s}}h\lg(eB)+c_{\mathrm{c1}}+c_{\mathrm{sc}}\right)+\frac{n}{eB}2^h\left(c_{\mathrm{c2}}+c_{\mathrm{of}}h\right)+c_{\mathrm{gh}}G(G2^h+h2^h)\right]\frac{k}{h}$$

$$=\left[n\left(c_{\mathrm{s}}\lg(eB)+\frac{c_{\mathrm{c1}}}{h}+\frac{c_{\mathrm{sc}}}{h}\right)+\frac{n}{eB}2^h\left(\frac{c_{\mathrm{c2}}}{h}+c_{\mathrm{of}}\right)+c_{\mathrm{gh}}G2^h\left(\frac{G}{h}+1\right)\right]k$$

Analysis

Dominant term using default values is sort within block:

$c_{\mathrm{s}}nh\lg(eB)$ or $c_{\mathrm{s}}2^{20}\times 4\times 10$

Note that this makes h irrelevant for computation:

All iterations: $\left(c_{\mathrm{s}}nh\lg(eB)\right)\frac{k}{h}=\left(c_{\mathrm{s}}n\lg(eB)\right)k.$

Block Size Effect

Smaller blocks reduce block sort time (which dominates).

Smaller blocks increase time spent on histograms and offsets.

Offset time: $c_{\text{of}} \frac{n}{eB} h 2^h$ or $c_{\text{of}} 2^{10} 2^6$.

Solving for block size that makes offset and block sort time equal:

$$c_{\text{s}} n h \lg(eB) = c_{\text{of}} \frac{n}{eB} h 2^h$$

$$eB \lg(eB) = \frac{c_{\text{of}}}{c_{\text{s}}} 2^h$$

For sample values $eB = 7.79$ or $B = 1.95$.

Block Size Analysis Conclusions

Dominant term reduced with smaller block size, until block size 1.95.

We know making block smaller than 32 (warp size) will hurt efficiency.

So block size should be 32.

Other benefits: less use of shared memory so more blocks can be active.

Computation vs. Communication

Computation vs. Memory Bandwidth

Should we focus on minimizing data access (communication)?

Should we focus on minimizing computation?

Some Background

For a Pascal GTX 1080 GPUs: 14 insn / B or 55 insn / int . . .
. . . assuming an instruction mix with a throughput of 128 thd / cyc / MP.

Analysis

Dominant term for computation: $c_s nh \lg(eB)$ (block sort time).

Dominant term for data access: $4n$.

Code is compute-bound if:

$$\frac{1}{55} c_s nh \lg(eB) > 4n$$
$$c_s h \lg(eB) > 220$$

With example values:

$$c_s 40 > 220$$

Computation and data accessed balanced if:

Block sort is very tightly coded so that $c_s < 5.5$.

Tile size (eB) is smaller.