Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

Распараллеливание циклов разного уровня вложенности и их анализ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Тимонина Артёма Вячеславовича

Научный руководитель:
доцент кафедры ИиП, Кабанова Л.В.
к.э.н., доцент
_____
подпись, дата

Зав. кафедрой:
к.ф.-м.н. Огнева М.В.
_____
подпись, дата

Саратов 2017

ВВЕДЕНИЕ

Популярные системы параллельного программирования для работы на одном компьютере, начали свое развитие сравнительно недавно. Ведь при разработке приложений перед разработчиком стает вопрос, как увеличить быстродействие с ресурсами которые имеются под рукой, а именно применением нескольких ядер компьютера, и к программисту на помощь приходит API Open MP. Разработанная в 1997 году технология OpenMP предоставляет в максимальной степени эффективно реализовать возможности многопроцессорных вычислительных систем с общей памятью, гарантируя использование общих данных для параллельно выполняемых потоков без каких-либо трудоемких межпроцессорных передач сообщений.

Одним из важных достоинств технологии OpenMP является реализация идеи «инкрементального программирования», когда разработчик постепенно находит участки в программе, содержащие ресурс параллелизма, с помощью предоставляемых механизмов делает их параллельными и затем переходит к анализу следующих участков. Таким образом данный подход упрощает процесс адаптации последовательных программ к параллельным ЭВМ.

OpenMP может использоваться совместно с другими технологиями параллельного программирования, например, с MPI. Обычно в этом случае MPI используется для распределения работы между несколькими вычислительными узлами, а OpenMP затем используется для распараллеливания на одном узле [5].

MPI – это библиотека функций, предоставляющая взаимодействие параллельных процессов с помощью механизма передачи сообщений. Библиотека содержит множество функций передачи сообщений типа точка-точка, расширенный набор функций для выполнения совместных операций и регулирование процессами параллельного приложения.

Цель бакалаврской работы – реализовать распараллеливание задачи с циклами с использованием технологий OpenMP и MPI, провести сравнительный анализ последовательного и параллельного алгоритма при работе с различными типами данных и различными итерациями.

Поставленная цель определила **следующие задачи**:

1. Рассмотреть технологию OpenMP и MPI для распараллеливания циклов.
2. Реализовать задачу распараллеливания нахождения суммы медленно сходящегося бесконечного произведения.
3. Сравнить последовательный и параллельный вариант с использованием различных типов данных.
4. Сравнить последовательный и параллельный вариант при различных вариантах итераций.

Теоретическая значимость состоит в рассмотрении технологий OpenMP и MPI для распараллеливания циклов.

Практическая в реализации задачи распараллеливания нахождения суммы медленно сходящегося бесконечного произведения.

Структура и объём работы Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и двух приложений. Общий объем работы – 52 страницы, из них 37 страниц – основное содержание, включая 8 рисунков и 7 таблиц, цифровой носитель в качестве приложения, список использованных источников информации – 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первая глава «Теоретические основы технологий для распараллеливания циклов**»** посвящен технологиям OpenMp и MPI.

Под параллельной программой в рамках OpenMP понимается программа, для которой в специально указываемых при помощи директив местах – параллельных фрагментах – исполняемый программный код может быть разделен на несколько раздельных командных потоков (threads). В общем виде программа представляется в виде набора последовательных (однопотоковых) и параллельных (многопотоковых) участков программного кода. Важно отметить, что разделение вычислений между потоками осуществляется под управлением соответствующих директив OpenMP. Равномерное распределение вычислительной нагрузки – балансировка (load balancing) – имеет принципиальное значение для получения максимально возможного ускорения выполнения параллельной программы [1,2].

Для выделения параллельных фрагментов программы следует использовать директиву parallel:

#pragma omp parallel [<параметр> …]

<блок_программы>

Для блока (как и для блоков всех других директив OpenMP) должно выполняться правило «один вход – один выход», т. Е. передача управления извне в блок и из блока за пределы блока не допускается.

Директива parallel является одной из основных директив OpenMP. Правила, определяющие действия директивы, состоят в следующем:

Когда программа достигает директиву parallel, создается набор (team) из N потоков; исходный поток программы является основным потоком этого набора (master thread) и имеет номер 0.

Программный код блока, следующий за директивой, дублируется или может быть разделен при помощи директив между потоками для параллельного выполнения.

В конце программного блока директивы обеспечивается синхронизация потоков – выполняется ожидание окончания вычислений всех потоков; далее все потоки завершаются – дальнейшие вычисления продолжает выполнять только основной поток.

