Производительность МД-алгоритмов на гибридных системах на чипе Nvidia Tegra K1 и X1

В.П. Никольский[1,2], В.С. Вечер[2,3], В.В. Стегайлов[2]

Национальный исследовательский университет «Высшая школа экономики»[1],
Объединенный институт высоких температур РАН[2],
Московский физико-технический институт[3]

В данной работе рассматривается эффективность использования гибридных систем на чипе для высокопроизводительных расчётов. Во-первых, производится построение модели производительности Roofline для рассматриваемых систем с помощью Empirical Roofline Toolkit, а также сравнение полученных результатов с теоретическими оценками. Во-вторых, на примере молекулярно-динамического пакета LAMMPS в различных конфигурациях демонстрируется производительность и эффективность новых устройств в контексте ранее собранных данных о производительности этого МД пакета на разных процессорах. Кроме того, описана попытка разделить compute-bound и memory-bound режимы работы МД алгоритма на исследуемых системах, что соответствует принципу Roofline.

Ключевые слова: Kepler, Maxwell, ARM, система на чипе, молекулярная динамика, Roofline

1. Введение

На сегодняшний день технологии производства микросхем близки к достижению физических пределов, дальше которых увеличение плотности транзисторов на кристале становится очень проблематичным. Дальшейшее наращивание вычислительной мощности суперкомпьютеров связано со значительным увеличением количества узлов и поиском более сложных технологий.

Одно из направлений развития — это повсеместное использование ускорителей, таких как ГПУ, то есть переход к гетерогенным системам. Хотя изначально ГПУ разрабатывались для массового рынка, привлекательное отношение «цена-производительность» привело к адаптации этих устройств для высокопроизводительных вычислений посредством разработки технологий GP-GPU. Другой тренд — это растущее внимание к показателям энергоэффективности и резкий рост влияния архитектуры ARM на серверный рынок. Уже несколько поколений ARM снабжаются конкуретоспособными модулями для работы с числами с плавающей запятой, и это открывает новые возможности их использования для высокопроизводительных вычислений [1].

При использовании сопроцессоров определенные ограничения вызывает то, что эти устройства физически разделены с ЦПУ. Коммуникация становится гораздо проще, если разместить ЦПУ и сопроцессор на одном чипе. Такие системы на чипе (SoC) уже представлены, например AMD Accelerated Processing Units, ARM Mali, Intel HD Graphics. Недавно и Nvidia объединила ранее названные два тренда в одном устройстве, что привело к выпуску Tegra K1, а вслед за ним Tegra X1. Это устройства, которые объединяют на одном узле несколько ядер ARM Cortex-A и Nvidia ГПУ. За девять лет разработки технологии Nvidia CUDA появлось много научных и инженерных программ, использующих её. Новая 64-битная архитектура ARM Cortex-A57 тоже демонстирует потенциал для высокопроизводительных вычислений (см. [2]). Учитывая низкий уровень энергопотребления, системы-на-чипе Tegra следует рассмотреть в качестве элементной базы для будущих решений в области HPC.

Ускоренная разработка оборудования увеличивает значимость тестов эффективности новых архитекутр. Тесты (супер)компьютерных систем имеет несколько десятилетий историй (см. [3–6]. Спектр используемых алгоритмов стал достаточно широким, поэтому основным

инструментом для разработки новых суперкомпьютеров должен быть тщательно отобранный набор тестов. Измерение и представление результатов тестов производительности параллельных компьютерных систем становятся всё более научно-обоснованными [7], включая измерение энергопотребления, которое крайне важно для разработки суперкомпьютеров экзафлопсного уровня [8].

2. Обзор литературы

Изучению новых процессоров ARMv8 в сравнении с Atom и серверными процессорами Intel посвящена работа [9]. В ней собран большой объем результатов тестов и проанализован с применением статистических методов. Для выявления узких мест каждой из архитектур применен новых подход.

Недавно в области HPC появилось несколько исследований систем-на-чипе Tegra. С использованием объединенной памяти, поддержка которой была введена в CUDA 6.5, Ukidave et al. показали увеличенную скорость передачи данных ЦПУ-ГПЦ на Tegra K1 по сравнению с обменом по шине PCIe [10]. Haidar et al. рассмотрели производительность и энергоэффективность при использовании библиотеки линейной алгебры MAGMA на платформе Jetson TK1 [11]. Stone et al. измеряли производительность основных пакетов вычислительной биологии на некоторых новых гетерогенных платформах (включая Jetson TK1 и TX1), которые объединяют многоядерные центральные процессоры ARM и ускорители GPU [12].

3. Тестовая конфигурация

3.1. Аппаратное обеспечение

3.1.1. Nvidia Jetson TK1

Nvidia Jetson TK1 это платформа для разработки, основанная на 32-битной системе-на-чипе (SoC) Tegra K1 с LPDDR3 памятью (работает на частоте 930 МГц). Комплекс центрального процессора Tegra K1 включает четырех ядра Cortex-A15, работающих на частоте до 2.3 ГГц, а также пятое маломощное ядро, предназначенное для замены основных ядер в режиме пониженной нагрузки для снижения потребления энергии и выделения тепла. Также на кристалле размещен один ГПУ Kepler Streaming multiprocessor (SM), имеющий частоту 852 МГц и 128 ядер CUDA. Каждое ядро Cortex-A15 имеет L1 кеш инструкций и L1 кеш данных каждый по 32 Кбайт. Кеш L2 имеет объем 2 Мбайт, и он используется всеми 4 ядрами. Программное окружение представлено Linux Ubuntu 14.01.1 LTS (GNU/Linux 3.10.40-gdacac96 armv7l). В качестве набора инструментов для компиляции имеется GCC ver. 4.8.4 и CUDA Toolkit 6.5.

