ДВУХУРОВНЕВОЕ РАСПАРАЛЛЕЛИВАНИЕ ДЛЯ ОПТИМИЗАЦИИ ВЫЧИСЛЕНИЙ НА СУПЕРКОМПЬЮТЕРЕ ПРИ РАСЧЕТЕ ЗАДАЧ ГАЗОВОЙ ДИНАМИКИ

А.А. Рыбаков

Межведомственный суперкомпьютерный центр РАН - филиал ФГУ ФНЦ НИИСИ РАН, Москва, rybakov@jscc.ru

Численные расчеты задач газовой динамики как правило связаны с большим объемом вычислений, поэтому при использовании суперкомпьютеров важную роль имеет как скорость вычислений, так и оптимальность загрузки вычислительных мощностей. При использовании часто используемых в расчетах блочно-структурированных сеток с достаточно большим количеством блоков можно добиться близкого к равномерному распределения блоков сетки по вычислительным узлам суперкомпьютера. При этом отдельный узел суперкомпьютера должен обрабатывать несколько блоков сетки, а взаимодействие между узлами осуществляется с помощью технологии MPI.

Внутри вычислительного узла обработка отдельного блока сетки выполняется с применением распараллеливания на несколько потоков с помощью OpenMP. Однако программный код обработки блока может содержать и последовательные участки. Таким образом, при последовательной обработке блоков сетки внутри вычислительного узла возникают потери производительности, связанные с простоем вычислительных мощностей. Выходом из данной ситуации служит использование двухуровневого распараллеливания внутри вычислительного узла. На первом уровне осуществляется распараллеливание по блокам, на втором - распараллеливание вычислений для одного конкретного блока (рис. 1).

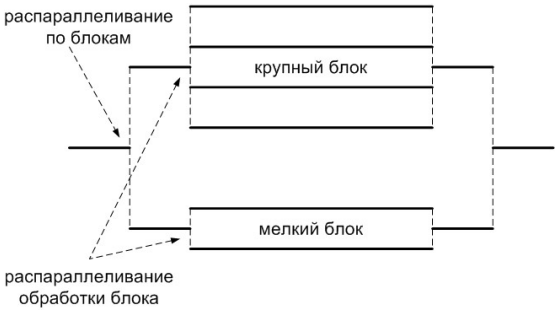

Рис. 1. Двухуровневая схема распараллеливания вычислений внутри вычислительного узла суперкомпьютера.

Для сбалансированной по времени обработки блоков сетки внутри одного вычислительного узла степень распараллеливания блока должна зависеть от его размера, то есть блок большего размера должен обрабатываться большим количеством потоков, чем меньший по размеру блок. Однако этого не достаточно. На рис. 2 показана ситуация, когда бесконтрольное расположение последовательных и параллельных секций обработки двух блоков внутри вычислительного узла приводит в появлению обширных зон низкой загрузки вычислительных мощностей.

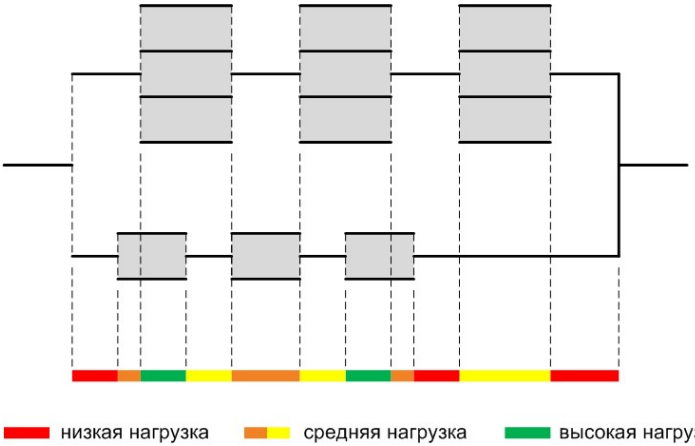

Рис. 2. Иллюстрация низкой загрузки вычислительных мощностей при параллельной обработке двух блоков сетки.

