ДЕКЛАРАТИВНЫЙ ЯЗЫК НОРМА И ПРОГРАММИРОВАНИЕ ДЛЯ НОВЫХ АРХИТЕКТУР: МНОГОЯДЕРНЫЕ СИСТЕМЫ

А.Н. Андрианов, А.Б. Бугеря, К.Н. Ефимкин, П.И. Колударов

Введение.

Наличие проблем, связанных с разработкой эффективных параллельных программ для многопроцессорных систем, является в настоящее время признанным фактом, который неоднократно обсуждался и продолжает обсуждаться, как с точки зрения природы этих проблем, так и с точки зрения ответа на вопрос – что делать для их решения.

Достаточно быстрое развитие новых аппаратных возможностей для поддержки параллельных вычислений, наблюдаемое в последнее время, еще больше усложняет проблему. Например, появление массово доступных многоядерных процессоров поставило вопрос об эффективном программировании для них. Практически одновременно появились массово доступные графические ускорители (графические процессоры), и опять возник вопрос об эффективном программировании для них. Агрессивное продвижение фирмами-производителями вычислительных систем, обладающих этими возможностями, часто дезориентирует прикладных специалистов, разрабатывающих параллельные вычислительные программы, толкает их на изменение средств разработки программ, хотя ясные и достаточно убедительные аргументы в пользу таких изменений отсутствуют. Так, применение технологии CUDA для эффективного программирования для графических ускорителей является, фактически, программированием на уровне ассемблера, с учетом тонких особенностей аппаратуры.

В настоящее время основной метод разработки эффективных параллельных программ для решения сложных вычислительных задач - это ручное программирование на языках Фортран или Си с использованием библиотеки передачи сообщений MPI (ставшей стандартом параллельного программирования “де-факто”), и с использованием стандарта OpenMP (встречалось реже, но в последнее время в связи с распространением многоядерных систем вновь приобрело актуальность).

Работы по созданию и продвижению новых средств и языков программирования (например, языков Chapel, X10, Fortress, UPC, Coarray Fortran, Charm++, различных сред и технологий программирования графических процессоров, способов программирования реконфигурируемых вычислителей, построенных на программируемой логике (FPGA), и прочее) ведутся весьма активно, однако проблема простой разработки параллельных программ и утилизации новых возможностей вычислительной техники остается в настоящее время не решенной.

Надежды на автоматическое распараллеливание последовательной программы также пока не оправдываются: для параллельных вычислительных систем с общей памятью распараллеливающие компиляторы не обеспечивают должное эффективное распараллеливание, а для параллельных вычислительных систем с распределенной памятью проводятся лишь исследовательские работы в данном направлении. Для архитектур типа графических процессоров решение задачи автоматического эффективного распараллеливания, по нашему мнению, вообще нереально в обозримой перспективе.

Декларативный подход. Язык НОРМА.

Один из возможных подходов к решению задачи автоматизации параллельного программирования вычислительных задач является подход с использованием декларативных (непроцедурных) языков. При использовании этого подхода решение вычислительной задачи программируется на непроцедурном языке (понятия, связанные с архитектурой параллельного компьютера, моделями параллелизма и т.п. при этом не используются), а затем компилятор автоматически строит параллельную программу (учитывая архитектуру целевого параллельного компьютера, модели параллелизма и т.п.). С учетом отмеченных выше проблем привлекательность этого подхода в настоящее время усиливается.

Идеи декларативного программирования были сформулированы еще в прошлом веке, теоретические исследования этого подхода для класса вычислительных задач проведены в пионерских работах И.Б. Задыхайло еще в 1963 году [1]. Непроцедурный язык НОРМА и система программирования НОРМА [2-4] разработаны в ИПМ им. М.В. Келдыша РАН также достаточно давно и предназначены для автоматизации решения вычислительных сеточных задач на параллельных компьютерах. Расчетные формулы записываются на языке НОРМА в математическом, привычном для прикладного специалиста виде, затем компилятор языка НОРМА генерирует параллельную программу на языках программирования Фортран или Си с использованием библиотеки передачи сообщений MPI.

Многолетний опыт использования системы программирования НОРМА при решении различных вычислительных задач [3] показал, по нашему мнению, целесообразность ее развития, как в направлении определения новых возможностей языка НОРМА, так и поддержки компиляции с учетом новых архитектурных возможностей параллельных систем.

В настоящее время нами ведутся работы по созданию версии компилятора для архитектур с двухуровневым параллелизмом (поле процессоров с распределенной памятью + каждый процессор многоядерный). Следует отметить, что проблема эффективного программирования для архитектур с двухуровневым параллелизмом не исчезает с появлением модных графических процессоров. Параллельное программирование для архитектур с двухуровневым параллелизмом требует совмещения особенностей двух моделей параллелизма – модели передачи сообщений и модели с общей памятью, а с технической точки зрения – использования библиотеки передачи сообщений MPI и стандарта OpenMP в рамках одной параллельной программы. Важно подчеркнуть, что это качественно более сложная проблема, не сводящаяся к механическому объединению способов программирования с использованием библиотеки MPI и стандарта OpenMP. Для её решения необходима разработка новых методов организации иерархии параллельных вычислений, балансировки вычислений, методов функциональной отладки и отладки эффективности и т.д. Без решения этих задач новые возможности вычислительной техники останутся не использованными.

