**Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования**

**"Уфимский университет науки и технологий"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Технологий параллельного программирования

**Отчет по лабораторной работе № 1,2**

**Тема:** «Параллельное вычисление суммы числового ряда»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-357 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Акмурзин М.Э. |  |  |  |
| Принял | Добровольцев А.С. |  |  |  |

**Уфа 2023**

**Цель:** на примере задачи сложения суммы ряда научиться использовать оптимизационные ключи различных компиляторов, а также инструмент для анализа вычислительных программ Intel Advisor.

**ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ**

**Компилятор Microsoft C/C++**

В состав Visual Studio включен компилятор языка C/C++ позволяющий создавать все, от простых консольных приложений, до универсальных приложений Windows, приложений Магазина Windows и компонентов .NET.

Ключи оптимизации:

/Od – отключение оптимизации (параметр по умолчанию),

/O1 – максимальная оптимизация с приоритетом к уменьшению размера кода программы,

/O2 – максимальная оптимизация с приоритетом к увеличению скорости работы программы,

/Ox – полная оптимизация.

/Qpar – автоматическое распараллеливание

/Arch:[SSE|SSE2|AVX|AVX2|AVX512] – позволяет использовать векторные инструкции

**Компилятор Intel C/C++**

Intel C++ Compiler Professional Edition раскрывает огромный потенциал следующего поколения многоядерных процессоров Intel. Professional Edition не только поставляется с широкими возможностями передовых оптимизаций компилятора, многопоточностью, и поддержкой процессора, в том числе автоматического процессора отправки, векторизации и предварительной выборки данных.

Ключи оптимизации:

/Od – отключение оптимизаций,

/O1 – оптимизация по размеру,

/O2 – максимальная оптимизация скорости работы программы,

/O3 – задействует оптимизации из /O2 и дополнительно более агрессивные методы оптимизации циклов и доступа к памяти.

/Ox – полная оптимизация

/QxHost - обеспечивает генерацию максимально современных векторных инструкций, поддерживаемых платформой

/Qparallel – автоматическое распараллеливание (происходит при выполнении определенных условий)

Инструмент Intel Advisor XE является помощником разработчика многопоточных приложений. Он дает советы разработчику по использованию многопоточных технологий в приложении, автоматизируя анализ исходного кода, необходимый для быстрого и корректного внедрения многопоточных технологий в приложении.

**ПРАКТИЧЕСКАЯ ЧАСТЬ**

**Задание:**

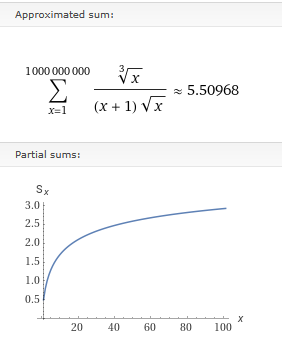
1. Написать последовательную версию программы вычисления суммы ряда, на языке C/С++. Предусмотреть замер времени выполнения основного вычислительного цикла, вывод на экран времени выполнения (в секундах) и вычисленной суммы.

Ряд:

2. Протестировать работоспособность программы при различных размерностях (N), проверить корректность путем сравнения с каким-либо интернет-сервисом, позволяющим вычислить сумму ряда.

3. Подобрать N при которых программа будет работать ~ 30 сек. Провести анализ времени ее выполнения при использовании компиляторов различных производителей и различных ключей оптимизации.

**Результат из стороннего источника.**

****

Замер времени:



Пусть N =1000000000

|  |  |
| --- | --- |
|  | t, c |
| Debug x86 | 96,392 |
| Debug x64 | 59,266 |
| Release x86 | 33,330 |
| Release x64 | 30,660 |

Таблица 1. Компилятор Microsoft

Из таблицы 1 видно, что минимальное время имеет Release x64. Поэтому в дальнейшем будем использовать именно этот режим.

|  |  |
| --- | --- |
|  | t, c |
| - Od | 57,691 |
| - O1 | 30,928 |
| - O2 | 28,978 |
| - Ox | 29,120 |

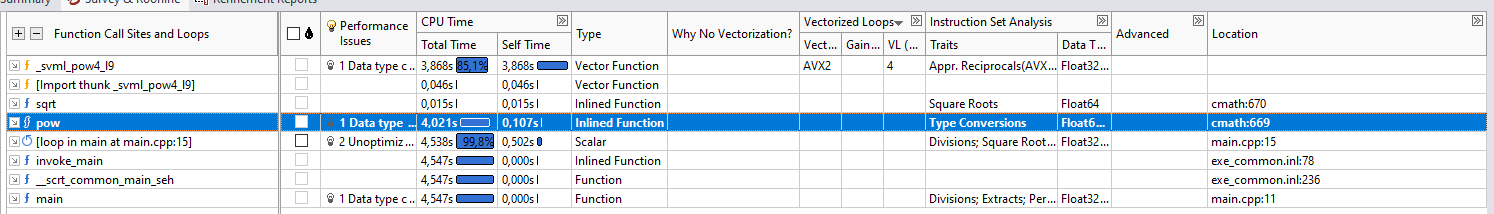
Таблица 2.