Выходом из данной ситуации является управление выполнением отдельных участков обработки блока для достижения более плотной загрузки вычислительных мощностей. Назовем вычисления по обработке одного блока сетки процедурой. Каждая конкретная процедура состоит из последовательности участков, которые мы будем называть атомами. Для каждого атома известна продолжительность его последовательного выполнения и оптимальное количество потоков в которых он может быть выполнен параллельно. Исполнение атома происходит непрерывно и не может быть приостановлено. Атомы одной процедуры исполняются друг за другом. Тогда жадная стратегия выделения ресурсов для выполнения всех процедур может выглядеть следующим образом: каждая процедура начинает выполняться в отдельном потоке, каждый готовый к выполнению атом каждой процедуры ждет освобождения необходимого ему количества потоков для параллельного выполнения, резервирует потоки, производит вычисления, после чего освобождает потоки. Схема выполнения отдельного атома приведена ниже:

```
void ExeAtom::GreedyExe(int *free_threads)
{
    int need_threads = <необходимое количество потоков>;
    bool is_start = false;

    while (true)
    {
        while (*free_threads < need_threads) <нулевое ожидание>;

        #pragma omp critical
        {
            if (*free_threads >= need_threads)
            {
                is_start = true;
                *free_threads -= need_threads;
            }
        }

        if (!is_start) continue;

        ParallelExe();

        #pragma omp critical
        {
            *free_threads += need_threads;
        }

        break;
    }
}
```

Данная стратегия может быть ослаблена, если разрешить атому использовать меньшее количество потоков, чем нужно ему для оптимального параллельного исполнения.

Для оценки эффективности жадной стратегии планирования исполнения атомов была выбрана модель процедуры с весом k, состоящая из трех параллельных атомов продолжительностью k^2/10 и оптимальным количеством потоков k, и двух последовательных атомов продолжительностью 1/10. При этом параллельные атомы в процедуре чередуются с последовательными. На рис. 3 приведены результаты запусков разных наборов процедур на процессоре Intel Xeon с 16 потоками. Жадная стратегия исполнения атома представлена с разными порогами снижения общего количества потоков исполнения (например порог 25% означает, что атом может попытаться выполниться не на оптимальном количестве потоков, а на количестве потоков на 25% меньшем от оптимального, что позволит начать исполнение раньше). Для сравнения приведены данные идеального исполнения процедур, когда общее количество потоков равно бесконечности.

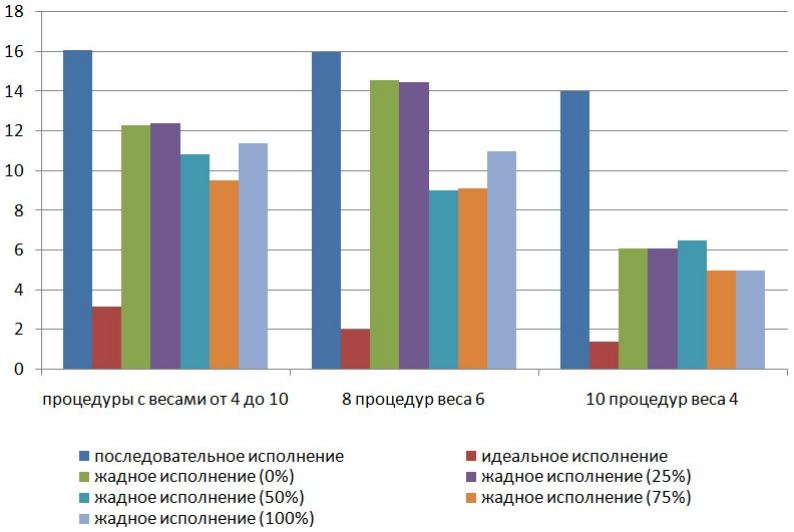

Рис. 3. Результаты сравнения разных стратегий выполнения наборов процедур для процессора с 16 потоками.

Из рис. 3 видно, что применение жадной стратегии исполнения процедур приводит к существенному снижению общего времени работы по сравнению с последовательным исполнением процедур. Особенно это актуально для запуска большого количества процедур с маленькими весами. Данная стратегия позволяет более компактно располагать обработку мелких блоков сетки в вычислительном узле, снижая общее время расчета.

Литература

1. J. Blazek. Computational fluid dynamics: principles and applications. Elsevier, 2001.
2. M. Farrashkhalvat, J. P. Miles. Basic structured grid generation with an introduction to unstructured grid generation. Butterworth-Heinemann, 2003.
3. К. Хьюз, Т. Хьюз. Параллельное и распределенное программирование с использованием C++. Вильямс, 2004.
4. M. Queen. Parallel programming in C with MPI and OpenMP. Mc-Grow Hill, 2004.
5. R. Chandra, R. Menon, L. Dagum, D. Kohr, D. Maydan, J. McDonald. Parallel programming in OpenMP. Morgan Kaufmann, 2000.