Разработка параллельной версии программы для сортировки данных

Богатенкова Анастасия, 328 гр.

Ноябрь, 2019

1 Алгоритм

Выбран алгоритм сортировки "пузырьком". Алгоритм пузырьковой сортировки в прямом виде достаточно сложен для распараллеливания – сравнение пар значений упорядочиваемого набора данных происходит строго последовательно.

```
// Последовательный алгоритм пузырьковой сортировки
void BubbleSort(double A[], int n) {
  for (int i = 0; i < n - 1; i++)
    for (int j = 0; j < n - i; j++)
      compare_exchange(A[j], A[j + 1]);
}
```

В связи с этим для параллельного применения был использован не сам этот алгоритм, а его модификация - метод чет-нечетной перестановки. Суть модификации состоит в том, что в алгоритм сортировки вводятся два разных правила выполнения итераций метода: в зависимости от четности или нечетности номера итерации сортировки для обработки выбираются элементы с четными или нечетными индексами соответственно, сравнение выделяемых значений всегда осуществляется с их правыми соседними элементами.

```
// Последовательный алгоритм чет-нечетной перестановки
void OddEvenSort(double A[], int n) {
 for (int i = 1; i < n; i++) {
   if (i % 2 == 1) {    // нечетная итерация
    for (int j = 0; j < n/2 - 2; j++)
      compare_exchange(A[2*j + 1], A[2*j + 2]);
    if (n % 2 == 1) // сравнение последней пары при нечетном n
      compare_exchange(A[n - 2], A[n - 1]);
   }
   else    // четная итерация
    for (int j = 1; j < n/2 - 1; j++)
      compare_exchange(A[2*j], A[2*j + 1]);
 }
}
```

Получение параллельного варианта для метода чет-нечетной перестановки уже не представляет каких-либо затруднений – несмотря на то, что четные и нечетные итерации должны выполняться строго последовательно, сравнения пар значений на итерациях сортировки являются независимыми и могут быть выполнены параллельно. Поскольку все вычислительные элементы имеют прямой доступ к каждому значению в сортируемом массиве, сравнение значений a_i и a_j может быть выполнено любым вычислительным элементом. При наличии нескольких вычислительных элементов появляется возможность одновременно выполнять операцию «Сравнить и переставить» над несколькими парами значений.

Таким образом, в OpenMP-версии в программу добавляются 2 клаузы omp parallel for, при помощи которых распараллеливаются циклы сравнения пар значений на четных и нечетных итерациях метода чет-нечетной перестановки.

2 Результаты замеров времени выполнения

Ниже приведены результаты замеров времени программ на суперкомпьютерах Bluegene и Polus: непосредственно в табличной форме и наглядно на 3D-графиках. Программа была запущена в конфигурациях:

- на Bluegene - 1, 2, 4 ядер;
- на Polus - 1, 2, 4, 8 ядер.

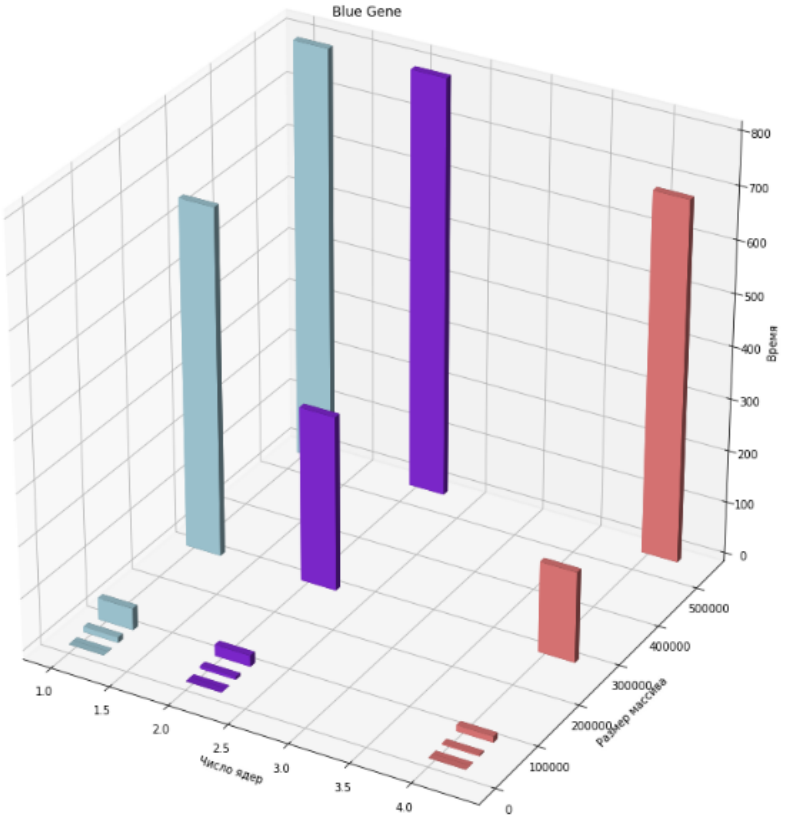
Также для сравнения OpenMP-версия программы была запущена на ноутбуке (OS: Linux Mint 19.1 Tessa x86 64; Kernel: 4.15.0-43-generic; CPU: Intel Celeron N2920 (4) @ 1.999GHz; Memory: 2430MB / 3827MB)

