



МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В.Ломоносова



Факультет вычислительной математики и кибернетики

Практикум по учебному курсу

Распределенные системы

ОТЧЕТ

о выполненном задании (Улучшение adi_3d.c)

студента 428 учебной группы факультета ВМК МГУ

Романова Александра Юрьевича

гор. Москва

2022 год

Содержание

1	Постановка задачи	2
2	Алгоритм	3
3	Реализация алгоритма	4
4	Инструкции по запуску	12
5	Временная оценка	20

1 Постановка задачи

Доработать MPI-программу, реализованную в рамках курса “Суперкомпьютеры и параллельная обработка данных”. Добавить контрольные точки для продолжения работы программы в случае сбоя. Реализовать один из 3-х сценариев работы после сбоя: а) продолжить работу программы только на “исправных” процессах; б) вместо процессов, вышедших из строя, создать новые MPI-процессы, которые необходимо использовать для продолжения расчетов; в) при запуске программы на счет сразу запустить некоторое дополнительное количество MPI-процессов, которые использовать в случае сбоя.

2 Алгоритм

Напомню условие прошлогодней задачи: Ставится задача по улучшению программы Adi-3D.

Требуется:

1. Оптимизировать в последовательном варианте функцию, производящую необходимые вычисления.
2. Распараллелить программу с помощью технологии OpenMP и MPI.
3. Исследовать масштабируемость полученной программы, построить графики зависимости времени выполнения программ от числа используемых ядер (процессов) и объёма входных данных.

Необходимые детали можно найти тут:

https://github.com/XelaraleX/CMC_SKPOD21/blob/main/adi_3d_MPI.c

Улучшение:

В данном алгоритме можно расставить контрольные точки на каждой итерации в цикле

```
for(int it = 0; it < itmax; it++)
```

Если хотя бы один из процессов (кроме мастера) во время выполнения очередной итерации упал, то будем перераспределять работу между оставшимися процессами. Если упадет мастер, то нужно будет перезапустить программу. Однако прогресс сохранится в файл A.txt. Чтобы при очередном запуске читать данные из файла, нужно присвоить константе READ_MATRIX_FROM_FILE значение, отличное от 0.

Чтобы программа не завершалась аварийно при падении процессов, зарегистрируем обработчик таким образом, чтобы при падении процессов функции коммуникаций возвращали ошибки

```
MPI_Comm_set_errhandler(comm, MPI_ERRORS_RETURN).
```

Будем смотреть в программе на возвращаемое значение таких функций и в случае неуспеха прерывать выполнение функции *wrap()* и возвращать -1 из неё. После этого будем создавать новый коммуникатор из выживших процессов и перезапускать итерацию.

В начале выполнения каждой итерации мастер будет читать матрицу из файла A.txt. В конце каждой итерации мастер будет сохранять полученную в результате выполнения матрицу в тот же файл.

3 Реализация алгоритма

```
1  #include <math.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <signal.h>
5  #include <mpi.h>
6
7  #define READ_MATRIX_FROM_FILE (0)
8
9  #define Max(a,b) ((a)>(b)?(a):(b))
10
11 void
12 init(int n, float *A)
13 {
14     for (int i = 0; i < n; i++) {
15         for (int j = 0; j < n; j++) {
16             for (int k = 0; k < n; k++) {
17                 if (i == 0 || i == n - 1 || j == 0 || j == n - 1 || k == 0 ||
18                     k == n - 1) {
19                     A[i * n * n + n * j + k] = 0.;
20                 } else {
21                     A[i * n * n + n * j + k] = (4. + i + j + k);
22                 }
23             }
24         }
25     }
26 }
27
28 float
29 relax_mpi(int n, float *A, int myrank, float *myA1, float *myA2,
30 int *cnt1, int *disp1, int *cnt2, int *disp2, MPI_Datatype COLRES, MPI_Comm comm)
31 {
32     static int count = 0;
33     ++count;
34
35     int myn = cnt1[myrank];
36     int arr_size = myn * n;
37     float eps = 0.;
38     float eps2 = 0.;
```

```

39
40     if (myrank == 0) {
41         FILE *input = fopen("A.txt", "r");
42         for (int i = 0; i < n * n * n; ++i) {
43             fscanf(input, "%f", &A[i]);
44         }
45         fclose(input);
46     }
47
48     if (count == 1 && myrank == 1) {
49         printf("%d: failed on %d launch\n", myrank, count); fflush(stdout);
50         raise(SIGKILL);
51     }
52
53     int err = MPI_Scatterv(A, cnt1, disp1, COLRES, myA1, arr_size, MPI_FLOAT, 0,
54                           comm);
55     if (err != MPI_SUCCESS) {
56         printf("%d: get fail on %d launch\n", myrank, count); fflush(stdout);
57         return -1;
58     }
59
60     int cnt = 0;
61     for (int j = 0; j < myn; j++) {
62         for (int i = 0; i < n; i++) {
63             if (i != 0 && i != n - 1) {
64                 myA1[cnt] = (myA1[cnt - 1] + myA1[cnt + 1]) / 2.;
65             }
66             cnt++;
67         }
68     }
69
70
71     if (count == 2 && (myrank == 1 || myrank == 2)) {
72         printf("%d: failed on %d launch\n", myrank, count); fflush(stdout);
73         raise(SIGKILL);
74     }
75
76     err = MPI_Gatherv(myA1, arr_size, MPI_FLOAT, A, cnt1, disp1,
77                      COLRES, 0, comm);
78     if (err != MPI_SUCCESS) {

