Сортировка слиянием. OpenMP

Николай Игоревич Хохлов

МФТИ, Долгопрудный

22 марта 2017 г.

Сортировка слиянием

```
void merge_sort(int *a, int na)
{
        if(na < 2) return;
        merge_sort(a, na / 2);
        merge_sort(a + na / 2, na - na / 2);
        int *b = (int*)malloc(sizeof(int) * na);
        merge(a, a + na / 2, b, na / 2, na - na / 2
        memcpy(a, b, sizeof(int) * na);
        free(b);
}
```

Варианты реализации OpenMP

Какие варианты реализации могут быть на OpenMP?

- task;
- section;
- алгоритмы коллективного взаимодействия.

#pragma omp task

#pragma omp task

Текущая нить выделяет в качестве задачи ассоциированный с директивой блок операторов. Задача может выполняться немедленно после создания или быть отложенной на неопределённое время и выполняться по частям. Размер таких частей, а также порядок выполнения частей разных отложенных задач определяется реализацией.
untied – опция означает, что в случае откладывания задача может быть продолжена любой нитью из числа выполняющих данную параллельную область; если данная опция не указана, то задача может быть продолжена только породившей её нитью.

Возможно ли применение данной директивы?

#pragma omp task

```
void parallel_sort(int *a, int na)
{
    if(na < 2) return;
    #pragma omp parallel
    {
        #pragma omp task untied
        parallel_sort(a, na / 2);
        #pragma omp task untied
        parallel_sort(a + na / 2, na - na / 2);
    }
    int *b = (int*)malloc(sizeof(int) * na);
    merge(a, a + na / 2, b, na / 2, na - na / 2);
    memcpy(a, b, sizeof(int) * na);
    free(b);
}
```

#pragma omp task

- По умолчанию нет вложенного параллелизма (нити создадуться один раз).
- На каждую сортировку будет создана задача – возможны проблемы с производительностью.
- Порядок выполнения задач не известен – возможны проблемы с корректностью работы.

#pragma omp sections

#pragma omp sections

Директива sections используется для задания конечного (неитеративного) параллелизма. Директива sections содержит набор структурированных блоков, которые распределяются по потокам в группе. Каждый структурированный блок исполняется один раз одним из потоков в группе.

#pragma omp sections

```
#pragma omp sections
{
#pragma omp section
    block1
#pragma omp section
    block2
}
```

#pragma omp sections

```
void parallel_sort(int *a, int na)
{
    if(na < 2) return;
    #pragma omp parallel
    {
        #pragma omp sections
        {
            #pragma omp section
            parallel_sort(a, na / 2);
            #pragma omp section
            parallel_sort(a + na / 2, na - na / 2);
        }
    }
    int *b = (int*)malloc(sizeof(int) * na);
    merge(a, a + na / 2, b, na / 2, na - na / 2);
    memcpy(a, b, sizeof(int) * na);
    free(b);
}
```

#pragma omp sections

- Создается две секции.
- Вложенного параллелизма нет.
- Максимальная работа на двух потоках.

Число ядер	Время, с	Ускорение
1	9.27	-
2	5.61	1.65
4	5.65	1.64

Необходимо увеличить число секций, выполняемых одновременно.

Вложенный параллелизм (nested)

void omp_set_nested(int nested)
Функция omp_set_nested() разрешает или запрещает вложенный параллелизм. В качестве значения параметра задаётся 0 или 1. Если вложенный параллелизм разрешён, то каждая нить, в которой встретится описание параллельной области, породит для её выполнения новую группу нитей. Сама породившая нить станет в новой группе нитью-мастером. Если система не поддерживает вложенный параллелизм, данная функция не будет иметь эффекта.
Добавляем в код omp_set_nested(1);

Число ядер	Время, с	Ускорение
1	43.18	-
2	ошибка	-
4	ошибка	-

Вложенный параллелизм (nested)

Почему увеличивается времы выполнения и выдаются ошибки?

- Каждая секция порождает параллельную секцию.
- Какждая итерация рекурсии порождает две секции.
- Происходит сильный рост числа потоков – ошибка выделения ресурсов.

Необходимо ограничить число порождаемых секций.

#pragma omp sections

```
void parallel_sort(int *a, int na, int nt)
{
    if(na < 2) return;
    if (nt < 2) {
        merge_sort(a, na);
        return;
    }
    #pragma omp parallel num_threads(2)
    {
        #pragma omp sections
        {
            #pragma omp section
            parallel_sort(a, na / 2, tn / 2);
            #pragma omp section
            parallel_sort(a + na / 2, na - na / 2, n
        }
    }
    int *b = (int*)malloc(sizeof(int) * na);
```

Вложенный параллелизм (nested)

Значение nt необходимо задать числу потоков. Можно использовать вызов omp_get_max_threads().

- Каждая параллельная секция состоит из двух потоков.
- Потоки порождаются до тех пор, пока их число не превысило максимальное число потоков.
- Когда создались все потоки – используется последовательная сортировка.

Число ядер	Время, с	Ускорение
1	9.27	-
2	5.62	1.65
4	3.46	2.67

Алгоритмы коллективного взаимодействия

Возможна реализация, используя алгоритмы коллективного взаимодействич по схеме гиперкуб.
Реализация дает несколько лучшее ускорение, чем реализация через sections и вложенный параллелизм.

Число ядер	Время sec/hyp, с	Ускорение sec/hyp
1	136.8/145.2	-
8	60.3/43.0	2.27/3.38
12	54.4/36.1	2.51/4.0