3.1.2. Nvidia Jetson TX1

Jetson TX1 основан на более новой 64-битной Tegra X1 SoC с памятью LPDDR4 (1600 МГц). Система включает 4 ядра Cortex-A57, работающие на частоте 1.9 GHz, 4 более слабых ядра Cortex-A53, подменяющие основные ядра при низкой нагрузке и два SM графического ускорителя Maxwell, работающего на частоте 988 ГГц и содержащего 256 ядер CUDA. Каждое ядро Cortex-A57 имеет L1 кеш объемом 48 Кбайт, 32 Кбайт кеша данных уровня L1. Также присутствует L2 кеш, 2 Мбайт которого распределены между всеми ядрами.

В качестве операционной системы установлен Linux Ubuntu 14.01.1 LTS с 64-битным ядром для архитектуры aarch64. Тем не менее остальное программное окружение 32-битное, такое же, как у 32-битного Nvidia Jetson TK1 (раздел 3.1.1), за исключением более нового CUDA Toolkit 7.0.

3.1.3. Сервер с ускорителем Nvidia Tesla K80

Для наглядного сравнения в данной работе использовался высокопроизводительный сервер на базе двух процессоров Intel Xeon E5-2697 v3. Это процессор архитектуры x86_64 Haswell, он имеет 14 ядер с поддержкой технологии hyper-threading, работающих на частоте 2.8 ГГц. На сервере установлен графический ускоритель Nvidia Tesla K80. Он имеет порядка 5 тысяч ядер, работающих на часототе около 627 МГц и способен увеличивать частоту в турборежиме. Ускоритель имеет 24 ГБ памяти, а на сервере имеется 132 ГБ DRAM.

3.2. Программное обеспечение

3.2.1. Empirical Roofline Toolkit

В данной работе использовался набор программ Empirical Roofline Toolkit версии 1.1.0, его исходный код доступен в репозитории Berkeley Lab на Bitbucket [13]. Для компиляции использовались ключи gcc '`-O3 -march=armv7-a -mtune=cortex-a15 -mfpu=neon-vfpv4 -ffast-math`' для Jetson TK1 и '`-O3 -macrh=armv8-a -mfpu=neon-vfpv4 -ffast-math`' на Jetson TX1. На Haswell для компиляции использовались ключи '`-O3 -macrh=haswell -mtune=haswell -mavx2 -mfma -ffast-math`'

3.2.2. LAMMPS

Для компиляции LAMMPS для Cortex-A15/A57 ядер из Tegra K1/X1 соответственно были использованы те же опции, что и для компиляции Empirical Roofline Toolkit. LAMMPS компилировался с пакетом USER-OMP для использования технологии OpenMP. Для пакета GPU были использованы опции компиляции CUDA-кода '`arch=sm_32`' для K1 и '`arch=sm_53`' для X1. Для пакета USER-CUDA использовался ключ '`arch=21`' в обоих случаях.

Существует отличие при компиляции LAMMPS с пакетом GPU и пакетом USER-CUDA. В случае GPU есть простой способ использовать любую желаемую архитектуру CUDA при компиляции, в то время как разработчики ныне не поддерживаемого пакета USER-CUDA предоставляют специальные скрипты и документацию только для сборки под несколько устаревших архитектур. Несмотря на это, USER-CUDA всё же был собран с новыми архитектурами sm_32 и sm_53 для проведения тестов на Jetson TX1 и Jetson TK1. Замеры показали, что производительность очень слабо зависит от архитектуры, используемой при компляции (разница близка к погрешности измерений), так что было принято решение использовать только 'arch=21' для последующих тестов как более доступную и распространенную настройку.

4. Модель производительности Roofline

4.1. Roofline

Модель производительности Roofline [14] (англ. «линия крыши», обозначает форму, профиль крыши) разработана в лаборатории Беркли Performance and algorithm research. Она позволяет на одном графике описать ограничения, связанные с производительностью процессора, пропускной способностью памяти, а также возможное влияние кешей и векторизации. Самой удобной и распространенной формой модели Roofline является изображение производительности как функции от машинной пиковой производительности, пиковой пропускной способности памяти и арифметической интенсивности.

Ключевым понятием для понимания модели Roofline является арифметическая интенсивность программы (arithmetic intesity). Определение этой величины звучит так: арифметическая интенсивность – это отношение общего количества арифметических операций (с плавающей запятой) к общему объему переданных данных в байтах. Для визуализации

модели нужно изобразить производительность компьютера в ГФлоп/с как функцию от арифметической интенсивности. Полученная кривая ограничит площадь, в которой лежат всевозможные производительности программ на этой машине. Причем при построении в двойном логарифмическом масштабе эти кривые отображаются в прямые линии, что очень наглядно. В качестве иллюстрации предлагаются графики из следующих частей, см. рис. 2.

В этой работе для построения модели Roofline использовался Empirical Roofline Toolkit. Это набор исходных кодов и скриптов, которые осуществляют тестирование системы и построение различных отчётов. Основная идея в следующем: в цикле организуют вычисление простейших арифметических операций над элементами массива определенной длины с измерением времени выполнения этой задачи. В процессе теста ERT варьируются величины ERT_FLOPS и размер рабочего массива (Working set size). Это позволяет построить зависимость производительности от арифметической интенсивности с учётом эффектов кешей.

4.2. Теоретические оценки

В этом разделе отдельно показано, каким образом в этом исследовании были сделаны оценки пиковых величин, связанных с производительностью.

