Библиотека для динамической адаптации программ к гетерогенным архитектурам вида CPU + GPU

М.А. Кривов[1], С.А. Гризан[2]

Московский Государственный Университет им. М.В. Ломоносова[1],
Сибирский Федеральный Университет[2]

За последние два года архитектура высокопроизводительных вычислительных систем претерпела ряд существенный изменений. Появившаяся в 2007 году технология NVidia CUDA позволила задействовать колоссальные вычислительные мощности существующих графических ускорителей, сделав возможным создание «домашних» суперкомпьтеров. Выпуск специализированных видеоплат серии NVidia Tesla позволил оснащать узлы классических кластеров данными ускорителями, в результате чего широкое распространение также стали получать так называемые гетерогенные суперкомпьютеры.

При реализации алгоритмов для систем с гетерогенными архитектурами с использованием современных технологий перед программистом встаёт ряд проблем, связанных с неоднородностью вычислителей. Одной из основных проблем является необходимость выбора оптимального вычислителя для каждой части задачи. Другими словами, без знания специфики архитектуры не всегда понятно, какую часть задачи на каком процессоре лучше считать. Согласно идеологии NVidia, на графические ускорители нужно переносить всё, что удаётся распараллелить под модель SIMT, оставляя центральному процессору только последовательные части программы. Однако, как показано в работах [1-2], при решении ряда задач, оптимально подходящих для видеокарт, центральный процессор класса Intel Xeon может показать схожую производительность. Соответственно, если же алгоритм «плохо» отображается на архитектуру видеокарт, то их использование может лишь замедлить программу.

В данной работа предложен подход, основная идея которого заключается в использовании динамических параметров, непосредственное значение которых будет определяться во время работы программы и, более того, будет изменяться в зависимости от типа вычислений и возможностей используемой аппаратной платформы. Для программиста это означает, что больше не требуется знание специфичных особенностей архитектуры. Если требуется определить значение какого-либо параметра, то вместо него можно указать специальную конструкцию-«заглушку» и, при необходимости, определить начальное значение.

Для подбора оптимальных значений определённых подобным образом параметров возможно два режима работы программы. В первом из них перед выполнением основных вычислений производится «обучение» на небольших входных данных. Второй режим ориентирован на итеративные программы, в которых одно и тоже вычислительное ядро выполняется много раз. Таким образом, каждую итерацию или запуск в режиме обучения можно рассматривать как один шаг оптимизационного метода, используемого для минимизации функции времени работы.

Данный подход был протестирован на задаче перемножения матриц размерности 1024 на 1024. При работе на системе с 4-ядерным центральным процессором (Intel Core 2 Quad Q6600) и двумя графическими ускорителями разной архитектуры (NVidia Tesla C1060 и NVidia Tesla C2050) за 7 итераций время работы приложения было уменьшено более чем в 8 раз.

Список литературы

1. Bordwekar R., Bondhugula U., Rao R., Believe it or Not! Multy-core CPUs Can Match GPU Performance for FLOP-intensive Application! IBM Research Report RC24982 (W1004-095), April 23, 2010.

2. Bordwekar R., Bondhugula U., Rao R., Can CPUs Match GPUs on Performance with Productivity?: Experiences with Optimizing a FLOP-intensive Application on CPUs and GPU, IBM Research Report RC25033 (W1008-020), August 5, 2010.