Разработка приложения для многопроцессорных вычислительных систем с помощью библиотеки OpenMP

Лабораторная работа

Ревизия: 0.1

Последняя модификация: Vlad Kovtun

Дата последней модификации: 10.05.2010 16:36:00

История изменений

28.04.2010 – Версия 0.1. Первичный документ. Влад Ковтун

Содержание

Лабораторная работа 7. Разработка приложения для многопроцессорных вычислительных систем с помощью библиотеки OpenMP

Вопросы

- Постановка задачи.
- Методические рекомендации.

Постановка задачи

Цель

Разработать консольное приложение анализа логов web-сервера Apache с использованием параллельных вычислений с помощью библиотеки OpenMP.

Задачи

1. Ознакомиться с архитектурой библиотеки OpenMP.

2. Ознакомиться с директивами OpenMP.

3. Ознакомиться с операторами OpenMP.

4. Ознакомиться с C и C++ интерфейсом API OpenMP.

5. Подключение библиотеки OpenMP к компилятору, например Microsoft Visual C++.

6. Разработать консольное приложение для анализа логов web-сервера Apache.

6.1. На основе файлов логов web-сервера, следует сформировать следующую статистику:

- количество уникальных посетителей за каждый день.
- наиболее популярные браузеры.
- наиболее популярные ОС.
- наиболее популярные страницы (ссылки).
- наиболее активные пользователи (IP).

7. Разработать тестовое приложение, которое формирует все возможные тестовые комбинации ключей командной строки для приложения.

8. Произвести оценки различных характеристик приложения.

8.1. Оценить производительность приложения, при различном количестве поток чтения и потоков обработки. Количество потоков следует изменять от 1 до 8. Для количества потоков выбранной библиотекой автоматически.

9. Оформить функциональную спецификацию на приложение. В функциональной спецификации обязательно указать, каким образом производится тестирование на корректность.

10. Оформить отчет к лабораторной работе.

Задание

Опишем основные ключи командной строки, которые следует реализовать.

Для управления приложением (первый подход) предлагается использовать ключи командной строки:

- /id:<filedir> – полный путь к директории, в которой хранятся файлы логов для анализа. Данный параметр является обязательным, если он не указан, то происходит вывод на экран соответствующего сообщения и подсказки по использованию данного приложения.
- /if:<filenames> - перечисление имен файлов логов через пробел, которые необходимо проанализировать. Файлы обязаны находится в директории указанной с помощью ключа /id. Данный ключ позволяет осуществлять выборочный анализ файлов логов.
- /o:<filename> – полный путь к файлу, либо имя файла, который будет хранить отчет. Если данный параметр не указывается, то вывод производится на экран.

- /th:<threads count> - количество потоков, которые будут производить анализ данных из файлов. По умолчанию используется 2 потока. Для выбора количества потоков самой библиотекой следует указать -1.
- /? - вывод информации о допустимых ключах командной строки.

Пример

Анализ 3 файлов логов web-сервера Apache, результат анализа выводится в файл report.dat.

C:/>analyser.exe /id:"c:/logs/" /if: access-www.1.log access-www.2.log access-www.3.log /o:report.dat

Вывод

Во время работы приложения, рекомендуется выводить информацию о статусе приложения, а также о корректности его работы на консоль.

Другими словами, необходимо анализировать, сколько процентов из каждого файла – проанализировано.

Допускается перенаправление вывода консоли в текстовый файл, например:

C:/>analyser.exe /id:"c:/logs/" /if: access-www.1.log access-www.2.log access-www.3.log /o:statistics.dat >report.txt

Требования

- Архитектура приложения строится по модульному принципу.
- За основу принимается стандартная библиотека C++ (в случае разработки на C++).
- Для реализации параллельных вычислений следует использовать библиотеку OpenMP.
- Рекомендуется использовать защищенные ресурсы и указатели.
- Обязательным является обработка исключений.
- Исходный код обязан быть комментирован. Для C++ следует использовать нотацию doxygen [2].

Методические указания для самостоятельной работы

При подготовке к лабораторной работе необходимо:

- Ознакомиться с понятиями поддержки нескольких процессоров в ОС. Известно, что ОС Windows поддерживает Symmetric Multiprocessing (SMP) Model [1].
- Подключение библиотеки OpenMP к компилятору Microsoft Visual C++ [1].
- Ознакомиться с функциями ОС Windows, используемыми для поддержки нескольких процессоров [1].
- Ознакомиться с директивами библиотеки OpenMP [1].
- Ознакомиться с операторами OpenMP [1].
- Ознакомиться с типами данных OpenMP[1].
- Ознакомиться с переменными окружения OpenMP [1].
- Ознакомиться с функциями OpenMP [1].