Пиковая пропускная способность L1 кеша процессов Tegra X1 и Tegra K1 может быть оценена по формуле $\text{freq}_{CPU}[\text{ГГц}] * 32[\text{бит}] * (1[\text{запись}] + 1[\text{чтение}])$. Теоретическая пропускная способность памяти значительно ниже: $\text{freq}_{DRAM}[\text{ГГц}] * 8[\text{байт}] * (1[\text{запись}] + 1[\text{чтение}])$. Для ГПУ системная пропускная способность L2 оценивается так: $[\text{количество SM}] * \text{freq}_{GPU}[\text{ГГц}] * 32[\text{банка памяти}] * 4[\text{байт}] * 0.5$. Это даёт 54.4 ГБ/с для Kepler SM в TK1 и 128 ГБ/с для двух Maxwell SM в TX1.

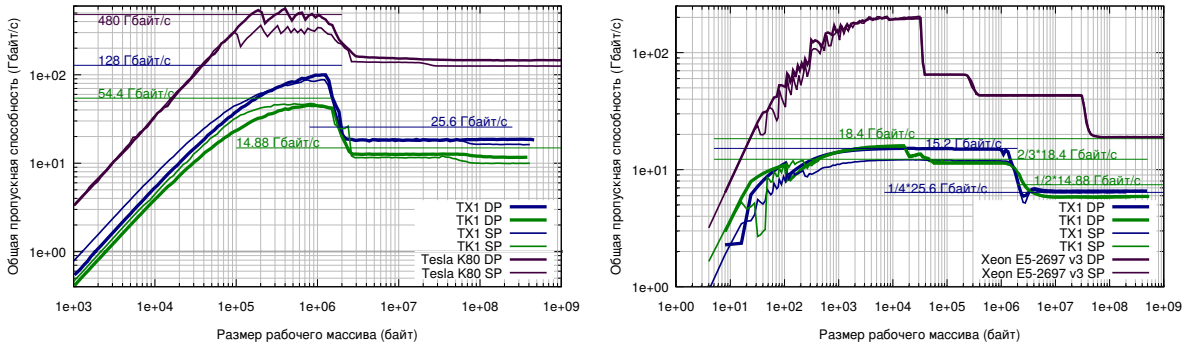

Рис. 1: График зависимости пропускной способности ГПУ (слева) и ЦПУ (справа) от размера обрабатываемого массива. Для сравнения приводятся графики сервера с процессором Xeon и ускорителем Tegra

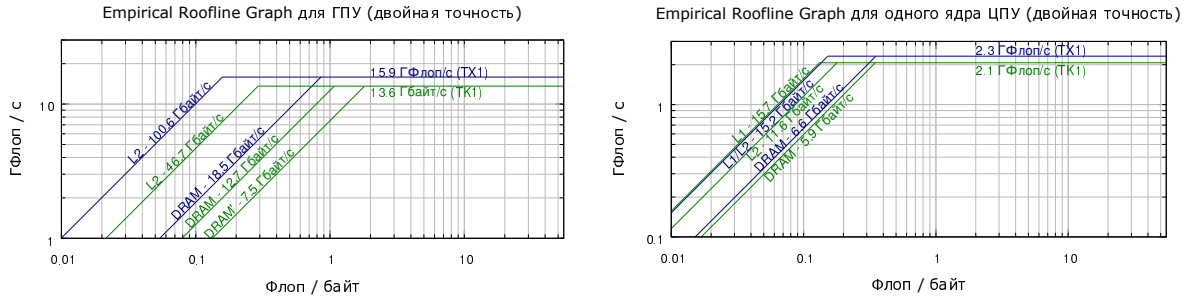

Рис. 2: Визуализация модели Roofline для ГПУ и ЦПУ аппаратных платформ Jetson TX1 и Jetson TK1

4.3. Анализ результатов

На рис. 1 изображены зависимости пропускной способности от объёма обрабатываемых данных на графическом ускорителе. На 1 ядре центрального процессора TK1 наблюдаются значения пропускной способности близкие к 85% от пикового значения для L1 кеша. Пропускная способность L2 примерно в 2/3 меньше. Значения пропускной способности на Tegra X1 находятся в очень хорошем соответствии с теоретическим значением и не обнаруживается разница между работой в L1 и L2 кешах. В одиночной точности Tegra X1 показывает пропускную способность кеша около 5/6 от значений при работе с двойной точностью.

Что касается устоявшейся пропускной способности (sustained bandwidth) основной памяти, результаты Roofline для одного ядра процессора показывают 1/2 и 1/4 от пиковых величин на TK1 и TX1 соответственно.

При запуске Roofline на ГПУ значения пропускной способности оказываются близки к теоретическим оценкам. Не наблюдается большой разницы между одиночной и двойной точностью. Рис. 1 демонстрирует небольшую разницу между размерами ГПУ L2 кеша на Kepler (1.5 Мбайт) и Maxwell (2 Мбайт). Пока остаётся непонятными, почему пропускная способность памяти немного понижается при размерах рабочей памяти более 50-70 Мбайт.

Рис. 2 показывает, что значения пиковой производительности для Cortex-A15 и Cortex-A57 очень близки: 0.9 Флоп/такт для A15 (соответствует предыдущей нашей оценке [15]) и 1.2 Флоп/такт на A57. Кроме того на Tegra X1 был произведен особый эксперимент с неофициальными 64-битными инструментами сборки (официальной поддержки ещё нет), и они показали такую же производительность. С использованием одиночной точности получается 5.0 ГФлоп/с на A15 и 5.1 ГФлоп/с на A57. Дизассемлирование бинарных файлов показывает, что в ядре Roofline используются FMA инструкции для получения R_{peak} на Cortex-A5/A15/A57, но требуется дополнительное перемещение данных, что снижает их эффективность. Дело в том, что в наборе иструкций ARM операции FMA присутствуют только в форме $a = a + b \cdot c$, в то время как в FMA3 (например, в Haswell) есть и другие две возможные операции: $a = a \cdot c + b$, $a = a \cdot b + c$. Эффективной векторизации с помощью NEON используемый компилятор не осуществляет, хотя это обеспечило бы дву- или четырёхкратный прирост производительности, и для этого есть возможность.