```

```

79         printf("%d: get fail on %d launch\n", myrank, count); fflush(stdout);
80         return -1;
81     }
82
83     arr_size = cnt2[myrank];
84     myn = arr_size / (n * n);
85
86
87     if (count == 3 && myrank == 3) {
88         printf("%d: failed on %d launch\n", myrank, count); fflush(stdout);
89         raise(SIGKILL);
90     }
91
92     err = MPI_Scatterv(A, cnt2, disp2, MPI_FLOAT,
93                      myA2, arr_size, MPI_FLOAT, 0, comm);
94     if (err != MPI_SUCCESS) {
95         printf("%d: get fail on %d launch\n", myrank, count); fflush(stdout);
96         return -1;
97     }
98
99     cnt = 0;
100    for (int i = 0; i < myn; i++) {
101        for (int j = 0; j < n; j++) {
102            for (int k = 0; k < n; k++) {
103                if (j != 0 && j != n - 1) {
104                    myA2[cnt] = (myA2[cnt - n] + myA2[cnt + n]) / 2.;
105                }
106                cnt++;
107            }
108        }
109    }
110    cnt = 0;
111    for (int i = 0; i < myn; i++) {
112        for (int j = 0; j < n; j++) {
113            for (int k = 0; k < n; k++) {
114                if (k != 0 && k != n - 1) {
115                    float e = myA2[cnt];
116                    myA2[cnt] = (myA2[cnt - 1] + myA2[cnt + 1]) / 2.;
117                    eps = Max(eps, fabs(e - myA2[cnt]));
118                }

```

```

119         cnt++;
120     }
121 }
122 }
123
124 if (count == 4 && myrank == 2) {
125     printf("%d: failed on %d launch\n", myrank, count); fflush(stdout);
126     raise(SIGKILL);
127 }
128
129 err = MPI_Gatherv(myA2, arr_size, MPI_FLOAT,
130                 A, cnt2, disp2, MPI_FLOAT, 0, comm);
131 if (err != MPI_SUCCESS) {
132     printf("%d: get fail on %d launch\n", myrank, count); fflush(stdout);
133     return -1;
134 }
135
136 err = MPI_Reduce(&eps, &eps2, 1, MPI_FLOAT, MPI_MAX, 0, comm);
137 if (err != MPI_SUCCESS) {
138     printf("%d: get fail on %d launch\n", myrank, count); fflush(stdout);
139     return -1;
140 }
141
142 err = MPI_Bcast(&eps2, 1, MPI_FLOAT, 0, comm);
143 if (err != MPI_SUCCESS) {
144     printf("%d: get fail on %d launch\n", myrank, count); fflush(stdout);
145     return -1;
146 }
147
148
149 if (myrank == 0) {
150     FILE *output = fopen("A.txt", "w");
151     for (int i = 0; i < arr_size; ++i) {
152         fprintf(output, "%f", A[i]);
153     }
154     fclose(output);
155 }
156
157 return eps2;
158 }

