КЕНТАВР: СРЕДСТВО АВТОМАТИЧЕСКОГО РАСПАРАЛЛЕЛИВАНИЯ ТРАФАРЕТНЫХ ПРОГРАММ НА ГИБРИДНЫЕ СУПЕРКОМПЬЮТЕРЫ

Ю.А. Климов, А.Ю. Орлов, А.Б. Шворин

Введение

В последние пару лет появилось множество гибридных суперкомпьютеров — машин с графическими ускорителями (Tsubame 2.0, Ломоносов, К-100), и такие машины будут появляться еще (Titan, Bluewaters). Программы, разработанные для последовательного исполнения или исполнения на кластере без ускорителей необходимо переписывать для гибридных суперкомпьютеров (заметим в скобках, что такая ситуация — необходимость переписывания программного кода под вновь появившуюся аппаратуру — возникает в отрасли далеко не в первый раз).

Разрабатывать программы для суперкомпьютеров с ускорителями вычислений значительно сложнее, чем для «классических» суперкомпьютеров. И графические ускорители — не исключение. Хотя само по себе программирование ускорителя делается на почти привычном языке, организация данных для эффективной работы скорее всего потребуется отличная от той, что использовалась ранее. Кроме этого, ускорители добавляют новые явные уровни в иерархию памяти современных суперкомпьютеров, которые приходится учитывать при реализации.

Существует класс программ, для которых достаточно легко и естественно можно специфицировать всю необходимую информацию, чтобы распараллеливание программы можно было выполнить автоматически, а именно — трафаретные (stencil) программы. Разрабатываемая авторами статьи набор инструментов Кентавр позволяет автоматизировать разработку параллельной версии трафаретной программы для гибридных суперкомпьютеров, обеспечивая, в первую очередь, прозрачные обмены данными между памятью ускорителя вычислений и системной памятью, а также между вычислительными узлами суперкомпьютера.

Сразу оговоримся, что данная статья в частности и проект Кентавр [6] в целом не направлены на написание эффективного кода и организацию эффективных (то есть, с высоким КПД) вычислений как на обычных процессорах, так и на графических ускорителях. Это отдельные задачи, которые для трафаретных вычислений могут быть эффективно разрешены. Проект Кентавр в первую очередь ориентирован на быструю разработку достаточно эффективных параллельных программ для гибридных суперкомпьютеров на основе последовательной программы, автоматизируя необходимые обмены и избавляя тем самым программиста от большого объема трудоемкой работы [2]. То есть можно сказать, что разработка проекта ведется под лозунгом «высокая продуктивность при неплохой эффективности».

В эпоху суперкомпьютеров, состоящих из классических процессоров, выгода от использования специального инструмента для распараллеливания трафаретных программ была не очень велика — для квалифицированного специалиста преобразование последовательной трафаретной программы в параллельную с использованием MPI является чисто технической типовой задачей. При этом можно ожидать, что специализированный инструмент будет иметь ряд недостатков, связанных в первую очередь с доступностью, переносимостью и, самое главное, взаимодействием с более универсальными средствами (как MPI) в тех случаях, когда возможностей инструмента оказывается недостаточно. Несмотря на это, проекты по распараллеливанию трафаретных программ все же имеют некоторую историю. По-видимому, самым успешным из них является библиотека LibGeoDecomp [3], основанная на шаблонах C++.

