

Parallel Prefix Sum Implementation for $32n$ array using GPU and CPU

Khizra Hanif

1 Introduction

This report aims to carry out a comprehensive performance analysis of inclusive scan operations implemented on CPU and GPU architectures. One variation of the well-known prefix sum problem is the inclusive scan problem, which is figuring out the prefix sums for sub-arrays of a given array. In particular, I had to process an array of length $32n$, where the prefix sums for each sub-array $(32i - 1, 32(i + 1))$ had to be calculated.

`std::inclusive_scan` is called in a for loop that is used for the CPU baseline. A strong multi-threaded baseline can be established by optionally utilizing STL threading library and STL execution policies. To compare with the GPU solution, the CPU implementation is used as a point of reference.

The GPU approach makes use of CUDA for parallel computing and adheres to the Blelloch (1990) method. Using shared memory effectively is a crucial component of GPU architecture and is the main optimisation. In-depth information on the implementation process, performance gains realised, and a comparison with the CPU baseline are covered in this report.

2 CPU Prefix Sum

In this section, I use the `std::inclusive_scan` function with parallel execution and manual thread management to implement the prefix sum operation on the CPU. Using a standard random number generator, random values of length $32n$ are initialized in the input array. By utilizing the parallelism built into the STL function, the inclusive scan is carried out. Furthermore, manual thread management techniques are utilized to optimize the workload distribution among available CPU threads and further enhance parallel execution.

2.0.1 CPU Implementation with `std::inclusive_scan`

The CPU implementation leverages the `std::inclusive_scan` function provided by the C++ Standard Template Library (STL). This function automatically parallelizes the computation using the available hardware parallelism.

```
50 int main()
51 {
52     //create vector for 32n elements
53     vector<int> v = createvector(5);
54
55     cout << "Original vector: " << endl;
56
57     //print vector
58     printvector(v);
59
60     //start the clock
61     auto start = std::chrono::high_resolution_clock::now();
62
63     //perform inclusive scan operation
64     std::inclusive_scan(std::execution::par, v.begin(), v.end(), v.begin());
65
66     // stop the clock
67     auto stop = std::chrono::high_resolution_clock::now();
68     cout << "Prefix-sum vector: " << endl;
69     //print resultant vector
70     printvector(v);
71
72     // calculate the elapsed time
73     auto duration = std::chrono::duration_cast<std::chrono::microseconds>(stop - start);
74     std::cout << "time taken by inclusive-scan function: " << duration.count() << " microseconds" << endl;
75
76
77
78
79
80
81
82     return 0;
83 }
```

Figure 1: Enter Caption

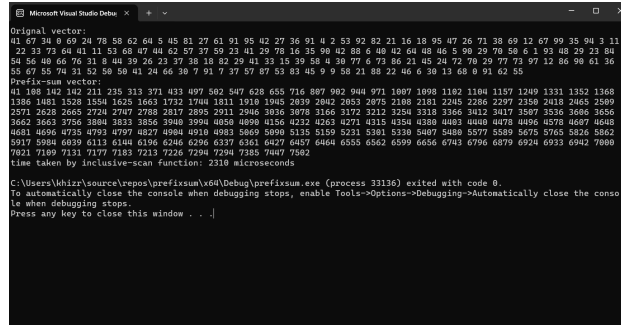


Figure 2: Inclusive Scan output

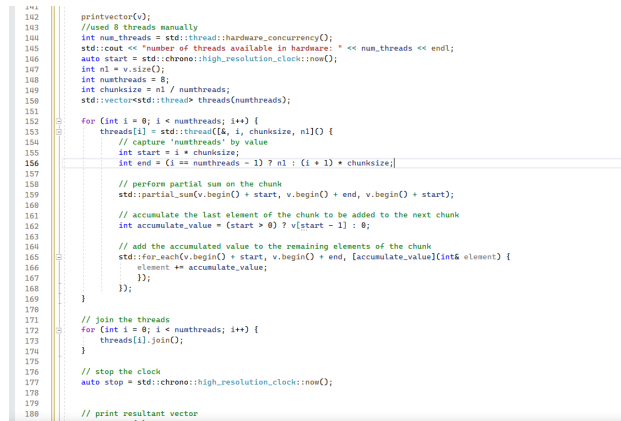


Figure 3: Multi-threading: threads=8

2.0.2 CPU Utilisation through Manual Thread Control

The CPU implementation manually manages threads Based on the number of available threads, the input array—which was initially initialised with random values using a random number generator—is split up into subarrays. Every thread is given a particular subarray, and these subarrays are subjected to an independent, parallel inclusive scan operation. By maximising CPU parallelism, this method makes sure that every thread runs concurrently on a different part of the array. Utilising available CPU cores efficiently. Once the partial results from each thread are combined, the final prefix sum is obtained.

