Ускорение алгоритмов и приложений MATLAB

Сара Вейт Заранек (Sarah Wait Zaranek), Билл Чоу (Bill Chou), Гаурав Шарма (Gaurav Sharma) и Хоман Зарринкуб (Houman Zarrinkoub), компания MathWorks

В этой статье описываются техники, которые применяются для ускорения алгоритмов и приложений MATLAB®.

Затрагиваются следующие темы:

Оценка производительности кода

Внедрение практик эффективного последовательного программирования

Работа с системными объектами

Параллельные вычисления на многоядерных процессорах и графических процессорах

Генерация кода C

Каждый раздел фокусируется на определенной технике, описывает технологию, лежащую в основе ускорения, и объясняет, когда она наиболее применима. Ваш опыт программирования, тип алгоритма, который вы хотите ускорить и доступное вам оборудование могут помочь в выборе определенных техник[1].

Оценка производительности кода

Перед тем, как вносить изменения в код для увеличения производительности, требуется определить, где нужно сфокусировать усилия. Два важных инструмента, поддерживающих этот процесс, это Code Analyzer (анализатор кода) и MATLAB Profiler (профилировщик MATLAB). Code Analyzer в редакторе MATLAB проверяет ваш код прямо во время написания. Code Analyzer выявляет потенциальные проблемы и рекомендует изменения для максимизации производительности и поддерживаемости (Рисунок 1). Отчеты Code Analyzer можно запускать над целыми папками, что позволяет видеть в одном документе все рекомендации Code Analyzer для набора файлов.

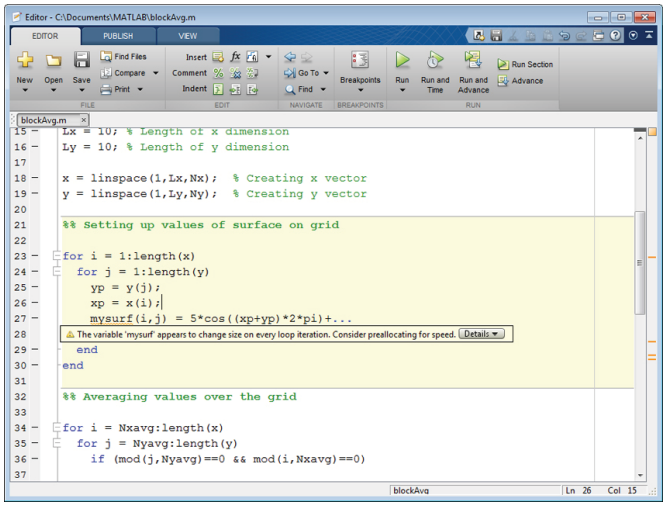

Рисунок 1. MATLAB Code Analyzer показывает рекомендуемые изменения в коде, улучшающие производительность.

Профилировщик показывает, сколько времени занимают операции в коде. Он показывает отчет, обобщающий выполнение вашего кода, включая список всех вызываемых функций, число вызовов каждой функции и общее время, проведенное в каждой функции (Рисунок 2). Профилировщик также предоставляет информацию о выполнении внутри самой функции – например, на какие строки кода приходится больше всего времени вычислений.

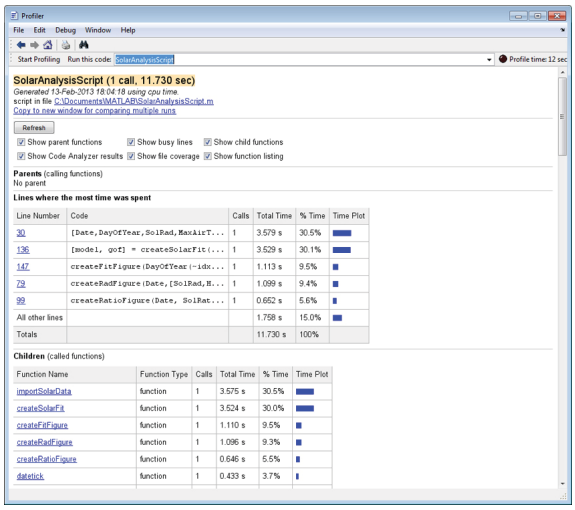

Рисунок 2. Общий отчет профилировщика.