Из таблицы 2 видно, что минимальное время имеет –O2.

Для ключа /О2 используем ключ /Qpar для распараллеливания.



Воспользуемся Intel Advisor, чтобы векторизовать циклы. Получим:



Можно сделать вывод, что встроенная функция pow, а также локальная переменная sum мешают векторизации кода.

**Рассмотрим компилятор Intel Classic C/C++.**

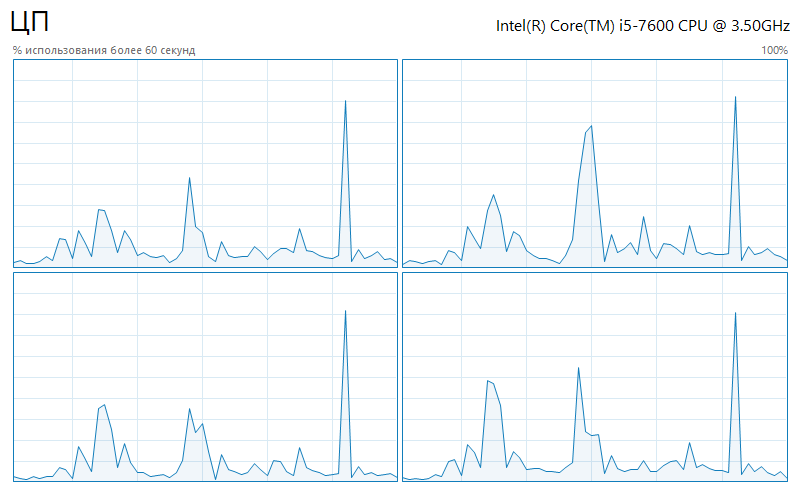
Для Release x64 рассмотрим различные ключи.

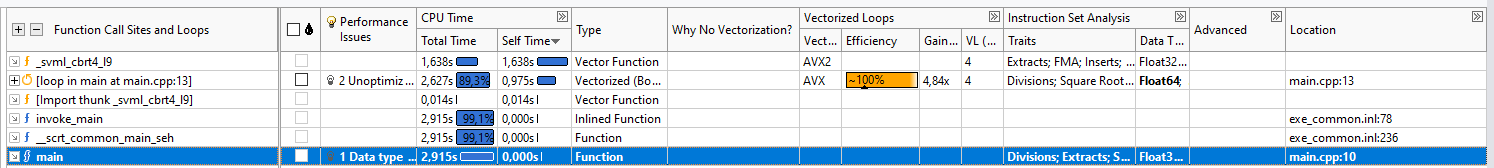
|  |  |
| --- | --- |
| Ключ | t, c |
| /Od | 22,783 |
| /O1 | 7,116 |
| /O2 | 4,290 |
| /Ox | 4,600 |
| /O3 | 4,257 |
| /O3 /QxHost | 3,091 |
| /O3/QxHost /Qparallel | 0,691 |

Таблица 4. Ключи компилятора Intel Classic C/C++

Из таблицы следует, что /O3 /QxHost /Qparallel – оптимальный вариант.

Мониторинг загрузки CPU:

 Воспользуемся Intel Advisor, чтобы векторизовать циклы. Получим:



Код векторизовался.

**Рассмотрим компилятор Intel Clang/LLVM C++.**

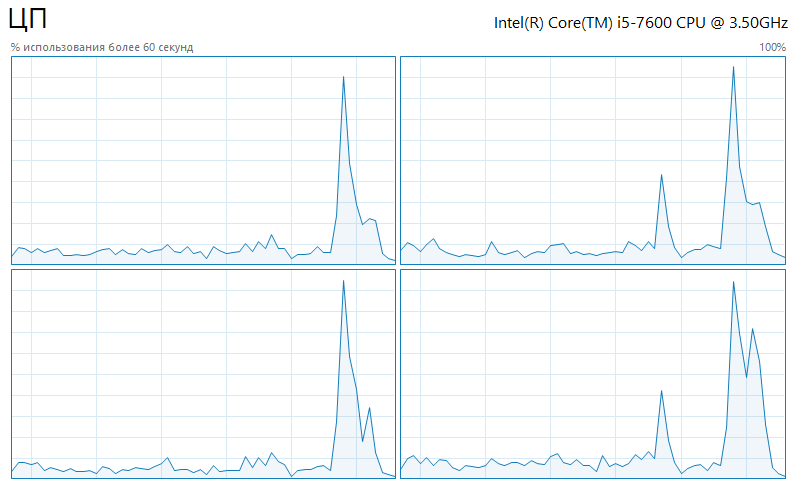
Для Release x64 рассмотрим различные ключи.