Под параллельной программой в рамках MPI понимается множество одновременно выполняемых процессов. Процессы могут выполняться на разных процессорах, но на одном процессоре могут располагаться и несколько процессов (в этом случае их исполнение осуществляется в режиме разделения времени). В предельном случае для выполнения параллельной программы может использоваться один процессор – как правило, такой способ применяется для начальной проверки правильности параллельной программы.

Каждый процесс параллельной программы порождается на основе копии одного и того же программного кода. Данный программный код, представленный в виде исполняемой программы, должен быть доступен в момент запуска параллельной программы на всех используемых процессорах. Исходный программный код для исполняемой программы разрабатывается на алгоритмических языках C или Fortran с применением той или иной реализации библиотеки MPI.

Синтаксис функции инициализации MPI_Init на языке C выглядит следующим образом:

int MPI_Init(int *argc, char ***argv), где

- argc — указатель на количество параметров командной строки,

- argv — параметры командной строки, применяемые для инициализации среды выполнения MPI программы.

Параметрами функции являются количество аргументов в командной строке и адрес указателя на массив символов текста самой командной строки. Последней вызываемой функцией MPI обязательно должна являться функция: int MPI_Finalize(void). Функция закрывает все MPI-процессы и ликвидирует все области связи. Как результат, можно отметить, что структура параллельной программы, разработанная с использованием MPI, должна иметь следующий вид:

```
#include<mpi.h>
int main(int argc,char **argv)
{  int rank, size;
 //Инициализация работы с MPI
 MPI_Init(&argc, &argv);
//Завершение работы с MPI
 MPI_Finalize();
 return 0; }
```

Следует отметить:

- файл mpi.h содержит определения именованных констант, прототипов функций и типов данных библиотеки MPI;

- функции MPI_Init и MPI_Finalize являются обязательными и должны быть выполнены (и только один раз) каждым процессом параллельной программы;

• перед вызовом MPI_Init может быть использована функция MPI_Initialized для определения того, был ли ранее выполнен вызов MPI_Init, а после вызова MPI_Finalize – MPI_Finalized аналогичного предназначения [6].

Проблемы распараллеливания циклов

Одна из основных задач, возникающих при параллельных вычислениях, связана с распараллеливанием циклов. При распараллеливании все итерации цикла могут выполняться одновременно, могут выполняться в произвольном порядке. Понятно, что не всякий цикл может быть распараллелен, - итерации должны быть независимыми, множества переменных, изменяемых на каждой итерации, не должны пересекаться, дабы избежать проблем с гонкой данных, блокировками, синхронизацией.

Предположим, что цикл допускает распараллеливание. Насколько просто его организовать? Возникают ли какие-нибудь проблемы, связанные с распараллеливанием? Ответ напрашивается – возникают. Отметим несколько серьезных проблем, которые приходится решать в каждом конкретном случае:

- **Накладные расходы.** Когда каждая итерация цикла выполняется в отдельном потоке, то расходы, связанные с привлечением потока (расходы памяти и времени) являются накладными расходами. Если доля накладных расходов мала в сравнении с выигрышем по времени, которое получается за счет параллельного выполнения, то ими можно пожертвовать. Но для «коротких» циклов, где число операций, выполняемых на каждой из итераций, мало, нужно прилагать особые усилия для уменьшения накладных расходов.

- **Балансировка нагрузки**. Итерация итерации рознь. Может оказаться, что некоторые итерации выполняются дольше, чем остальные. В этом случае необходимо предпринимать специальные меры для балансировки нагрузки на используемые потоки.

- **Управление итерациями**. При выполнении одной или нескольких итераций могут возникать условия, требующие завершения цикла. Причины досрочного завершения могут быть разными – достижение требуемого результата, обнаружение ситуации, при которой продолжение цикла должно быть прервано, возникновение исключительной ситуации, прерывающей выполнение итерации. Во всех этих случаях нужно разумным способом завершить уже выполняемые итерации и не порождать выполнение итераций, еще не начавших свое выполнение [8,9].

Вторая глава «Реализация алгоритма бесконечного произведения**»**

Реализовать программу, иллюстрирующую ускорение нахождения частичной суммы медленно сходящегося бесконечного произведения

$$\prod(x) = \prod_{k=1}^{\infty}(1 - \frac{k}{2k^3 - x^2}) \approx \prod_{k=1}^{100000}(1 - \frac{k}{2k^3 - x^2})$$

при помощи технологии OpenMP и MPI. Произвести сравнительный анализ последовательной и параллельной версии. Так же, сравнить 3 типа данных: double, long double и float.