После выявления узких мест в коде, можно сконцентрироваться на способах улучшения производительности этих определенных участков кода. По мере внедрения оптимизаций и техник ускорения вашего алгоритма, профилировщик может использоваться для измерения улучшений.

Внедрение практик эффективного последовательного программирования

Часто хорошая практика заключается в оптимизации вашего последовательного кода для производительности до того, как начинать рассматривать параллельные вычисления, генерацию кода или другие подходы. Две эффективные техники программирования для ускорения кода MATLAB это предварительное размещение (preallocation) и векторизация (vectorization).

С использованием предварительного размещения осуществляется инициализация массива окончательного размера, требуемого для этого массива. Предварительное размещение помогает избегать динамического изменения размера массивов, особенно если код содержит циклы for и while. Поскольку массивы в MATLAB находятся в непрерывных блоках памяти, повторное изменение размера массива часто требует от MATLAB траты времени на поиск большего блока памяти и последующего перемещения массива в этот блок памяти. Используя предварительное размещение массива, можно избежать этих ненужных операций и улучшить общее время выполнения.

Векторизация это процесс конвертации кода от использования циклов к использованию матричных и векторных операций. MATLAB использует процессорно-оптимизированные библиотеки матричных и векторных вычислений. В результате, часто можно увеличить производительность посредством векторизации кода.

Векторизованные вычисления MATLAB, использующие большие массивы, могут быть хорошими кандидатами для ускорения с использованием графического процессора (GPU). В случаях, когда циклы for не могут быть векторизованы, часто можно использовать параллельные циклы for (parfor) или генерацию кода C для ускорения алгоритма. Обратитесь к разделам, касающимся параллельных расчетов и генерации кода C из MATLAB для получения большей информации об этих техниках.

Узнайте больше о профилировании, предварительном распределении, векторизации и других техниках последовательного программирования для улучшения производительности.

Работа с системными объектами

Можно использовать System objects™ (системные объекты) для ускорения кода MATLAB - в первую очередь в приложениях обработки сигналов и систем связи. Системные объекты это объектно-ориентированные реализации алгоритмов MATLAB, доступные в системных тулбоксах, включая Communications System Toolbox™ и DSP System Toolbox™. Используя эти системные объекты, вы развязываете объявление (создание системного объекта) от выполнения алгоритма, заложенного в системном объекте. Такая развязка приводит к более эффективным расчетам в цикле, поскольку позволяет осуществлять обработку параметров и инициализацию только один раз. Можно создавать и настраивать экземпляры системных объектов вне цикла, а затем вызывать метод step (шаг) внутри цикла для выполнения объекта.

Большинство системных объектов в DSP System Toolbox и Communications System Toolbox реализованы в виде исполняемых файлов MATLAB (MEX-файлов). Такая реализация может ускорять симуляцию, поскольку многие алгоритмические оптимизации включены в реализации MEX этих объектов. Обратитесь к разделу по генерации кода C из MATLAB для получения большей информации о MEX-файлах.

Узнайте больше о потоковой обработке с использованием системных объектов, создании ваших собственных системных объектов и в MATLAB с использованием системных объектов.

Параллельные вычисления

Техники, описанные выше, фокусировались на способах оптимизации последовательного кода MATLAB. Также можно улучшить производительность с использованием дополнительных вычислительных мощностей. Продукты для параллельных расчетов в MATLAB предоставляют техники вычислений, позволяющие получить преимущество от многоядерных процессоров, компьютерных кластеров и графических процессоров.

Использование MATLAB на многоядерных процессорах и кластерах

Parallel Computing Toolbox™ позволяет запускать несколько работников (т.н. workers, вычислительных движков) MATLAB на многоядерных рабочих станциях. Ваши приложения можно ускорять, распределяя вычисления по этим работникам. Такой подход предоставляет больше управления над параллельными вычислениями, чем встроенная многопоточность, используемая в MATLAB. Последняя чаще используется для более грубых задач – таких, как подбор параметров или симуляции Монте-Карло. Для еще большего ускорения, параллельные приложения, использующие работников MATLAB, могут масштабироваться на компьютерный кластер или облако с использованием MATLAB Distributed Computing Server™.

