Parallel Sorting Assignment Report

Parallel Bucketsort via MPI

BY

Miss Kanrawee Chiamsakul 6188049

Mr. Tharit Chantanalertvilai 6188068

Present

Assoc. Prof. Dr. Sudsanguan Ngamsuriyaroj

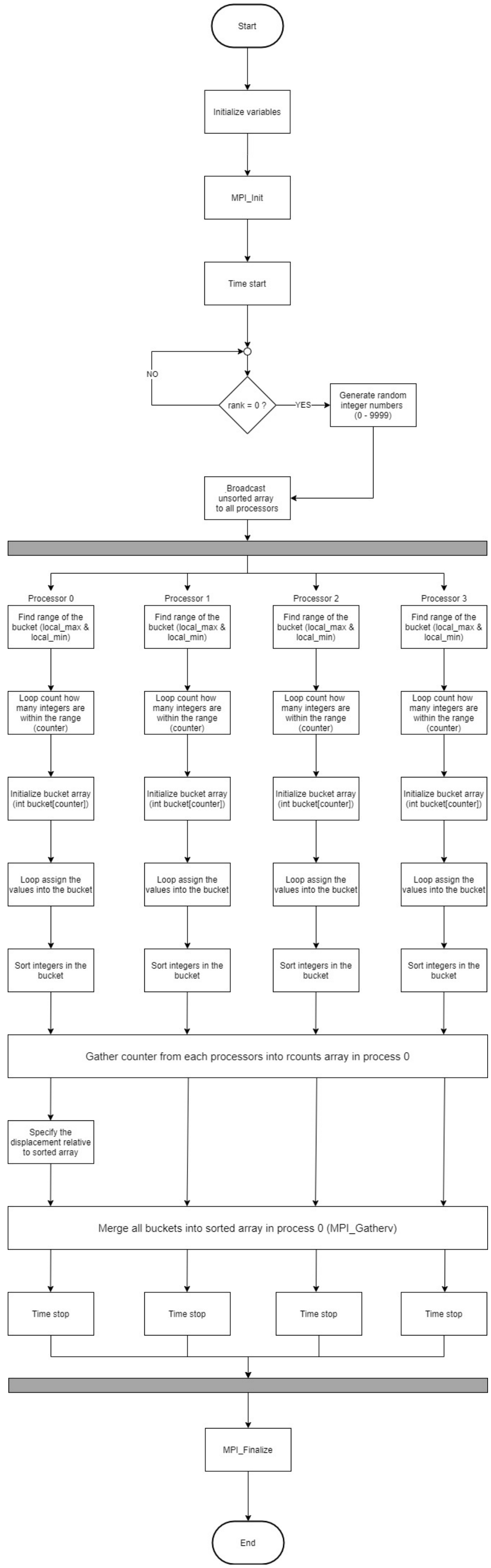
ITCS443 Parallel and Distributed Systems

Faculty of Information and Communication Technology

Mahidol University

2020

Explanation of the Source Code as a Flowchart



Testing Results with Sample Capture Screen Shots

1. 10,000 random integer numbers

```
/project $ mpicc -o out bucketsort.c
/project $ mpirun -np 1 ./out
Process 0 took 0.319473 seconds to run.
/project $ mpirun -np 2 ./out
Process 1 took 0.163924 seconds to run.
Process 0 took 0.171179 seconds to run.
/project $ mpirun -np 3 ./out
Process 2 took 0.105898 seconds to run.
Process 1 took 0.139083 seconds to run.
Process 0 took 0.141575 seconds to run.
/project $ mpirun -np 4 ./out
Process 1 took 0.105823 seconds to run.
Process 2 took 0.123137 seconds to run.
Process 3 took 0.127459 seconds to run.
Process 0 took 0.142905 seconds to run.
```

