Оптимизируем для процессоров Intel®: Как распараллелить Вашу программу?

Дмитрий Сергеев
Intel

Содержание

- Введение
- Насколько можно улучшить?
- Определение “слабого звена”
- Проблемы распараллеливания
- Проблемы вычислений

Введение

- Распараллеливание программы делается в первую очередь для улучшения производительности
- Почему все получилось “не так”
  - Много сериального кода или несбалансированное выполнение
  - Долгая синхронизация
  - Дополнительные вычисления
  - Изменилось распределение данных
  - Больше данных на поток
  - Данные не влезают в кэш
  - Увеличилась латентность памяти
- Цель
  - Быстро определить почему “не получилось”
  - Узнать, что надо исправить в и не допустить в первую очередь
  - Возможные решения уже существующих проблем

С чего начать?

- У вас есть приложение которое:
  - Имеет сериальную версию (ST)
  - Параллельную (MT)
  - Время выполнения стабильно (или иная метрика производительности)
- Предварительно сделано:
  - Тестируемая задача репрезентативна (мы оптимизируем то что актуально)
  - Скомпилировали с ключами оптимизации
  - Версия ST уже оптимизирована (векторизация)
  - Результат проверяем
  - Нет явных проблем с MT версией – дедлоки и т.д. …

Насколько можно улучшить?

- ST = 60 s
- 4-потока = 20 s

- Ускорение
  - Speedup = 60 / 20 = 3
  - Gain = 1 – 20 / 60 = 67%

- Что осталось?
  - От 20 секунд к 15 = 1 - 15 / 20 = 25%
  - От ускорения 3 к максимуму 4 = (4 – 3) / 4 = 25%
  - Обычно эти оставшиеся 25% самые сложные …

Решить – нужно ли оптимизировать дальше

Определяем “слабое звено”

Все время выполнения потоков параллельной программы можно разбить на 3 части:

- Вычисления
- Синхронизация
- Простой

Собрать профиль с использованием PTU (Vtune) и Trace Collector Analyzer

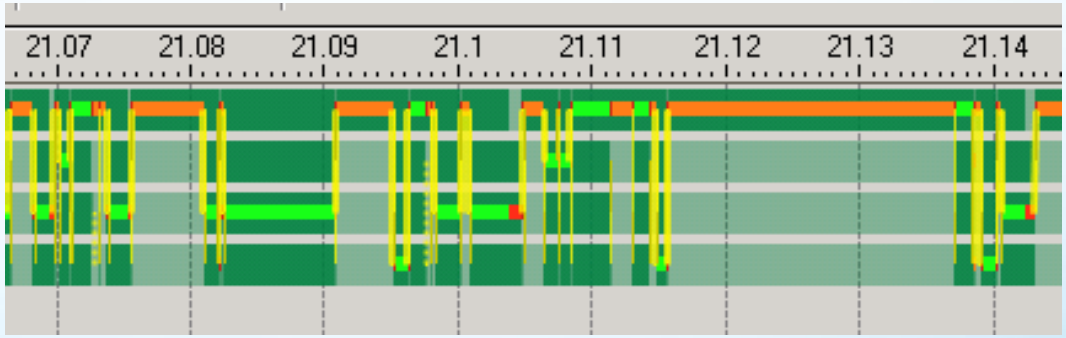
Проблемы распараллеливания:
Много синхронизации и/или простоя

Причины:

- Сериальный код
- Баланс нагрузки
- Механизм распараллеливания

Сериальный код

- Не все распараллелилось
  - Инициализация
  - Завершение; Объединение результатов
  - Части алгоритма
- Как найти
  - Thread Profiler
  - PTU - #cycles в сериальных функциях



Thread Profiler view

- Например: Если есть 10% сериального кода, то на 4 потоках максимум что мы можем достичь – 3; на 8 потоках – 4.7 (Amdahl's law)
- Решение: надо распараллелить ☺

Баланс нагрузки

- Симптом
  - Какой то поток ждёт остальных на барьере (или другом синхронизирующем объекте)
- Общие причины
  - Некоторые потоки делают больше работы чем остальные
  - Функциональная декомпозиция: функции разной длины
  - Декомпозиция по данным: несбалансировано
- Как найти
  - Thread Profiler
  - PTU™ - #cycles в функциях для каждого потока
- Решения
  - Попробовать OpenMP* полиси (распределения потоков, задач, шедулинг)
  - Очереди задач
  - Балансировать руками

Механизм распараллеливания

Чтобы определить что проблема в синхронизирующих объектах, нужно собрать:

- **MEM_UOP_RETIRED.LOCK_LOADS_NP**
- **BUS_LOCK_CLOCKS.CACHE_LOCK_DURATION**

Или используем Thread Profiler для определения проблем синхронизации

Проблема вычислений

Увеличение вычислительных циклов в MT версии против ST.