Однако сложность программирования суперкомпьютеров с ускорителями заставляет пересмотреть имеющийся арсенал средств и предоставить автоматизацию там, где раньше это было нецелесообразно. Помимо обсуждаемого в данной работе инструмента Кентавр, за последний год появилось еще по крайней мере два проекта, нацеленных на распараллеливание трафаретных программ на гибридные суперкомпьютеры: 1) был реализован соответствующий back-end в рамках уже упомянутого проекта LibGeoDecomp; 2) для использования на суперкомпьютере Tsubame 2.0 был запущен проект Physis [4]. Также хочется отметить проект Microsoft Research Accelerator [5], который, хотя и не имеет пока выхода для гибридных суперкомпьютеров, ставит своей целью строить эффективные программы для совершенно различных типов аппаратуры, исходя из высокоуровневого описания трафаретных программ. В данной работе не будут обсуждаться сравнительные характеристики этих проектов в силу того, что все эти проекты еще молоды и относительно незрелы. На данный момент нам достаточно показать, что тема вызывает горячий интерес в мире и продемонстрировать возможности нашего подхода. Скажем только, что отличительной особенностью подхода Кентавра является отсутствие принципиальных ограничений на конфигурации сеток, возможность работы с неструктурными и адаптивными стеками.

Трафаретные вычисления

Существует широкий класс задач, где имеется пространство ячеек, над которым пошагово проводятся вычисления таким образом, что (новое) состояние данной ячейки на текущем шаге зависит от (старых)

состояний некоторого подмножества ячеек на предыдущем шаге. Сюда входят, например, сеточные задачи вычислительной механики, где состояние определяется значениями в узлах (ребрах, ячейках) сетки.

Важный подкласс образуют так называемые трафаретные вычисления (stencil codes) [7]. Его особенностью является то, что множество ячеек, от которых зависит состояние данной ячейки, устроено некоторым регулярным образом, а именно — пусть пространство представлено многомерным массивом, и значение следующего состояния данной ячейки массива зависит только от текущего состояния «соседних» ячеек. Понятие соседства определяется так: ячейка Y является соседней по отношению к ячейке X, если ее координаты (индексы в массиве) получаются сдвигом координат X на один из векторов из заданного набора. Этот набор векторов называется трафаретом и должен быть задан заранее. Трафаретные вычисления заключаются в том, что для каждой ячейки ее новое состояние определяется как функция от (старых) состояний ее соседей. Отметим, что вычисления для каждой ячейки происходят независимо от вычислений для других ячеек.

В качестве вырожденного примера трафаретной задачи можно привести вычислительную схему, где новое состояние ячейки зависит лишь от ее же старого состояния; здесь трафарет состоит из одного элемента (0, …,0). Другой, более содержательный, пример — схема Якоби для решения уравнения теплопроводности: в двумерном случае трафаретом является набор из четырех элементов {(0,1), (0,-1), (1,0), (-1,0)} (рис. 1). Код этой задачи, представленный ниже, в каждом шаге вычислений (во внешнем цикле) содержит последовательное применение двух трафаретов.

```
/* Шаги по итерациям */
for (s = 0; s < N_STEP; s++) {
   /* Трафаретный цикл. Трафарет: {(0,1), (0,-1), (1,0), (-1,0)}. */
   for (i = 1; i < (MX-1); i++) {
   for (j = 1; j < (MY-1); j++) {
      f_new[i][j] = (f[i][j+1]+f[i][j-1]+f[i+1][j]+f[i-1][j])*0.25;
   }}
   /* Трафаретный цикл. Трафарет: {(0,0)}. */
   for (i = 1; i < (MX-1); i++) {
   for (j = 1; j < (MY-1); j++) {
      f[i][j] = f_new[i][j];
   }}
}
```

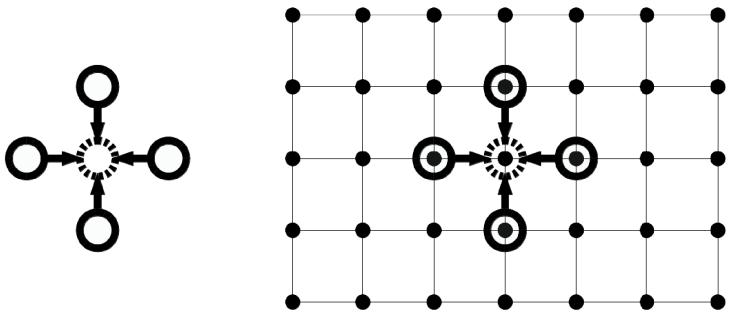

Рис. 1. Трафарет {(0,1), (0,-1), (1,0), (-1,0)} и его наложение на узел сетки.