2. 100,000 random integer numbers

```
/project $ mpicc -o out bucketsort.c
/project $ mpirun -np 1 ./out
Process 0 took 18.340747 seconds to run.
/project $ mpirun -np 2 ./out
Process 1 took 9.201187 seconds to run.
Process 0 took 9.211407 seconds to run.
/project $ mpirun -np 3 ./out
Process 2 took 6.090886 seconds to run.
Process 1 took 6.092352 seconds to run.
Process 0 took 6.102874 seconds to run.
/project $ mpirun -np 4 ./out
Process 1 took 4.557874 seconds to run.
Process 2 took 4.658165 seconds to run.
Process 3 took 4.694629 seconds to run.
Process 0 took 4.728061 seconds to run.
```

Parallel Image [Running] - Oracle VM VirtualBox

```
 9308 9308 9309 9310 9311 9313 9313 9314 9315 9319 9321 9323 9325 9326 9327 9328 9330 9331 9331 9333
 9333 9334 9334 9336 9337 9338 9339 9339 9340 9340 9340 9341 9342 9342 9342 9343 9345 9345 9346 9346
 9347 9347 9348 9350 9352 9352 9352 9354 9356 9357 9358 9358 9360 9361 9362 9363 9364 9364 9365 9366
 9367 9369 9369 9370 9370 9371 9372 9372 9374 9375 9376 9376 9377 9377 9379 9381 9381 9381 9382 9385
 9387 9387 9388 9388 9390 9391 9391 9391 9392 9392 9392 9395 9395 9396 9396 9396 9397 9397 9398 9398
 9400 9401 9401 9402 9402 9404 9404 9405 9408 9408 9409 9410 9410 9410 9411 9411 9411 9412 9413 9415
 9415 9416 9416 9417 9417 9418 9421 9425 9425 9426 9426 9427 9427 9429 9429 9430 9431 9433 9433 9435
 9436 9436 9438 9440 9441 9442 9444 9445 9446 9448 9449 9449 9452 9452 9452 9454 9455 9455 9456 9458
 9458 9458 9458 9459 9460 9460 9460 9461 9461 9462 9463 9463 9463 9464 9465 9467 9468 9469 9471 9471
 9471 9472 9472 9473 9473 9475 9476 9478 9478 9479 9481 9481 9482 9484 9485 9486 9489 9490 9491 9491
 9492 9493 9494 9496 9497 9498 9500 9500 9500 9502 9502 9503 9503 9504 9504 9506 9507 9507 9508 9508
 9509 9509 9510 9511 9511 9513 9514 9517 9517 9518 9518 9518 9520 9521 9521 9522 9523 9523 9525 9525
 9526 9526 9527 9528 9530 9532 9532 9534 9535 9535 9535 9535 9536 9537 9537 9539 9539 9540 9541 9542
 9542 9543 9545 9545 9545 9547 9548 9548 9549 9549 9550 9552 9552 9552 9554 9555 9555 9558 9558 9560
 9562 9565 9566 9567 9567 9568 9568 9569 9570 9570 9577 9578 9578 9579 9579 9581 9581 9585 9585 9586
 9586 9587 9588 9588 9588 9588 9589 9590 9591 9592 9595 9595 9597 9597 9598 9598 9598 9598 9599 9600
 9600 9602 9604 9606 9607 9608 9609 9611 9614 9614 9615 9619 9620 9621 9622 9622 9623 9623 9626 9627
 9627 9627 9629 9630 9630 9631 9632 9633 9634 9635 9635 9635 9635 9636 9640 9641 9641 9643 9645 9646
 9646 9646 9648 9649 9649 9649 9650 9653 9656 9656 9657 9657 9657 9658 9660 9660 9660 9661 9662 9663
 9664 9664 9666 9667 9668 9669 9670 9671 9671 9672 9674 9675 9676 9677 9679 9680 9680 9683 9686 9686
 9688 9688 9691 9691 9692 9694 9696 9696 9699 9699 9700 9700 9700 9701 9704 9705 9706 9706 9706 9706
 9707 9707 9709 9710 9710 9710 9713 9714 9715 9716 9716 9717 9717 9718 9718 9719 9720 9721 9721 9723
 9723 9727 9728 9731 9732 9732 9734 9734 9734 9736 9738 9742 9743 9745 9745 9746 9747 9748 9748 9749
 9751 9754 9755 9756 9759 9761 9761 9762 9763 9763 9764 9765 9766 9766 9768 9769 9769 9772 9774 9776
 9777 9777 9779 9781 9781 9781 9782 9782 9783 9784 9786 9786 9787 9787 9789 9790 9790 9790 9792 9793
 9793 9794 9794 9797 9799 9799 9802 9803 9805 9807 9807 9809 9809 9809 9810 9812 9813 9814 9816 9816
 9818 9818 9819 9819 9820 9821 9821 9821 9821 9823 9825 9825 9826 9826 9826 9827 9828 9828 9828 9828
 9829 9829 9830 9832 9832 9832 9834 9836 9837 9838 9838 9838 9838 9839 9842 9842 9843 9844 9845 9846
 9847 9848 9848 9849 9849 9852 9853 9854 9854 9855 9856 9858 9860 9860 9861 9862 9863 9863 9864 9864
 9865 9865 9866 9866 9869 9870 9870 9873 9876 9876 9877 9878 9879 9880 9883 9883 9884 9885 9886 9889
 9889 9891 9892 9894 9894 9895 9895 9897 9898 9899 9899 9899 9899 9899 9899 9903 9903 9903 9904 9904
 9905 9908 9908 9909 9910 9912 9912 9914 9915 9915 9916 9918 9920 9921 9922 9923 9924 9924 9925 9927
 9927 9927 9927 9928 9928 9929 9931 9933 9933 9933 9933 9934 9935 9935 9936 9937 9937 9938 9938 9938
 9939 9940 9941 9943 9944 9946 9946 9947 9950 9951 9953 9955 9956 9958 9960 9960 9960 9961 9961 9962
 9963 9964 9964 9965 9965 9967 9967 9967 9969 9970 9970 9975 9977 9979 9981 9984 9986 9987 9988 9989
 9989 9989 9991 9992 9994 9996 9997 9997 9997 9998 9999 9999 /project $
/project $
```

