ИСПОЛЬЗОВАНИЕ МНОГОЯДЕРНЫХ ПРОЦЕССОРОВ НА ЗАДАЧАХ ДИСКРЕТНОГО МОДЕЛИРОВАНИЯ

А.А. Корж, Д.В. Макагон, А.Б. Шворин

В данной работе рассматривается параллельная реализация модели СКСН (суперкомпьютера стратегического назначения) «Ангара» — сверхпроизводительной вычислительной системы, основанной на отечественном мультитредовом процессоре J7/J10 с аппаратной поддержкой распределенной общей памяти. Для реализации модели используется модель вычислений, управляемых сообщениями (message-driven), а именно язык программирования Charm++. Исследуется производительность модели на многоядерных вычислительных узлах с микропроцессорами AMD Barcelona и Intel Clovertown.

ВВЕДЕНИЕ

В последние несколько лет широкое распространение получили коммерческие многоядерные процессоры. Известно, что это было вызвано в основном технологическими сложностями, такими как ограниченная скорость распространения сигнала в кристалле, ограничивающими рост вычислительной мощности одного ядра, но позволяющими разместить несколько ядер в одном кристалле. Однако до сих пор остро стоит проблема эффективного использования нескольких ядер. Возможные традиционные варианты программирования, такие как POSIX-треды или OpenMP оказываются порой неудобными или неэффективными. Использование парадигмы MPI слишком усложняет программирование и не вполне соответствует многоядерным процессорам, хотя следует отметить, что современные реализации MPI по нашим измерениям очень хорошо оптимизированы в случае обменов между ядрами одного узла, без использования коммуникационной сети.

Мы решили попробовать исследовать эффективность применения многоядерных процессоров на имитационной модели, написанной с помощью параллельного языка программирования Charm++. Выбор языка был обусловлен требованием возможности запуска модели СКСН для большого количества узлов (до 32 тысяч) и высокой продуктивностью параллельного программирования.

ЯЗЫК ПРОГРАММИРОВАНИЯ CHARM++ И МОДЕЛЬ ВЫЧИСЛЕНИЙ, УПРАВЛЯЕМЫХ СООБЩЕНИЯМИ

Charm++ [1] — переносимый объектно-ориентированный параллельный язык программирования, реализованный как библиотека для C++. Модель вычислений, управляемых сообщениями (message-driven), отличается от традиционных параллельных моделей программирования, таких как модель передачи сообщений и модели с общей памятью, и помогает писать программы, слабо чувствительные к задержкам в сети.

В процессе работы программы создается множество объектов (chare-объекты), каждый из которых имеет набор entry-методов, которые могут быть асинхронно вызваны на удаленных процессорах. Вычисления (исполнение методов) инициируются приходом сообщений. В свою очередь, эти вычисления, вызванные получением сообщений, могут создавать новые сообщения, предназначенные потенциально удаленным процессорам, таким образом, вызывая новые вычисления на других процессорах. Коммуникации между объектами осуществляются только посредством асинхронных вызовов неблокируемых методов, не возвращающих значения. Распределение объектов по процессорам производится динамически рантаймом системы.

КРАТКОЕ ОПИСАНИЕ ИМИТАЦИОННОЙ МОДЕЛИ

На Charm++ была реализована модель СКСН «Ангара» [3]. Моделируемый суперкомпьютер представляет собой множество процессоров, подключенных к маршрутизаторам, которые в свою очередь соединены друг с другом в некоторую топологию. Каждый узел помещается в отдельный объект и состоит из маршрутизатора и нескольких процессоров (для простоты изложения, будем рассматривать простейший случай сети тип тор, в котором узел состоит из одного маршрутизатора и подключенного к нему процессора). В таком случае схема моделирования упрощается, так как процессор обменивается данными только с собственным маршрутизатором, локально расположенным в том же объекте, а маршрутизаторы обмениваются сообщениями между различными chare-объектами.

Процессор и маршрутизатор моделируются потактово, причем их тактовые частоты могут различаться. Модель процессора не зависит от модели сети и является динамически подключаемой, в частности, возможно подключение моделей с различной детализацией, а стало быть, и производительностью.

