

Project Report

Project description: In this project we will implement a GPU version of the parallel radix partition using CUDA. Radix partition is a partition mechanism commonly used in parallel hash join. It reorders the input keys so that the keys that have the same radix value (or hash value) are gathered into a contiguous memory space that forms a partition. Here we perform three operations implementing three kernels namely: **Histogram, Prefix-scan and Reorder.**

The output array consists of all the partitions and the partitions with smaller radix values are located in front of those with larger radix values.

Histogram Kernel:

The histogram that is calculated here is somewhat similar to the Project 2 implementation and uses shared memory to store an array of partitions which consist of the keys who have same radix value and belong to the same partition.

I have used here **output privatization** to reduce race conditions that may occur within the threads when they try to access the same global memory location. I have used private copies of partitions on the shared memory which stores the histogram of radix values(**here the histogram consists of 32 threads per block that span over the entire key array**). Here we use a **bit field extraction** where a start index and number of bits to be extracted are provided along with the element of the input key array to hash the keys to a particular radix value.

This histogram of the radix values is stored on the shared memory and is then atomic added to the actual histogram.

Prefix Scan Kernel:

The basic goal here is to calculate a prefix scan of the histogram array obtained from the histogram kernel so that we can provide it as an input to the Reorder kernel which will then reorder them with the smaller radix values on the front side than the larger radix value.

We specifically consider here the exclusive scan which doesn't consider here the last element of the array while scanning.

Prefix scan is a type of parallel scan which uses an idea to build a balanced binary tree on the input data and sweep it to and from the root to compute the prefix sum.

We try avoiding the extra $\log n$ time required by the naive scan approach.

So this algorithm is an adaptation of **Blelloch's scan algorithm** where the basic steps followed are the input is loaded in to the **shared memory(sharedpartitions[])**, then a in place sum is build followed by removing the last element and writing back results to the **device memory (Hist_dev_pre[])**.

ReOrder Kernel:

In Reorder kernel the main goal is to implement and arrange the radix value partition starting from the smaller values to all the way to the larger values. This is done again using a bfe function where we calculate a hash value and increment that position of the Hist_pre array by 1. This gives us the offset value which is then indexed in the final reorder output array and the actual key is assigned to this offset reordering all the keys with respect to their radix values.

Result Snippets:**1. Array of Keys Size = 20 , Number of partitions = 8**

```
[[sbhadale@c4cuda05 cuda]$ ./p3 20 8
*****
Running Time of Histogram Kernel: 0.04707 msec i.e 0.00005 *****
3 3 3 3 2 2 2 2

*****
Running Time of prefixScan Kernel: 0.01795 msec i.e 0.00002 sec*****
0 3 6 9 12 14 16 18

*****
Running Time of ReOrder Kernel: 0.01408 msec i.e 0.00001 sec *****
0 8 16 1 9 17 10 18 2 19 11 3 4 12 13 5 14 6 15 7

*****
Total Running Time of All Kernels: 0.07910 msec i.e 0.00008 sec*****
[sbhadale@c4cuda05 cuda]$ ]
```

