

REPORT: Optimising the communication pattern of the Manager-worker version of the Mandelbrot calculation program

Duc Dat Hoang

High-performance Computing ECMM461

1 Introduction

This report details the modifications made to a Mandelbrot set calculation program that employs the Message Passing Interface (MPI) for parallelization. The implementation of the program utilizes a manager-worker pattern to distribute computational tasks among MPI processes, specifically focusing on the calculation of the Mandelbrot set in a two-dimensional array representing points in the complex plane. Additionally, the report presents a comparative analysis of the original and modified programs' performance, focusing on their parallel scaling characteristics. By measuring and plotting the parallel speed-up relative to the number of MPI processes for both versions, we assess the impact of the modifications on the program's efficiency. Through this analysis, we aim to quantify the performance improvements of the modified program.

P/s: For details on the file structure and instructions on how to run the program, please refer to the README.md file or visit the following link: <https://github.com/dathd6/Mandelbrot-MPI>.

2 Description of modifications to the program

```
27  /* Task 1.1: Initial variables
28  | *      - Create buffer space for sending and receiving data
29  | *      - Missing data value (negative value)
30  | * */
31  float buffer[N_IM + 3];
32  const int MISSING_DATA_VALUE = -1;
```

Figure 1: Allocate *buffer* space to facilitate data transmission and reception. In the complex plane, one column is designated for the imaginary axis (N_IM), resulting in a buffer size of $N_IM + 3$. Within this buffer, the entry at position $N_IM + 1$ signifies the current rank of the process, while the entry at $N_IM + 2$ serves as a placeholder for either the missing data indicator or the present value i which show the column of values that buffer are stored. Concurrently, the missing data value is set to -1.

```
40  /* Set to true to write out results*/
41  const bool doIO = false;    // [Coursework] Set to TRUE
42  const bool verbose = false; // [Coursework] Set to TRUE
```