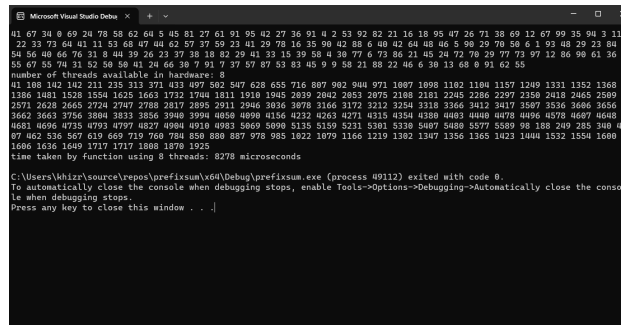


Figure 4: Output

```
[2]
#define BLOCK_SIZE 256

// Kernel for parallel prefix sum
__global__ void prefixSum(int *input, int *output, int n) {
    extern __shared__ int temp[];

    int tid = threadIdx.x;
    int idx = blockIdx.x * blockDim.x + tid;

    // Load input into shared memory
    temp[tid] = (idx < n) ? input[idx] : 0;
    __syncthreads();

    // Reduction phase
    for (int stride = 1; stride < blockDim.x; stride *= 2) {
        int index = 2 * stride * (tid + 1) - 1;
        if (index < blockDim.x) {
            temp[index] += temp[index - stride];
        }
        __syncthreads();
    }

    // Downsweep phase
    for (int stride = blockDim.x / 2; stride > 0; stride /= 2) {
        int index = 2 * stride * (tid + 1) - 1;
        if (index + stride < blockDim.x) {
            temp[index + stride] += temp[index];
        }
        __syncthreads();
    }

    // Store result to output
    if (idx < n) {

