Отчёт о выполнении практического задания по курсу «Распределённые системы»

Михальцов Данила, 428 группа ВМК МГУ

Задание 1: Круговой алгоритм выбора координатора

Постановка задания

> Реализовать программу для выбора координатора среди 36 процессов, находящихся в узлах транспьютерной матрицы размером 6*6, использующую круговой алгоритм.
> Все необходимые межпроцессорные взаимодействия реализовать при помощи пересылок MPI типа точка-точка.
> Получить временную оценку работы алгоритма. Оценить сколько времени потребуют выборы координатора, если время старта равно 100, время передачи байта равно 1 (Ts=100,Tb=1). Процессорные операции, включая чтение из памяти и запись в память считаются бесконечно быстрыми.

Описание и реализация алгоритма

У нас есть транспьютерная матрица размером 6 на 6. В качестве порядка её обхода был выбран следующий: $0 \to 1 \to \ldots \to 35 \to 0$. Заранее определяется инициатор запуска алгоритма. В начале определяется, какие процессы завершатся досрочно, имитируя неисправность: с вероятностью 0.42 процесс выйдет из строя. Первый круг алгоритма состоит в следующем: процессы принимают сообщение «ВЫБОРЫ» и отмечают себя в массиве. Затем они передают дальше первому откликнувшемуся (за < 1 секунду) процессу сообщение «ВЫБОРЫ». Если окажется, что в массиве данный процесс уже отмечен, значит, был сделан круг, и нужно подвести итоги выборов с помощью рассылки по кругу сообщений «КООРДИНАТОР». Таким образом, все процессы договариваются о новом координаторе.

Оценка времени работы

Дано: `Ts = 100`, `Tb = 1`, `N = 36`, процессорные операции считаем «бесплатными». Оценим худший случай: пусть ни один процесс не вышел из строя. Тогда на первом круге будет отправлено N сообщений с массивами из N int, а на втором круге N сообщений с одним int. Итоговое количество байт n на один процесс равно `N * sizeof(int) + sizeof(int)`.

Итого получаем оценку:

```
T = N * (Ts + n * Tb) = 36 * (100 * 2 + (36 * 4 + 4) * 1) = 12528
```

Пример вывода программы

Процесс, инициирующий выборы имеет rank=5. В итоге координатором назначен процесс с rank=35, т.к. это максимальный rank среди всех живых.

```
$ source ../dockervars.sh
$ mpicc main.c -o main
$ mpirun --oversubscribe -n 36 main

Alive processes:
10 24 20 18 12 34 16 14 22 26 0 2 21 5 15 17 19 1 33 35 3
Stating the algorithm
 5: Didn't get a confirmation from  6.
 5: Didn't get a confirmation from  7.
 5: Didn't get a confirmation from  8.
 5: Didn't get a confirmation from  9.
10: Got array from  5
10: Didn't get a confirmation from 11.
12: Got array from 10
12: Didn't get a confirmation from 13.
14: Got array from 12
15: Got array from 14
16: Got array from 15
17: Got array from 16
18: Got array from 17
19: Got array from 18
20: Got array from 19
21: Got array from 20
22: Got array from 21
22: Didn't get a confirmation from 23.
24: Got array from 22
24: Didn't get a confirmation from 25.
26: Got array from 24
26: Didn't get a confirmation from 27.
26: Didn't get a confirmation from 28.
26: Didn't get a confirmation from 29.
26: Didn't get a confirmation from 30.
26: Didn't get a confirmation from 31.
26: Didn't get a confirmation from 32.
33: Got array from 26
34: Got array from 33
35: Got array from 34
 0: Got array from 35
 1: Got array from  0
 2: Got array from  1
 3: Got array from  2
 3: Didn't get a confirmation from  4.
 5: Got array from  3
 5: New coordinator: 35
10: New coordinator: 35
12: New coordinator: 35
14: New coordinator: 35
15: New coordinator: 35
16: New coordinator: 35
17: New coordinator: 35
18: New coordinator: 35
19: New coordinator: 35
20: New coordinator: 35
21: New coordinator: 35
22: New coordinator: 35
```

```
24: New coordinator: 35
26: New coordinator: 35
33: New coordinator: 35
34: New coordinator: 35
35: New coordinator: 35
 0: New coordinator: 35
 1: New coordinator: 35
 2: New coordinator: 35
 3: New coordinator: 35
```

Задание 2: Fault tolerance модификация RedBlack3D

Постановка задания

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.
Подготовить отчет о выполнении задания, включающий описание алгоритма, детали реализации, а также временные оценки работы алгоритма.

Алгоритм

Алгоритм из прошлого семестра – RedBlack 3D. Описание и деталии реализации алгоритма описаны в отчёте по СКиПОД, далее будут использоваться обозначения из него (например, sausages – "сосиски").

Модификация

- Каждый из K процессов отвечает за N//K "сосисок" – параллелепипедов размера 1x1xN (последний может брать на себя больше).
- После каждой итерации фиксируется текущее состояние используемых данных
- В случае возникновении ошибки, вызывается обработчик `verbose_errhandler(...)`. В нём происходит перераспределение работы на оставшиеся K-1 процессов поровну. Все процессы откатываются к последней контрольной точке и продолжают работу. Для передачи управления используется `longjump`.

Пример вывода программы

Запустим программу без убийства какого-либо процесса (`PROCESS_RANK_TO_BE_KILLED = -1, N=10, K=5`):

```
$ source ../dockervars.sh
$ mpicc main.c -o main
$ mpirun --oversubscribe -n 5 --with-ft ulfm main
```

```
rank 1, from 2 to 3
rank 2, from 3 to 4
rank 0, from 1 to 2
rank 3, from 4 to 5
rank 4, from 5 to 9
sum = 18.943777
0.033295, num_workers = 5
```

Теперь запустим её с убийством процесса с `rank=2` (PROCESS_RANK_TO_BE_KILLED = 2, N=10, K=5):

```
rank 1, from 2 to 3
rank 0, from 1 to 2
rank 2, from 3 to 4
rank 3, from 4 to 5
rank 4, from 5 to 9
killed.
rank 1, from 3 to 5
rank 2, from 5 to 7
rank 3, from 7 to 9
rank 0, from 1 to 3
sum = 18.893315
0.073041, num_workers = 4
```

Как можно видеть, количество работающих процессов сократилось, однако подсчёт результата был корректно завершён, однако понадобилось чуть больше времени.