Figure 2: When conducting the test for task 1, assign the `doIO` value as `TRUE`. For task 2, change the `doIO` value to `FALSE`.

```

85  /* Task 1.5: do_communication function is no longer required
86  * */
87  // void do_communication(int myRank) {
88  //
89  //     MPI_Group worldGroup, workerGroup;
90  //     MPI_Comm workerComm;
91  //     int zeroArray = {0};
92
93  //     // ...
94  // }
95
96  // ...
97
98  // ...
99
100  // ...
101
102  // ...
103
104  // ...
105
106  // ...
107
108  // ...
109
110  // ...
111
112  // ...
113
114  // ...
115
116  // ...
117
118  // ...
119
120  // ...
121
122  // ...
123
124  // ...
125
126  // ...
127
128  // ...
129
130  // ...
131
132  // ...
133
134  // ...
135
136  // ...
137
138  // ...
139
140  // ...
141
142  // ...
143
144  // ...
145
146  // ...
147
148  // ...
149
150  // ...
151
152  // ...
153
154  // ...
155
156  // ...
157
158  // ...
159
160  // ...
161
162  // ...
163
164  // ...
165
166  // ...
167
168  // ...
169
170  // ...
171
172  // ...
173
174  // ...
175
176  // ...
177
178  // ...
179
180  // ...
181
182  // ...
183
184  // ...
185
186  // ...
187
188  // ...
189
190  // ...
191
192  // ...
193
194  // ...
195
196  // ...
197
198  // ...
199
200  // ...
201
202  // ...
203
204  // ...
205
206  // ...
207
208  // ...
209
210  // ...
211
212  // ...
213
214  // ...
215
216  // ...
217
218  // ...
219
220  // ...
221
222  // ...
223
224  // ...
225
226  // ...
227
228  // ...
229
230  // ...
231
232  // ...
233
234  // ...
235
236  // ...
237
238  // ...
239
240  // ...
241
242  // ...
243
244  // ...
245
246  // ...
247
248  // ...
249
250  // ...
251
252  // ...
253
254  // ...
255
256  // ...
257
258  // ...
259
260  // ...
261
262  // ...
263
264  // ...
265
266  // ...
267
268  // ...
269
270  // ...
271
272  // ...
273
274  // ...
275
276  // ...
277
278  // ...
279
280  // ...
281
282  // ...
283
284  // ...
285
286  // ...
287
288  // ...
289
290  // ...
291
292  // ...
293
294  // ...
295
296  // ...
297
298  // ...
299
300  // ...
301
302  // ...
303
304  // ...
305
306  // ...
307
308  // ...
309
310  // ...
311
312  // ...
313
314  // ...
315
316  // ...
317
318  // ...
319
320  // ...
321
322  // ...
323
324  // ...
325
326  // ...
327
328  // ...
329
330  // ...
331
332  // ...
333
334  // ...
335
336  // ...
337
338  // ...
339
340  // ...
341
342  // ...
343
344  // ...
345
346  // ...
347
348  // ...
349
350  // ...
351
352  // ...
353
354  // ...
355
356  // ...
357
358  // ...
359
360  // ...
361
362  // ...
363
364  // ...
365
366  // ...
367
368  // ...
369
370  // ...
371
372  // ...
373
374  // ...
375
376  // ...
377
378  // ...
379
380  // ...
381
382  // ...
383
384  // ...
385
386  // ...
387
388  // ...
389
390  // ...
391
392  // ...
393
394  // ...
395
396  // ...
397
398  // ...
399
400  // ...
401
402  // ...
403
404  // ...
405
406  // ...
407
408  // ...
409
410  // ...
411
412  // ...
413
414  // ...
415
416  // ...
417
418  // ...
419
420  // ...
421
422  // ...
423
424  // ...
425
426  // ...
427
428  // ...
429
430  // ...
431
432  // ...
433
434  // ...
435
436  // ...
437
438  // ...
439
440  // ...
441
442  // ...
443
444  // ...
445
446  // ...
447
448  // ...
449
450  // ...
451
452  // ...
453
454  // ...
455
456  // ...
457
458  // ...
459
460  // ...
461
462  // ...
463
464  // ...
465
466  // ...
467
468  // ...
469
470  // ...
471
472  // ...
473
474  // ...
475
476  // ...
477
478  // ...
479
480  // ...
481
482  // ...
483
484  // ...
485
486  // ...
487
488  // ...
489
490  // ...
491
492  // ...
493
494  // ...
495
496  // ...
497
498  // ...
499
500  // ...
501
502  // ...
503
504  // ...
505
506  // ...
507
508  // ...
509
510  // ...
511
512  // ...
513
514  // ...
515
516  // ...
517
518  // ...
519
520  // ...
521
522  // ...
523
524  // ...
525
526  // ...
527
528  // ...
529
530  // ...
531
532  // ...
533
534  // ...
535
536  // ...
537
538  // ...
539
540  // ...
541
542  // ...
543
544  // ...
545
546  // ...
547
548  // ...
549
550  // ...
551
552  // ...
553
554  // ...
555
556  // ...
557
558  // ...
559
560  // ...
561
562  // ...
563
564  // ...
565
566  // ...
567
568  // ...
569
570  // ...
571
572  // ...
573
574  // ...
575
576  // ...
577
578  // ...
579
580  // ...
581
582  // ...
583
584  // ...
585
586  // ...
587
588  // ...
589
590  // ...
591
592  // ...
593
594  // ...
595
596  // ...
597
598  // ...
599
600  // ...
601
602  // ...
603
604  // ...
605
606  // ...
607
608  // ...
609
610  // ...
611
612  // ...
613
614  // ...
615
616  // ...
617
618  // ...
619
620  // ...
621
622  // ...
623
624  // ...
625
626  // ...
627
628  // ...
629
630  // ...
631
632  // ...
633
634  // ...
635
636  // ...
637
638  // ...
639
640  // ...
641
642  // ...
643
644  // ...
645
646  // ...
647
648  // ...
649
650  // ...
651
652  // ...
653
654  // ...
655
656  // ...
657
658  // ...
659
660  // ...
661
662  // ...
663
664  // ...
665
666  // ...
667
668  // ...
669
670  // ...
671
672  // ...
673
674  // ...
675
676  // ...
677
678  // ...
679
680  // ...
681
682  // ...
683
684  // ...
685
686  // ...
687
688  // ...
689
690  // ...
691
692  // ...
693
694  // ...
695
696  // ...
697
698  // ...
699
700  // ...
701
702  // ...
703
704  // ...
705
706  // ...
707
708  // ...
709
710  // ...
711
712  // ...
713
714  // ...
715
716  // ...
717
718  // ...
719
720  // ...
721
722  // ...
723
724  // ...
725
726  // ...
727
728  // ...
729
730  // ...
731
732  // ...
733
734  // ...
735
736  // ...
737
738  // ...
739
740  // ...
741
742  // ...
743
744  // ...
745
746  // ...
747
748  // ...
749
750  // ...
751
752  // ...
753
754  // ...
755
756  // ...
757
758  // ...
759
760  // ...
761
762  // ...
763
764  // ...
765
766  // ...
767
768  // ...
769
770  // ...
771
772  // ...
773
774  // ...
775
776  // ...
777
778  // ...
779
780  // ...
781
782  // ...
783
784  // ...
785
786  // ...
787
788  // ...
789
790  // ...
791
792  // ...
793
794  // ...
795
796  // ...
797
798  // ...
799
800  // ...
801
802  // ...
803
804  // ...
805
806  // ...
807
808  // ...
809
810  // ...
811
812  // ...
813
814  // ...
815
816  // ...
817
818  // ...
819
820  // ...
821
82
```