Отметим также, что декларативный подход также очень перспективен и с точки зрения автоматизации программирования графических процессоров (или каких-либо гибридных архитектур).

О компиляторе языка НОРМА+.

Начата разработка новой версии языка НОРМА, получившей название НОРМА+, и компилятора для нее. Задача состоит в том, чтобы при полной поддержке совместимости программ, написанных на языке НОРМА, обеспечить следующие основные новые возможности:

- Поддержка в качестве целевой платформы исполняемой параллельной программы многоядерных вычислительных систем и вычислительных систем с двухуровневым параллелизмом. Таким образом, учитывая имеющуюся поддержку распределенных вычислительных систем, в качестве целевой платформы можно будет использовать любую из широко распространённых архитектур параллельных вычислительных машин.
- Поддержка различных конфигураций распараллеливания данных, и, возможно, поддержка динамического конфигурирования распараллеливания данных при запуске программы.
- Поддержка динамического распределения памяти в исполняемой программе.
- Введение понятия «безопасного переприсваивания» в языке НОРМА+.
- Ведение глобальных описаний и глобальных переменных в язык НОРМА+.

Для реализации поставленных выше задач начато создание компилятора для языка НОРМА+ с новой архитектурой, рассчитанной на различные архитектуры целевых платформ создаваемых параллельных программ. Разработка компилятора ведется на языке Си++, компилятор предназначен для применения как в среде UNIX, так и в среде Windows.

Компилятор программ, написанных на языке НОРМА+ (далее - компилятор НОРМА+), представляет собой приложение с интерфейсом командной строки. Компилятор производит синтаксический и семантический анализ текста программы на языке НОРМА+ и строит Абстрактное Синтаксическое Дерево (АСД), представляющее анализируемую программу в виде структурированного дерева. Затем выполняется процесс кодогенерации. В зависимости от заданных опций, компилятор НОРМА+ генерирует исполняемую программу на одном из языков программирования низкого уровня (Си или Фортран) в одной из моделей параллельного программирования (MPI, OpenMP, MPI совместно с OpenMP).

Все обращения к АСД для записи, изменения или чтения информации производятся через специально созданный интерфейс набора классов на языке Си++. АСД содержит полную информацию об исходной синтаксически и семантически корректной программе, либо информацию для модуля обработки ошибок.

Процесс кодогенерации состоит из нескольких этапов:

1. Первым начинает работу модуль общего генератора исполняемой программы. Он порождает на основе исходной программы на языке НОРМА+ исполняемые операторы, определяет последовательность их выполнения, возможность параллельного исполнения и т.п. Модуль общего генератора исполняемой программы строит внутреннее представление исполняемой программы в виде обобщенного дерева, которое содержит конструкции общего типа, которые не зависят ни от выбранной модели параллелизма, ни от выходного языка. Все эти конструкции затем будут уточняться и дополняться другими модулями кодогенератора компилятора НОРМА+.
2. После того, как общая структура исполняемой программы сгенерирована, вызывается модуль независимого оптимизатора. Он оптимизирует исполняемую программу в целом и решает такие задачи, как переопределение порядка вычисления операторов для оптимизации использования памяти.
3. Затем, в зависимости от заданных опций, вызывается кодогенератор для одной из трёх поддерживаемых моделей параллелизма: передачи сообщений (MPI), общая память (OpenMP) или гибридная модель (MPI совместно с OpenMP). Заметим, что структурная схема компилятора НОРМА+ рассчитана на подключение новых модулей кодогенераторов для новых моделей параллелизма. Модуль кодогенератора для конкретной модели параллелизма производит действия по модификации исполняемой программы с учётом выбранной модели параллелизма. Например, в случае MPI производится распределение данных по процессорам, генерация операторов обмена данными и т.п. На выходе получается всё то же внутреннее представление исполняемой программы, но только конкретизированное для выбранной модели параллелизма.

4. Затем, в зависимости от заданных опций, вызывается кодогенератор для одного из языков программирования (Си или Фортран), который переводит операторы внутреннего представления исполняемой программы в операторы выбранного языка программирования. Текст исполняемой программы в выбранной модели параллелизма на выбранном языке программирования записывается в выходной файл и на этом процесс компиляции заканчивается.

Работа выполнена при финансовой поддержке гранта РФФИ N09-01-00329.

ЛИТЕРАТУРА:

1. И.Б. Задыхайло. Организация циклического процесса счета по параметрической записи специального вида. Журн. выч. мат. и мат. физ., т.3, N2, 1963, с.337-357.
2. Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Задыхайло И.Б. НОРМА. Описание языка. Рабочий стандарт. — М.: Препринт ИПМ им.М.В.Келдыша РАН, 1995, № 120, 52с.
3. Андрианов А.Н., Бугеря А.Б., Ефимкин К.Н., Колударов П.И. Система параллельного программирования НОРМА – подход, результаты разработки, применение. В сборнике Труды X Байкальской Всероссийской конференции “Информационные и математические технологии в науке, технике и образовании”. Часть 1. Иркутск, 2005, 5с.
4. Система НОРМА. http://www.keldysh.ru/pages/norma/