Некоторые тулбоксы, включая Optimization Toolbox™ и Statistics Toolbox™, предлагают алгоритмы, которые могут использовать параллельные расчеты на нескольких работниках для ускорения вычислений[2]. В большинстве случаев параллельные алгоритмы можно использовать, просто включив опцию в настройках. Например, чтобы запустить fmincon из Optimization Toolbox параллельно, требуется выставить опцию 'UseParallel' в значение 'always'.

Parallel Computing Toolbox предоставляет высокоуровневые конструкции программирования, такие, как parfor. С использованием parfor можно ускорять циклы for в коде MATLAB путем деления итераций цикла на одновременное выполнение несколькими работниками MATLAB (Рисунок 3).

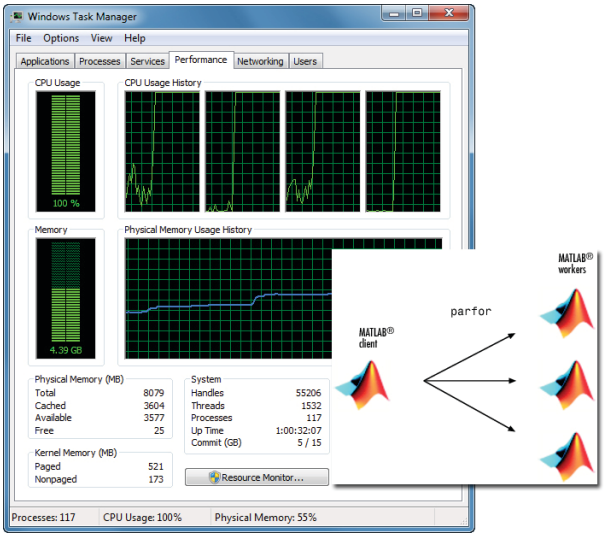

Рисунок 3. Итерации цикла parfor, распределенные между несколькими работниками MATLAB на многоядерном компьютере.

Чтобы использовать parfor, итерации цикла должны быть независимыми друг от друга. Чтобы ускорить зависимые циклы или содержащие состояния, можно попробовать организовать цикл таким образом, чтобы он стал независимым от порядка выполнения. Как вариант, можно распараллелить внешний цикл, который содержит цикл for. Если это невозможно, следует либо оптимизировать тело цикла for, либо рассмотреть возможность генерации кода C.

При передаче данных между клиентом (рабочим окружением MATLAB) и работниками MATLAB в циклах parfor, возникают накладные расходы. Это означает, что можно не получить преимуществ при использовании parfor для небольшого числа простых вычислений. Если это так, следует сосредоточиться на распараллеливании внешнего цикла for, содержащего более простой цикл for.

Команда batch может использоваться для распределения независимых наборов вычислений по работникам MATLAB для не интерактивных пакетных вычислений. Такой подход особенно полезен, когда эти вычисления занимают длительное время, и вам требуется освободить основное рабочее окружение MATLAB для другой работы.

Использование графических процессоров

Изначально используемые для обработки графики, графические процессоры (GPU) могут также применяться для научных расчетов при обработке сигналов, финансовых вычислений, производстве энергии и других областях. Можно осуществлять вычисления на графических процессорах NVIDIA напрямую из MATLAB. FFT, IFFT, функции линейной алгебры и более 100 встроенных функций MATLAB могут выполняться непосредственно на GPU. Эти перегруженные функции работают на GPU или на центральном процессоре (CPU), в зависимости от типа данных передаваемых аргументов. При передаче входного аргумента типа GPUArray (специальный тип массива, доступный в Parallel Computing Toolbox), эти функции автоматически будут выполняться на GPU (Рисунок 4). Некоторые тулбоксы, включая Communications System Toolbox и Signal Processing Toolbox™, также предлагают алгоритмы, выполняющиеся на GPU.

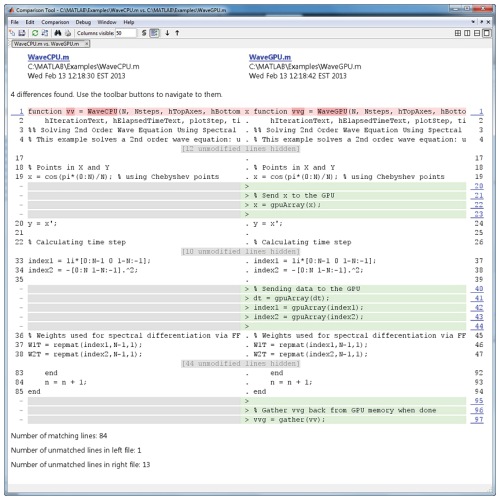

Рисунок 4. Последовательный код (слева) и код, запускаемый на GPU (справа) с использованием перегруженных функций для GPUArray.