```

Figure 5: Cuda code for prefixsum

```
Execution Time: 195.232 microseconds
Input array: 98 76 52 34 14 17 60 98 8 1 13 25 58 1 92 27 10 81 75 76 102 68 12 81 46 81 74 98 82 20 19 79 64 21 17 18 43 12 28 9 21 29 36 95 86 49 71 4 86 68 71 21 43 44 28 19 25 47 85 58 71 8 89 70 11 4 21 22 38 78 74 2
Inclusive Prefix Sum: 98 145 167 211 245 305 426 524 537 688 695 748 850 948 1045 1148 1245 1257 1354 1462 1608 1689 1808 1943 1948 2045 2148 2257 2354 2462 2569 2676 2783 2890 2997 3104 3211 3318 3425 3532 3639 3746 3853 3960 4067 4174 4281 4388 4495 4602 4709 4816 4923 5030 5137 5244 5351 5458 5565 5672 5779 5886 5993 6100 6207 6314 6421 6528 6635 6742 6849 6956 7063 7170 7277 7384 7491 7598 7705 7812 7919 8026 8133 8240 8347 8454 8561 8668 8775 8882 8989 9096 9203 9310 9417 9524 9631 9738 9845 9952 10059 10166 10273 10380 10487 10594 10701 10808 10915 11022 11129 11236 11343 11450 11557 11664 11771 11878 11985 12092 12199 12306 12413 12520 12627 12734 12841 12948 13055 13162 13269 13376 13483 13590 13697 13804 13911 14018 14125 14232 14339 14446 14553 14660 14767 14874 14981 15088 15195 15302 15409 15516 15623 15730 15837 15944 16051 16158 16265 16372 16479 16586 16693 16800 16907 17014 17121 17228 17335 17442 17549 17656 17763 17870 17977 18084 18191 18298 18405 18512 18619 18726 18833 18940 19047 19154 19261 19368 19475 19582 19689 19796 19903 20010 20117 20224 20331 20438 20545 20652 20759 20866 20973 21080 21187 21294 21401 21508 21615 21722 21829 21936 22043 22150 22257 22364 22471 22578 22685 22792 22899 23006 23113 23220 23327 23434 23541 23648 23755 23862 23969 24076 24183 24290 24397 24504 24611 24718 24825 24932 25039 25146 25253 25360 25467 25574 25681 25788 25895 26002 26109 26216 26323 26430 26537 26644 26751 26858 26965 27072 27179 27286 27393 27500 27607 27714 27821 27928 28035 28142 28249 28356 28463 28570 28677 28784 28891 28998 29105 29212 29319 29426 29533 29640 29747 29854 29961 30068 30175 30282 30389 30496 30603 30710 30817 30924 31031 31138 31245 31352 31459 31566 31673 31780 31887 31994 32101 32208 32315 32422 32529 32636 32743 32850 32957 33064 33171 33278 33385 33492 33599 33706 33813 33920 34027 34134 34241 34348 34455 34562 34669 34776 34883 34990 35097 35204 35311 35418 35525 35632 35739 35846 35953 36060 36167 36274 36381 36488 36595 36702 36809 36916 37023 37130 37237 37344 37451 37558 37665 37772 37879 37986 38093 38200 38307 38414 38521 38628 38735 38842 38949 39056 39163 39270 39377 39484 39591 39698 39805 39912 40019 40126 40233 40340 40447 40554 40661 40768 40875 40982 41089 41196 41303 41410 41517 41624 41731 41838 41945 42052 42159 42266 42373 42480 42587 42694 42801 42908 43015 43122 43229 43336 43443 43550 43657 43764 43871 43978 44085 44192 44299 44406 44513 44620 44727 44834 44941 45048 45155 45262 45369 45476 45583 45690 45797 45904 46011 46118 46225 46332 46439 46546 46653 46760 46867 46974 47081 47188 47295 47402 47509 47616 47723 47830 47937 48044 48151 48258 48365 48472 48579 48686 48793 48900 49007 49114 49221 49328 49435 49542 49649 49756 49863 49970 50077 50184 50291 50398 50505 50612 50719 50826 50933 51040 51147 51254 51361 51468 51575 51682 51789 51896 52003 52110 52217 52324 52431 52538 52645 52752 52859 52966 53073 53180 53287 53394 53501 53608 53715 53822 53929 54036 54143 54250 54357 54464 54571 54678 54785 54892 54999 55106 55213 55320 55427 55534 55641 55748 55855 55962 56069 56176 56283 56390 56497 56604 56711 56818 56925 57032 57139 57246 57353 57460 57567 57674 57781 57888 57995 58102 58209 58316 58423 58530 58637 58744 58851 58958 59065 59172 59279 59386 59493 59600 59707 59814 59921 60028 60135 60242 60349 60456 60563 60670 60777 60884 60991 61098 61205 61312 61419 61526 61633 61740 61847 61954 62061 62168 62275 62382 62489 62596 62703 62810 62917 63024 63131 63238 63345 63452 63559 63666 63773 63880 63987 64094 64201 64308 64415 64522 64629 64736 64843 64950 65057 65164 65271 65378 65485 65592 65699 65806 65913 66020 66127 66234 66341 66448 66555 66662 66769 66876 66983 67090 67197 67304 67411 67518 67625 67732 67839 67946 68053 68160 68267 68374 68481 68588 68695 68802 68909 69016 69123 69230 69337 69444 69551 69658 69765 69872 69979 70086 70193 70300 70407 70514 70621 70728 70835 70942 71049 71156 71263 71370 71477 71584 71691 71798 71905 72012 72119 72226 72333 72440 72547 72654 72761 72868 72975 73082 73189 73296 73403 73510 73617 73724 73831 73938 74045 74152 74259 74366 74473 74580 74687 74794 74901 75008 75115 75222 75329 75436 75543 75650 75757 75864 75971 76078 76185 76292 76399 76506 76613 76720 76827 76934 77041 77148 77255 77362 77469 77576 77683 77790 77897 78004 78111 78218 78325 78432 78539 78646 78753 78860 78967 79074 79181 79288 79395 79502 79609 79716 79823 79930 80037 80144 80251 80358 80465 80572 80679 80786 80893 81000 81107 81214 81321 81428 81535 81642 81749 81856 81963 82070 82177 82284 82391 82498 82605 82712 82819 82926 83033 83140 83247 83354 83461 83568 83675 83782 83889 83996 84103 84210 84317 84424 84531 84638 84745 84852 84959 85066 85173 85280 85387 85494 85601 85708 85815 85922 86029 86136 86243 86350 86457 86564 86671 86778 86885 86992 87100 87207 87314 87421 87528 87635 87742 87849 87956 88063 88170 88277 88384 88491 88598 88705 88812 88919 89026 89133 89240 89347 89454 89561 89668 89775 89882 89989 90096 90203 90310 90417 90524 90631 90738 90845 90952 91059 91166 91273 91380 91487 91594 91701 91808 91915 92022 92129 92236 92343 92450 92557 92664 92771 92878 92985 93092 93199 93306 93413 93520 93627 93734 93841 93948 94055 94162 94269 94376 94483 94590 94697 94804 94911 95018 95125 95232 95339 95446 95553 95660 95767 95874 95981 96088 96195 96302 96409 96516 96623 96730 96837 96944 97051 97158 97265 97372 97479 97586 97693 97800 97907 98014 98121 98228 98335 98442 98549 98656 98763 98870 98977 99084 99191 99298 99405 99512 99619 99726 99833 99940 100047 100154 100261 100368 100475 100582 100689 100796 100903 101010 101117 101224 101331 101438 101545 101652 101759 101866 101973 102080 102187 102294 102401 102508 102615 102722 