Запуск теста на GPU получил значения пиковой производительности в одиночной точности 209.9 ГФлоп/с на Kepler SM и 485.1 ГФлоп/с на двух Maxwell SM в составе TX1. Эти значения близки к теоретическим пиковым значениям 218 ГФлоп/с и 511 ГФлоп/с соответственно. При использовании двойной точности получаются значения 13.6 ГФлоп/с и 15.9 ГФлоп/с, которые соотносятся с ожидаемым значением около 16 ГФлопс/такт (производительность FP64 относится к производительности FP32 как 1/24 для K1 Kepler и 1/32 для TX1 Maxwell).

5. Классическая молекулярная динамика и LAMMPS

5.1. Метод молекулярной динамики

Молекулярно-динамический метод основан на решении Ньютоновских уравнений движения отдельных частиц и является исследовательским инструментом величайшей важности. Вычислительная скорость и эффективность параллелизации - это главные факторы, которые ограничивают пространственные и временные масштабны, доступные для МД-расчёта. Достигнутые пределы это триллионы атомов [16] и милисекунды времени [17], при типичном МД шаге порядка 1 фемтосекунды.

Узким местом в МД алгоритме является вычислительная сложность расчёта межатомных потенциалов. Гибридные архитектуры рассматриваются как основная возможность для использования на суперкомпьютерных узлах. Алгоритмы МД на графических ускорителях являются частным случаем портирования МД алгоритмов на SIMD архитектуры, такие как

Intel Xeon Phi.

5.2. LAMMPS и гибридные архитектуры

В данной работе используется пакет LAMMPS, это гибкий инструмент для построения моделей классической МД в материаловедении, химии и биологии. LAMMPS это не единственный МД пакет, который портирован на гибридные архитектуры (например HOOMD [18] изначально проектировался с расчётом на ГПУ ускорители). Первыми реализациями МД алгоритма для гибридных архитектур в LAMMPS были пакеты GPU [19, 20] и USER-CUDA [21].

Пакет GPU поддерживает и CUDA, и OpenCL. Он был спроектирован для того, чтобы использовать распространенную конфигурацию, когда один или более графический ускоритель подключен к одному или более многоядерному процессору. Особенностью этой реализации стало перемещение данных с ускорителя на хост после каждого временного шага. В отличае от USER-CUDA, пакет GPU позволяет запустить несколько потоков центрального процессора на один ГПУ, а также допускает расчёт сил на ГПУ и ЦПУ одновременно. Впрочем, в наших экспериментах конфигурации, отличные от 1 ГПУ на 1 ядро процессора, приводили к потере производительности. В этой работе пакет компилировался в CUDA версии для всех трёх поддерживаемых режимов точности вычислений (одиночный, двойной и смешанный).

Пакет USER-CUDA поддерживает только устройства, совместимые с CUDA. В отличие от пакета GPU, USER-CUDA позволяет производить все вычисления LAMMPS для множества шагов на графическом устройстве (за исключением MPI обменов между узлами). Таким образом, информация об атомах не передаётся лишний раз туда и обратно между ГПУ и ЦПУ. Списки соседей также строятся на ГПУ, в то время как пакет GPU позволяет выбрать для этих целей ГПУ или ЦПУ. Впрочем, вызов не-ГПУ операторов во входном скрипте LAMMPS приводит к выгрузке данных в основную память, что может привести к потерям производительности. USER-CUDA также был протестирован в режимах одиночной, двойной и смешанной точности.

5.3. Тестовая модель

В качестве теста в данной работе использовалась модель Леннард-Джонсовской жидкости с плотностью $0.8442\sigma^{-3}$, количеством атомов 108000, расчитываемая на 250 шагах по схеме интегрирования NVE. Подобный бенчмарк крайне распространён среди работ, посвященных МД моделированию, и обычно используется радиус отсечения 2.5σ.

Этот последний параметр заслуживает отдельного обсуждения. Обычно отсечение межатомного взаимодействия используется в алгоритме классической МД для существенного снижения вычислительной сложности. Он определяет количество ближайший соседей каждой частицы, соответственно и количество вычислений с тем же набором данных. Пользуясь терминологией модели Roofline, можно сказать, что арифметическая интенсивность алгоритма прямо пропорциональна количеству ближайших соседей. А также производительность пропорциональна отношению ближайших соседей к общему времени работы МД алгоритма. Для рассмотрения такой зависимости были собраны данные о зависимости, и они отображены на рисунке 3. Тесты обеспечивают пакет USER-CUDA в режимах двойной и одинарной точности, пакет USER-OMP в двойной точности и KOKKOS в режимах CUDA и OpenMP. Так как известно, что на каждое взаимодействие приходится 23 Флоп, а количество взаимодействий на атом равно количеству соседей (либо половине этого количества, если в алгоритме используется 3-ий закон Ньютона), то от величины «соседей в секунду» легко перейти к величине Флоп/с, что позволяет рассматривать зависимость как аналог Roofline.

В случае USER-OMP при малом количестве соседей алгоритм является memory-bound, и с увеличением количества соседей переходит к compute-bound. Алгоритм пакета USER-CUDA

является memory-bound при малом количестве соседей. Поэтому отношения производительности в двойной и однарной точности (x2.5 для Tegra K1 и x2.8 для Tegra X1) определяются отношением пропускных способностей (в одинарной точности объём передаваемых данных в 2 раза меньше по сравнению с данными в двойной точности и пропускная способность L2 кеша ГПУ также несколько выше в одиночной точности). Можно увидеть, что производительность GPU в 1.8 раза выше, чем производительность 4-ех ядер ЦПУ, полученная с помощью пакета USER-OMP. Это очень близко к отношению пиковой производительности ГПУ Maxwell к 4 ядрам Cortex-A57: $15.9/(4 * 2.3) \approx 1.7$. Пакет USER-CUDA показывает довольно низкую производительность при одиночной точности: алгоритм всё время остаётся зависимым от скорости памяти из-за того, что увеличение количества соседей приводит к усложнению доступа к памяти.

