Отображение DVMH-программ на кластеры с ускорителями*

М.Н. Притула

ФГБУН Институт прикладной математики им. М.В. Келдыша РАН

В 2011 году для новых гетерогенных и гибридных суперкомпьютерных систем в Институте прикладной математики им. М.В. Келдыша РАН были разработаны языки программирования высокого уровня, представляющие собой стандартные языки Фортран и Си, расширенные директивами. В статье дается представление о логической структуре DVMH-программы и ходе ее выполнения на кластере с ускорителями; описываются методы и режимы отображения DVMH-программы на кластер с ускорителями, включая динамическое планирование выполнения DVMH-программы на многоядерных процессорах и ускорителях. Освещаются также возможности сравнительной отладки и особенности ее реализации в системе поддержки выполнения DVMH-программ.

1. Введение

В последнее время появляется много вычислительных кластеров с установленными в их узлах ускорителями. В основном, это графические процессоры компании NVIDIA. В 2012 году начинают появляться кластеры с ускорителями другой архитектуры – Xeon Phi от компании Intel. Так, в списке Top500 [2] самых высокопроизводительных суперкомпьютеров мира, объявленном в ноябре 2012 года, 62 машины имеют в своем составе ускорители, из них 50 машин имеют ускорители NVIDIA, 7 – Intel, 3 – AMD/ATI, 2 – IBM. Данная тенденция заметно усложняет процесс программирования кластеров, так как требует освоения на достаточном уровне сразу нескольких моделей и языков программирования. Традиционным подходом можно назвать использование технологии MPI для разделения работы между узлами кластера, а затем технологий CUDA (или OpenCL) и OpenMP для загрузки всех ядер центрального и графического процессоров.

С целью упрощения программирования распределенных вычислительных систем, были предложены высокоуровневые языки программирования, основанные на расширении директивами стандартных языков такие, как HPF [3], Fortran-DVM [1,9], C-DVM [1,10]. Также были предложены модели программирования и соответствующие основанные на директивах расширения языков для возможности использования ускорителей такие, как HMPP [4], PGI Accelerator Programming Model [5], OpenACC [6], hiCUDA [7].

В 2011 году в Институте прикладной математики им. М.В. Келдыша РАН была расширена модель DVM для поддержки кластеров с ускорителями [8]. Это расширение названо DVMH и позволяет с небольшими изменениями перевести DVM-программу для кластера в DVMH-программу для кластера с ускорителями.

Одним из важных аспектов функционирования такой программной модели, как DVMH является вопрос отображения исходной программы на все уровни параллелизма и разнородные вычислительные устройства. Важными задачами механизма отображения является обеспечение корректного исполнения всех поддерживаемых языком конструкций на разнородных вычислительных устройствах, балансировка нагрузки между вычислительными устройствами, а также выбор оптимального способа исполнения каждого участка кода на том или ином устройстве.

2. Схема выполнения DVMH-программы

Под DVMH-программой будем понимать программу на языке Fortran-DVMH или программу на языке C-DVMH.

* Работа выполнена при финансовой поддержке гранта Президента РФ НШ-4307.2012.9 и грантов РФФИ №11-01-00246, 12-01-33003-мол_а_вед, 12-07-31204-мол_а.

Выполнение DVMH-программы можно представить, как выполнение последовательности вычислительных регионов и участков между ними, которые будем называть внерегионным пространством. Код во внерегионном пространстве выполняется на центральном процессоре, тогда как для вычислительных регионов возможно их исполнение на разнородных вычислительных устройствах. Внутри, равно как и вне регионов могут быть как параллельные циклы, так и последовательные участки программы.

Параллелизм в DVMH-программах проявляется на нескольких уровнях:

1. Распределение данных и вычислений по MPI-процессам. Этот уровень задается специальными директивами распределения или перераспределения данных и спецификациями параллельных подзадач и циклов.
2. Распределение данных и вычислений по вычислительным устройствам при входе в вычислительный регион.
3. Параллельная обработка в рамках конкретного вычислительного устройства. Этот уровень появляется при входе в параллельный цикл, находящийся внутри региона.

Наличие этих уровней дает возможность органично отобразить программу на кластер с многоядерными процессорами и ускорителями в узлах.

