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

Кафедра информатики и программирования

Распараллеливание алгоритмов сортировки 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
Бодрова Александра Андреевича

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

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

Саратов 2017

Введение. Идея параллельных вычислений возникла практически одновременно с появлением ЭВМ. Первые шаги в этом направлении были сделаны еще в конце 50-х годов прошлого века. Первоначально эти идеи реализовались лишь в передовых для своего времени суперкомпьютерах, а сегодня уже не редкость применение многоядерного процессора в сотовом телефоне. Большинство серийно выпускаемых процессоров для персональных ЭВМ также являются многоядерными.

Параллельные вычисления имеют свою специфику и создание эффективных параллельных программ невозможно без изучения основных принципов их построения и работы. Эти знанияявляются необходимыми для современного специалиста в области информационных технологий. Математическое моделирование, искусственный интеллект, статистическая обработкаданных, компьютерная обработка изображений и видео – вотлишь несколько областей, где использование параллельных вычислений помогает решать задачи более эффективно.

Множество проблем параллельного программирования до сих пор не нашли хорошего решения в современных теориях, языках, системах и технологиях параллельного программирования. По этой причине параллельная реализация алгоритмов наталкивается на значительные трудности. Разработка, отладка и сопровождение параллельных программ оказываются весьма сложным делом.

Наиболеераспространены сейчас системы параллельного программирования на основе MPI (MessagePassingInterface). Идея MPI предполагает представление параллельной программы в виде множества параллельно исполняющихся процессов (программы процессов обычно разработаны на обычном последовательном языке программирования C или Фортран), взаимодействующих друг с другом в ходе исполнения для передачи данных с помощью коммуникационных процедур. Они и составляют библиотеку MPI.

Еще одним из наиболее популярных средств программирования для компьютеров с общей памятью, базирующихся на традиционных языках программирования и использовании специальных комментариев, в настоящее время является технология OpenMP. Разработанная в 1997 году технология OpenMP позволяет в максимальной степени эффективно реализовать возможности многопроцессорных вычислительных систем с общей памятью, обеспечивая использование общих данных для параллельно выполняемых потоков без каких-либо трудоемких межпроцессорных передач сообщений.

Сортировка является одной из наиболее часто выполняемыхопераций с набором данных. Практически ни одна программа не обходится без сортировки. Большинство алгоритмов используют сортировку как один из этапов своей работы. Учитывая распространенность задачи сортировки, желание ускорить ее решение путем использования параллельных вычислений вполнеочевидно

Цель бакалаврской работы – реализация параллельных алгоритмов Timsort, пузырьковой сортировки и сортировки слиянием на основе технологий OpenMP и MPI и анализ полученных результатов.

Поставленная цель определила **следующие задачи**:

1. Рассмотреть технологию OpenMP и MPI для распараллеливания сортировок.
2. Изучить возможности распараллеливания алгоритмов сортировки.
3. Реализовать параллельный алгоритм сортировки Timsort, пузырьковой сортировки, сортировки слиянием.
4. Сравнить время выполнения последовательного и параллельного алгоритма.
5. Рассчитать и проанализировать ускорение и эффективность.

Теоретическая значимостьсостоит в рассмотрение технологий для распараллеливания сортировок.

Практическая значимость состоит в реализации алгоритмов сортировки, Timsort, чет – нечетной перестановки и сортировки слиянием. Осуществлению вычислительных экспериментов, по расчету времени и нахождению ускорения и эффективности параллельных алгоритмов сортировки.

Бакалаврская работа состоит из введения, двухразделов, заключения, списка использованных источников и девятиприложений. Общий объем работы – 158 страниц, из них 68 страниц – основное содержание, включая 16 рисунков и 10 таблиц, цифровой носитель в качестве приложения, список использованных источников информации –20 наименований.

Перваяглава«Теоретические основы технологий для распараллеливания сортировок»посвященатехнологиям OpenMpи MPI и описанию алгоритмов сортировки.

OpenMP— механизм написания параллельных программ для систем с общей памятью. Состоит из набора директив компилятора и библиотечных функций. Позволяет достаточно легко создавать многопоточные приложения на C/C++, Fortran. Поддерживается производителями аппаратуры (Intel, HP, SGI, Sun, IBM).

MPI — это библиотека функций, обеспечивающая взаимодействие параллельных процессов с помощью механизма передачи сообщений. Библиотека включает в себя множество функций передачи сообщений типа точка-точка, развитый набор функций для выполнения коллективных операций и управления процессами параллельного приложения [12].