При сравнении результатов SoC с производительностью на обычном сервере следует обратить внимание, что производительность не возрастает в соотвествии с отношением производительности, а в большей степени определяется пропускной способностью. Кроме того следует уделить внимание тому факту, что при увеличении радиуса обрезки разница между одиночной и двойной точностью на TX1 уменьшается, в то время как на сервере явно увеличивается и расходится.

При большом значении радиуса обрезки производительность пакета KOKKOS с CUDA становится близка к производительности USER-CUDA, в то время как при малыз значениях он явно проигрывает. На ЦПУ KOKKOS не показыавет хороших результатов при любых значениях радиуса.

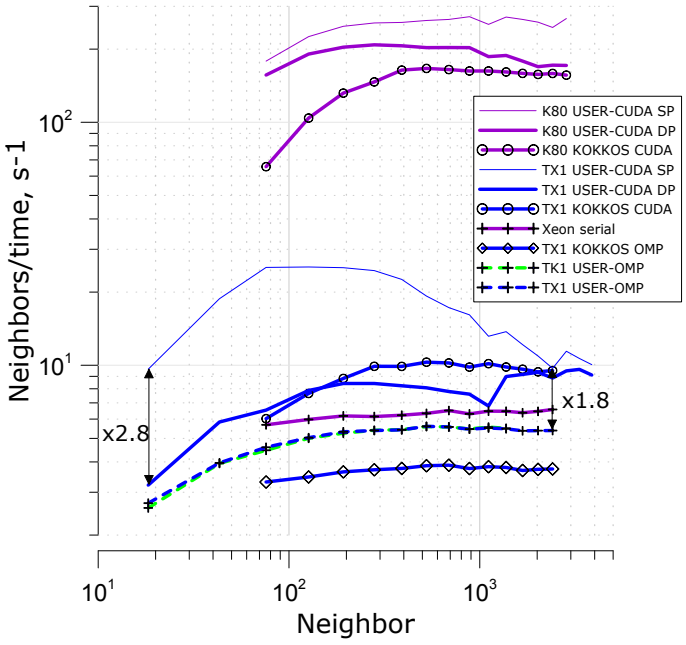

Рис. 3: Производительность МД алгоритма как функция от количества ближайших атомов в модели Л-Дж жидкости: можно рассматривать как аналог Roofline

5.4. Производительность LAMMPS относительно R_{peak}

Для конечного пользователя высокопроизводительных систем важнейшим параметром является время решения задачи. Это приводит нас к очевидному решению использовать величину «время на один шаг МД интегрирования» в качестве одного из параметров метрики производительности. Другой параметр должен характеризовать аппаратное обеспечение. Для сравнения существенно разного аппаратного обеспечения пиковая производительность

будет использована в качестве второго параметра метрики. График 4 показывает сравнение

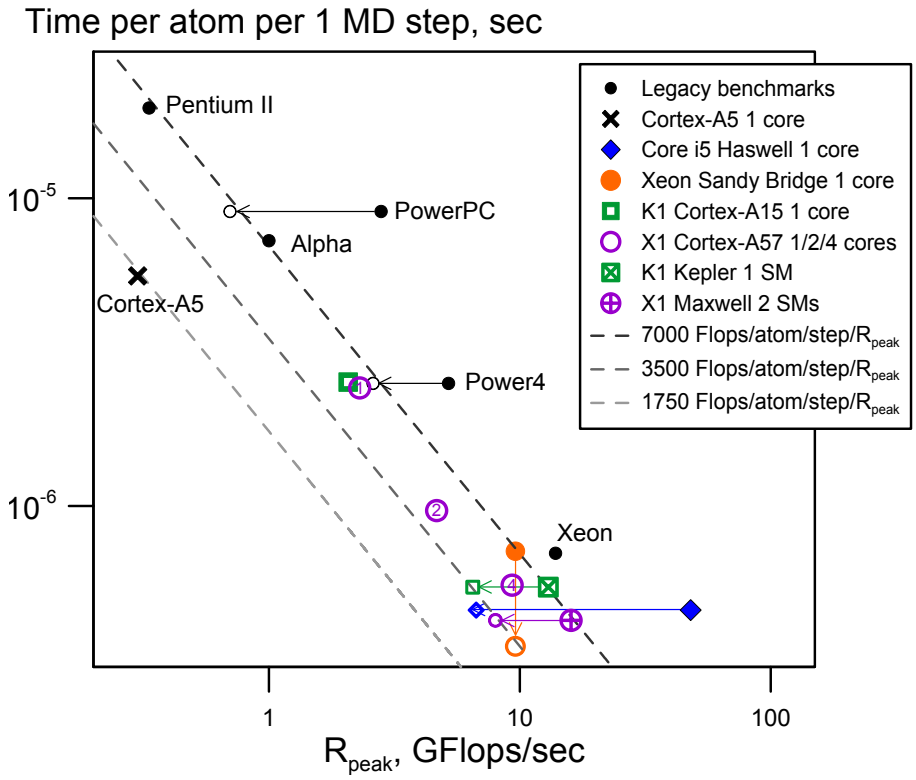

Рис. 4: Время на атом на 1 МД шаг для теста Л-Дж жидкости (радиус обрезки 2.5σ)