Исполнение DVMH-программы начинается синхронно всеми запущенными процессами в модели SPMD. На каждый процесс выделяется одна основная последовательная нить исполнения.

При входе в вычислительный регион каждый процесс независимо выполняет дополнительное к межпроцессному распределение данных, используемых данным вычислительным регионом, по вычислительным устройствам, выбранным для выполнения региона. На этом этапе производится динамическое планирование с целью балансировки нагрузки и минимизации временных затрат на перемещения данных.

При входе в параллельный цикл внутри региона каждый процесс разделяет работу в соответствии с распределением данных по вычислительным устройствам. Затем он выбирает для каждого устройства метод обработки части цикла на конкретном устройстве, называемый обработчиком, а также оптимизационные параметры для выбранного обработчика. На этом этапе производится динамическая настройка оптимизационных параметров, включая количество ЦПУ-нитей для обработчиков на ЦПУ, а также размер и форму блока нитей для CUDA-обработчиков с целью минимизации времени исполнения на каждом отдельном устройстве.

Параллельный цикл внутри региона при исполнении распадается на несколько частей, каждая из которых обрабатывается некоторым обработчиком на некотором вычислительном устройстве. На этом этапе собирается основная информация для отладки эффективности DVMH-программы.

При выходе из вычислительного региона собирается дополнительная информация для целей динамического планирования выполнения региона, а также может быть осуществлена сравнительная отладка с целью контроля корректности результатов, полученных при счете на ускорителях.

Напомним также, что перемещения данных между регионами регулируются в директивах регионов соответствующими указаниями для используемых в них переменных, а перемещения данных во внерегионном пространстве – с помощью специальных исполнительных директив актуализации.

3. Отображение данных

В DVMH-программах все распределенные данные распределяются блочно: каждое распределенное измерение делится несколькими точками на отрезки. При этом есть возможность по каждому измерению распределенного массива задавать или равномерное блочное распределение, или распределение взвешенными блоками, т.е. с учетом заданного вектора весов. Эти указания непосредственно выполняются при распределении данных между MPI-процессами. Вложенные распределения, появляющиеся при входе в вычислительный регион, строятся с использованием этой информации, но, в силу разнородности вычислительных устройств, могут иметь отличающуюся от внешней схему распределения.

В DVM-системе разрабатываются три режима распределения данных по вычислительным устройствам в точках входа в регионы:

1. Простой статический режим.
2. Динамический режим с подбором схемы распределения.
3. Статический режим с использованием подобранной схемы распределения.

Рассмотрим подробнее эти режимы распределения.

3.1 Простой статический режим

В этом режиме в каждом регионе распределение производится одинаково. Пользователем задается вектор весов вычислительных устройств, имеющихся в каждом узле кластера (или они могут быть грубо определены автоматически), затем эти веса накладываются на параметры внешнего распределения данных, которое по каждому распределенному измерению может быть распределено как равномерно, так и с заданным вектором весов. В таком режиме сводятся к минимуму перемещения данных в связи с их перераспределением, но не учитывается различное соотношение производительности вычислительных устройств на разных участках кода.

3.2 Динамический режим с подбором схемы распределения

В этом режиме в каждом регионе распределение выбирается на основе постоянно пополняющейся истории запусков данного региона и его соседей, определяющихся динамически.

Каждый регион в данном режиме рассматривается в виде нескольких родственных инстанций, каждая из которых определяется парой (регион, соответствие данных), где под соответствием данных понимается соответствие используемых в коде региона локальных переменных реальным переменным (массивам или скалярам).

Для каждой инстанции определяются:

- Динамически предшествующие инстанции, являющиеся поставщиками актуальных входных данных.
- Зависимость времени работы от распределения данных, причем принимаются во внимание все инстанции одного и того же региона с учетом их разных соответствий данных.
- Общее время выполнения, как сумма всех времен выполнения инстанции.
- Количество вхождений

Зависимость времени работы от распределения данных строится в виде табличной функции времени от распределений распределенных массивов, которая является суммой таких же табличных функций, построенных для параллельных циклов, последовательных участков и хост-секций, содержащихся внутри данной инстанции региона.