Тестирование

Тестирование приложения осуществляется в несколько этапов:

- Разработка Unit Test в рамках проекта. Данный подход позволит проверить корректность реализации алгоритма.
- Воспользоваться тестовыми пакетными файлами для проверки корректности функционирования.
- Разработать простейшее приложение, которое формирует большие тестовые пакетные файлы, которые затем использовать для тестирования основного приложения.

Теоретические сведения

Функции ОС Windows для поддержки нескольких процессоров [1].

Функция	Описание
GetCurrentProcessorNumber	Возвращает число процессоров, которые используются текущим запущенным потоком во время выполнения этой функции.
GetLogicalProcessorInformation	Получает информацию логических процессорах и связанного с ними аппаратного обеспечения.
GetProcessAffinityMask	Получает маску родственности процессоров для заданного процесса и системную маску для системы.
SetProcessAffinityMask	Устанавливает маску родственности процессоров для потоков заданного процесса.
SetThreadIdealProcessor	Устанавливает предпочтительный процессор для потока.
SetThreadAffinityMask	Устанавливает маску родственности процессоров для заданного потока.

Директивы OpenMP [1].

Директива	Описание
atomic	Указывает, что область памяти будет обновляться автоматически.
barrier	Синхронизирует все потоки в одну команду; все потоки приостанавливаются, до тех пор, пока все потоки выполняют барьер.
critical	Указывает, что код может быть выполнен лишь единственным потоком (исключает параллельную обработку).
flush (OpenMP)	Указывает, что все потоки имеют один и тот же образ для всех общедоступных объектов.
for (OpenMP)	Указывает выполнять все действия внутри цикла for параллельно распределив между потоками.
master	Указывает, что только master (ведущему) потоку следует выполнять секцию программы.
ordered (OpenMP Directives)	Указывает, что код в распараллеленном цикле for следует выполнять как последовательный цикл.
parallel	Определяет параллельную область, которая является кодом, который будет выполнен параллельно несколькими потоками.
sections (OpenMP)	Идентифицирует секции кода, которые должны быть распределены между всеми потоками.
single	Позволяет указать, что секцию кода следует выполнять в одном потоке, не обязательно в master (ведущем) потоке.

threadprivate	Указывает, что переменная является личной для потока.

Операторы OpenMP [1].

Оператор	Описание
copyin	Позволяет всем потокам производить доступ к значению master (ведущего) потока, для **threadprivate** переменных.
copyprivate	Указывает, что одна или более переменных следует сделать доступной среди всех потоков.
default (OpenMP)	Указывает поведение переменной вне области видимости в параллельной области.
firstprivate	Указывает, что каждый поток обязан иметь свой собственный экземпляр переменной и что переменную следует инициализировать значением переменной, потому что она существует перед началом параллельного выполнения кода.
if (OpenMP)	Указывает следует ли циклу выполняться параллельно или последовательно.
lastprivate	Указывает что переменная заключенная в версии контекста установлена равной личной версии какого бы ни было потока выполняющего финальную итерацию (для цикла типа for) или последнюю секцию (#pragma секции).
nowait	Отменяет барьер указанный в директиве.
num_threads	Указывает число потоков в команде потоков.
ordered (OpenMP Clauses)	Обязателен для параллельного оператора for (OpenMP) если директива ordered (OpenMP Directives) использована в цикле.
private (OpenMP)	Указывает, что каждому потоку следует иметь его личный экземпляр переменной.
reduction	Указывает, что одна или несколько переменных, являющихся личными для каждого потока являются предметом для операции привдения в конце параллельной области.
schedule	Применяется для директивы for (OpenMP).
shared (OpenMP)	Указывает, что одна или несколько переменных следует сделать доступными для всех потоков.

Примеры исходного кода с использованием директив, операторов, типов данных и функций OpenMP приведены в [1, 2]

Литература

1. Microsoft Developer Network. URL: http://www.msdn.com
2. Open MP. URL: http://openmp.org

3. Формирование документации к исходному коду с помощью средства doxygen. URL: www.nrjetix.com/r-and-d/lectures