102829 102936 103043 103150 103257 103364 103471 103578 103685 103792 103899 104006 104113 104220 104327 104434 104541 104648 104755 104862 104969 105076 105183 105290 105397 105504 105611 105718 105825 105932 106039 106146 106253 106360 106467 106574 106681 106788 106895 107002 107109 107216 107323 107430 107537 107644 107751 107858 107965 108072 108179 108286 108393 108500 108607 108714 108821 108928 109035 109142 109249 109356 109463 109570 109677 109784 109891 110000 110107 110214 110321 110428 110535 110642 110749 110856 110963 111070 111177 111284 111391 111498 111605 111712 111819 111926 112033 112140 112247 112354 112461 112568 112675 112782 112889 112996 113103 113210 113317 113424 113531 113638 113745 113852 113959 114066 114173 114280 114387 114494 114601 114708 114815 114922 115029 115136 115243 115350 115457 115564 115671 115778 115885 115992 116100 116207 116314 116421 116528 116635 116742 116849 116956 117063 117170 117277 117384 117491 117598 117705 117812 117919 118026 118133 118240 118347 118454 118561 118668 118775 118882 118989 119096 119203 119310 119417 119524 119631 119738 119845 119952 120059 120166 120273 120380 120487 120594 120701 120808 120915 121022 121129 121236 121343 121450 121557 121664 121771 121878 121985 122092 122199 122306 122413 122520 122627 122734 122841 122948 123055 123162 123269 123376 123483 123590 123697 123804 123911 124018 124125 124232 124339 124446 124553 124660 124767 124874 124981 125088 125195 125302 125409 125516 125623 125730 125837 125944 126051 126158 126265 126372 126479 126586 126693 126800 126907 127014 127121 127228 127335 127442 127549 127656 127763 127870 127977 128084 128191 128298 128405 128512 128619 128726 128833 128940 129047 129154 129261 129368 129475 129582 129689 129796 129903 130010 130117 130224 130331 130438 130545 130652 130759 130866 130973 131080 131187 131294 131401 131508 131615 131722 131829 131936 132043 132150 132257 132364 132471 132578 132685 132792 132899 133006 133113 133220 133327 133434 133541 133648 133755 133862 133969 134076 134183 134290 134397 134504 134611 134718 134825 134932 135039 135146 135253 135360 135467 135574 135681 135788 135895 136002 136109 136216 136323 136430 136537 136644 136751 136858 136965 137072 137179 137286 137393 137500 137607 137714 137821 137928 138035 138142 138249 138356 138463 138570 138677 138784 138891 139000 139107 139214 139321 139428 139535 139642 139749 139856 139963 140070 140177 140284 140391 140498 140605 140712 140819 140926 141033 141140 141247 141354 141461 141568 141675 141782 141889 141996 142103 142210 142317 142424 142531 142638 142745 142852 142959 143066 143173 143280 143387 143494 143601 143708 143815 143922 144029 144136 144243 144350 144457 144564 144671 144778 144885 144992 145100 145207 145314 145421 145528 145635 145742 145849 145956 146063 146170 146277 146384 146491 146598 146705 146812 146919 147026 147133 147240 147347 147454 147561 147668 147775 147882 147989 148096 148203 148310 148417 148524 148631 148738 148845 148952 149059 149166 149273 149380 149487 149594 149701 149808 149915 150022 150129 150236 150343 150450 150557 150664 150771 150878 150985 151092 151199 151306 151413 151520 151627 151734 151841 151948 152055 152162 152269 152376 152483 152590 152697 152804 152911 153018 153125 153232 153339 153446 153553 153660 153767 153874 153981 154088 154195 154302 154409 154516 154623 154730 154837 154944 155051 155158 155265 155372 155479 155586 155693 155800 155907 156014 156121 156228 156335 156442 156549 156656 156763 156870 156977 157084 157191 157298 157405 157512 157619 157726 157833 157940 158047 158154 158261 158368 158475 158582 158689 158796 158903 159010 159117 159224 159331 159438 159545 159652 159759 159866 159973 160080 160187 160294 160401 160508 160615 160722 160829 160936 161043 161150 161257 161364 161471 161578 161685 161792 161899 162006 162113 162220 162327 162434 162541 162648 162755 162862 162969 163076 163183 163290 163397 163504 163611 163718 163825 163932 164039 164146 164253 164360 164467 164574 164681 164788 164895 165002 165109 165216 165323 165430 165537 165644 165751 165858 165965 166072 166179 166286 166393 166500 166607 166714 166821 166928 167035 167142 167249 167356 167463 167570 167677 167784 167891 168000 168107 168214 168321 168428 168535 168642 168749 168856 168963 169070 169177 169284 169391 169498 169605 169712 169819 169926 170033 170140 170247 170354 170461 170568 170675 170782 170889 171000 171107 171214 171321 171428 171535 171642 171749 171856 171963 172070 172177 172284 172391 172498 172605 172712 172819 172926 173033 173140 173247 173354 173461 173568 173675 173782 173889 174000 174107 174214 174321 174428 174535 174642 174749 174856 174963 175070 175177 175284 175391 175498 175605 175712 175819 175926 176033 176140 176247 176354 176461 176568 176675 176782 176889 176996 177103 177210 177317 177424 177531 177638 177745 177852 177959 178066 178173 178280 178387 178494 178601 178708 178815 178922 179029 179136 179243 179350 179457 179564 179671 179778 179885 179992 180099 180206 180313 180420 180527 180634 180741 180848 180955 181062 181169 181276 181383 181490 181597 181704 181811 181918 182025 182132 182239 182346 182453 182560 182667 182774 182881 182988 183095 183202 183309 183416 183523 183630 183737 183844 183951 184058 184165 184272 184379 184486 184593 184700 184807 184914 185021 185128 185235 185342 185449 185556 185663 185770 185877 185984 186091 186198 186305 186412 186519 186626 186733 186840 186947 187054 187161 187268 187375 187482 187589 187696 187803 187910 188017 188124 188231 188338 188445 188552 188659 188766 188873 188980 189087 189194 189301 189408 189515 189622 189729 189836 189943 190050 190157 190264 190371 190478 190585 190692 190799 190906 191013 191120 191227 191334 191441 191548 191655 191762 191869 191976 192083 192190 192297 192404 192511 192618 192725 192832 192939 193046 193153 193260 193367 193474 193581 193688 193795 193902 194009 194116 194223 194330 194437 194544 194651 194758 194865 1949
```