Figure 3: Comment out the call to ‘do_communication’ as this is no longer required

```

230 // Worker Processes
231 else {
232     /* Task 1.2 */
233     buffer[N_IM + 1] = myRank; // Store current process to buffer
234     buffer[N_IM + 2] = MISSING_DATA_VALUE; // Task 1.3: initial missing data value
235     while (true) {
236         // Send request for work
237         MPI_Send(&buffer, N_IM + 3, MPI_INT, 0, 100 + myRank, MPI_COMM_WORLD);
238         // Receive i value to work on
239         MPI_Recv(&i, 1, MPI_INT, 0, 100, MPI_COMM_WORLD, &status);
240         if (i == endFlag) {
241             break;
242         } else {
243             calc_vals(i);
244         }
245     } // while(true)
246 } // else worker process

```

Figure 4: **Worker Processes** - At the beginning of the operation, each parallel process will initialize a buffer. This buffer will contain the process's rank and a placeholder for missing data, signifying that the buffer's values have yet to be computed. Each worker process will then transmit a buffer array of size $N_{IM} + 3$ to the manager process to store the values in `nIter`.

```
50 void calc_vals(int i) {
    62 buffer[N_IM + 2] = i; // Set value column value to buffer
    63 /* Loop over imaginary axis */
    64 for (j = 0; j < N_IM + 1; j++) {
    65     z0 = z_Re[i] + z_Im[j] * I;
    66     z = z0;
    67     /* Iterate up to a maximum number or bail out if mod(z) > 2 */
    68     k = 0;
    69     while (k < maxIter) {
    70         buffer[j] = k; // Store all value of current column i in buffer
    71         if (cabs(z) > 2.0)
    72             break;
    73         z = z * z + z0;
    74         k++;
    75     }
    76 }
```

Figure 5: **Worker Processes** - Each worker, while computing the column of values for the current i , assigns the value of i to the position $N_IM + 2$ in the buffer to signify the calculation has been implemented. The previous code updates the `nIter` array at line 70, while the modified version updates the buffer to transfer to the manager.

```

204 // Hand out work to worker processes
205 for (i = 0; i < N_RE + 1; i++) {
206     // Receive request for work
207     /* Task 1.4: Receiving the buffer array to store in the correct column of the nIter */
208     MPI_Recv(&buffer, N_IM + 3, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status); // Receive the message
209     nextProc = buffer[N_IM + 1]; // Next process
210     int c_values = buffer[N_IM + 2]; // Unpack the column of values
211     if (c_values != MISSING_DATA_VALUE) { // Task 1.3: If there are no data -> manager process discard the data
212         for (j = 0; j < N_IM + 1; j++) {
213             nIter[c_values][j] = buffer[j]; // stores it in the correct column of the nIter array
214         }
215     }
216     // Send i value to requesting process
217     MPI_Send(&i, 1, MPI_INT, nextProc, 100, MPI_COMM_WORLD);
218 }

```

Figure 6: **Manager Process** - The manager receives a buffer array of size $N_IM + 3$. It checks the value at index $N_IM + 2$ within the buffer to determine if the data for the current buffer has been computed. If the data is calculated, the manager updates the $nIter$ array for the current i to match the received buffer.

```

218 // Tell all the worker processes to finish (once for each worker process = nProcs-1)
219 for (i = 0; i < nProcs - 1; i++) {
220     // Receive request for work
221     /* Task 1.4: Get the process rank from buffer and tell the worker to finish its process
222      * Receiving the last buffer from the array to store in the correct column of the nIter
223      */
224     MPI_Recv(&buffer, N_IM + 3, MPI_INT, MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, &status); // Receive the message
225     nextProc = buffer[N_IM + 1]; // Next process
226     int c_values = buffer[N_IM + 2]; // Unpack the column of values
227     if (c_values != MISSING_DATA_VALUE) { // Task 1.3: If there are no data -> manager process discard the data
228         for (j = 0; j < N_IM + 1; j++) {
229             nIter[c_values][j] = buffer[j]; // stores it in the correct column of the nIter array
230         }
231     }
232     // Send endFlag to finish
233     MPI_Send(&endFlag, 1, MPI_INT, nextProc, 100, MPI_COMM_WORLD);
234 }