|  |  |
| --- | --- |
| Ключ | t, c |
| /Od | 19,037 |
| /O1 | 7,027 |
| /O2 | 6,765 |
| /Ox | 6,808 |
| /O3 | 6,796 |
| /O2 /QxHost | 4,493 |

Таблица 4. Ключи компилятора Intel Clang/LVVM C++

Из таблицы следует, что /O2/QxHost – оптимальный вариант.

Мониторинг загрузки CPU:



**Рассмотрим компилятор GNU C++.**

|  |  |
| --- | --- |
| Ключ | t, c |
| /O0 | 24,021 |
| /O1 | 17,569 |
| /O2 | 16,823 |
| /O3 | 16,890 |

Компилятор GNU C/C++

**Вывод**

В ходе лабораторной работы научился использовать оптимизационные ключи различных компиляторов, а также инструмент для анализа вычислительных программ Intel Advisor.

**Лабораторная работа №2**

Возьмем оптимизированную программу и лучшую последовательную конфигурацию на оптимизированной программе и подберём N такое, что работы программы ≈ 30 секунд, N =1

*Применение OpenMP.*

Подключим поддержку OpenMP и при помощи директивы *#pragma omp parallel* for распараллелим циклы

Проверим программу на ошибки, используя Intel Inspector:

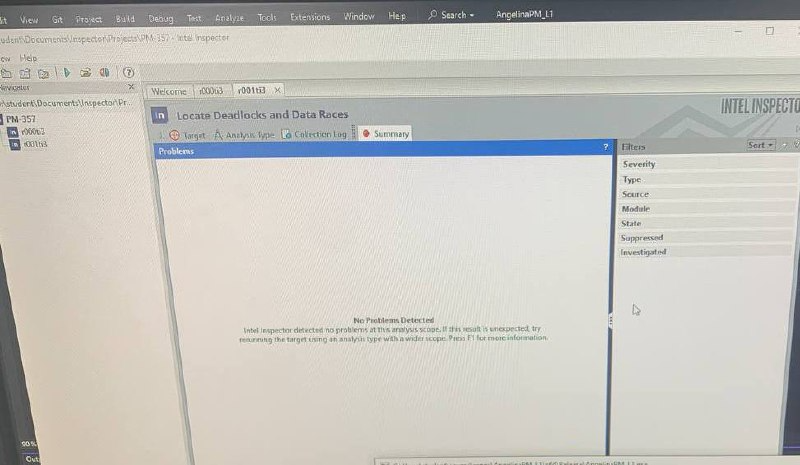


Рисунок *1*. Проверка на ошибки Intel Inspector

Посмотрим качество распараллеливания через Intel VTune:

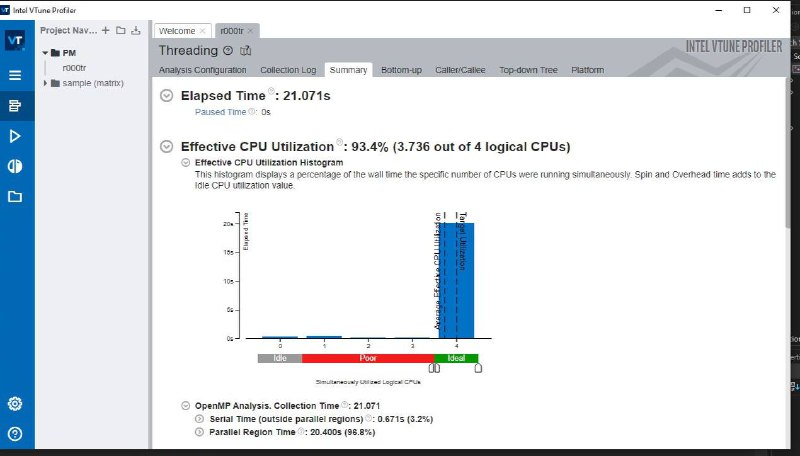


Рисунок *2*. Качество распараллеливания в Intel VTune

Перепишем программу, сохраняя корректность вычисления:

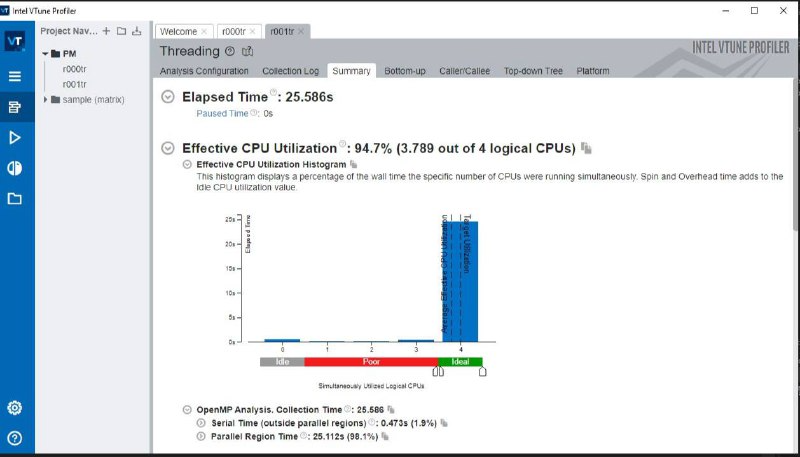


Рисунок *3*. Качество распараллеливания в Intel VTune

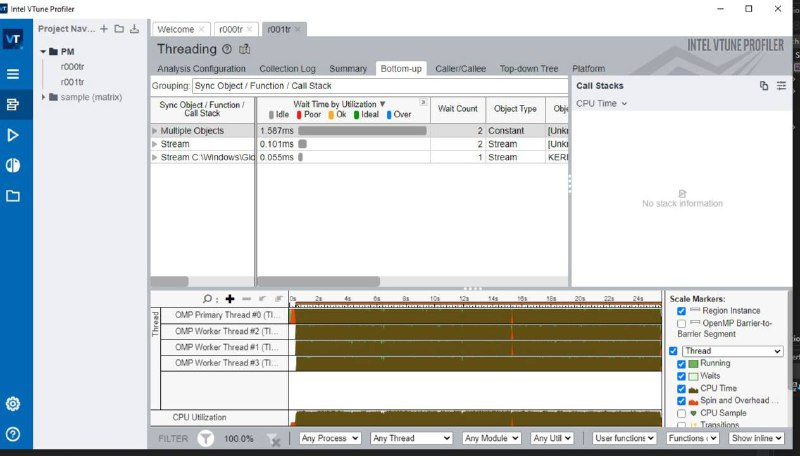


Рисунок *4*. Качество распараллеливания в Intel VTune

Заметим, что при повышении качества распараллеливания, как следствие, увеличилась скорость выполнения программы

Используя наилучшую версию программы (с лучшей конфигурацией, программа дополнительно оптимизирована для распараллеливания) оценим ускорение (S) и эффективность (E) параллельной программы при различном количестве потоков (изменяя параметры выражения omp\_set\_num\_threads() от 1 до 4 тем самым контролируя число потоков p).

|  |  |  |  |
| --- | --- | --- | --- |
| Количество потоков | Время работы | Ускорение | Эффективность |
| 1 | 29.3589 | 1 | 1 |
| 2 | 14.9824 | 1.96 | 0.98 |
| 3 | 10.0975 | 2.9 | 0.96 |
| 4 | 7.68768 | 3.8 | 0.95 |

Как видно из приведённых тестов, с увеличением количества ядер программа заметно ускоряется. Наглядно покажем ускорение (S) и его эффективность (E) от количества ядер. Для того, чтобы найти ускорение, нужно найти отношение времени выполнения параллельной программы на одном процессоре (ядре) ко времени выполнения параллельной программы на 𝑝 процессорах называется ускорением при использовании 𝑝 процессоров:

Для оценки эффективности нужно найти отношение ускорения к количествупроцессоров p:

График *1*. Оценка ускорения

График *2*. Оценка эффективности

По данным графикам видим, что при увеличении числа потоков ускорение увеличивается, а эффективность параллельной программы слабо уменьшается.

**Вывод**

В ходе выполнения лабораторной работы на примере задачи сложения суммы ряда были изучены: работа оптимизационных ключей различных компиляторов, инструмент для анализа вычислительных программ Intel Advisor, разработка простейших параллельных программ средствами OpenMP, использование инструмента для проверки корректности программ Intel Inspector и инструмента для профилирования производительности программ Intel VTune Profiler