Entry-методы объекта-узла принимают сообщения от соседних узлов. Схема работы такова: узел выполняет один такт работы, затем отправляет сообщения соседним узлам и, наконец, синхронизируется со всеми другими узлами на барьере. Данная схема является достаточно примитивной, однако ее реализация была необходима для отладки и получения первых результатов. В настоящее время реализуется и отлаживается

асинхронная схема моделирования, позволяющая сэкономить на барьерных синхронизациях, используя тот факт, что задержка передачи сообщений между узлами составляет несколько десятков тактов.

УСЛОВИЯ ЭКСПЕРИМЕНТА

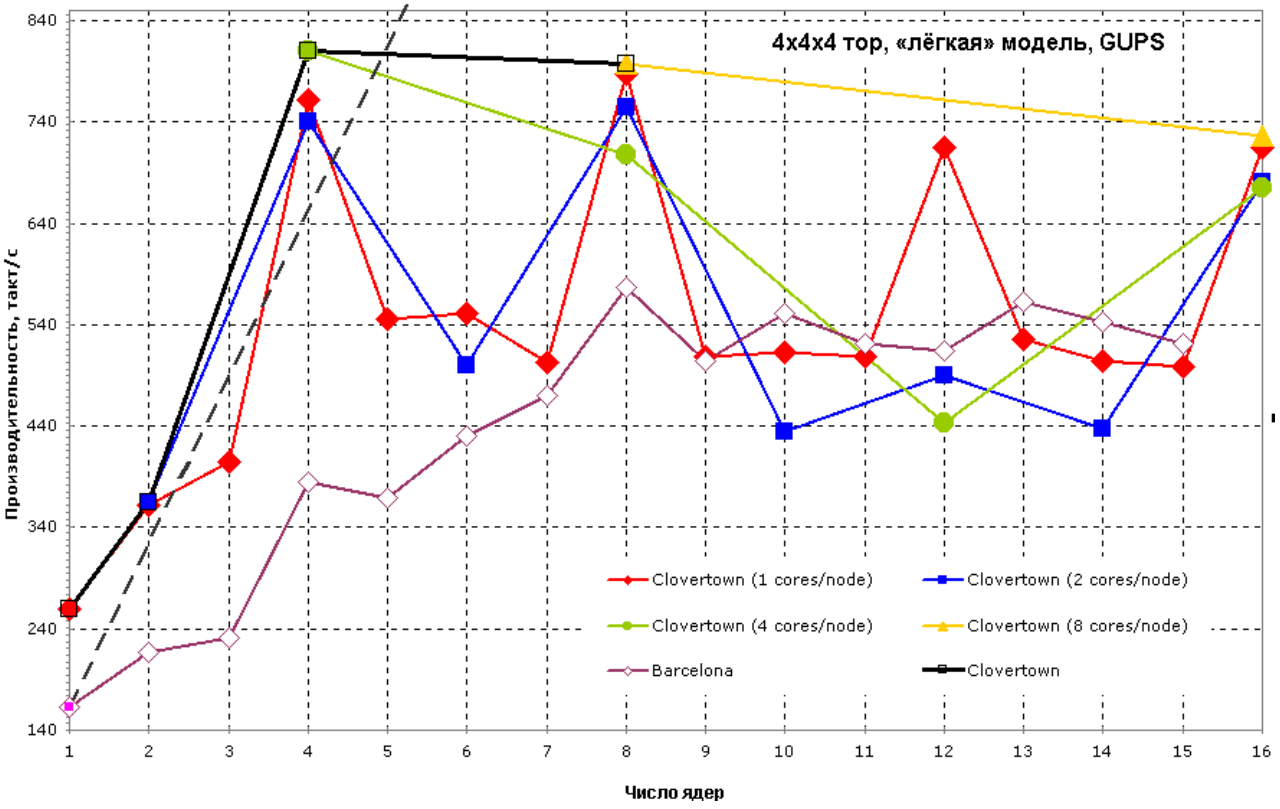
Рассматривается два варианта модели «Ангары»: «лёгкая» и «тяжелая», что соответствует сложности моделируемой аппаратуры. Тяжелый вариант соответствует 8-ми ядерному микропроцессору J10, а легкий — двухядерному J7 [3]. Время на процессоре Barcelona симуляции одного такта для одного узла составляет около 50 мкс для «лёгкой» модели и 180 мкс для «тяжелой». Для Clovertown, соответственно, 30 мкс и 100 мкс.

