УДК 004.02

ОРГАНИЗАЦИЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В МНОГОЯДЕРНЫХ ПРОЦЕССОРАХ

Тузко Я. Н., Соколова О. О., Акишин Б. А.
Донской государственный технический университет, Ростов-на-Дону, Российская Федерация
yaroslav-tuzko-2015@yandex.ru
fallout101.lp@mail.ru
akiboralex@mail.ru

Проведен анализ существующих технологий и методов организации параллельных вычислений, отмечены сложности разработки и программирования параллельных алгоритмов. С помощью пакета MATLAB на примере задачи перемножения матриц большой размерности была исследована зависимость времени решения от размерности при использовании последовательного и параллельного (на два ядра) алгоритмов.

Ключевые слова: параллельные вычисления, программно-аппаратная архитектура, перемножения матриц, программирование в MATLAB.

UDC 004.02

ORGANIZATION OF PARALLEL COMPUTING IN MULTI-CORE PROCESSORS

Tuzko Ya. N., Sokolova O. O., Akishin B. A.
Don State Technical University, Rostov-on-Don, Russian Federation

yaroslav-tuzko-2015@yandex.ru
fallout101.lp@mail.ru
akiboralex@mail.ru

The analysis of the existing technologies and methods of organization of parallel computations is carried out, the difficulties in developing and programming parallel algorithms are noted in this article. The dependence of the solution time on the dimensionality was investigated using the serial and parallel algorithms (for two kernels), as well as the example of the matrix multiplication problem of large dimension, with the help of MATLAB package.

Keywords: parallel computing, software and hardware architecture, matrix multiplication, programming in MATLAB.

Введение. В условиях перехода производителей персональных компьютеров к процессорам с многоядерной архитектурой становится возможным и логичным использование их для решения ряда вычислительных задач достаточно большой размерности. Для этого нужны так называемые алгоритмы параллельных расчетов, которые позволяют оптимально загрузить работой все ядра процессора. Параллельное программирование становится неотъемлемым инструментом каждого продвинутого программиста и исследователя.

Но существует ряд сложностей в работе с алгоритмами параллельных расчётов. Одной из важных является то, чтобы алгоритм задачи было возможно разделить на два алгоритма, которые можно решить отдельно, то есть параллельно. Сложностью и недостатком параллельного расчета является его неспособность к решению абсолютно любой задачи. Это связано с тем, что некоторые задачи не решаются методом параллельных расчётов.

Не менее важным и сложным в данном методе является создание работающей системы, которая осуществила бы работу алгоритма параллельных расчётов.

Создание программ и алгоритмов для параллельных систем вызывает большую сложность, нежели для последовательных, так как взаимодействие и синхронизация между вычислительными и информационными процессами представляют определенные проблемы для получения наибольшей производительности в алгоритмах параллельных расчётов. Благодаря параллельным методам вычисления можно значительно ускорить и оптимизировать процесс решения задачи при условии,

что для вычислений требуется гораздо больше времени, чем для отправки, получения и обработки данных.

Программа обмена сообщений (MPI) занимает позицию основной и классической в методе параллельных расчётов. Однако освоить его удается не каждому. Поэтому создатели прикладных математических пакетов, среди которых, в частности, MathWork, Inc. — разработчик пакета MATLAB, озаботились внедрением параллельных и распределенных вычислений в свои пакеты [1, 2]. Что же реализовано к настоящему времени? Цель данной работы — провести анализ существующих технологий и методов организации параллельных вычислений, исследовать зависимость времени решения задач от размерности при использовании последовательного и параллельного алгоритмов.

Технология CUDA. CUDA (Compute Unified Device Architecture) — программно-аппаратная архитектура параллельных расчётов, с помощью которой реализуется заметное повышение производительности ЭВМ за счёт использования графических процессоров GPU фирмы Nvidia. Первоначальная версия CUDA была продемонстрирована в 2007 году [3]. CUDA SDK способна создавать алгоритмы на упрощённом языке Си, параллельно организованные на процессорах Nvidia. Устройство программно-аппаратной архитектуры оставляет возможность изменять доступ к графическому ускорителю и работать с его памятью.

Технология MPI. MPI (Message Passing Interface) — информационно-передающая технология, позволяющая осуществлять связь между активными процессами, которые выполняют алгоритмы одной и той же задачи [4]. Стоит заметить, что при использовании алгоритма параллельных расчётов нередко используют именно интерфейс MPI, поэтому он считается распространённым.

Его исполнение можно найти на различных языках программирования, таких как Фортран 77/90, Java, Си, Си++ и Python. Чаще всего он используется при создании программ для систем компьютеров и суперкомпьютеров. В MPI общение процессов происходит за счёт обмена информации между процессами. Необходимые данные передаются от одного процесса к другому.

