Report Oblig 4 IN3030

Haiyuec

Processor:

Intel Xeon E3-1240 v2

Base frequency: 3.40 GHz

Max Turbo frequency: 3.80 GHz

4 cores 8 Threads

How to run the program:

```
javac *.java
```

```
java  Main  {N} {seed} {numBits} {test-flag}(optional)
```

test-flags:

- test-seq:
  Uses the Oblig4Precode to print the sequential result to a file.
- test-para:
  Uses the Oblig4Precode to print the parallel result to a file.

Implementation explanation:

- Step 1:
  - Find the max value in the array in parallel. Each thread gets a section of the array and find the local max value in their respective sections. Then they put their results in their own index in the result bucket. Call await() on the cyclic barrier.
  - The main thread wakes up and find the global max value from the result bucket.
  - Using the global max, the main thread finds the amount of bits per digit.
- Step 2:
  - The main thread computes the mask_len and bit shift and update it for all the threads.
  - The threads computes the number of values that has a certain value of the current digit and sets it in the int[][] all_count.

    - The main thread compute the local index/pointer table for all the threads. The threads would use this table to move elements from a to b. The threads only moves the elements that are in their assigned section of the array, so the table is different for all the threads. In a scenario of 1 bit per digit, this could be for example Thread-0 begin to move elements with value of 0 to index 0, and Thread-1 would move the elements that have the digit value of 0 to index 25. Because in Thread-0 section it has 25 elements with digit value 0, therefore Thread-1 must put its elements behind Thread-0's elements. This conserves the stability of the counting sort subroutine of radix sort.
- Step 3:
    - Loop step 2 for all the digits.

Runtime data:

Runtime measured in ms

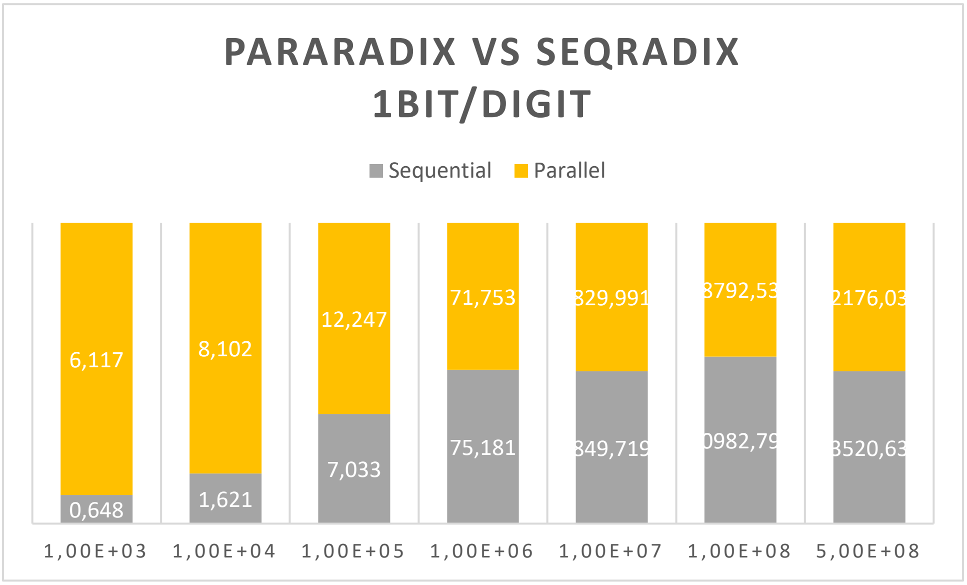
bits per digit	N	Sequential	Parallel
1	1.00E+03	0.648	6.117
	1.00E+04	1.621	8.102
	1.00E+05	7.033	12.247
	1.00E+06	75.181	71.753
	1.00E+07	849.719	829.991
	1.00E+08	10982.798	8792.53
	5.00E+08	53520.639	52176.038