Сортировка слиянием

Алгоритм использует принцип “разделяй и властвуй” и состоит из трех этапов:

1. Исходная последовательность делится на две части, примерно одинакового размера.
2. Каждая из последовательностей сортируется отдельно, например этим же алгоритмом. Последовательность длиной 1 считается

отсортированной.

3. Две отсортированные последовательности соединяются в одну.

Сложность алгоритма для худшего случая составит $O(n\log_2 n)$, такая же сложность получается и в среднем. Основным недостатком алгоритма является необходимость дополнительной памяти при слиянии двух упорядоченных последовательностей, в общем случае необходима дополнительная память под N элементов [2].

Алгоритм может работать в двух вариантах:

- “сверху-вниз” (нисходящая), когда исходная последовательность делится на две равные части, каждая из которых затем также делится пополам и т. д.
- “снизу-вверх”, когда соседние элементы последовательности (нечетный с четным) объединяются в отсортированные пары, которые затем объединяются в “четверки” и т. д., пока в итоге не получится итоговая последовательность.

Параллельная версия данного алгоритма получается путем параллельного выполнения сортировок последовательностей, полученных при разбиении. Операция слияния двух последовательностей выполняется последовательно на одном процессоре. Если в нашем распоряжении имеется P процессоров, то исходную последовательность можно сразу поделить на P равных частей, отсортировать их последовательным алгоритмом на каждом из процессоров и затем объединять друг с другом, пока не получится итоговая последовательность. Особенностью данного алгоритма является возможность сортировки больших объемов данных, расположенных на внешних носителях с последовательным доступом (например, расположенных в виде файлов на жестком диске).

Timsort

Timsort— гибридный алгоритм сортировки, сочетающий сортировку вставками и сортировку слиянием, опубликованный в 2002 году Тимом

Петерсом. В настоящее время Timsort является стандартным алгоритмом сортировки в Python, OpenJDK 7 и реализован в Android JDK 1.5.

Алгоритм построен на той идее, что в реальном мире сортируемый массив данных часто содержат в себе упорядоченные (не важно, по возрастанию или по убыванию) подмассивы. Это и вправду часто так. На таких данных Timsort лучше всех остальных алгоритмов.

Timsort — это не полностью самостоятельный алгоритм, а гибрид, эффективная комбинация нескольких других алгоритмов, приправленная собственными идеями. Очень коротко суть алгоритма можно объяснить так:

1. По специальному алгоритму разделяем входной массив на подмассивы.
2. Сортируем каждый подмассив обычной сортировкой вставками.
3. Собираем отсортированныеподмассивы в единый массив с помощью модифицированной сортировки слиянием.

Рассмотрим более подробно модификацию процедуры слияния подмассивов.

Алгоритм Timsort предлагает модификацию, которую он называет «галоп». Суть в следующем:

1. Начинаем процедуру слияния, как было показано выше.
2. На каждой операции копирования элемента из временного или большего подмассива в результирующий запоминаем, из какого именно подмассива был элемент.
3. Если уже некоторое количество элементов было взято из одного и того же массива — предполагается, что и дальше придётся брать данные из него. Чтобы подтвердить эту идею, нужно перейти в режим «галопа», т.е. бежать по массиву - претенденту на поставку следующей большой порции данных бинарным поиском текущего элемента из второго соединяемого массива. Бинарный поиск эффективнее линейного, а потому операций поиска будет намного меньше.
4. Найдя, наконец, момент, когда данные из текущего массива

поставщика больше не подходят (или дойдя до конца массива), можно, наконец, скопировать их все разом.

Пузырьковая сортировка

Последовательный алгоритм пузырьковой сортировки сравнивает и обменивает соседние элементы в последовательности, которую нужно отсортировать. Для последовательности $(a_{i,}, a_j, ..., a_n)$ алгоритм сначала выполняет $n-1$ базовых операций "сравнения-обмена" для последовательных пар элементов $(a_{i1}, a_2), (a_2, a_3), ..., (a_{n-1}, a_n)$. В результате после первой итерации алгоритма самый большой элемент перемещается ("всплывает") в конец последовательности. Далее последний элемент в преобразованной последовательности может быть исключен из рассмотрения, и описанная выше процедура применяется к оставшейся части последовательности $(a_1^{'}, a_2^{'}, ..., a_{n-1}^{'})$. Как можно увидеть, последовательность будет отсортирована после $n-1$ итерации [11].

Алгоритм пузырьковой сортировки обладает достаточно большой трудоемкостью, и уже при сравнительно небольшом размере массива время, необходимое на выполнение сортировки, становится значительным.