Два общих правила помогут вам понять, что ваша вычислительно интенсивная задача хорошо подходит для GPU. Во-первых, вы увидите лучшую производительность на GPU, когда все ядра заняты вычислениями, используя свойственную для GPU параллельную природу. Код, использующий векторизованные вычисления MATLAB над большими массивами или поддерживающие GPU функции из тулбоксов попадают в эту категорию. Во-вторых, время, требуемое для работы алгоритма на GPU, должно быть значительно больше, чем время, требуемое для передачи данных между CPU и GPU во время выполнения алгоритма.

Для более продвинутого использования GPU, если вы знакомы с программированием CUDA, вы можете запускать существующие ядра CUDA на GPU напрямую из MATLAB. Затем вы можете использовать возможности анализа данных и визуализации в MATLAB, при этом сохраняя непосредственный контроль над вашим алгоритмом GPU.

Узнайте больше об использовании parfor и batch, о запуске MATLAB на многоядерных и многопроцессорных машинах, GPU вычислениях в MATLAB и тулбоксах, содержащих встроенные параллельные и GPU алгоритмы.

Генерация кода C из кода MATLAB

Замена частей вашего кода MATLAB на автоматически сгенерированный исполняемый файл MATLAB (MEX-функцию) может привести к ускорению. Используя MATLAB Coder™, можно сгенерировать читаемый и портируемый код C и скомпилировать его в MEX-функцию, которая заменит эквивалентную секцию вашего алгоритма MATLAB (Рисунок 5). Также можно воспользоваться преимуществами многоядерных процессоров, генерируя MEX-функции и конструкции parfor.

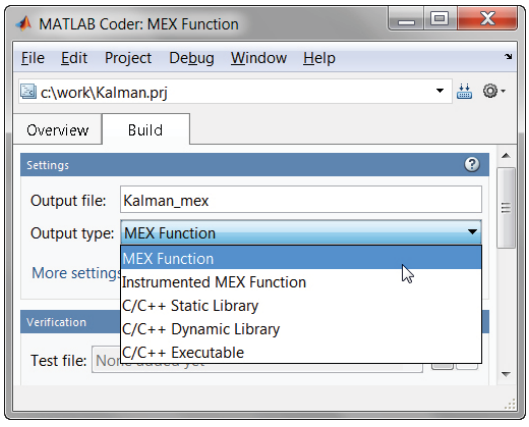

Рисунок 5. Меню MATLAB Coder для генерации MEX-функции.

Ускорение, которого можно достичь, зависит от природы алгоритма. Лучший способ оценить ускорение заключается в генерации MEX-функции при помощи MATLAB Coder и измерении ускорения. Если алгоритм содержит типы данных единичной точности, типы данных в фиксированной точке, циклы с состояниями или код, который не векторизуется, вы, скорее всего, увидите ускорение. С другой стороны, если алгоритм содержит вычисления MATLAB, которые уже неявно многопоточные – такие, как fft и svd, функции, вызывающие библиотеки IPP или BLAS, функции, оптимизированные для выполнения в MATLAB на ПК, такие, как FFT или алгоритмы, где можно векторизовать код, ускорение менее вероятно. Попробуйте MATLAB Coder, следуйте лучшим практикам для генерации кода C и проконсультируйтесь с техническими экспертами MathWorks для нахождения лучших методов ускорения вашего алгоритма с использованием такого подхода.

Многое из языка MATLAB и нескольких тулбоксов поддерживается для генерации кода. MATLAB Coder предлагает автоматизированные инструменты для оценки готовности вашего алгоритма к генерации кода и проведения вас по всем шагам генерации кода C (Рисунок 6).