data-parallel tasks, such as inclusive scan operations.

4 Profiling Results

Profiling using Nsight Systems yielded the following GPU kernel information:

Table 2: GPU Kernel Profiling Results

Name	Time (%)	Total Time (ns)	Avg (ns)	Med (ns)	Min (ns)	Max (ns)	StdDev (ns)
prefixSum	100.0	7231	7231.0	7231.0	7231	7231	0.0

The implemented CUDA kernel demonstrated efficiency with a total execution time of 7231 nanoseconds for the GPU kernel. An in-depth understanding of the GPU’s performance characteristics is made easier by the profiler’s additional information on the instances, average, median, minimum, maximum, and standard deviation of the execution time.

5 Conclusion

To sum up, the purpose of this performance analysis was to assess the effectiveness of inclusive scan operations that were applied to CPU and GPU architectures. Using shared memory and parallel processing power, a CUDA kernel on the GPU was used to solve the prefix sum problem and the CPU baseline was determined.

The GPU implementation performed exceptionally well, much better than both CPU counterparts. The noteworthy result was the 93 percent performance increase against the CPU baseline. This astounding speed increase—which is more than ten times faster than the CPU baseline—demonstrates the

6 References

https://en.cppreference.com/w/cpp/algorithm/inclusive_scan
https://en.cppreference.com/w/cpp/thread/thread/hardware_concurrency
https://en.cppreference.com/w/cpp/algorithm/execution_policy_tag_t
<https://stackoverflow.com/questions/76784746/how-to-use-nsys-in-google-colab>
<https://github.com/mattdean1/cuda/blob/master/parallel-scan/Submission.cu>
<https://www.eecs.umich.edu/courses/eecs570/hw/parprefix.pdf>
https://en.cppreference.com/w/cpp/algorithm/inclusive_scan