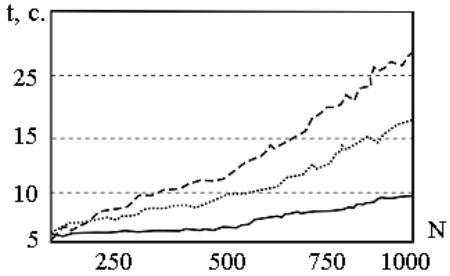

Рис. 1 Длительность вычислений по пакету B-CALM (пунктир), алгоритму на основе канонической формы для уравнений Максвелла (точки) и предложенному алгоритму (непрерывная кривая).

Отмечая более чем трехкратное превосходство предложенного алгоритма по сравнению с B-CALM и более чем двукратное по сравнению с предыдущей авторской разработкой, объясним эти отличия во-первых переходом к векторному представлению операций. Разработчики B-CALM работают с потоками, что, видимо, менее соответствует конвейерной архитектуре GPU. Во-вторых, повторным использованием сдвоенных сумм и переходом к другому типу векторных операций (от покомпонентного произведения векторов в [9] к saxpy). Как известно [10], одна векторная операция типа saxpy содержит больше скалярных операций, чем покомпонентное произведение.

Благодарности

Публикация подготовлена в рамках работ по гранту РФФИ № 14-07-00291 А.

Заключение

Программная реализация предложенного векторного алгоритма решения сеточных уравнений явной разностной схемы для дифференциального уравнения Даламбера позволила ускорить вычисления на видеопроцессоре по сравнению с известными авторам реализациями за счет перехода к использованию векторных нотации алгоритма и процедур библиотеки CUBLAS.

Литература

1. Taflove, A. Computational Electrodynamics: The Finite-Difference Time-Domain Method / A. Taflove, S. Hagness. ed. 3-d. – Boston: Arthech House Publishers, 2005. – 852 p.

2. Дифракционная нанофотоника / под редакцией В.А. Сойфера. - М.: Физматлит, 2011. — 680 с.

3. Климов, В.В. Наноплазмоника/ В.В. Климов. - М.: Физматлит, 2010. — 480 с.

4. Lourtioz, J.-M. Photonic Crystals. Towards Nanoscale Photonic Devices/ J.-M. Lourtioz, H. Benistry, V. Berger, J.-M. Gerard, D. Maystre, A. Tchelnokov. - Springer, 2008 514 p.

5. Козлова, Е.С. Моделирование распространения короткого двумерного импульса света / Е.С. Козлова, В.В. Котляр // Компьютерная оптика. – 2012. – Т. 36, No 2. – С. 158–164

6. Wahl, P. B-CALM: An open-source GPU-based 3D-FDTD with multi-pole dispersion for plasmonics / P. Wahl, D.-S. LyGagnon, Ch. Debaes, D.A.B. Miller, H. Thienpont // Optical and Quantum Electronics. – 2012. – Vol. 44. – P. 285-290.

7. Неганов, В.А. Линейная макроскопическая электродинамика. Том 1 / В.А.Неганов, С.Б.Раевский, Г.П.Яровой. - М: Радио и Связь, 2000. – 512 с.

8. Самарский, А.А. Теория разностных схем/ А.А. Самарский. - М.: Наука, 1977. — 656 с.

9. Воротникова, Д.Г. Алгоритмы с «длинными» векторами решения сеточных уравнений явных разностных схем / Д.Г. Воротникова, Д.Л. Головашкин // Компьютерная оптика. – 2015. – Т. 39, № 1. – С. 87-93.

10. Golub, G.H. Matrix Computations / G.H. Golub, Ch.F. Van Loan. – Baltimore: Johns Hopkins University Press, 1989. – 747 p.

С.В. Востокин

ПРИМЕНЕНИЕ ПРЕДМЕТНЫХ ЯЗЫКОВ ДЛЯ АВТОМАТИЗАЦИИ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ

(Самарский национальный исследовательский университет
имени академика С.П. Королёва)

Предметные языки (domain-specific language, DSL) широко применяются для автоматизации научных расчетов. Наиболее известные из них: MATLAB (векторные вычисления); Wolfram Language (комбинация символьной и численной математики); Maple (символьные вычисления); Mathcad (интерактивные вычисления, визуализация). Преимущество данных языков в предметной области численного моделирования состоит в том, что их синтаксис и семантика адаптированы к традиционной математической нотации, что существенно снижает трудоёмкость программирования.

Однако, если требуется оптимизация кода, что характерно для области высокопроизводительных научных вычислений, предметные языки уступают языкам общего назначения высокого уровня C/C++. Поэтому критичные к производительности части кода перечисленных выше предметных языков реализуются именно на языках C/C++.