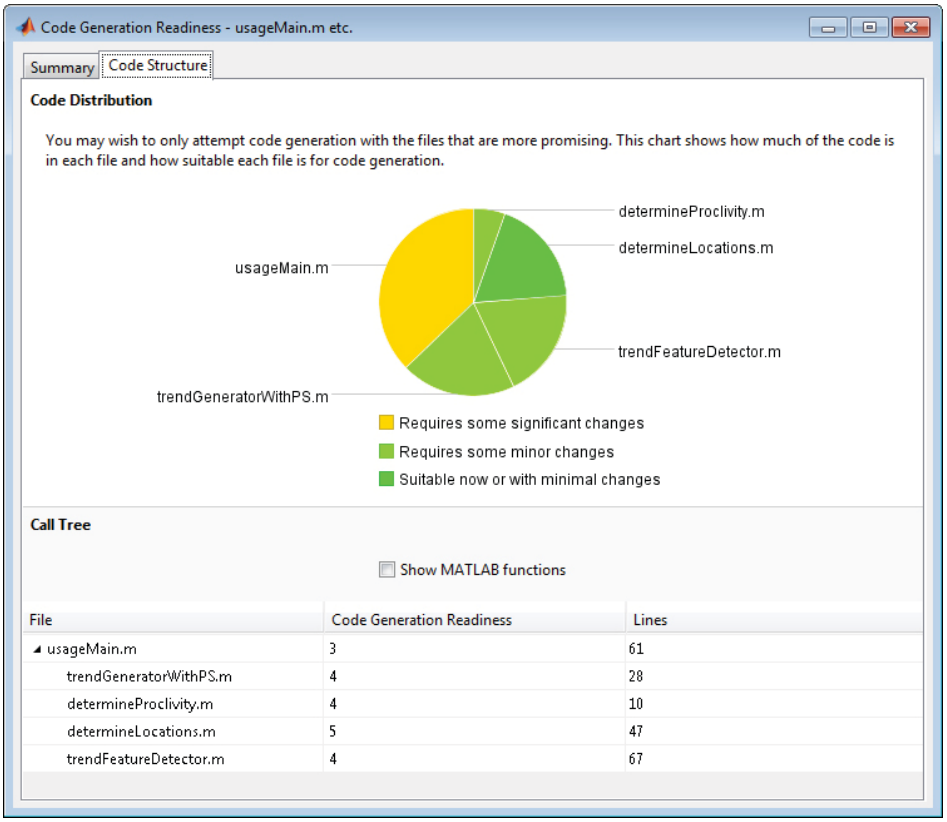

Рисунок 6. Инструмент MATLAB для оценки готовности к генерации кода и руководства, помогающие вам начать генерировать код C.

Узнайте больше о переходе от MATLAB к коду C и о том, как быстро начать использовать MATLAB Coder.

Возможности по увеличению производительности

Приложения MATLAB можно ускорить при помощи написания более эффективных алгоритмов, параллельных вычислений и генерации кода. Каждый метод предоставляет диапазон возможных ускорений, в зависимости от задачи и используемого оборудования. Бенчмарки и примеры ускорения, приведенные здесь, дают общее понимание доступных методов ускорения.

Узнайте больше об увеличении производительности при помощи parfor, различных методах расчета, поддерживаемых на GPU, встроенной поддержке GPU для системных объектов и генерации кода C.

Комбинирование техник

Часто можно достичь дополнительного ускорения путем комбинирования техник, описанных в этой статье. Например, циклы parfor могут вызывать MEX-функции на C, генерация кода поддерживается для многих системных объектов, а векторизованный код MATLAB может быть адаптирован для запуска на GPU.

Узнайте больше о совместном использовании нескольких техник для ускорения симуляции алгоритмов систем связи и разработки системы 4G LTE.

[1] В этой статье не затронуты ограничения, накладываемые памятью – например, подкачка или файловый I/O. Для получения дополнительной информации по этим темам, обратитесь к стратегии по эффективному использованию памяти и импорту и экспорту данных.

[2] При использовании с Parallel Computing Toolbox.

ДЛЯ ЗАМЕТОК

Дополнительная информация и контакты

Информация о продуктах
matlab.ru/products

Пробная версия
matlab.ru/trial

Запрос цены
matlab.ru/price

Техническая поддержка
matlab.ru/support

Тренинги
matlab.ru/training

Контакты
matlab.ru
E-mail: matlab@sl-matlab.ru
Тел.: +7 (495) 232-00-23, **доб.** 0609
Адрес: 115114 Москва,
Дербеневская наб., д. 7, стр. 8

MathWorks®
Accelerating the pace of engineering and science