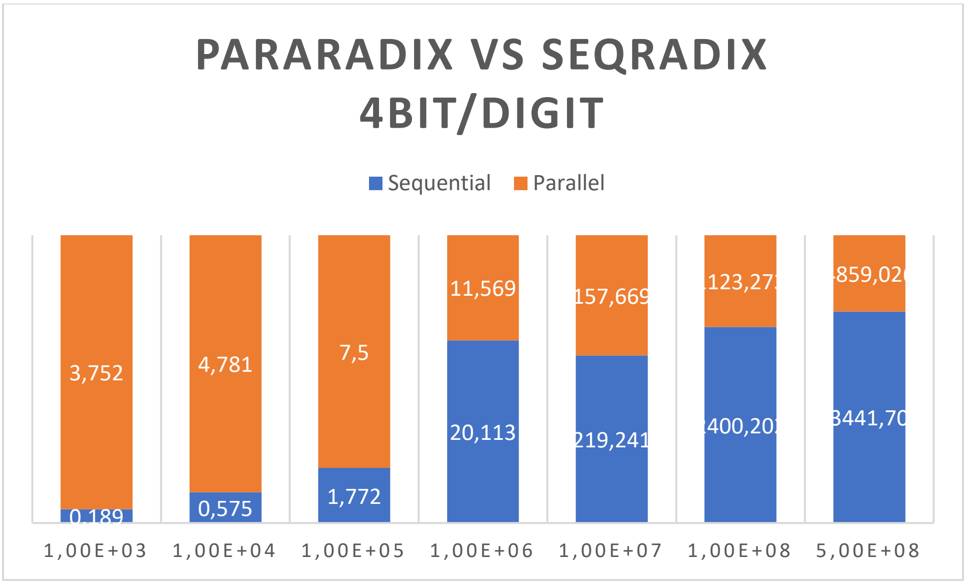
bits per digit	N	Sequential	Parallel
4	1.00E+03	0.189	3.752
	1.00E+04	0.575	4.781
	1.00E+05	1.772	7.5
	1.00E+06	20.113	11.569
	1.00E+07	219.241	157.669
	1.00E+08	2400.202	1123.273
	5.00E+08	13441.709	4859.026