```



```

159
160 void
161 wrap(
162     int n, float * A, int itmax, float mineps, MPI_Comm comm)
163 {
164     int size;
165     MPI_Comm_size(comm, &size);
166
167     int myrank;
168     MPI_Comm_rank(comm, &myrank);
169
170
171     int *cnt1, * disp1;
172     cnt1 = calloc(size, sizeof(*cnt1));
173     disp1 = calloc(size, sizeof(*disp1));
174     int * cnt2, * disp2;
175     cnt2 = calloc(size, sizeof(*cnt2));
176     disp2 = calloc(size, sizeof(*disp2));
177     MPI_Datatype COL, COLRES;
178     MPI_Type_vector(n, 1, n * n, MPI_FLOAT, &COL);
179     MPI_Type_commit(&COL);
180     MPI_Type_create_resized(COL, 0, sizeof(float), &COLRES);
181     MPI_Type_commit(&COLRES);
182
183     cnt1[0] = (n * n / size + (0 < (n * n) % size));
184     disp1[0] = 0;
185     for (int i = 1; i < size; i++) {
186         cnt1[i] = cnt1[i - 1];
187         if (i == (n * n) % size) {
188             cnt1[i]--;
189         }
190         disp1[i] = disp1[i - 1] + cnt1[i - 1];
191     }
192     cnt2[0] = (n / size + (0 < n % size)) * n * n;
193     disp2[0] = 0;
194     for (int i = 1; i < size; i++) {
195         cnt2[i] = cnt2[i - 1];
196         if (i == n % size) {
197             cnt2[i] -= n * n;
198         }

```

```

199         disp2[i] = disp2[i - 1] + cnt2[i - 1];
200     }
201     int arr_size = cnt1[myrank] * n;
202     float *myA1 = calloc(arr_size, sizeof(*myA1));
203     arr_size = cnt2[myrank];
204     float *myA2 = calloc(arr_size, sizeof(*myA2));
205     for(int it = 0; it < itmax; it++)
206     {
207         float eps = relax_mpi(n, A, myrank, myA1, myA2,
208                               cnt1, disp1, cnt2, disp2, COLRES, comm);
209         if (eps < 0) {
210             // repeat this iteration with new comm
211             free(myA1);
212             free(myA2);
213             MPI_Type_free(&COL);
214             MPI_Type_free(&COLRES);
215             free(cnt1);
216             free(disp1);
217             free(cnt2);
218             free(disp2);
219
220             MPIX_Comm_shrink(comm, &comm);
221
222             wrap(n, A, itmax, mineps, comm);
223             return;
224         }
225         if (eps < mineps) {
226             break;
227         }
228     }
229     free(myA1);
230     free(myA2);
231     MPI_Type_free(&COL);
232     MPI_Type_free(&COLRES);
233     free(cnt1);
234     free(disp1);
235     free(cnt2);
236     free(disp2);
237 }
238

```

```

239 void
240 verify(int n, float *A)
241 {
242     float s = 0.;
243     for (int i = 1; i < n - 1; i++) {
244         for (int j = 1; j < n - 1; j++) {
245             for (int k = 1; k < n - 1; k++) {
246                 s += A[i * n * n + n * j + k] * (i + 1) * (j + 1) *
247                     (k + 1) / (n * n * n);
248             }
249         }
250     }
251     printf("S = %f\n", s);
252 }
253
254 int main (int argc, char *argv[])
255 {
256     MPI_Init(&argc, &argv);
257
258     const float mineps = 0.1e-7;
259     const int itmax = 100;
260     int numproc = 64;
261     int myrank;
262     MPI_Comm comm = MPI_COMM_WORLD;
263     MPI_Comm_set_errhandler(comm, MPI_ERRORS_RETURN);
264     MPI_Comm_rank(comm, &myrank);
265     FILE * f = NULL;
266     if (myrank == 0) {
267         f = fopen("output_proc64", "w");
268     }
269     int n = 16;
270     double start, end;
271     float *A;
272     if (myrank == 0) {
273         A = calloc(n * n * n, sizeof(*A));
274         if (!READ_MATRIX_FROM_FILE) {
275             init(n, A);
276             FILE *output = fopen("A.txt", "w");
277             for (int i = 0; i < n * n * n; ++i) {
278                 fprintf(output, "%f", A[i]);