- Объем вычислений
- Изменённое разделение данных
- Больше данных на поток
- Не хватает разделяемого кэша
- Увеличилась латентность памяти

Как определить проблемы с вычислениями

Измеряем количество событий для сериальной версии и для версии на 1, 2, ... N-потоков

- Объем вычислений – **INST_RETIRED.ANY**
- Изменённое разделение данных –
  - **OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT_MESF.HIT_NO_FWD/HITM**
  - **MEM_LOAD_LLC_HIT_RETIRED.XSNP_HITM_PS**
  - **MEM_LOAD_MISC_RETIRED.LLC_MISS_PS**
  - **MEM_TRANS_RETIRED.LOAD_LATENCY_GT_32**
- Больше данных на поток
  - **MEM_LOAD_RETIRED.L2_HIT_PS**
  - **MEM_LOAD_RETIRED.L3_HIT_PS**
  - **MEM_LOAD_MISC_RETIRED.LLC_MISS_PS**
  - **L1D.REPLACEMENT**
- Не хватает разделяемого кэша
  - **MEM_LOAD_MISC_RETIRED.LLC_MISS_PS**
- Латентность памяти
  - **MEM_TRANS_RETIRED.LOAD_LATENCY_GT_128**

Объем вычислений

- Общие причины
  - Добавлен код для поддержки MT (инициализация, объединение результатов)
  - Распределение данных внутри вычислений
    - Все потоки сканируют входные данные
- Как найти
  - Сравнить количество выполненных инструкций в ST и MT версиях
  - Определить их положение в модулях, хотспотах, исходниках
- Возможные решения
  - Пытаться избегать долгих инициализаций и объединения данных
  - Перераспределить данные

Изменённое разделение данных - проблемы

- Общие причины
  - Поставщик ➔ потребитель
    - Например, в сериальной версии данные в L1D
    - Если префетч работает – никаких проблем нет
  - Ложное разделение данных
    - Один поток пишет – другие читают в/из ту же линию
    - Два потока пишут в туже линию (границы структур)
    - Один пишет, в другом префетчер подгружает эту линию

Изменённое разделение данных – как найти

- Запись:
  - **OFFCORE_RESPONSE.DEMAND_RFO.LLC_HIT_MESF.HIT_NO_FWD/ HITM –** Запись не/модифицированных данных соседними ядрами
- Загрузки:
  - **MEM_LOAD_LLC_HIT_RETIRED.XSNP_HITM_PS** – Загрузка модифицированной линии соседними ядрами.
- Или: использовать PTU Data Access view
  - Собрать **MEM_TRANS_RETIRED.LOAD_LATENCY_GT_32**
  - Исследовать линии исходного кода подсвеченные розовым

Изменённое разделение данных – как решить

- Возможные решения
  - Поставщик-потребитель
    - Потребитель: последовательный доступ и достаточное время для префетчинга
    - Потребитель: локализовать данные в LLC
    - Producer: Использовать циклический буфер > L2
  - Ложное разделение данных
    - Глобальные и статические переменные
      - Использовать вложения
      - Использовать массивы структур вместо структур массивов
    - Динамические объекты
      - Заменить/перегрузить функцию аллокации
        - Для каждого потока свой пул
        - Использовать аллокаторы из TBB
      - Для каждого потока свой диапазон выходных адресов
  - Синхронизация
    - Уменьшить количество синхронизационных точек

Больший объем данных на поток

- Общие причины
  - MT алгоритм требует больше данных
  - В каждом потоке свой буфер + аккумулирующий буфер
- Как найти
  - Сравнить сериальный и N-потоковый запуск. Стало ли больше загрузок из кэшей?
    - **MEM_LOAD_RETIRED.L2_HIT_PS**
    - **MEM_LOAD_RETIRED.L3_HIT_PS**
    - **MEM_LOAD_MISC_RETIRED.LLC_MISS_PS**
- Решение
  - Переделать алгоритм для большей локальности данных

Не хватает разделяемого кэша

- Общие причины
  - На каждый поток данных больше чем:
    (Размер кэше – размер данных)/N
    для N потоков которые используют LLC
    - Потоки вытесняют друг друга из кэша
- Как найти
  - Сравнить сериальный и N-потоковый запуск

MEM_LOAD_MISC_RETIRED.LLC_MISS_PS
/ INST_RETIRED.ANY >= 0.002

  - Стало ли отношение больше?
- Решения
  - Сериальный код: улучшить локальность данных
  - Улучшить разделение данных между потоками

Увеличение латентности памяти

- Общие причины
  - Изменённое разделение данных
  - Больший объем данных на поток
  - Не хватает разделяемого кэша
  - Траффик из памяти стал более плотным, так как общее количество циклов увеличилось
  - На NUMA может быть доступ к удалённой DRAM
- Как найти
  - Собрать **MEM_TRANS_RETIRED.LOAD_LATENCY_GT_128** с PTU
  - В “data access view” проверить, что средняя латентность загрузки существенно увеличилась
- Возможные решения
  - Сериальный код: улучшить локальность данных
  - Избегать удалённой памяти на NUMA системах
  - “Performance Analysis Guide for Processors Based on Nehalem” by David Levinthal;

http://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the "Intel® Compiler User and Reference Guides" under "Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

http://intel.com/software/products