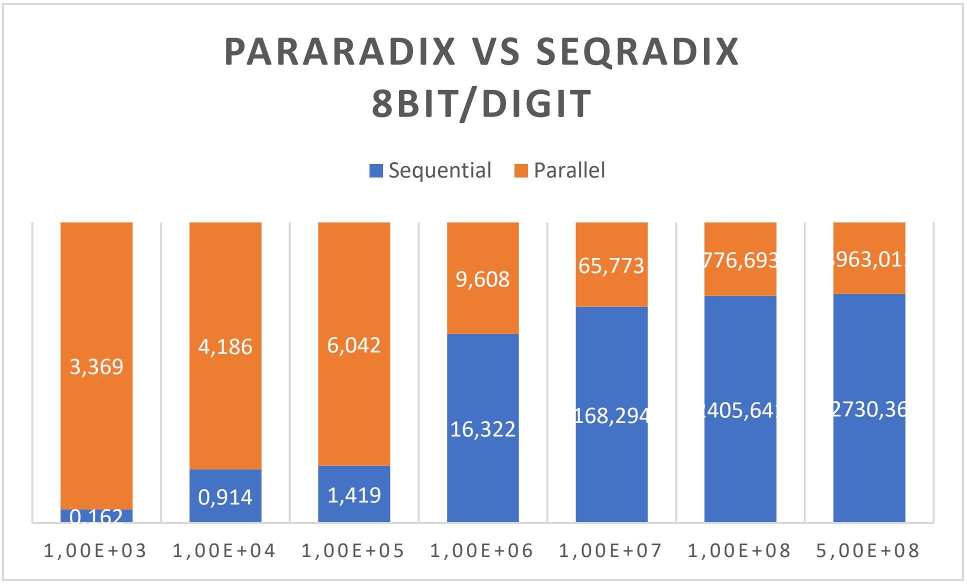
bits per digit	N	Sequential	Parallel

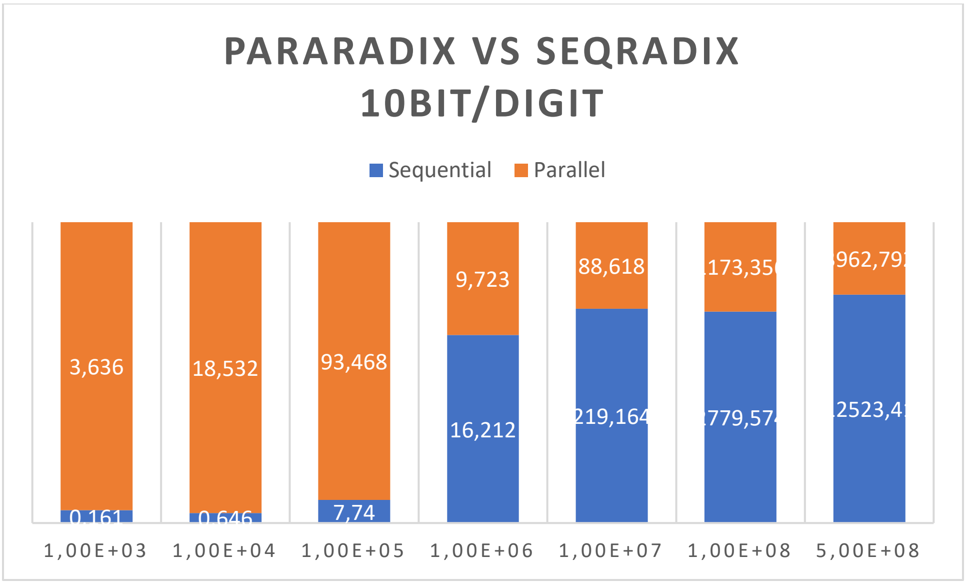
8	1.00E+03	0.162	3.369
	1.00E+04	0.914	4.186
	1.00E+05	1.419	6.042
	1.00E+06	16.322	9.608
	1.00E+07	168.294	65.773
	1.00E+08	2405.641	776.693
	5.00E+08	12730.367	3963.011

bits per digit	N	Sequential	Parallel
10	1.00E+03	0.161	3.636
	1.00E+04	0.646	18.532
	1.00E+05	7.74	93.468
	1.00E+06	16.212	9.723
	1.00E+07	219.164	88.618
	1.00E+08	2779.574	1173.356
	5.00E+08	12523.41	3962.792




PARARADIX VS SEQRADIX
4BIT/DIGIT
Sequential
Parallel
3,752
4,781
7,5
11,569
157,669
123,27
859,02
0,189
0,575
1,772
20,113
219,241
400,20
3441,70
1,00E+03
1,00E+04
1,00E+05
1,00E+06
1,00E+07
1,00E+08
5,00E+08


PARARADIX VS SEQRADIX
8BIT/DIGIT
Sequential
Parallel
3,369
4,186
6,042
9,608
65,773
776,693
963,01
0,162
0,914
1,419
16,322
168,294
405,64
2730,36
1,00E+03
1,00E+04
1,00E+05
1,00E+06
1,00E+07
1,00E+08
5,00E+08



Conclusion:

We can see that it is by no surprise that the parallel version is performing worse than the sequential version for N is low. The parallel version uses more resources on synchronization and startup and therefore its performance is worse. When N is high, the performance of the parallel version increases greatly compared to the sequential version. I think that this is because that (almost) every step in the algorithm is parallelized and the number of synchronization is constant per iteration. The parallel version is therefore capable to gain performance compared to the sequential version every iteration.

I found it a bit tricky in the start on how to construct a pointer table for the threads to move elements from a to b, such that it was more efficient compared to the sequential counter part and is stable. I was quite puzzled by Arne Maus' hint on this problem, as if his method was used, there would have been essentially no speed up at all.