В сообщении находится пакет с основными данными (полезные данные) и дополнительными. Дополнительные данные необходимы для идентификации данных и дальнейшего их использования. Также существуют:

— отправитель — ранг (номер в группе) отправителя сообщения;

— получатель — ранг получателя;

— признак — может использоваться для разделения различных видов сообщений;

— коммуникатор — код группы процессов.

В результате обмена данными процесс может быть блокирующимся и неблокирующимся. Неблокирующийся процесс служит для определения готовности к совершению операции и готовности к начатию данной операции.

Существует ещё один вид обмена данными процессов. Это удалённое управление памятью (RAM). Благодаря этому виду связи появляется возможность удалённого считывания или изменения памяти процесса. Процесс, действующий локально, действует с памятью в обе стороны, то есть он может некоторую область памяти переносить в свою и возвращать назад. Также, например, путем суммирования локальный процесс может по-разному сочетать данные для передачи их в удалённый процесс. После выполнения удалённых операций, которые являются неблокирующимися, следует вызывать блокирующиеся функции синхронизации до и после выполнения операции (использование MPI вызывает затруднение у многих прикладных программистов).

Параллельные вычисления в MATLAB. К последним версиям MATLAB (например, R2014a) подключен набор инструментов Parallel Computing Toolbox. В нём созданы инструменты для алгоритмов параллельных расчётов, такие как параллельные числовые алгоритмы, параллель-

ные циклы, функции передачи сообщений, распределенные массивы. Благодаря им в среде MATLAB можно создать программы, которые не нуждались бы в программировании на аппаратном уровне и использовании сетевых архитектур (табл.1). Таким образом, с изменением приложения с последовательным алгоритмом в приложение с параллельным алгоритмом в MATLAB происходит минимальное изменение исходного кода. Стоит отметить, что для этой операции глубоких знаний о CUDA или MPI не требуется и даже можно не задействовать язык программирования низкого уровня [5].

Таблица 1

Основные функции параллельного MATLAB

parfor	Создает параллельный расчёт
pmode	Запуск Parallel Command Window
labReceive	Принятие информации
labSend	Передача информации
labSendReceive	Синхронная передача и принятие информации
pload	Перемещение файла в параллельный цикл
psave	Закрепление данных из параллельного расчёта
labProbe	Проверка лаборатории на предмет приёма данных

Основная функция Toolbox — параллельный цикл *parfor*. Часть программы выполняется на Клиенте MATLAB, а другая — на Рабочих MATLAB. Клиент MATLAB — это устройство с активным *parfor*. Рабочий MATLAB — это процесс MATLAB, осуществляющий расчёт. Необходимые данные, анализируемые *parfor*, отправляют от Клиента к Рабочим. В этом блоке выполняется большая часть вычислений. После этого вычисленные значения посылаются обратно Клиенту и группируются. Рабочие MATLAB совершают операции независимо друг от друга. Так как операции совершаются независимо, то итерации не всегда могут быть синхронизованы, но для простых задач нет необходимости в синхронизации итераций. Однако при количестве операций, превосходящих Рабочих, некоторые Рабочие совершают сразу несколько итерации.

Стоит заметить, что благодаря функции *pmode* можно обратиться к процессам Рабочих непосредственно из командного окна Parallel Command Window. *Pmode* является своеобразным диалоговым окном для каждой лаборатории, совершающей операции. Здесь возможно просмотреть результаты, вводить команды, обращаться к рабочему пространству каждой **Лаборатории** (под понятием Лаборатория в MATLAB понимается объединение нескольких Рабочих) и т.д.

Parallel Computing Toolbox предоставляет до 12 локальных Рабочих для запуска приложений локально на многоядерном ПК; без изменения кода то же самое приложение можно запускать на **Кластере** (cluster — это несколько компьютеров, соединенных по сети и предназначенных для общей цели) или сервисе распределённых вычислений (с помощью MATLAB Distributed Computing Server). Параллельные приложения можно запускать интерактивно или в пакетном режиме. Планировщик (scheduler) задает очереди и назначает задания Рабочим.

Оценка эффективности параллельных расчетов. Рассмотрим задачу умножения матриц. Ее алгоритм (умножение «строка на столбец») допускает распараллеливание. В табл. 2 представлены коды алгоритмов последовательного и параллельного перемножение матриц в MATLAB.

Таблица 2

Последовательное и параллельное перемножение матриц

Последовательный цикл	Параллельный цикл
clear; n = 700; tic; for i = 1:n M = magic(n); R = rand(n); A(i) = sum(M(i,:).*R(n+1-i,:)); end toc	matlabpool open local 2 clear; n = 700; tic; parfor i = 1:n M = magic(n); R = rand(n); A(i) = sum(M(i,:).*R(n+1-i,:)); end toc matlabpool close

Здесь встроенная функция **magic(n)** создаёт определённую матрицу, которая имеет одинаковое количество символов в строке и столбцах, то есть квадратная, **M** размера $n \times n$, элементами которой являются целые числа, и суммы элементов которой по строкам и столбцам равны, а функция **rand(n)** формирует квадратную матрицу **R** размера $n \times n$, элементами которой являются случайные величины, распределенные по равномерному закону в интервале (0, 1). Функция **tic** начинает, а функция **toc** заканчивает отсчет времени.