Трафаретная задача может в общем случае содержать несколько трафаретов, каждый из которых применяется независимо.

Многие сеточные методы на регулярных неадаптивных сетках также являются по сути трафаретными. Поэтому позволим себе трактовать понятие трафарета и трафаретных вычислений более общо. А именно — не будем ограничиваться многомерным массивом как представлением пространства. Пусть оно по-прежнему состоит из элементов (ячеек), но обладает заданной структурой. И пусть имеется несколько преобразований — трансляций — пространства в себя, определенных, возможно, не на всем пространстве, но, по крайней мере, на большей его части. Тогда, согласно более общему определению, трафаретом будет называться набор трансляций, а соответствующее этому трафарету вычисление заключается в том, что новое состояние ячейки X зависит от ячеек, получающихся при применении к X трафарета (при условии, что все трансляции определены для данной ячейки X) (рис. 2).

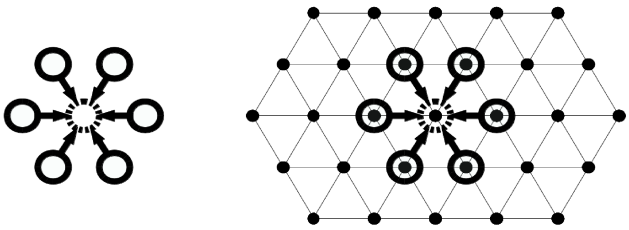

Рис. 2. Трафарет для непрямоугольной сетки и его наложение на узел сетки.

Распараллеливание трафаретных программ

Программы, состоящие в основном из циклов с независимыми витками, легко и эффективно распараллеливаются на системах с общей памятью (например, с помощью системы OpenMP). В общем случае, однако, перенос таких программы на суперкомпьютеры с распределенной памятью может быть весьма нетривиальным делом.

Класс трафаретных программ обладает тем свойством, что заранее (до начала основных вычислений) известно, к каким данным могут происходить обращения на каждом шаге цикла. Это свойство позволяет создавать довольно эффективные реализации такого рода задач на многоузловой аппаратуре — удается заранее распределить данные и вычисления между узлами суперкомпьютера и обеспечить обмены необходимыми данными между узлами.

Реализация проекта Кентавр исходит из следующих принципов.

1. Программист рассматривает свою программу как последовательную, за исключением вызовов функций редукции.
2. Шаблоны (трафареты) должны специфицироваться декларативно, как высокоуровневые свойства программы (алгоритма).
3. Должна быть возможна работа с неструктурными, адаптивными сетками, заданными на областях произвольной формы.
4. Должна быть возможна балансировка вычислений.
5. Должно быть возможно совмещение вычислений с обменами (как между узлами, так и между ускорителями и CPU).
6. Программа должна работать на суперкомпьютере с ускорителями CUDA. В будущих версиях возможно расширение на другие типы ускорителей.
7. На данный момент корректная спецификация трафаретов является задачей программиста, однако в будущих версиях возможно извлечение этой информации автоматически путем анализа кода циклов.

Набор инструментов Кентавр

Разрабатываемый авторами статьи набор инструментов Кентавр [6] для автоматизации распараллеливания ориентирован прежде всего на трафаретные задачи. Что, впрочем, не ограничивает его применение и на более общем классе задач, для которых заранее (как минимум, по завершении инициализации) известна схема передачи данных. Этот набор предназначен для быстрого создания достаточно эффективной параллельной программы для исполнения на гибридных суперкомпьютерах.

Пользователь Кентавра рассматривает свою программу как последовательную в целом (аналогично OpenMP). Дополнительно он задает трафареты — высокоуровневые свойства программы (алгоритма), описывающие зависимости при доступе к элементам массивов.