```

```

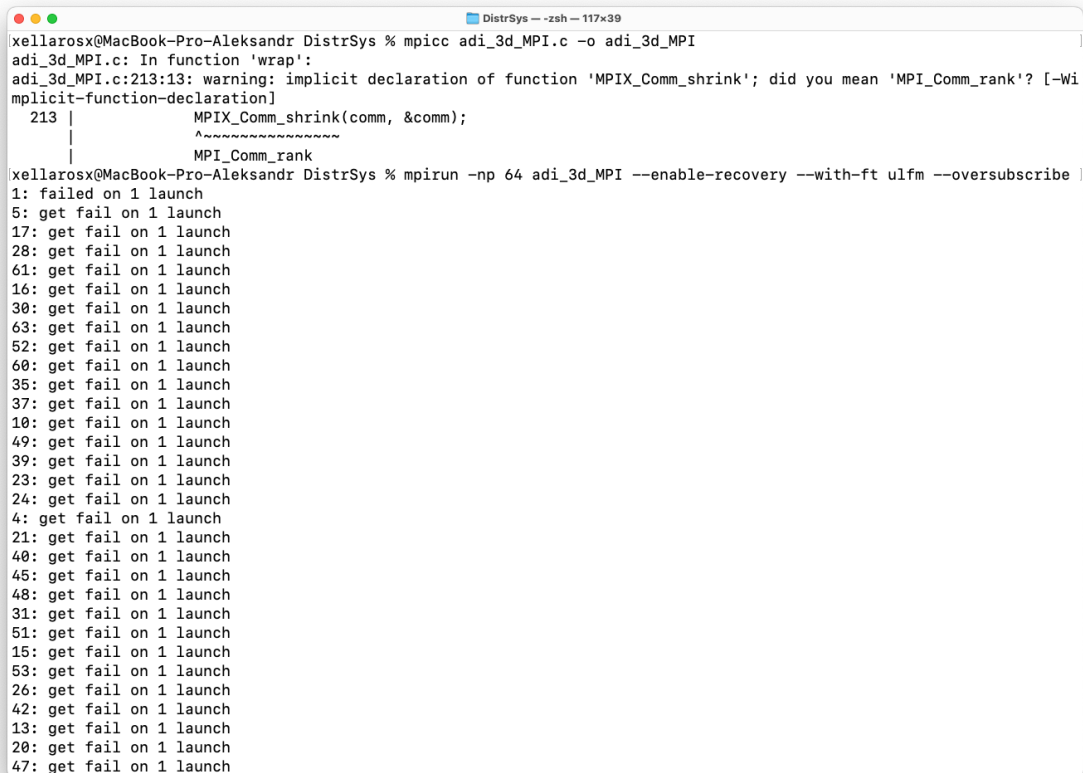
279         }
280         fclose(output);
281     }
282 }
283 MPI_Barrier(comm);
284 if (myrank == 0) {
285     start = MPI_Wtime();
286 }
287
288 wrap(n, A, itmax, mineps, comm);
289
290 MPI_Barrier(comm);
291 if (myrank == 0) {
292     end = MPI_Wtime();
293     fprintf(f, "%d\t%d\t%lf\n", n, numproc, end - start);
294     // verify(n, A); // we comment so as not to waste
295     // time on checking as it has already been checked
296     free(A);
297 }
298 if (myrank == 0) {
299     fclose(f);
300 }
301 MPI_Finalize();
302 return 0;
303 }

```

4 Инструкции по запуску

Для запуска необходимо:

- `mpicc adi_3d_MPI.c -o adi_3d_MPI`
- `mpirun -np 64 adi_3d_MPI --enable-recovery --with-ft ulfm --oversubscribe`



```
xcellarosx@MacBook-Pro-Aleksandr DistrSys % mpicc adi_3d_MPI.c -o adi_3d_MPI
adi_3d_MPI.c: In function 'wrap':
adi_3d_MPI.c:213:13: warning: implicit declaration of function 'MPIX_Comm_shrink'; did you mean 'MPI_Comm_rank'? [-Wimplicit-function-declaration]
  213 |             MPIX_Comm_shrink(comm, &comm);
      |             ~~~~~^~~~~~
      |             MPI_Comm_rank
xcellarosx@MacBook-Pro-Aleksandr DistrSys % mpirun -np 64 adi_3d_MPI --enable-recovery --with-ft ulfm --oversubscribe
1: failed on 1 launch
5: get fail on 1 launch
17: get fail on 1 launch
28: get fail on 1 launch
61: get fail on 1 launch
16: get fail on 1 launch
30: get fail on 1 launch
63: get fail on 1 launch
52: get fail on 1 launch
60: get fail on 1 launch
35: get fail on 1 launch
37: get fail on 1 launch
10: get fail on 1 launch
49: get fail on 1 launch
39: get fail on 1 launch
23: get fail on 1 launch
24: get fail on 1 launch
4: get fail on 1 launch
21: get fail on 1 launch
40: get fail on 1 launch
45: get fail on 1 launch
48: get fail on 1 launch
31: get fail on 1 launch
51: get fail on 1 launch
15: get fail on 1 launch
53: get fail on 1 launch
26: get fail on 1 launch
42: get fail on 1 launch
13: get fail on 1 launch
20: get fail on 1 launch
47: get fail on 1 launch
```