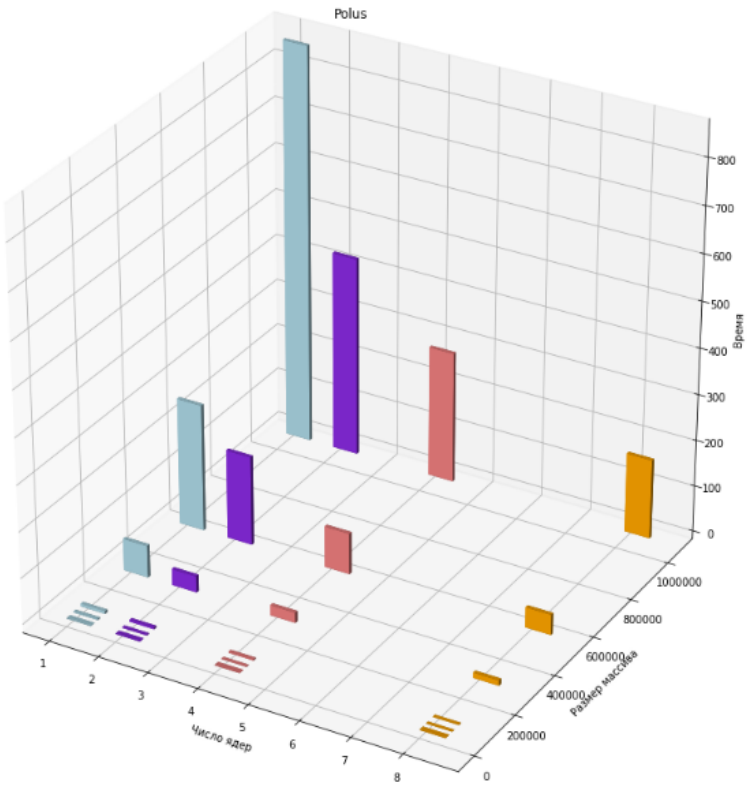
Каждая конфигурация была запущена 5 раз. Ниже приведены усредненные результаты.

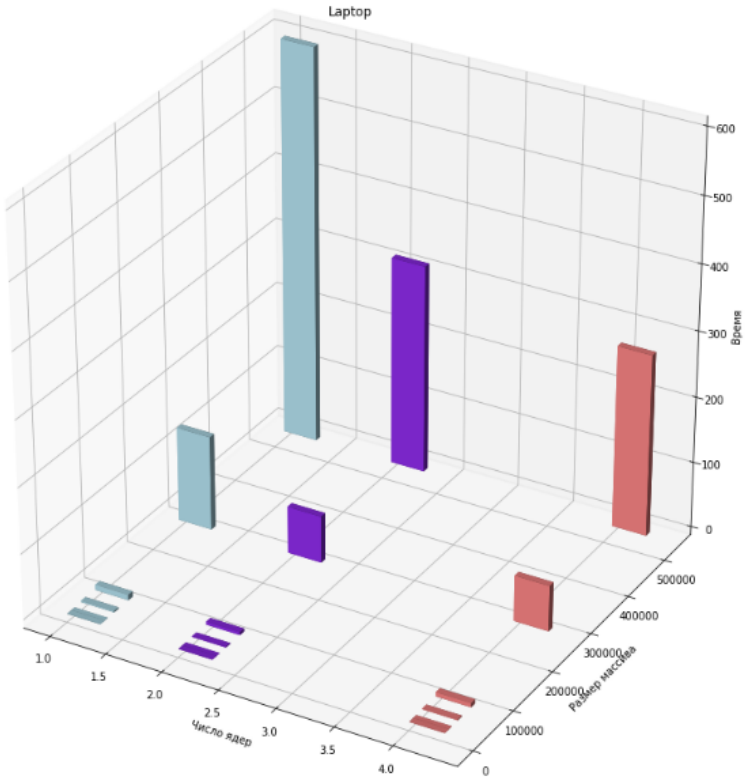
Blue Gene	компиляция: mpixlc_r -qsmp=omp openmp.c -o openmp -O2 запуск: mpisubmit.bg -n 5 -w 00:00:15 openmp -- число_нитей размер_массива		
размер массива \ число нитей	**1**	**2**	**4**
1024	0.01	0.015	0.013
8192	0.65	0.4	0.25
32768	10.46	5.5	3.03
65536	41.72	21.4	11.4
262144	665.5	334.98	173.75
524288	>15 мин	>15 мин	690.3

Polus	компиляция: xlc_r -qsmp=omp openmp.c -o openmp -O2 запуск: mpisubmit.pl -p 1 -w 00:15 openmp -- число_нитей размер_массива			
размер массива \ число нитей	**1**	**2**	**4**	**8**
1024	0.0014	0.0017	0.0025	0.0038
8192	0.19	0.049	0.09	0.09
32768	2.1	1.31	0.84	0.29
65536	5.79	3.86	2.35	0.81
262144	69.31	35.36	22.38	11.68
524288	274.6	192.96	89.24	44.11
1048576	860.26	432.42	284.11	172.13

Laptop	компиляция: gcc openmp.c -o openmp -O2 -fopenmp						
размер массива \ число нитей	**1**	**2**	**4**				
1024	0.006	0.006	0.015				
8192	0.14	0.09	0.11				
32768	2.06	1.16	1.3				
65536	8.15	4.4	6.27				
262144	143.15	71.4	69.39				
524288	601.69	317.76	276.76				




Polus
Время
0
100
200
300
400
500
600
700
800
Число ядер
1
2
3
4
5
6
7
8
Размер массива
0
200000
400000
600000
800000
1000000


Laptop
Время
Размер массива
Число ядер
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
100000
200000
300000
400000
500000
0
100
200
300
400
500
600