В этом режиме периодически включается построение субоптимальной схемы распределения данных во всех инстанциях регионов на основе накопленных сведений в целом для программы, анализируя целиком последовательность инстанций регионов и их характеристики выполнения. При построении таких схем учитываются как внутренние показатели инстанций регионов в виде зависимости времени работы от распределения данных, так и последовательность исполнения инстанций с целью минимизации в том числе и временных затрат на перераспределение данных. После построения такой схемы, она применяется и происходит дальнейшее накопление характеристик и информации о регионах.

Есть возможность записи построенных схем распределения в файл для использования в последующих запусках программы, как в этом режиме, так и в третьем режиме. Также в качестве начального приближения может быть использован вектор весов вычислительных устройств в том же виде, как и для простого статического режима.

Из данного режима возможен переход в третий режим в любой точке выполнения программы, что обеспечивает упрощенную схему подбора и использования схемы распределения без необходимости сохранения параметров в файл и повторного запуска программы.

3.3 Статический режим с использованием подобранной схемы распределения

В этом режиме в каждом регионе распределение выбирается на основе предоставленной схемы распределения, построенной при работе программы во втором режиме, причем есть возможность, как перейти в этот режим непосредственно из второго, так и использовать схему распределения из файла, полученного в результате работы программы во втором режиме. При использовании схемы распределения из файла не гарантируется ее корректное использование в случае, если параметры программы были изменены, в особенности это касается тех параметров, которые влияют на путь выполнения программы (выбор другого метода расчета, отключение или включение этапов расчета).

4. Отображение вычислений

При отображении на разнородные вычислители параллельных циклов внутри вычислительных регионов необходимо решать задачи двух типов:

- Выбор метода обработки (обработчика) для конкретного вычислительного устройства, а также параметров для конкретного обработчика.
- Выполнение отображения на параллельную архитектуру.

Выбор обработчика и его параметров производится для всех вычислительных устройств в каждой инстанции вычислительного региона, содержащего данный параллельный цикл.

Так как вычислители разнородные, то разберем отдельно эти вопросы для ЦПУ и графических процессоров с архитектурой CUDA.

4.1 Отображение на ЦПУ

Обработчиков для ЦПУ может быть несколько, но, как правило, он присутствует в единственном числе и может быть параметризован количеством нитей и методом планирования их расписания.

В простом статическом режиме выбирается ЦПУ-обработчик по-умолчанию, а в других режимах – или исходя из имеющейся схемы распределения, или лучший по результатам произведенных запусков.

В простом статическом режиме количество нитей выбирается максимально доступное. В других режимах – максимальное из тех, которые дают лучшее время, чем меньшее количество нитей.

Метод планирования расписания нитей используется динамический или статический (в терминах OpenMP) и выбирается динамически, исходя из равномерности загрузки нитей при статическом планировании.

Выполнение отображения на ЦПУ делается простейшим образом с использованием технологии OpenMP.

4.2 Отображение на CUDA-устройство

Обработчиков для CUDA-устройств может быть несколько – могут быть применены разные подходы к оптимизации на уровне кода CUDA-ядра, использованы различные целевые архитектуры при их компиляции. Каждый обработчик может быть параметризован размерами CUDA-блока нитей и способом обработки редукционных операций.

В простом статическом режиме для всех CUDA-устройств выбирается CUDA-обработчик по-умолчанию, а в других режимах – или исходя из имеющейся схемы распределения, или лучший по результатам произведенных запусков.

Размеры CUDA-блока нитей для параллельного цикла могут быть заданы в тексте программы, и в этом случае будут использоваться эти размеры. Если же размеры CUDA-блока нитей не были заданы в тексте программы, то в простом статическом режиме блок выбирается фиксированным, заданным в настройках DVM-системы. В других режимах – или исходя из имеющейся схемы распределения, или лучший по результатам произведенных запусков.

Способ обработки редукционных операций может быть двух видов – с большим потреблением памяти устройства, но с более быстрым алгоритмом или с меньшим потреблением памяти устройства, но с менее быстрым алгоритмом. Если этот параметр поддерживается обработчиком, то способ выбирается динамически на основании имеющегося в данный момент объема свободной памяти устройства.