Рис. 1: Пример работы программы

Полный вывод:

```
1: failed on 1 launch
57: get fail on 1 launch
61: get fail on 1 launch
33: get fail on 1 launch
```

63: get fail on 1 launch
49: get fail on 1 launch
17: get fail on 1 launch
9: get fail on 1 launch
5: get fail on 1 launch
2: get fail on 1 launch
10: get fail on 1 launch
50: get fail on 1 launch
8: get fail on 1 launch
0: get fail on 1 launch
6: get fail on 1 launch
38: get fail on 1 launch
7: get fail on 1 launch
31: get fail on 1 launch
40: get fail on 1 launch
36: get fail on 1 launch
43: get fail on 1 launch
16: get fail on 1 launch
52: get fail on 1 launch
41: get fail on 1 launch
55: get fail on 1 launch
20: get fail on 1 launch
47: get fail on 1 launch
23: get fail on 1 launch
15: get fail on 1 launch
18: get fail on 1 launch
54: get fail on 1 launch
39: get fail on 1 launch
13: get fail on 1 launch
62: get fail on 1 launch
35: get fail on 1 launch
28: get fail on 1 launch
4: get fail on 1 launch
46: get fail on 1 launch
37: get fail on 1 launch
29: get fail on 1 launch
30: get fail on 1 launch
48: get fail on 1 launch
25: get fail on 1 launch
27: get fail on 1 launch

42: get fail on 1 launch
53: get fail on 1 launch
45: get fail on 1 launch
59: get fail on 1 launch
56: get fail on 1 launch
24: get fail on 1 launch
26: get fail on 1 launch
12: get fail on 1 launch
19: get fail on 1 launch
22: get fail on 1 launch
21: get fail on 1 launch
32: get fail on 1 launch
60: get fail on 1 launch
44: get fail on 1 launch
58: get fail on 1 launch
11: get fail on 1 launch
14: get fail on 1 launch
51: get fail on 1 launch
3: get fail on 1 launch
34: get fail on 1 launch
2: failed on 2 launch
1: failed on 2 launch
49: get fail on 2 launch
61: get fail on 2 launch
3: get fail on 2 launch
9: get fail on 2 launch
33: get fail on 2 launch
17: get fail on 2 launch
4: get fail on 2 launch
5: get fail on 2 launch
18: get fail on 2 launch
57: get fail on 2 launch
62: get fail on 2 launch
50: get fail on 2 launch
0: get fail on 2 launch
58: get fail on 2 launch
10: get fail on 2 launch
34: get fail on 2 launch
7: get fail on 2 launch
6: get fail on 2 launch

8: get fail on 2 launch
14: get fail on 2 launch
15: get fail on 2 launch
37: get fail on 2 launch
11: get fail on 2 launch
35: get fail on 2 launch
38: get fail on 2 launch
31: get fail on 2 launch
46: get fail on 2 launch
42: get fail on 2 launch
29: get fail on 2 launch
56: get fail on 2 launch
43: get fail on 2 launch
21: get fail on 2 launch
28: get fail on 2 launch
22: get fail on 2 launch
39: get fail on 2 launch
30: get fail on 2 launch
20: get fail on 2 launch
52: get fail on 2 launch
36: get fail on 2 launch
55: get fail on 2 launch
51: get fail on 2 launch
53: get fail on 2 launch
19: get fail on 2 launch
41: get fail on 2 launch
45: get fail on 2 launch
32: get fail on 2 launch
12: get fail on 2 launch
24: get fail on 2 launch
54: get fail on 2 launch
60: get fail on 2 launch
59: get fail on 2 launch
25: get fail on 2 launch
16: get fail on 2 launch
44: get fail on 2 launch
23: get fail on 2 launch
26: get fail on 2 launch
47: get fail on 2 launch
40: get fail on 2 launch