времени расчёта модели МД теста Л-Дж жидкости на различных микропроцессорах. Результаты теста на одном ядре ЦПУ для процессоров Intel Pentium II Over-Drive 333 МГц, DEC Alpha 500 МГц, PowerPC 440 700 МГц, Power4 1.3 ГГц and Intel Xeon 3.47 ГГц взяты с сайта LAMMPS и отображены на графике черными кругами. Время определяется структурой микропроцессорного ядра и способностью компилятора создавать эффективный исполняемый код. Все точки отвечают одной и той же вычислительной задаче (Л-Дж жидкость) и одному и тому же C++ коду программы LAMMPS, но разным компиляторам для каждой системы. Величина R_{peak} для старых процессоров взята из опубликованных технических спецификаций. Результаты Intel и DEC Alpha хорошо согласуются с выражением $t \approx 1/R_{peak}$. К тому же, если отмасштабировать пиковую производительность ЦПУ IBM Power так, чтобы исключить FMA операции, каждая из которых обеспечивает 2 Флоп. Также для PowerPC 440 исключаются предположительно неиспользуемые операции SIMD, тогда все точки ложатся на общую зависимость ~ 7000 Флоп/атом/шаг/R_{peak}.

Мы можем сравнить это соответствие с другой [22] оценкой 55[соседей]$*$23[Флоп на пару] = 1265 Флоп/атом/шаг, или вдвое меньше 633 Флоп/атом/шаг для не-ГПУ алгоритмов, которые учитывают 3-ий закон Ньютона. Это означает, что МД алгоритм в LAMMPS при исполнении на ЦПУ архитектуры x86 и некоторых устаревших архитектурах использует $633/7000 \approx 10\%$ от общего R_{peak}.

Результаты экспериментов, сделанные на суперкомпьютерах МВС-10П МСЦ РАН без (полный оранжевый кружок) и с (пустой оранжевый кружок) ручной векторизацией также представлены на рис. 4. В этом случае LAMMPS компилируется с помощью Intel C++ compiler с подключением пакета USER-INTEL, который ориентирован на эффективное использование архитектуры Intel Xeon. Такой тип оптимизации кода приводит к ускорению вычислений до 2-ух раз и приводит к уровню ~ 3500 Флоп/атом/шаг/R_{peak}, что соответ-

ствует уровню потребления порядка 20% от R_{peak}. Запуск на Intel Core i5 (Haswell) показан на рис. 4 с соответствующим мастабированием R_{peak}. Анализ показал, что при определении производительности Roofline используются с максимальной эффективностью используются векторные инструкции FMA из набора AVX2, обеспечивая восьмикратное ускорение. В то же время в LAMMPS автовекторизация не происходит, и используются только две FMA инструкции на расчёт каждого Л-Дж взаимодействия.

Пиковая производительность ARM Cortex-A не задокументирована производителями. Соответствующие значения R_{peak} для Nvidia Tegra K1/X1, а также ODROID-C1 получены с помощью тестов Empirical Roofline Toolkit (часть 4). При использовании gcc автовекторизация для ARM не осуществляется ни в случае LAMMPS, ни даже в случае Roofline. LAMMPS демонстрирует при запуске на Cortex-A5 эффективность порядка $633/1750 \sim 40\%$, что примерно в 4 раза выше, чем эффективность других ЦПУ.

Выдающие результаты ODROID-C1 вызвали интерес провести дополнительные проверки. Чтобы убедиться в том, что Cortex-A5 не подменяет двойную точность одинарной, была определена машинная точность, и оказалось, что она соответствует стандарту IEEE 754 как в случае одинарной точности, так и в случае двойной.

Результаты, полученные с пакетом USER-CUDA в двойной точности должны быть отмасштабированы (изображено стрелками на рис. 4), чтобы исключить влияние FMA, поддерживаемых некоторыми ГПУ. В этом случае пакет USER-CUDA делает около 3500 Флоп/атом/шаг, что соотвествует использованию около 50% от ГПУ R_{peak}.

Стоит отметить, что время теста на 4 ядрах Cortex-A57 с использованием OpenMP версии LAMMPS очень близко к USER-CUDA на ГПУ.

6. Выводы

В этой работе была сделана оценка производительности систем Jetson TK1 и новейшей Jetson TX1 с использованием модели Roofline и инструмента Empirical Roofline Toolkit. Рассмотрены случаи одинарной и двойной точности. Полученные результаты пропускной способности оперативной памяти и кэш-памяти сопоставлены с теоретическими значениями.

Для тестов производительности конкретного примера прикладных расчетов использовались три разные реализации молекулярно-динамического алгоритма из пакета LAMMPS: OpenMP версия МД алгоритма и два варианта ГПУ-алгоритмов. Предложен метод варьирования вычислительной интенсивности МД задачи путем варьирования радиуса обрезки парного потенциала. Таким образом, был показан переход МД расчета из memory-bound в compute-bound режим в каждом из трех случаев. Дана интерпретация производительности МД алгоритмов в соответствии с результатами Empirical Roofline Toolkit.

Полученные в этой работе результаты были сопоставлены с раннее накопленными результатами для различных архитектур. В анализе использовалась метрика «время решения к R_{peak}», которая позволяет проанализировать эффективность использования ресурсов вычислений с плавающей точкой при разных комбинациях аппаратного обеспечения и ПО. Наибольшая эффективность достигается в случае архитектуры Cortex-A5.

Использование технологий OpenMP или CUDA в случае МД алгоритмов обеспечивает сопоставимое время при использовании двойной точности для расчёта Л-Дж жидкости. В одинарной точности CUDA-алгоритмы на этих системах сильно ограничены возможностями памяти и используют очень малую часть от достижимой пиковой производительности. Можно сделать вывод, что новые ядра ARM Cortex-A имею эффективность близкую к ГПУ Nvidia Kepler и Maxwell для решения задач классической молекулярной динамики с короткодействующим потенциалом. Для использования всех доступных ресурсов новых процессоров серии Cortex-A (SIMD операций NEON) требуется специальная адаптация МД алгоритма.