Алгоритм пузырьковой сортировки в прямом виде достаточно сложен для распараллеливания — сравнение пар значений упорядочиваемого набора данных происходит строго последовательно. В связи с этим для параллельного применения обычно используется не сам этот алгоритм, а его модификация, известная в литературе как метод чет — нечетной перестановки (odd-eventransposition). Суть модификации состоит в том, что в алгоритм сортировки вводятся два разных правила выполнения итераций метода — в зависимости от четности или нечетности номера итерации сортировки для обработки выбираются элементы с четными или нечетными индексами соответственно, сравнение выделяемых значений всегда осуществляется с их правыми соседними элементами. Таким образом, на всех нечетных итерациях сравниваются пары $(a_1, a_2), (a_2, a_3), ..., (a_{n-1}, a_n)$ (при

четном n), а на четных итерациях обрабатываются элементы $(a_2, a_3), (a_4, a_5), ..., (a_{n-2}, a_{n-1})$. После n-кратного повторения итераций сортировки исходный набор данных оказывается упорядоченным [11].

Вторая глава«Реализация параллельных алгоритмов сортировки»посвященареализации алгоритмов сортировки Timsort, чет – нечетной перестановки и сортировки слиянием.

Результаты вычислительных экспериментов алгоритма Timsort.

Эксперимент проводился на базе процессора Intel® Core™ i5 5675®, 3.10 ГГц, , 8 Гб RAM под управлением операционной системы MicrosoftWindows 8.1. Разработка программ проводилась в среде MicrosoftVisualStudio 2015, для компиляции использовался C++.

Экспериментальный анализ проводился для сравнения последовательного и параллельного алгоритма сортировки Timsort.Произведена оценка их временной сложности.Найдена оценкаэффективностипараллельного алгоритма данной сортировки.

Производилась сортировка массива, который содержит от 100000 до 5000000 элементов, которые создаются случайным образом.Для параллельного варианта алгоритма эксперимент проводился с использованием 2-х, 4-х потоков.

Приоценки временной сложности последовательного варианта алгоритма Timsort, разница во времени между сортировкой 100000 элементов массива,которая выполняется за 0,022 секунды и 5000000, которая выполняется за 1,401 была незначительной – 1,3 секунды.

Распараллеливание сортировки Timsort с помощью технологий OpenMpи MPI, показало, при оценке их временной сложности, что параллельный вариант программы, реализованный с помощью технологии MPI, показывает лучшее время при сортировках массива, но разница в результатах не является большой и не превышает 1 секунды.

Результаты вычислительных экспериментов алгоритма чет – нечетной перестановки.

Эксперимент проводился на базе процессора Intel® Core™ i5 5675®, 3.10 ГГц, , 8 Гб RAM под управлением операционной системы MicrosoftWindows 8.1. Разработка программ проводилась в среде MicrosoftVisualStudio 2015, для компиляции использовался C++.

Экспериментальный анализ проводился для сравнения последовательного и параллельного алгоритма сортировки чет – нечетной перестановки.Произведена оценка их временной сложности.Найдена оценкаэффективностипараллельного алгоритма данной сортировки.

Производилась сортировка массива, который содержит от 100000 до 5000000 элементов, которые создаются случайным образом.Для параллельного варианта алгоритма эксперимент проводился с использованием 2-х, 4-х потоков.

Приоценки временной сложности последовательного варианта алгоритма чет – нечетной перестановки, разница во времени между сортировкой 100000 элементов массива, которая выполняется за 0,036 секунды и 5000000, которая выполняется за 2,826 секунды была незначительной – 2,8 секунды.

Распараллеливание сортировки чет – нечетной перестановки с помощью технологий OpenMpи MPI, показало, при оценке их временной сложности, что параллельный вариант программы, реализованный с помощью технологии MPI, показывает лучшее время при сортировках массива, но разница в результатах не является большой и не превышает 1 секунды.

Результаты вычислительных экспериментов алгоритма сортировки слиянием.

Эксперимент проводился на базе процессора Intel® Core™ i5 5675®, 3.10 ГГц, , 8 Гб RAM под управлением операционной системы MicrosoftWindows 8.1. Разработка программ проводилась в среде MicrosoftVisualStudio 2015, для компиляции использовался C++.

Экспериментальный анализ проводился для сравнения последовательного и параллельного алгоритма сортировки слиянием.Произведена оценка их временной сложности.Найдена оценкаэффективностипараллельного алгоритма данной сортировки.