48: get fail on 2 launch
27: get fail on 2 launch
13: get fail on 2 launch
3: failed on 3 launch
7: get fail on 3 launch
51: get fail on 3 launch
1: get fail on 3 launch
10: get fail on 3 launch
9: get fail on 3 launch
4: get fail on 3 launch
35: get fail on 3 launch
59: get fail on 3 launch
5: get fail on 3 launch
19: get fail on 3 launch
2: get fail on 3 launch
8: get fail on 3 launch
12: get fail on 3 launch
13: get fail on 3 launch
28: get fail on 3 launch
33: get fail on 3 launch
15: get fail on 3 launch
47: get fail on 3 launch
29: get fail on 3 launch
41: get fail on 3 launch
54: get fail on 3 launch
36: get fail on 3 launch
38: get fail on 3 launch
44: get fail on 3 launch
37: get fail on 3 launch
26: get fail on 3 launch
40: get fail on 3 launch
18: get fail on 3 launch
6: get fail on 3 launch
27: get fail on 3 launch
17: get fail on 3 launch
49: get fail on 3 launch
34: get fail on 3 launch
50: get fail on 3 launch
31: get fail on 3 launch
52: get fail on 3 launch

39: get fail on 3 launch
60: get fail on 3 launch
20: get fail on 3 launch
46: get fail on 3 launch
57: get fail on 3 launch
56: get fail on 3 launch
58: get fail on 3 launch
32: get fail on 3 launch
14: get fail on 3 launch
53: get fail on 3 launch
25: get fail on 3 launch
48: get fail on 3 launch
43: get fail on 3 launch
42: get fail on 3 launch
23: get fail on 3 launch
30: get fail on 3 launch
24: get fail on 3 launch
45: get fail on 3 launch
22: get fail on 3 launch
21: get fail on 3 launch
16: get fail on 3 launch
0: get fail on 3 launch
11: get fail on 3 launch
55: get fail on 3 launch
2: failed on 4 launch
17: get fail on 4 launch
9: get fail on 4 launch
6: get fail on 4 launch
3: get fail on 4 launch
8: get fail on 4 launch
7: get fail on 4 launch
33: get fail on 4 launch
1: get fail on 4 launch
0: get fail on 4 launch
49: get fail on 4 launch
57: get fail on 4 launch
4: get fail on 4 launch
5: get fail on 4 launch
46: get fail on 4 launch
14: get fail on 4 launch

53: get fail on 4 launch
50: get fail on 4 launch
27: get fail on 4 launch
34: get fail on 4 launch
36: get fail on 4 launch
18: get fail on 4 launch
32: get fail on 4 launch
51: get fail on 4 launch
31: get fail on 4 launch
19: get fail on 4 launch
58: get fail on 4 launch
12: get fail on 4 launch
35: get fail on 4 launch
39: get fail on 4 launch
48: get fail on 4 launch
41: get fail on 4 launch
42: get fail on 4 launch
45: get fail on 4 launch
20: get fail on 4 launch
37: get fail on 4 launch
23: get fail on 4 launch
13: get fail on 4 launch
30: get fail on 4 launch
59: get fail on 4 launch
55: get fail on 4 launch
26: get fail on 4 launch
52: get fail on 4 launch
29: get fail on 4 launch
38: get fail on 4 launch
47: get fail on 4 launch
11: get fail on 4 launch
21: get fail on 4 launch
16: get fail on 4 launch
24: get fail on 4 launch
44: get fail on 4 launch
56: get fail on 4 launch
15: get fail on 4 launch
28: get fail on 4 launch
22: get fail on 4 launch
43: get fail on 4 launch

40: get fail on 4 launch
25: get fail on 4 launch
10: get fail on 4 launch
54: get fail on 4 launch

5 Временная оценка

Установка контрольных точек отразится на производительности.

1) Операции чтения из файла и запись в файл мастером будут занимать некоторое время, зависящее от размеров исходной матрицы A . В это время все остальные процессы будут простаивать в ожидании своего кусочка от мастера. Таким образом, в лучшем случае программа замедлится на $it_{max} * T_{read} + it_{max} * T_{wright}$ (при условии что мы не достигли нужной точности ϵ_{rs} до последней итерации), где

it_{max} — максимальное число итераций;

T_{read} — время чтения матрицы A из файла;

T_{wright} — время записи матрицы A в файл;

2) Если на какой-либо итерации произойдет падение одного или нескольких процессов (кроме мастера), то итерация будет перезапущена. Это значит, что чем больше итераций, на которых падают процессы, тем дольше будет выполняться программа. Таким образом, в худшем случае программа будет выполняться $(numproc - 1) * T_{it}$ (время выполнения одной итерации) дольше чем последовательный алгоритм (когда все $(numproc - 1)$ вспомогательных процессов упадут)

3) Если на какой-либо итерации произойдет падение мастера, то программу придется перезапустить. Однако, поскольку прогресс будет сохранен в файл, выполнение новой программы можно будет начать с последней успешно завершившейся итерации.