Speedup Graph for 1 to 4 processors

$$S(p) = \frac{Sequential\ execution\ time}{Parallel\ execution\ time} = \frac{t_s}{t_p}$$

As t_p is the total runtime of the slowest process, we can calculate the Speedup as follows:

1. 10,000 random integer numbers

$$S(1) = \frac{0.319473}{0.319473} = 1$$

$$S(2) = \frac{0.319473}{0.171179} = 1.866$$

$$S(3) = \frac{0.319473}{0.141575} = 2.257$$

$$S(4) = \frac{0.319473}{0.142905} = 2.236$$

2. 100,000 random integer numbers

$$S(1) = \frac{18.340747}{18.340747} = 1$$

$$S(1) = \frac{18.340747}{9.211407} = 1.991$$

$$S(1) = \frac{18.340747}{6.102874} = 3.005$$

$$S(1) = \frac{18.340747}{4.728061} = 3.879$$

3. 1,000,000 random integer numbers

There is no data to show for 1,000,000 random integer numbers as the memory of the VMware is insufficient.

Note: 1 int array = 4 bytes therefore if we generate 1,000,000 array size, 4MB of memory is needed. Also, we broadcast the copy of the array to 4 processors so, 12MB of memory is needed in total.

```
/project $ mpicc -o out bucketsort.c
/project $ mpirun -np 4 ./out

===================================================================================
=   BAD TERMINATION OF ONE OF YOUR APPLICATION PROCESSES
=   PID 83 RUNNING AT 172.18.0.3
=   EXIT CODE: 139
=   CLEANING UP REMAINING PROCESSES
=   YOU CAN IGNORE THE BELOW CLEANUP MESSAGES
===================================================================================
```