Аппаратное обеспечение, использованное в данной работе, было приобретено при

поддержке МФТИ и ВШЭ. Авторы благодарят компания Forsite за доступ к Nvidia Tesla K80. Работа поддержана грантом РНФ 14-05-00124.

Литература

1. G. Mitra, B. Johnston, A.P. Rendell, E. McCreath, and Jun Zhou. Use of simd vector operations to accelerate application code performance on low-powered arm and intel platforms. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1107–1116, May 2013.

2. Kristopher Keipert, Gaurav Mitra, Vaibhav Sunriyal, Sarom S. Leang, Masha Sosonkina, Alistair P. Rendell, and Mark S. Gordon. Energy-efficient computational chemistry: Comparison of x86 and ARM systems. *J. Chem. Theory Comput.*, 11(11):5055–5061, 2015.

3. Harold J. Curnow and Brian A. Wichmann. A synthetic benchmark. *The Computer Journal*, 19(1):43 – 49, 1976.

4. E. Strohmaier and Shan Hongzhang. Apex-Map: A global data access benchmark to analyze HPC systems and parallel programming paradigms. *Proceedings of the ACM/IEEE SC 2005 Conference*, 2005.

5. M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, , and R. W. Numrich. Improving performance via mini-applications. Technical report, Sandia National Laboratories, Tech. Rep., 2009.

6. S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, Apr. 2009.

7. T. Hoefler and R. Belli. Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC'15*, page 73:1–73:12, 2015.

8. T. Scogland, J. Azose, D. Rohr, S. Rivoire, and D. Hackenberg N. Bates. Node variability in large-scale power measurements: Perspectives from the Green500, Top500 and EEHPCWG. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '15*, 2015.

9. Michael A. Laurenzano, Ananta Tiwari, Allyson Cauble-Chantrenne, Adam Jundt, William A. Ward, Roy Campbell, and Laura Carrington. Characterization and bottleneck analysis of a 64-bit ARMv8 platform. *(preprint)*, 2016.

10. Y. Ukidave, D. Kaeli, U. Gupta, and K. Keville. Performance of the NVIDIA Jetson TK1 in HPC. *2015 IEEE International Conference on Cluster Computing*, page 533–534, Sept. 2015.

11. A. Haidar, S. Tomov, P. Luszczek, and J. Dongarra. Magma embedded: Towards a dense linear algebra library for energy efficient extreme computing. *High Performance Extreme Computing Conference (HPEC) IEEE*, pages 1–6, 2015.

12. J. E. Stone, M. J. Hallock, J. C. Phillips, J. R. Peterson, Z. Luthey-Schulten, and K. Schulten. Evaluation of emerging energy-efficient heterogeneous computing platforms for biomolecular and cellular simulation workloads. *International Parallel and Distributed Processing Symposium Workshop (IPDPSW) IEEE*, 2016.

13. Y. Lo, S. Williams, Van Straalen B., T. Ligocki, M. Cordery, N. Wright, M. Hall, and L. Oliker. *Roofline Model Toolkit: A practical tool for architectural and program analysis*,

volume 8966, chapter High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation, pages 129–148. Springer, 2015.

14. S. Williams. *Performance Tuning of Scientific Applications*. CRC Press, 2010.

15. В. Никольский и В. Стегайлов. Эффективность процессоров архитектуры ARM для расчетов классической молекулярной динамики. *Вычислительные методы и программирование*, 16(4):578–585, 2015.

16. Wolfgang Eckhardt, Alexander Heinecke, Reinhold Bader, Matthias Brehm, Nicolay Hammer, Herbert Huber, Hans-Georg Kleinhenz, Jadran Vrabec, Hans Hasse, Martin Horsch, Martin Bernreuther, ColinW. Glass, Christoph Niethammer, Arndt Bode, and Hans-Joachim Bungartz. 591 TFLOPS multi-trillion particles simulation on SuperMUC. *Supercomputing*, volume 7905 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2013.

17. Stefano Piana, John L Klepeis, and David E Shaw. Assessing the accuracy of physical models used in protein-folding simulations: quantitative evidence from long molecular dynamics simulations. *Current Opinion in Structural Biology*, 24:98 – 105, 2014.

18. Rio Yokota and Lorena A. Barba. Hierarchical n-body simulations with autotuning for heterogeneous systems. *Computing in Science & Engineering*, 14(3):30–39, 2012.

19. W. Michael Brown, Peng Wang, Steven J. Plimpton, and Arnold N. Tharrington. Implementing molecular dynamics on hybrid high performance computers – short range forces. *Computer Physics Communications*, 182(4):898 – 911, 2011.

20. W. Michael Brown, Axel Kohlmeyer, Steven J. Plimpton, and Arnold N. Tharrington. Implementing molecular dynamics on hybrid high performance computers – particle–particle particle-mesh. *Computer Physics Communications*, 183(3):449 – 459, 2012.

21. C. R. Trott, L. Winterfeld, and P. S. Crozier. General-purpose molecular dynamics simulations on GPU-based clusters. *ArXiv e-prints*, September 2010.

22. Lammps web site — benchmarks. `http://lammps.sandia.gov/bench.html`

Performance of MD-algorithms on hybrid systems-on-a-chip Nvidia Tegra K1 and X1

V.P. Nikolskiy[1,2], V.S. Vecher[2,3], V.V. Stegailov[2]

National Research University "Higher School of Economics"[1],
Joint Institute for High Temperatures of the RAS[2],
Moscow Institute of Physics and Technology[3]

In this paper we consider the efficiency of hybrid systems-on-a-chip for high-performance calculations. Firstly, we build the Roofline performance models for the subject systems using Empirical Roofline Toolkit and compare the results with the theoretical estimates. Secondly, we use LAMMPS as an example of the molecular dynamic package to demonstrate its performance and efficiency in various configurations running on the new devices. The results are discussed in the context of previously collected data on the performance of this MD package on different processors. Besides we describe an attempt to divide the compute-bound and memory-bound modes of MD algorithm what comply with the principle of Roofline.