2. Array of Keys Size = 1000 , Number of partitions = 4

```
[sbhadale@c4cuda05 cuda]$ ./p3 1000 4
*****
Running Time of Histogram Kernel: 0.04726 msec i.e 0.00005 *****
250 250 250 250

*****
Running Time of prefixScan Kernel: 0.01718 msec i.e 0.00002 sec*****
0 250 500 750

*****
Running Time of ReOrder Kernel: 0.01587 msec i.e 0.00002 sec *****
980 400 592 712 88 236 32 564 576 328 516 292 412 876 436 792 484 532 168 252 188 664 772 144 464 752 160 184 280 536 324 112 288 352 8 448 24 684 212 300 100 296 96 136 60 500 496 528 948 244 472 404 624
824 368 592 240 724 568 124 788 468 548 628 488 322 988 612 164 812 176 108 216 520 314 414 834 28 12 152 46 68 692 552 264 584 940 320 284 548 668 308 448 988 952 237 344 620 188
340 392 444 672 648 104 348 744 972 684 784 468 608 964 48 708 396 992 72 936 768 968 380 756 856 508 688 892 976 872 788 140 736 984 248 272 924 128 696 676 120 64 384 25
916 768 592 848 832 124 788 468 548 628 488 322 284 280 488 852 192 408 0 980 644 156 376 884 716 668 996 44 652 420 820 372 732 764 776 880 476 312 364 636 224 804 556 584 92 788 840 336 796 746
388 596 94 720 688 268 616 632 648 76 928 196 868 428 452 816 864 504 844 52 288 800 540 148 132 260 828 80 84 704 888 968 466 28 276 868 544 744 656 384 69 61 801 17 981 793 589 229 761 645 749 317 413
617 285 281 661 333 421 213 825 557 621 717 197 569 881 989 37 429 513 797 113 677 349 477 485 453 961 449 185 481 485 693 489 613 445 505 637 121 897 149 401 169 129 101 89 589 725 261 385 973 997 7
69 1 581 241 861 293 245 437 817 21 765 869 529 821 301 953 873 353 933 937 237 773 789 469 721 181 145 917 665 921 193 425 913 13 781 153 249 653 781 25 949 161 165 461 993 65 669 889 389 117 157 753 189
173 649 137 561 629 945 341 373 837 133 537 493 289 781 813 417 593 733 329 313 833 497 93 365 337 565 729 433 465 233 321 841 893 33 789 49 253 885 757 685 969 273 657 981 297 522 597 989 577 581 177 92
9 525 977 845 9 125 829 225 981 889 853 699 77 85 201 457 97 221 785 265 865 189 689 393 573 217 385 877 673 745 857 681 41 713 965 489 737 585 925 473 57 81 389 381 967 885 377 941 73 397 441 285 849 281
541 108 517 553 641 269 685 369 601 345 545 777 984 549 277 29 433 326 625 533 53 5 141 361 357 257 45 741 694 934 114 222 886 818 892 678 474 778 786 934 662 814 494 454 870 742 678 854 156 698 680 10 7
02 464 462 318 858 382 818 86 42 579 938 630 358 598 430 386 426 438 282 58 170 614 270 358 838 204 286 794 246 746 774 210 274 638 714 738 546 294 478 238 894 82 186 94 558 898 794 362 582 594 698 418 27
8 942 282 534 858 102 162 314 338 166 242 554 846 518 458 838 394 266 750 862 692 562 78 766 482 718 366 214 646 710 502 118 054 226 410 926 758 942 38 499 966 399 446 918 338 398 62 738 686 98 522 586 32
6 734 322 994 234 262 878 874 310 510 194 606 834 46 806 186 50 354 538 2 298 574 586 610 882 238 26 914 556 634 154 842 138 6 54 866 618 74 334 66 146 382 486 199 134 582 578 110 918 726 722 414 22 7
78 442 822 90 998 174 250 986 178 198 642 988 34 826 950 798 122 614 142 158 902 406 530 182 978 974 594 138 434 622 542 14 922 990 422 30 342 899 79 626 126 754 782 674 786 218 459 254 498 526 378 492 34
6 566 982 386 654 762 378 666 978 298 466 986 470 18 658 374, 871 767 23 539 315 467 731 103 491 755 671 63 593 331 851 57 775 159 885 223 298 167 111 527 883 815 935 763 611 879 279 727 251 579 911 495
339 235 3 675 423 271 263 735 99 543 559 19 385 856 563 687 391 531 239 43 715 595 623 631 291 207 39 679 639 255 451 619 823 783 947 739 883 139 463 483 699 807 367 275 978 643 931 435 127 447 567 711 6
23 231 899 811 371 723 191 951 123 831 939 183 487 919 915 691 443 859 395 387 799 215 843 211 363 895 267 707 743 999 807 411 79 555 175 751 955 219 319 471 195 647 819 27 87 31 535 439 359 47 927 287 91
163 847 583 55 587 187 71 479 199 379 431 779 475 627 467 867 351 511 403 59 495 355 259 827 35 663 883 591 399 327 943 923 455 119 283 99 667 787 995 427 747 383 335 387 987 459 783 143 983 95 807 773 9
71 599 247 75 571 839 823 575 887 299 147 987 375 51 547 11 759 243 891 343 311 603 659 115 83 959 171 131 875 615 516 619 979 551 179 719 963 695 967 155 419 107 791 499 283 227 415 15 347 151 983 135 65
1 683 795 655

*****
Total Running Time of All Kernels: 0.08032 msec i.e 0.00008 sec*****
[sbhadale@c4cuda05 cuda]$ ]
```

3. Array of Keys Size = 1000 , Number of partitions = 32

4. Array of Keys Size = 10000 , Number of partitions = 32

[the above and below images are continuation of each other but they consist of all the values as can fit them in a single screen shot (same applies to Results 5 and 6)]

5. Array of Keys Size = 100000 , Number of partitions = 32

6. Array of Keys Size = 1000000 , Number of partitions = 32