Первым этапом в осуществлении кода, который имеет в себе параллельный цикл *parfor*, нужно запустить следующую функцию в MATLAB:

matlabpool open local 2

Она позволяет закрепить для повторов цикла определённое количество компьютеров — **Рабочих**, которые выполняют вычисления (в данном примере их два). По окончании работы с вычислениями необходимо запустить следующую функцию: matlabpool close.

Расчеты проводились при различных размерностях матриц **n** от 10 до 1000. Результаты приведены в табл. 3 и отображены на графике (рис. 1).

Таблица 3

Зависимость времени решения от размерности матриц

n	**Последовательное умножение, сек.**	**Параллельное умножение, сек.**
10	0.00	1.42
30	0.01	1.24
50	0.02	1.33
100	0.37	1.43
200	2.35	3.5
300	9.23	7.64
400	22.13	17.75
500	44.12	36.10
700	136.18	108.43
1000	355.06	285.11

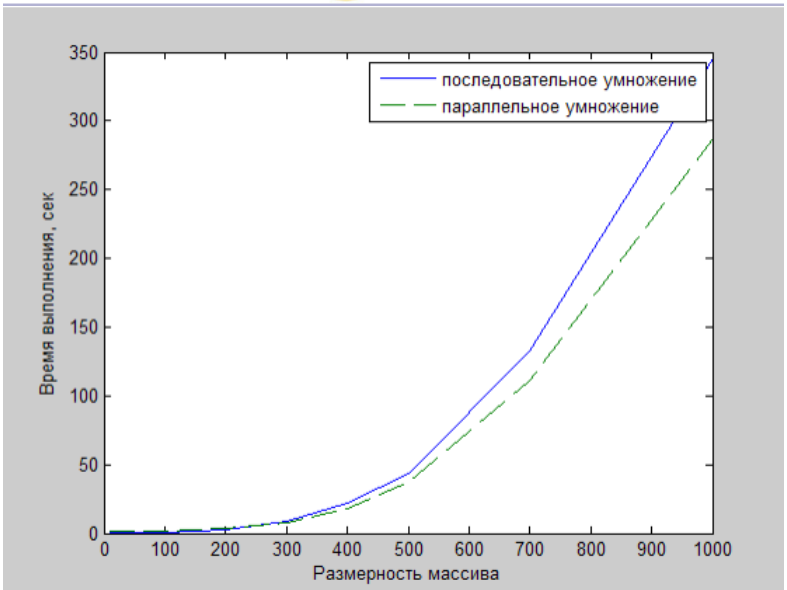

Рис.1. График зависимости времени выполнения от размерности перемножаемых матриц.

Проанализировав результаты работы алгоритмов, можно сделать вывод, что эффективность алгоритма параллельных вычислений зависит от размерности матрицы. Это значит, что в данной задаче наиболее эффективен параллельный метод, но с оговоркой на то, что значения матрицы будут браться от 300. Однако последовательный метод оказывается эффективен в значениях матрицы от 0 до 300.

Таким образом, при малых размерах матриц параллельные вычисления нецелесообразны. Это нужно учитывать при решении конкретных задач.

Выводы. На ПК, имеющих процессоры с большим количеством матриц, можно решать задачи достаточно большой размерности, допускающие распараллеливание, таких как:

— решение систем линейных алгебраических уравнений (СЛАУ),

— вычисление кратных интегралов,

— численное решение систем обыкновенных дифференциальных уравнений (ДУ) и уравнений в частных производных,

— моделирование методом Монте-Карло и другие.

Библиографический список

1. Распределенные и параллельные вычисления / Введение [Электронный ресурс] / Викиучебник. — Режим доступа: http://ru.wikibooks.nym.su/wiki/Распределенные_и_ параллельные_вычисления/Введение (дата обращения: 10.02.18).

2. Введение в параллельные вычисления [Электронный ресурс] / ИНТУИТ. — Режим доступа: https://www.intuit.ru/studies/courses/5938/1074/lecture/16443 (дата обращения: 10.02.18).

3. CUDA [Электронный ресурс] / Википедия. — Режим доступа: https://ru.wikipedia.org/wiki/CUDA (дата обращения: 11.02.18).

4. Message_Passing_Interface [Электронный ресурс] / Википедия. — Режим доступа: https://ru.wikipedia.org/wiki/Message_Passing_Interface (дата обращения: 11.02.18).

5. Что такое вычисление на GPU [Электронный ресурс] / NVIDIA. — Режим доступа: http://www.nvidia.ru/object/tesla-matlab-accelerations-ru.html (дата обращения: 13.02.18).