Для ускоренной разработки программного обеспечения были предложены методики, позволяющие сочетать предметные языки и языки общего назначения в одном проекте [1,2]. В данной работе рассматривается применение этого подхода для автоматизации высокопроизводительных научных вычислений на базе облачного сервиса Templet Web [3].

Автоматизация программирования при составлении задачи в системе Templet Web основана на концепциях скелетного программирования [4,5], предметных языков и использовании специализированных визуальных редакторов предметных языков в составе интегрированной среды разработки (IDE).

В системе Templet Web используются несколько предметных языков для решения перечисленных ниже задач автоматизации научных вычислений:

- организации вычислений в парадигме «портфель задач» [6];
- применения акторной модели вычислений, реализованной в общей и распределённой памяти [7];
- ввода данных;
- вывода данных;
- вызова функций стандартных математических библиотек;
- описания серий экспериментов.

Для пояснения назначения предметных языков в системе Templet Web ниже приведён скелет программы параллельного умножения матриц в распределённой и разделяемой памяти. Под скелетом программы понимается проект на языке программирования высокого уровня (C/C++), который, в отличие от проекта программы на псевдокоде, компилируется.

Предметные языки в исследуемой технологии автоматизации используются, чтобы пояснить, что конкретно понимается под не полностью определёнными или неопределёнными объектами в проекте программы (её скелете). Информация, передаваемая на предметном языке, в дальнейшем применяется для автоматического преобразования скелета программы в исполняемый код.

```
const int N=10;
double a[N][N],b[N][N],c[N][N];

struct Result{
  void save(ostream&s){ s<<num;
     for(int j=0;j<N;j++)s<<c[num][j]; }
  void rest(istream&s){ s>>num;
     for(int j=0;j<N;j++)s>>c[num][j]; }
    int num;// номер вычисленной строки
};
struct Task{
    void save(ostream&s){ s<<num; }
    void rest(istream&s){ s>>num; }
    int num;// номер вычисляемой строки
};
void Proc(Task&t,Result&r){
    int i=t.num;// параллельное вычисление строки матрицы C
    for(int j=0;j<N;j++){
        c[i][j]=0.0;
        for(int k=0;k<N;k++)c[i][j]+=a[i][k]*b[k][j];
    }
    r.num=i;
}
struct Bag{
```



```
    Bag(int argc, char* argv[]);
    void run();
    bool isTask(){return cur<N;}
    void put(Result&){ }
    void get(Task&t){t.num=cur++;}
    int cur;//номер текущей строки в матрице C
};
int main(int argc, char* argv[])
{
    Bag bag(argc,argv);
    // инициализация
    input(a,b);
    bag.cur=0;
    // параллельное умножение матриц
    bag.run();
    // вывод результата параллельного умножения
    output(c);

    start_time();
    strassen(a,b,c); // теперь умножаем методом Штрассена
    end_time();
    return 0;
}
```

В приведённом примере структуры `Result`, `Task`, `Bag` и функция `Proc` являются составными частями реализации метода организации вычислений «портфель задач». Это указано в аннотации на предметном языке, которая сопровождает приведённый код скелета программы умножения матриц (форма описания скелетов на DSL в данной статье не приводится).

Описание скелета программы содержит способ именования составных элементов метода «портфель задач», способ реализации в общей или распределённой памяти, количество используемых узлов и/или потоков выполнения и другие детали.

Скелет программы также содержит неопределённые процедуры `input`, `output` для ввода и вывода данных. Аннотация на предметном языке описывает привязку этих процедур к имеющимся в системе Templet Web пользовательским интерфейсам ввода-вывода данных.

В коде скелета программы имеется неопределённая процедура `strassen`. Это ссылка на реализацию соответствующего алгоритма умножения матриц Штрассена. Пользователь на предметном языке в визуальном редакторе может выбрать конкретную библиотечную реализацию алгоритма. При этом ему не нужно знать сигнатуру реальной функции в библиотеке и детали связи с этой библиотекой. Аналогично можно сделать привязку к другим библиотечным процедурам. Например, в задаче анализа хаотического движения гироскопов [8] интересны разные варианты численного интегрирования пользовательских функций.

Предметный язык описания серий экспериментов в системе Templet Web предназначен для автоматического построения кода серии экспериментов из

программы для единичного эксперимента. Например, в коде скелета программы умножения матриц параметр `N` может изменяться в пределах, описанных на DSL. Также можно автоматически выполнять статистическую обработку результатов при замере производительности. Например, пара функций `start_time()`, `end_time()` может обозначать секцию кода с замером времени. Полученное в серии время можно автоматически усреднить, определить минимум и максимум, СКО, построить гистограмму функции распределения вероятности времени исполения.