Производилась сортировка массива, который содержит от 100000 до 5000000 элементов, которые создаются случайным образом.Для параллельного варианта алгоритма эксперимент проводился с использованием 2-х, 4-х потоков.

Приоценки временной сложности последовательного варианта алгоритма сортировки слиянием, разница во времени между сортировкой 100000 элементов массива, которая выполняется за 0,035 секунды и 5000000, которая выполняется за 2,329 секунды была незначительной – 2,3 секунды.

Распараллеливание сортировки слиянием с помощью технологий OpenMpи MPI, показало, при оценке их временной сложности, что параллельный вариант программы, реализованный с помощью технологии MPI, показывает лучшее время при сортировках массива, но разница в результатах не является большой и не превышает 1 секунды.

На основании экспериментально полученных данных, можно сделать вывод, что одному процессору системы, для сортировки того же массива требуется больше времени, чем системе распределяющий работу на несколько процессоров. Следует подчеркнуть, что с увеличением числа используемых процессоров время решения задачи устойчиво сокращается, что вполне соответствует основной преследуемой цели - минимизации времени сортировки массива.

Ускорение полученного параллельного метода по отношению к наиболее эффективному последовательному является достаточно хорошим результатом.По полученным результатам видно, что параллельный алгоритм может давать хорошее ускорение, но использовать для этого множество процессов неэффективно, получается, что при увеличении числа процессоров эффективность падает.

ЗАКЛЮЧЕНИЕ

В ходе данной работы, целями которой являлось, реализация параллельных алгоритмовTimsort, пузырьковой сортировки и сортировки слиянием на основе технологийOpenMP и MPI и анализ полученных результатов.

Были выполнены следующие задачи:

- Рассмотрена технология OpenMP и MPI для распараллеливания сортировок.
- Изучены возможности распараллеливания алгоритмов сортировки.
- Реализованы параллельный алгоритм Timsort, пузырьковой сортировки, сортировки слиянием.
- Произведено сравнение времени выполнения последовательного и параллельного алгоритма.
- Произведен расчёт и анализ эффективности и ускорения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Пальян Р.Л., Гаврилов Д.А., Леус А.В., Филимонов А.В. Эффективность и ускорение параллельных программ, Московский физико-технический институт, 2011 – 3 с
2. Алексеев А.Г.,Йовенко А.Р. Параллельное программирование. Учебное пособие, Чебоксары, 2014 – 28 с
3. Андрейченко Д.К. , В.М. Велиев В.М., Ерофтиев А.А., Портенко М.С. Теоретические основы параллельного программирования, Саратов, 2015 - 100 с
4. Гергель В.П. Параллельное программирование с использованием OpenMP, Новосибирск, 2007. – 32с
5. Антонов А.С. Параллельное программирование с использованием технологии OpenMP. – М.: Изд-во МГУ, 2009. – 77 с
6. Антонов А.С. Параллельное программирование с использованием технологии MPI: Учебное пособие. – М.: Изд-во МГУ, 2004. – 71 с.
7. Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI. – Минск:БГУ, 2002.
8. Букатов А.А., Дацюк В.Н., Жегуло А.И. Многопроцессорные системы и параллельное программирование. – Ростов-на-Дону: РГУ, 2000.
9. Немнюгин С.А. Средства программирования для многопроцессорных вычислительных систем. – СПб: СПбГУ, 2007
10. M. Quinn Parallel Programming in C with MPI and OpenMP / M. Quinn - , 2003.
11. Г. Р. Эндрюс. Основы многопоточного, параллельного и распределенного программирования / Г.Р. Эндрюс – Вильямс, 2003.
12. Статья «Математическое моделирование: параллельные алгоритмы» [Электронный ресурс]. URL: http://ssd.sscc.ru/ru/content/параллельные-алгоритмы-сортировки/ (дата обращения: 17.02.2017).

13. 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.
14. 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.
15. Бурова И.Г., Демьянович Ю.К. Алгоритмы параллельных вычислений и программирование: Курс лекций. – СПб.: Изд-во С-Пб. ун-та, 2007. – 206 с
16. Барский А.Б., Параллельные процессы в вычислительных системах: планирование и организация,М.:1990
17. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб: БХВ-Петербург, 2002. – 608 с.
18. Гергель, В. П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем: Учебник. / В. П. Гергель. — М.: Издательство Московского университета, 2010. — 544 с.
19. Бахтин А.В. Технология параллельного программирования OpenMP, Москва, 2012.
20. Антонов А.С. Введение в параллельные вычисления: Методическое пособие. – М.: Изд-во Физфака МГУ, 2002.