На основе трафаретов Кентавр определяет распределение данных по узлам суперкомпьютера (с учетом необходимых теневых граней) и распределение вычислений. Обеспечивает создание распределенных массивов с данными и передачи необходимых данных как между узлами суперкомпьютера, так и между процессорами и ускорителями вычислений. Расположение данных производится по классической методике [1], в соответствии с которой основные циклы обхода сетки в последовательном случае без изменений используются для обхода локальных областей на каждом из узлов суперкомпьютера.

В итоге пользователю Кентавра предоставляются библиотечные средства для описания конкретной используемой области и сетки. Массивы с данными также должны заводиться с помощью специальных вызовов. Для каждого цикла необходимо задать трафарет, описывающий, к каким данным производится обращение в данном цикле. В текущей версии дополнительно особым образом требуется аннотировать заголовки циклов, чтобы система могла сгенерировать дополнительный код, необходимый для реализации обменов. В будущих версиях планируется избавиться от дополнительного аннотирования. Внутри циклов дополнительные преобразования в общем случае не требуются.

Для исполнения на графическом ускорителе вычислений каждое тело цикла дополнительно компилируется в код для графического ускорителя. По заданным трафаретам определяются данные, необходимые для вычислений, которые будут передаваться на ускоритель и обратно.

Использование Кентавр позволяет без изменений в программе пользователя проводить дополнительные оптимизации для повышения производительности: например, совмещение передачи данных и вычислений и балансировку нагрузки на узлы суперкомпьютера путем изменения распределения данных.

Заключение

Трафаретные программы обладают тем свойством, что вся информация о зависимостях между данными в процессе вычислений может быть специфицирована на этапе создания программы достаточно просто и естественно. После того как такая информация задана, распараллеливание (как и другие преобразования) программы можно успешно выполнять в автоматическом режиме.

В частности, проект Кентавр [6] посвящен отображению трафаретных программ на гибридные суперкомпьютеры [2]. В рамках проекта не рассматриваются вопросы эффективного задействования конкретных вычислительных устройств (процессоров, ускорителей), однако в полной мере решаются вопросы распределения и передачи данных между ними.

Работа выполняется при поддержке Министерства образования и науки Российской Федерации (госконтракт № 07.514.11.4014).

ЛИТЕРАТУРА:

1. Андрианов А.Н., Ефимкин К.Н. Подход к параллельной реализации численных методов на неструктурированных сетках // Вычислительные методы и программирование: новые вычислительные технологии. 2007. Т. 8. № 2. С. 6-17.
2. Климов Ю.А., Орлов А.Ю., Шворин А.Б. Перспективные подходы к созданию масштабируемых приложений для суперкомпьютеров гибридной архитектуры // Программные системы: теория и приложения: электронный научный журнал. 2011. № 4 (8). С. 45-59. URL: http://psta.psiras.ru/read/psta2011_4_45-59.pdf.
3. Andreas Schäfer, and Dietmar Fey. LibGeoDecomp: A Grid-Enabled Library for Geometric Decomposition Codes // Proceedings of the 15th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface. Pp. 285-294. Springer-Verlag Berlin, Heidelberg. 2008.
4. Naoya Maruyama, Tatsuo Nomura, Kento Sato, and Satoshi Matsuoka. Physis: An Implicitly Parallel Programming Model for Stencil Computations on Large-Scale GPU-Accelerated Supercomputers // Proceedings of the 2011 ACM/IEEE Conference on Supercomputing (SC'11). Pp. 1-12. Seattle, WA, USA. Nov 2011.
5. Satnam Singh. Computing without Processors // ACM Queue 9, 6, Pages 50:50--50:63. June 2011. URL: http://queue.acm.org/detail.cfm?id=2000516
6. Проект Кентавр // URL: http://centaur.botik.ru/
7. Stencil codes // URL: http://en.wikipedia.org/wiki/Stencil_codes