----титул

Здравствуйте, меня зовут максим максимов. Моеей бакалаврской работой под руководством вадима велериевича монахова является портирование трехуровневой системы подключения математических библиотек на платформу .NET

Подавляющее число математических библиотек, используемых учеными-физиками, реализовоны на языке fortran. Это связано с тем, что fortran долгое время не имел альтернативы в своей области – проведение высокопроизводительных научных вычислений. Эти библиотеки хорошо отлажены, эффективны, документированы. Однако язык фортран обладает ненадежным синтаксисом, что порождает много потенциальных проблем.

Существуют современные и популярные платформы программирования, которые позволяют писать программы на языках с низким порогом вхождения(такие как java, c#), однако они слабо приспособлены для высокопроизводительных вычислений.

В связи с этим появилась тенденция перевода библиотек на современные языковые средства. Стоит отметить, что прямой перевод кода порождает множество проблем и недостатков, такие как высокая трудозатратность, повторяющийся код, а также возможна потеря производительности.

В данной работе рассматривается портирование библиотеки LAPACK и использующей ею библиотеки BLAS на платформу .NET. Это библиотека содержит набор подпрограмм для решения наиболее часто встречающихся задач линейной алгебры.

На нашей кафедре была разработана библиотека JavaLapack, которая является портом этой библиотеки на другую платформу – Java. Платформа Java является конкурирующей к .NET. Платформа .NET охватывает огромное количество программистом и ученых по всему миру, также она имеет очень высокие темпы развития.

---введение

В связи с этим мною решалось множество задач, такие как вызов неуправляемого кода из общеязыковой среды выполнения, изучение способов вызова dll, написанных на неуправляемых языках. Также производилось сравнение производительности.

---трехуровневая система

Для решения обозначенных выше проблем, на нашей кафедре была разработана система подключения библиотек, которая состоит из трех основных составляющих – это ядро – скомпилированная библиотека, оболочка, которая скрывает детали реализации ядра и ряд функций системного характера, работу с памятью и последний, верхний слой, предоставляет удобный доступ к библиотек для специалистов предметной области.

В решении JavaLapack для верхнего слоя использовался язык Java, для среднего – C++, а для связи этих слоев – технология Java Native Interface, которая позволяет вызывать код написанные на C++ и ассемблере под управлением JVM.

---архитектура1

В решении NLAPACK для связи верхнего слоя с нижним было использовано расширение для VC++ - Visual C++ with Managed Extensions(далее просто Managed C++). Он включает в себя набор грамматических и синтаксических расширений, которые позволяют связывать управляемый код(т.е. код, исполняющийся под управление общеязыковой среды выполнения) с неуправляемым(обычный машинный код).

Managed C++ имеет ряд преимуществ такие как высокая производительность, разрешает полный контроль над низкоуровневыми затратными операциями, позволяет смешивать управляемый и неуправляемый код даже в одном файле.

---архитектура2

Библиотеки LAPACK и BLAS были скомпилированы при помощи компилятора MinGW, дальше, при помощи стандартных средств, входящих в набор с visual studio, были сгенерированы библиотеки импорта. Подробнее- в описании.

Благодаря хорошо спланированной архитектуре, удалось преиспользовать большую часть уже имеющегося C++-кода от решения JavaLapack. Связывание неуправлямых C++-классов было осуществлено методом управляемых оберток, который подробно описан в описании.

Проект получился очень большим и подробный рассказ об архитектуре может занять очень много времени. Перейдем непосредственно к обсуждению результатов измерения производительности.

---Сравнение производительности0

При измерении производительности для уменьшения погрешности была использована следующая методика: генерирование набора данных, пробный запуск функции, генерация нового набора входных данных, тестовый запуск функции заданное число раз.

Тестирование производилось на ряде важных задач линейной алгебре, такие как svd-разложение, решение СЛАУ.

---Сравнение производительности JN СЛАУ 1

На графике предоставлено зависимость времени решения слау от размера входной матрицы

Давайте рассмотрим зависимость более детально

---Сравнение производительности JN СЛАУ RE

Как мы видим, производительность системы NLapack находится на уровне системы JavaLapack

---Сравнение производительности JN СЛАУ IM

Тоже самое можно наблюдать и для комплексных матриц

---Сравнение производительности JN SVD 1

Далее произведено сравнение производительности для SVD – разложения с меньшей точностью.

---Сравнение производительности JN SVD 2

Однако, мы также видим, что производительность находится на уровне решения JavaLapack.

--- Сравнение производительности .NET

Давайте произведем сравнение производительности с уже имеющимися библиотеками на платформе .NET. Было опробовано более 10 библиотек, выбраны 3 лучшие из них. Рассотрим производительность умножения матриц.

--- Сравнение производительности .NET

В логарифмическом масштабе. Как мы видим, решение NLapack серьезно превосходит тестируемые мной математических библиотек на платформе .NET. Библиотека CSLAPACK особенно интересна тем, что что они являются переводом процедур численных методов библиотеки LAPACK, написанных на Fortran на C#. Это наглядно демонстрирует преимущества использования неуправляемого кода.

Полученные результаты показывают, что производительность NLapack заметно превосходит производительность математических библиотек, имеющихся в настоящее время для платформы .NET.