В текущей версии системы Templet Web, развёрнутой по адресу templet.ssau.ru/templet на базе вычислительного кластера «Сергей Королёв» СГАУ, в настоящее время поддерживаются предметный язык Templet и скелеты программ «портфель задач», «конвейер». В дальнейшем планируется развивать данную технологию на основе подсистемы интегрированной среды разработки (IDE), встроенной в сервис.

Литература

1. Ward, M.P. Language-oriented programming // Software-Concepts and Tools, vol. 15, no. 4, pp. 147–161, 1994.

2. Dmitriev, S. Language oriented programming: The next programming paradigm // JetBrains onBoard, vol. 1, no. 2, pp. 1–13, 2004.

3. Артамонов, Ю.С. Применение облачного сервиса Templet Web при проведении лабораторных практикумов на суперкомпьютере «Сергей Королев» [Текст] / Ю.С. Артамонов, С.В. Востокин // X Международная научно-практическая конференция «Современные информационные технологии и ИТ-образование», МГУ, Москва, 2015. Том 2. – С. 409 - 414.

4. Cole, M. Bringing skeletons out of the closet: a pragmatic manifesto for skeletal parallel programming // Parallel computing, vol. 30, no. 3, pp. 389–406, 2004.

5. González-Vélez, H. A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers / H. González-Vélez, M. Leyton // Software: Practice and Experience, vol. 40, no. 12, pp. 1135–1160, 2010.

6. Литвинов, В.Г. Разработка и применение вычислительной модели типовых решений. Пример использования «портфеля задач» для обучения нейронной сети HRBF // Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. Науки, 3(36) (2014), 183–195.

7. Востокин, С.В. Templet: язык разметки для параллельного программирования // Известия Самарского научного центра РАН. Том 17, №2(5), 2015. С.947-955.

8. Востокин С.В. Программный комплекс анализа многомерных динамических систем и процессов на суперкомпьютере «Сергей Королёв» / С.В. Востокин, А.В. Дорошин, Ю.С. Артамонов, Ю.П. Назаров // Управление движением и навигация летательных аппаратов: сборник трудов XVI Всероссийского семинара по управлению движением и навигацией летательных аппаратов. - Самара: Издательство СНЦ РАН, 2013. - с.60-63.

Я.В. Голубева

РАЗРАБОТКА МЕТОДИКИ ИССЛЕДОВАНИЯ АЛГОРИТМОВ БАЛАНСИРОВКИ НАГРУЗКИ В ПАРАЛЛЕЛЬНОЙ РЕАЛИЗАЦИИ МЕТОДА ВЕТВЕЙ И ГРАНИЦ[3]

(Вычислительный центр им. А.А. Дородницына Федерального исследовательского центра «Информатика и управление» оссийской академии наук)

Одним из основных методов решения задач глобальной оптимизации является метод ветвей и границ [1]. Идея этого метода заключается в том, что множество допустимых решений разбивается на подмножества, и те подмножества (подзадачи), в которых оптимальных решений не существует, отбрасываются. Метод ветвей и границ имеет древовидную структуру, где вершины – это полученные в результате разбиений подзадачи (см. рис. 1). Вершина, отмеченная буквой Т – это отбрасываемая вершина, а вершины, отмеченные буквой К – это задачи-кандидаты, которые могут привести к оптимальному решению.

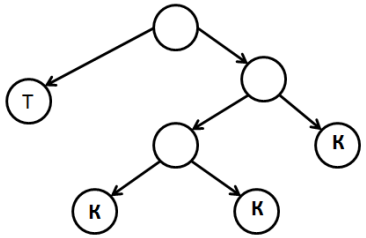

Рис. 1. Древовидная структура Метода ветвей и границ

Такая структура метода создает благоприятные условия для его параллельной реализации, так как поиск оптимальных решений в отдельных ветвях этого дерева может производиться независимо на разных вычислительных узлах. В то же время при параллельной реализации возникают определенные сложности. В частности, возникает проблема балансировки нагрузки. Она заключается в том, что невозможно статически равномерно распределить нагрузку между процессорами, так как дерево является неоднородным и его структура не известна заранее. Это может привести к тому, что в определенный момент времени одни процессоры уже закончат выполнять разветвления и будут простаивать, а другие всё еще будет продолжать решать выделенные им подзадачи.

[3] Работа выполнена при поддержке РФФИ (проекты № 16-07-00873 А и № 16-07-00458 А).