Эксперименты проводились на вычислительном узле на базе процессора Intel® Core™ i7 3630QM, 2.40 ГГц, , 8 Гб RAM под управлением операционной системы Microsoft Windows 10. Разработка программ проводилась в среде Microsoft Visual Studio 2015, для компиляции использовался C++.

В ходе экспериментов было выявлено, что последовательный алгоритм с использованием типа данных long double работает за 77,6867 секунд, а его параллельная версия за 19,3754 секунд, так мы получаем ускорение в 4 раза. При использовании управлении итерациями static и типа данных double время работы составило 19,6867 секунд. При использовании управлении итерациями dynamic и типа данных double время работы составило 10,3452 секунд. При использовании управлении итерациями dynamic и типа данных guided время работы составило 10,2288 секунд. Самый быстрый способ управления распределения итераций цикла между потоками guided. При использовании технологии OpenMP в последовательной и параллельной версии алгоритма, было выявлено, что порядок итераций не влияет на результат. Были выведены четные и не четные итерации, которые также не влияют на результат.

При использовании технологии MPI, результаты показали, что самый быстрый тип данных float, который работает в последовательном варианте за 64,98 секунд, а в параллельном 9,57. Наиболее долгим типом данных является long double, который в последовательном варианте работает за 85,6867 секунд, а в параллельном 11,3754 секунды. При различных вариантах итераций, результат не изменился.

ЗАКЛЮЧЕНИЕ

В ходе данной работы, целями которой являлось реализовать распараллеливание задачи с циклами с использованием технологии OpenMP и MPI, провести сравнительный анализ последовательного и параллельного алгоритма при работе с различными типами данных и различными итерациями. Были выполнены следующие задачи:

Рассмотрена технология OpenMP и MPI для распараллеливания циклов.

Реализована задача распараллеливания нахождения суммы медленно сходящегося бесконечного произведения.

Произведен сравнительный анализ последовательной и параллельной программы с использованием различных типов данных.

Произведен сравнительный анализ последовательной и параллельной программы при различных вариантах итераций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Антонов А.С. Параллельное программирование с использованием технологии OpenMp. – М.: Изд-во МГУ, 2009.

2 Бахтин А.В. Технология параллельного программирования OpenMP, Москва, 2012.

3 Гергель В.П. Параллельное программирование с использованием OpenMP, Новосибирск, 2007.

4 Д.К. Андрейченко, В.М. Велиев, А.А. Ерофтиев, М.С. Портенко Теоретические основы параллельного программирования, Саратов 2014.

5 Жалнин Р.В., Панюшкина Е.Н., Пескова Е.Е., Шаманаев П.А. Основы параллельного программирования с использованием технологий MPI и OpenMP. Саранск, 2013.

6 Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI. – Минск:БГУ, 2002.

7 Букатов А.А., Дацюк В.Н., Жегуло А.И. Многопроцессорные системы и параллельное программирование. – Ростов-на-Дону: РГУ, 2000.

8 Немнюгин С.А. Средства программирования для многопроцессорных вычислительных систем. – СПб: СПбГУ, 2007

9 Г. Р. Эндрюс. Основы многопоточного, параллельного и распределенного программирования / Г.Р. Эндрюс – Вильямс, 2003

10 Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. - 608 с.

11 Антонов А.С. Параллельное программирование с использованием технологии MPI: Учебное пособие. -М.: Изд-во МГУ, 2004. - 71 с.

12 Ильин В. А. и др. Математический анализ. Продолжение курса. – М., 1987.

13 Кудрявцев Л. Д. Курс математического анализа. – М., 1988.

14 Фихтенгольц Г. М. Курс дифференциального и интегрального исчисления, том 2. – М., 1970.

15 А.М. Сальников, Е.А. Ярошенко, О.С. Гребенник, С.В. Спиридонов. Введение в параллельные вычисления. Основы программирования на языке Си с использование интерфейса MPI. – М., 2009.

16 Гришагин В.А., Свистунов А.Н. Параллельное программирование на основе MPI: Учебное пособие – Нижний Новгород: Изд-во ННГУ им. Н.И. Лобачевского, 2005. – 93 с.

17 Алексеев А.Г.,Йовенко А.Р. Параллельное программирование. Учебное пособие, Чебоксары, 2014 – 28 с

18 Жалнин Р.В., Панюшкина Е.Н., Пескова Е.Е., Шаманаев П.А. Основы параллельного программирования с использованием технологий MPI и OpenMp – Изд-во СВМО Саранск, 2013, 7с.

19 Малышкин В.Э. Параллельное программирование мультикомпьютеров, Новосибирск, 2006. – 452 с

20 Барский А.Б., Параллельные процессы в вычислительных системах: планирование и организация,М.:1990