Ниже приведены результаты моделирования сети из 64 узлов (тор 4x4x4) на тесте GUPS (Giga-updates per second) [2,4]. Данный тест характеризуется довольно тяжелой нагрузкой на подсистему распределенной общей памяти, а тем самым нагрузка на моделируемую сеть оказывается близка к максимальной, отрицательно влияя на производительность модели, которая зависит, прежде всего, от количества моделируемых сетевых обменов.

Было использовано стандартное распределение моделируемых узлов (т. е. chare-объектов) по ядрам, принятое в Charm++. Этим объясняются особенности графиков в точках, где число ядер кратно четырем, — там результаты несколько лучше. При использовании специального распределения, учитывающего связность моделируемой сети, графики сглаживаются за счет небольшого улучшения результатов в «плохих» точках.

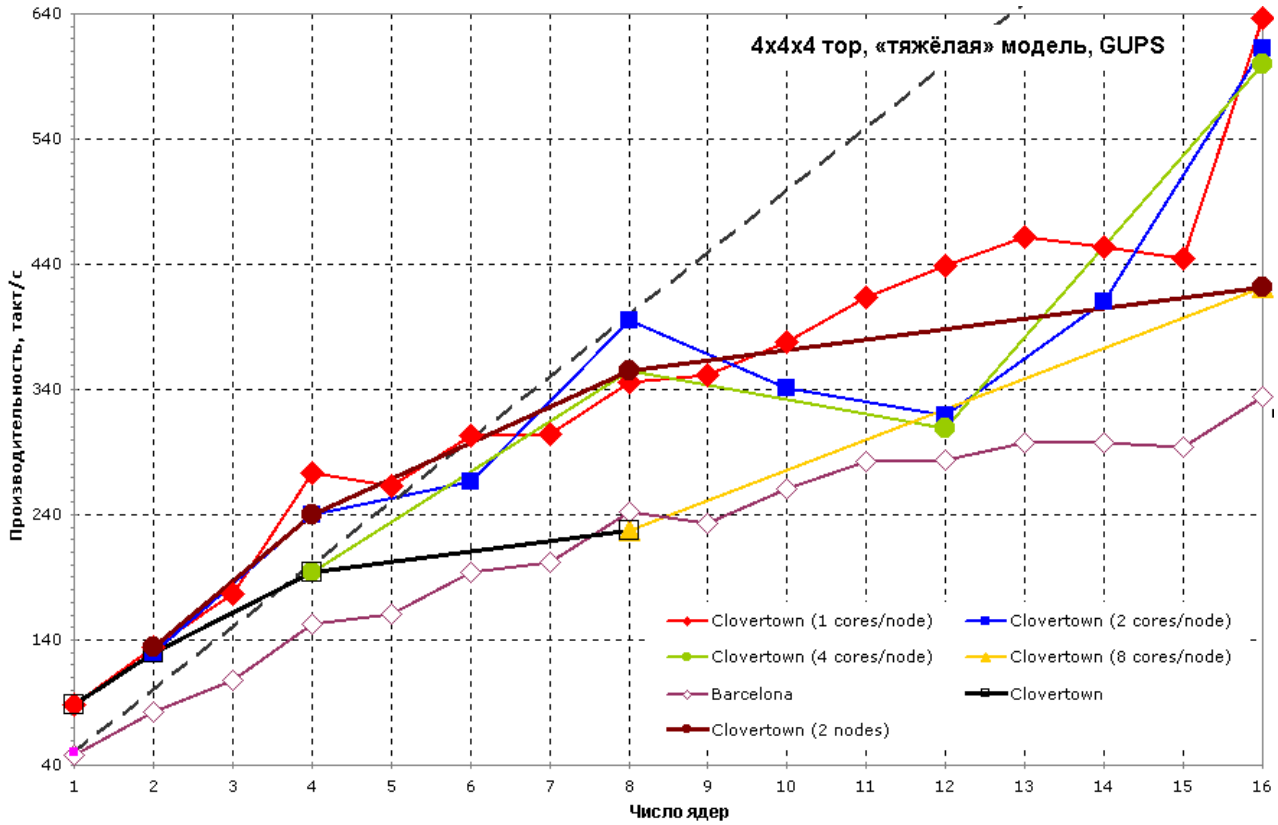
Аппаратная характеристика используемых платформ следующая: для процессора AMD Barcelona использовался один узел с 16 ядрами (четыре 4-ядерных процессора AMD Quad Core 8350 2.0 GHz), объем оперативной памяти 16ГБ, для процессора Intel Clovertown использовался кластер Межведомственного Суперкомпьютерного Центра РАН МВС-100000М, состоящий из узлов, связанных сетью Infiniband DDR. В каждом узле 8 ядер (два 4-ядерных процессора Intel Xeon X5365 3.0 GHz), объем оперативной памяти 4 ГБ.