Keywords: Kepler, Maxwell, ARM, system on a chip, molecular dynamics, Roofline

References

1. G. Mitra, B. Johnston, A.P. Rendell, E. McCreath, and Jun Zhou. Use of simd vector operations to accelerate application code performance on low-powered arm and intel platforms. In *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International*, pages 1107–1116, May 2013.

2. Kristopher Keipert, Gaurav Mitra, Vaibhav Sunriyal, Sarom S. Leang, Masha Sosonkina, Alistair P. Rendell, and Mark S. Gordon. Energy-efficient computational chemistry: Comparison of x86 and ARM systems. *J. Chem. Theory Comput.*, 11(11):5055–5061, 2015.

3. Harold J. Curnow and Brian A. Wichmann. A synthetic benchmark. *The Computer Journal*, 19(1):43 – 49, 1976.

4. E. Strohmaier and Shan Hongzhang. Apex-Map: A global data access benchmark to analyze HPC systems and parallel programming paradigms. *Proceedings of the ACM/IEEE SC 2005 Conference*, 2005.

5. M. A. Heroux, D. W. Doerfler, P. S. Crozier, J. M. Willenbring, H. C. Edwards, A. Williams, M. Rajan, E. R. Keiter, H. K. Thornquist, , and R. W. Numrich. Improving performance via mini-applications. Technical report, Sandia National Laboratories, Tech. Rep., 2009.

6. S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, Apr. 2009.

7. T. Hoefler and R. Belli. Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC'15*, page 73:1–73:12, 2015.

8. T. Scogland, J. Azose, D. Rohr, S. Rivoire, and D. Hackenberg N. Bates. Node variability in large-scale power measurements: Perspectives from the Green500, Top500 and EEHPCWG. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '15*, 2015.

9. Michael A. Laurenzano, Ananta Tiwari, Allyson Cauble-Chantrenne, Adam Jundt, William A. Ward, Roy Campbell, and Laura Carrington. Characterization and bottleneck analysis of a 64-bit ARMv8 platform. *(preprint)*, 2016.

10. Y. Ukidave, D. Kaeli, U. Gupta, and K. Keville. Performance of the NVIDIA Jetson TK1 in HPC. *2015 IEEE International Conference on Cluster Computing*, page 533–534, Sept. 2015.

11. A. Haidar, S. Tomov, P. Luszczek, and J. Dongarra. Magma embedded: Towards a dense linear algebra library for energy efficient extreme computing. *High Performance Extreme Computing Conference (HPEC) IEEE*, pages 1–6, 2015.

12. J. E. Stone, M. J. Hallock, J. C. Phillips, J. R. Peterson, Z. Luthey-Schulten, and K. Schulten. Evaluation of emerging energy-efficient heterogeneous computing platforms for biomolecular and cellular simulation workloads. *International Parallel and Distributed Processing Symposium Workshop (IPDPSW) IEEE*, 2016.

13. Y. Lo, S. Williams, Van Straalen B., T. Ligocki, M. Cordery, N. Wright, M. Hall, and L. Oliker. *Roofline Model Toolkit: A practical tool for architectural and program analysis*, volume 8966, chapter High Performance Computing Systems. Performance Modeling, Benchmarking, and Simulation, pages 129–148. Springer, 2015.

14. S. Williams. *Performance Tuning of Scientific Applications.* CRC Press, 2010.

15. V. Nikolskiy, V. Stegailov. *Efficiency of ARM processors for classical molecular dynamics calculations* Journal of Physics: Conference Series, 2016, 681, 012049.

16. Wolfgang Eckhardt, Alexander Heinecke, Reinhold Bader, Matthias Brehm, Nicolay Hammer, Herbert Huber, Hans-Georg Kleinhenz, Jadran Vrabec, Hans Hasse, Martin Horsch, Martin Bernreuther, ColinW. Glass, Christoph Niethammer, Arndt Bode, and Hans-Joachim Bungartz. 591 TFLOPS multi-trillion particles simulation on SuperMUC. In JulianMartin Kunkel, Thomas Ludwig, and HansWerner Meuer, editors, *Supercomputing*, volume 7905 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2013.

17. Stefano Piana, John L Klepeis, and David E Shaw. Assessing the accuracy of physical models used in protein-folding simulations: quantitative evidence from long molecular dynamics simulations. *Current Opinion in Structural Biology*, 24:98 – 105, 2014.

18. Rio Yokota and Lorena A. Barba. Hierarchical n-body simulations with autotuning for heterogeneous systems. *Computing in Science & Engineering*, 14(3):30–39, 2012.

19. W. Michael Brown, Peng Wang, Steven J. Plimpton, and Arnold N. Tharrington. Implementing molecular dynamics on hybrid high performance computers – short range forces. *Computer Physics Communications*, 182(4):898 – 911, 2011.

20. W. Michael Brown, Axel Kohlmeyer, Steven J. Plimpton, and Arnold N. Tharrington. Implementing molecular dynamics on hybrid high performance computers – particle–particle particle-mesh. *Computer Physics Communications*, 183(3):449 – 459, 2012.

21. C. R. Trott, L. Winterfeld, and P. S. Crozier. General-purpose molecular dynamics simulations on GPU-based clusters. *ArXiv e-prints*, September 2010.

22. Lammps web site — benchmarks. `http://lammps.sandia.gov/bench.html`