```

Figure 7: **Manager Process** - The manager collects the final buffer array from each worker process, which was sent in the last iteration but has not yet been received. This buffer is then stored in the corresponding position i of the $nIter$ array. The manager instructs all worker processes to terminate by extracting the rank from the buffer and sending an `endFlag` to each worker process, thereby stopping their respective while loops.

```

252 /* Write out results */
253 if (doIO && myRank == 0) { /* Task 1.6: Write out the results from the manager process */
254     if (verbose) {
255         printf("Writing out results from process %d \n", myRank);
256     }
257     write_to_file("mandelbrot.dat");
258 }

```

Figure 8: Write out the results from the manager process instead of the rank 1 process.

3 Scaling test

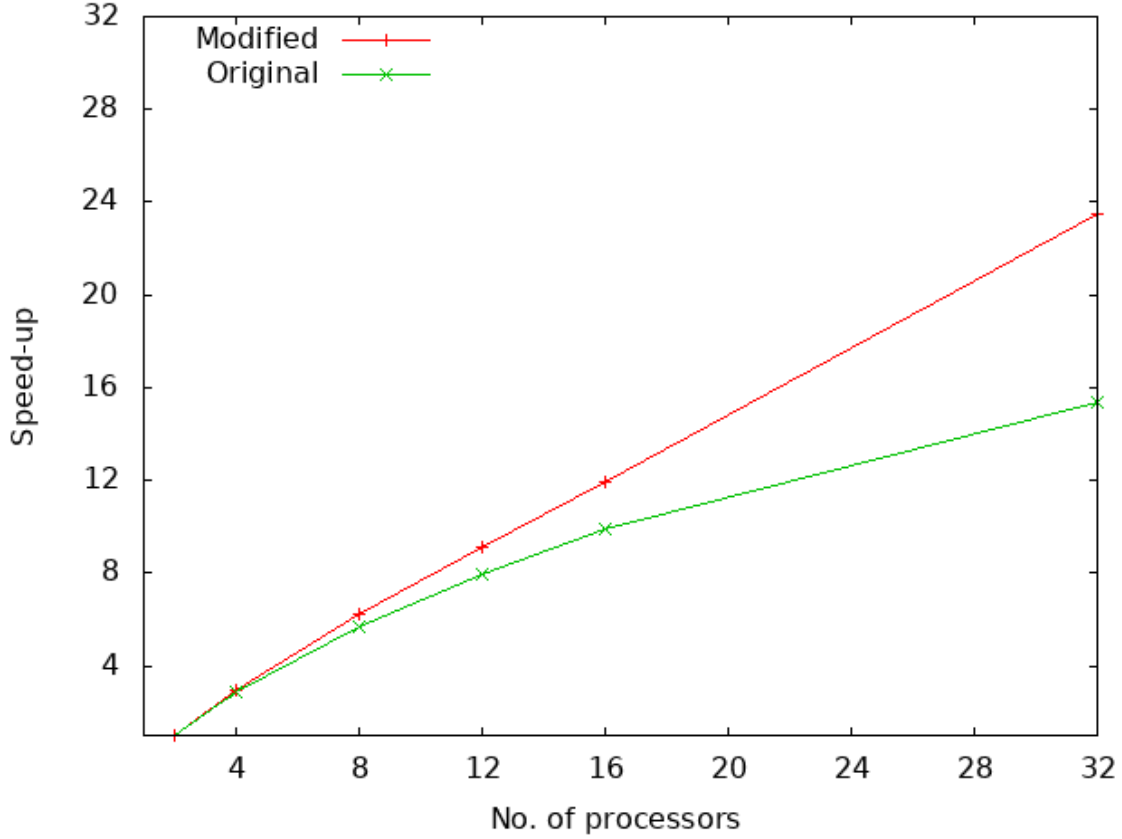


Figure 9: Scaling test plot show parallel-speed up against number of MPI processes for both the original program (green line) and modified version (red line).

4 Discussion

According to figure 9, The modified version clearly perform better than original program at every point measured. Additionally, the slope of the modified version's line is steeper than that of the original version, especially noticeable from around 8 processors onward. This suggests that the modified version is benefiting more efficiently from the added processors. This could be explained by the previous approach leads to the transmission of excess data, as each worker sends the entire computational domain, including regions it did not compute. To address this inefficiency, we modified the communication pattern within the program. Rather than waiting until the end of all computations to aggregate results, worker processes now send their computed column of values to the manager immediately after each is completed. This change aims to reduce the data transmitted between processes, potentially enhancing the program's overall parallel efficiency.