РЕЗУЛЬТАТЫ И ВЫВОДЫ



На графиках видно, что «тяжелая» модель распараллеливается лучше, чем «лёгкая». Это вполне закономерный результат, поскольку в обоих случаях затраты на синхронизацию одинаковы, а вычислений больше в «тяжелой» модели. Также можно увидеть, что для легкой модели лучший результат на микропроцессоре Intel достигается при использовании 4-х ядер (ускорение 3.11 раз), большее количество ядер ни в одном узле, ни через сеть не дает прироста производительности. Причем использование ядер, расположенных в одном кристалле, оказывается незначительно лучше, чем четыре ядра, расположенных в разных кристаллах. Для процессора Barcelona лучшим является использование 8 ядер, на которых достигается ускорение в 3.53 раза. Это означает, что данная задача является небольшой, и большее число ядер не в состоянии повысить скорость ее работы. Проведя несложные расчеты, получаем, что для 4-х ядер Clovertown на

каждое приходится примерно 0.5 мс работы на такт между барьерными синхронизациями. Для 8 ядер AMD Barcelona — 0.4 мс.



Так же как для легкой модели, зависимость от количества используемых ядер на узел (для MVS100k) довольно слабая (смотрим на 4, 8, 16 ядер), причем, чем меньше ядер на узел, тем, как правило, выше производительность. За исключением случая полной загрузки узла (8 ядер), в котором производительность на тяжелой модели оказывается значительно ниже. Лучший результат на Clovertown достигается на 16 ядрах (ускорение 7.14 раз) используемых через сеть. Те же 16 ядер в двух узлах дают всего лишь ускорение в 4.79 раз. Таким образом эффект локальности здесь оказывается слабее, чем конфликты ядер одного кристалла при работе с памятью. Для процессора Barcelona максимальное ускорение достигается на 16 ядрах и составляет 6.81 раз. В сравнении с этим, один узел на базе процессоров Clovertown показывает масштабируемость намного хуже, причем изначально более быстрый Clovertown на 8 ядрах проигрывает Barcelona не только по масштабируемости, но и в абсолютной производительности. Представляется, что тяжелая модель более насыщена операциями доступа к памяти, и таким образом на процессоре Intel проявляются конфликты по памяти, в то время у имеющих собственный внутрикристальный контроллер памяти процессоров Barcelona данный эффект проявляется в меньшей мере.

Проведя аналогичные вычисления, получаем, что на тяжелой модели для 16 ядер Clovertown приходится 0.4 мс работы каждому ядру на такт, для 16 ядер Barcelona — 0.7 мс. Таким образом, можно сделать вывод, что необходимым условием масштабируемости является количество работы между барьерными синхронизациями более чем 500 мкс на ядро. Одним из способов увеличения масштабируемости является утяжеление модели, однако это не решает проблему низкой производительности легких моделей. Введение асинхронности в модель позволит уменьшить число барьерных синхронизаций, таким образом, увеличив количество работы, которое нужно сделать одному ядру между двумя барьерами.

ЛИТЕРАТУРА:

1. L. V. Kale and Sanjeev Krishnan, Charm++: Parallel Programming with Message-Driven Objects, Book Chapter in "Parallel Programming using C++", by Gregory V. Wilson and Paul Lu. MIT Press, 1996. pp 175-213. http://charm.cs.uiuc.edu/
2. http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/.
3. Слуцкин А., Эйсымонт Л. Российский суперкомпьютер с глобально адресуемой памятью // Открытые системы. 2007. №9. 20–21.
4. Волков Д., Фролов А. Оценка быстродействия нерегулярного доступа к памяти// Открытые системы. 2008. №1.