Распределенные системы. Отчёт по практическому заданию 1

Андрей Лебедев, группа 424

1 Постановка задачи

Требуется реализовать операцию редукции `MPI_MAXLOC` (определение глобального максимума и первого процесса, у которого он встречается) на двумерной транспьютерной матрице 8×8. Каждый процесс хранит локальное значение `data` и свой `rank`. При равных значениях `data` приоритет имеет процесс с меньшим `rank`. В итоге необходимо, чтобы все процессы получили `(res_data, res_rank)`, соответствующие глобальному максимуму.

- Использовать только точечные пересылки (`MPI_Send / MPI_Recv`).
- Время старта одной пересылки (`Ts`) — 100, передача одного байта (`Tb`) — 1. Нужно оценить, как долго будет выполняться глобальная редукция.
- Процессорные операции считаются бесконечно быстрыми.

2 Основные функции и их описание

2.1 topology.c

- `MPI_Comm create_2d_topology(int *rank)` — инициализирует `MPI`, определяет двумерный коммуникатор размера `N_ROW` × `N_COL`, возвращает его, а также узнаёт `rank`.
- `void get_2d_coords(MPI_Comm comm, int rank, int coords[2])` — возвращает координаты `(row,col)` процесса в созданном 2D-коммуникаторе.

2.2 data.c

- `int generate_value(int rank)` — генерирует псевдослучайное число на основе `rank`.
- `void gather_and_print_generated_data(...)` — собирает у всех процессов `(rank, coords, data)` через `MPI_Gather` и выводит общей таблицей.
- `void gather_and_print_final_data(...)` — аналогично собирает `(rank, res_rank, res_coords, res_data)` и печатает одной таблицей.

2.3 reduction.c

Здесь реализуются «ручные» коллективные операции.

- `void reduce_2d_rc(...)` — сначала редукция по строкам: каждый лидер строки собирает максимум `data` у процессов своей строки. Затем редукция по столбцам: процесс `(0,0)` собирает максимум у лидеров строк. В итоге `(0,0)` знает глобальный максимум.
- `void broadcast_2d_rc(...)` — обратная рассылка глобального максимума всем: сначала процесс `(0,0)` шлёт лидерам строк, затем каждый лидер строки шлёт всем процессам в своей строке.

3 Запуск программы

Пример команды для компиляции программы:

```
$ make
```

`Makefile` автоматически вызывает `mpicc` с нужными флагами и собирает исполняемый файл `main`.

Запуск программы с эмуляцией 64 процессов:

```
$ mpirun --oversubscribe -n 64 ./main
```

- `-oversubscribe` используется, чтобы разрешить запуск больше процессов, чем доступно физических ядер процессора при тестировании на локальной машине.
- `-n 64` задаёт количество процессов.

4 Пример работы программы

Запустим программу на матрице `N_ROW` $= 8$ и `N_COL` $= 8$. Ниже приведён сокращённый фрагмент вывода, где:

- В разделе «GENERATED DATA» видны псевдослучайные числа, которые сгенерировал каждый из 64 процессов.
- В разделе «FINAL RESULTS» показано, что во всех процессах совпадают `res_rank` и `res_data`, равные глобальному максимуму (для примера предполагается, что максимальное значение оказалось у процесса `PID` $= 63$).

Листинг 1: Сокращенный пример вывода

```
1 GENERATED DATA (all processes):
2
3  PID  coords   data
4
5    0  ( 0, 0)    123456
6    1  ( 0, 1)     51234
```

```
...
  62  ( 7, 6)    654321
  63  ( 7, 7)    999999


FINAL RESULTS (each process sees the same global max):

 PID  res_rank res_coords res_data

   0        63  ( 7, 7)       999999
   1        63  ( 7, 7)       999999
...
  62        63  ( 7, 7)       999999
  63        63  ( 7, 7)       999999

```

Как видно, глобальным максимумом оказалось значение 999999, полученное процессом PID = 63. После редукции и распространения это значение и соответствующий res_rank стали известны всем процессам.

5 Временная оценка

По условию, для всех процессов операция редукции запускается одновременно, а время передачи сообщения из k байт есть $\texttt{Ts} + k \cdot \texttt{Tb}$. Поскольку мы пересылаем всего несколько целых, величина k мала. Основное время — это $\texttt{Ts} = 100$ на каждую отправку.
Предложенный алгоритм:

1. Редукция по строкам. N_ROW строк, в каждой из которых $\texttt{N\_COL} - 1$ пересылка (каждый не-лидер посылает лидеру). Затем $\texttt{N\_ROW} - 1$ пересылка для сбора в $(0, 0)$. Итого $\texttt{N\_ROW} \cdot (\texttt{N\_COL} - 1) + (\texttt{N\_ROW} - 1)$ отправок.

2. Broadcast. Аналогичное число отправок: $(\texttt{N\_ROW} - 1)$ (от (0,0) к лидерам строк) плюс $\texttt{N\_ROW} \cdot (\texttt{N\_COL} - 1)$ (распространение по строкам).

В сумме получается:

$$2\times\Big[\texttt{N\_ROW}\cdot(\texttt{N\_COL}-1)+(\texttt{N\_ROW}-1)\Big] = 2\times\big[\texttt{N\_ROW}\cdot\texttt{N\_COL}-\texttt{N\_ROW}+\texttt{N\_ROW}-1\big] = 2\times\big[\texttt{N\_ROW}\cdot\texttt{N\_COL}-1\big].$$

$\texttt{N\_ROW} = \texttt{N\_COL} = 8$. Тогда $\texttt{N\_ROW} \cdot \texttt{N\_COL} = 64$. Получаем $2 \times (64 - 1) = 2 \times 63 = 126$ отправок.
Каждая отправка длится $\texttt{Ts} = 100$, итого $126 \times 100 = 12600$ условных единиц времени.
Таким образом, **время выполнения редукции** — порядка $O(\texttt{N\_ROW} \cdot \texttt{N\_COL})$ отправок, умноженных на Ts. При $\texttt{N\_ROW} = 8$, $\texttt{N\_COL} = 8$, ориентировочно около **12600** условных единиц.