Выполнение отображения на CUDA-устройство выполняется с использованием технологии CUDA на усмотрение обработчика. Стандартным практикуемым подходом является отображение ровно одного витка параллельного цикла на одну CUDA-нить, причем пространство витков параллельного цикла разбивается на блоки методом замощения, при котором внутренние три измерения цикла ставятся в соответствие с измерениями блока нитей, а решетка блоков полагается одномерной и отвечает за покрытие всего пространства витков блоками. Для циклов с регулярными зависимостями применяются особого вида CUDA-обработчики, обрабатывающие пространство витков гиперплоскостями.

5. Сравнительная отладка вычислительных регионов

Для DVMH-программ есть возможность сравнительной отладки регионов, это специальный режим работы вычислительного региона, при котором вычисления повторяются на ЦПУ и других вычислительных устройствах с целью сравнения значений выходных переменных при завершении вычислительного региона. Такой механизм позволяет выявлять и локализовывать ошибки, проявляющиеся при работе на ускорителях.

Включение и использование этого механизма не требует от программиста менять программу или что-либо дополнительно сообщать о своей DVMH-программе. Реализация этого механизма основывается на том, что, во-первых, вычислительный регион может быть выполнен одновременно независимо на нескольких вычислительных устройствах и, во-вторых, системе поддержки выполнения DVMH-программ известен исчерпывающий набор входных и выходных данных региона, так как он ей необходим для управления перемещениями данных между устройствами.

В сравнение включаются все выходные данные вычислительного региона. При этом целочисленные данные сравниваются на совпадение, а вещественные числа сравниваются с заданной точностью по абсолютной и относительной погрешности. В случае нахождения расхождений пользователю выдается информация о них, и далее в программе используется та версия данных, которая была получена при счете на центральном процессоре.

6. Выводы

Новый подход к созданию прикладного программного обеспечения, разработанный в Институте прикладной математики им. М.В. Келдыша РАН, существенно упрощает создание прикладных программ для суперкомпьютерных систем с ускорителями. Языки модели DVMH обеспечивают высокий уровень переносимости прикладного ПО на системы с другими архитектурами ускорителей, поскольку перенос не требует изменения программы.

Кроме того, DVMH-программы обладают способностью гибко подстраиваться под конкретную вычислительную аппаратуру, на которой эти программы запущены.

Литература

1. DVM-система [Электронный ресурс] – : [web-сайт] – Режим доступа: http://www.keldysh.ru/dvm/ – 01.12.2012
2. Top500 List – November 2012 | TOP500 Supercomputer Sites [Электронный ресурс] – : [web-сайт] – Режим доступа: http://top500.org/list/2012/11/ – 01.12.2012
3. High Performance Fortran [Электронный ресурс] – : [web-сайт] – Режим доступа: http://hpff.rice.edu/ – 01.12.2012

4. Romain Dolbeau, Stéphane Bihan, and François Bodin. HMPP™: A Hybrid Multi-core Parallel Programming Environment.
URL: http://www.caps-entreprise.com/wp-content/uploads/2012/08/caps-hmpp-gpgpu-Boston-Workshop-Oct-2007.pdf (дата обращения 02.12.2012)

5. The Portland Group. PGI Accelerator Programming Model for Fortran & C.
URL: http://www.pgroup.com/lit/whitepapers/pgi_accel_prog_model_1.3.pdf (дата обращения 02.12.2012)

6. OpenACC [Электронный ресурс] – : [web-сайт] – Режим доступа: http://www.openacc-standard.org/ – 01.12.2012

7. T. D. Han and T. S. Abdelrahman. *hi*CUDA: High-Level GPGPU Programming. IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 1, pp. 78-90, Jan. 2011

8. В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами. – Вестник Южно-Уральского государственного университета, серия "Математическое моделирование и программирование", №18 (277), выпуск 12 – Челябинск: Издательский центр ЮУрГУ, 2012, с. 82-92

9. Н.А. Коновалов, В.А. Крюков, С.Н. Михайлов, А.А. Погребцов. Fortran DVM – язык разработки мобильных параллельных программ. – Программирование, № 1, 1995, стр. 49-54

10. Н.А. Коновалов, В.А. Крюков, Ю.Л. Сазанов. C-DVM – язык разработки мобильных параллельных программ. – Программирование, № 1, 1999, стр. 54-65