ОБМЕН ДАННЫМИ В РАСПРЕДЕЛЕННЫХ ПРОГРАММАХ, СОЗДАННЫХ В ТЕХНОЛОГИИ ГРАФО-СИМВОЛИЧЕСКОГО ПРОГРАММИРОВАНИЯ

В.В. Жидченко, П.В. Аболмасов

Самарский государственный аэрокосмический университет им. С.П. Королёва

Приведены концепции технологии графо-символического программирования, позволяющей разрабатывать параллельные программы в визуальной форме. Описан способ реализации модели общей памяти, используемый в технологии при создании программ для распределенных вычислительных систем.

Введение

В настоящее время параллельные суперкомпьютеры рассматриваются как один из основных инструментов для решения задач моделирования при проектировании сложных систем. Однако на практике их применение затрудняется высокой сложностью разработки программ для суперкомпьютерных систем. Так, создание программы для распределенного кластера требует от разработчика глубоких знаний в области программирования, как минимум, знание стандарта MPI и языка C. Ясно, что специалист в предметной области, например проектирования авиационных двигателей, не всегда обладает такими знаниями (а может быть, и не должен обладать). В результате появляется тандем специалиста в предметной области и программиста, а вместе с ним – проблемы коммуникаций, взаимопонимания и следующее за ними увеличение сроков разработки. В такой ситуации представляется актуальным создание простого, наглядного способа описания параллельных алгоритмов и средства автоматической генерации параллельных программ для целевой платформы, скрывающих от пользователя сложности реализации.

Технология графо-символического программирования (технология ГСП), разработанная и развиваемая на кафедре программных систем СГАУ, ставит своей целью решение указанной задачи [1]. Технология ГСП определяется как технология проектирования и кодирования алгоритмов программного обеспечения на основе графического представления программ, с целью полной или частичной автоматизации процессов проектирования, кодирования и тестирования программного обеспечения.

Существенную трудность при создании программы для распределенной вычислительной системы представляет задача управления потоками данных. В сравнении с моделью распределенной памяти, модель общей памяти является более привычной для прикладного программирования. Основополагающей целью при проектировании механизма организации памяти в технологии ГСП являлось желание избавить разработчика от необходимости ручного управления данными в вычислительных системах с распределенной памятью.

В настоящей статье описывается реализация модели общей памяти в технологии ГСП. Для реализации поставленной задачи был использован шаблон проектирования «свойство», который не реализован в языке C++, но который может быть реализован доступными средствами языка.

Концептуальное описание технологии ГСП

Модель параллельной программы в технологии ГСП (граф-модель или граф-программа) представляется четверкой <D, F, P, G>, где D – множество данных некоторой предметной области, F – множество операторов, определенных над данными предметной области, P – множество предикатов, действующих над структурами данных предметной области, G = {A, Ψ, Φ, R} – ориентированный помеченный граф, называемый агрегатом (рис. 1). A = {A1, A2, …, An} – множество вершин графа. Каждая вершина Ai помечена локальным оператором fi(D)∈F. На графе задано множество дуг управления Ψ = { Ψ1i1, Ψ1i2, …, Ψjm} и множество дуг синхронизации Φ = {Φ1i1, Φ1i2, …, Φjl}. R – отношение над множествами вершин и дуг, определяющее способ их связи. Дуга управления, соединяющая любые две вершины Ai и Aj, имеет три метки: предикат pij(D) ∈P, приоритет k(Ψij) ∈N и тип дуги T(Ψij) ∈N. Каждая дуга синхронизации Φij помечена сообщением μij∈N. В классической (последовательной) модели вычислительного процесса, используемой в технологии ГСП, отсутствуют дуги синхронизации, которые вводятся в модель параллельного вычислительного процесса для решения задач синхронизации между его различными участками.

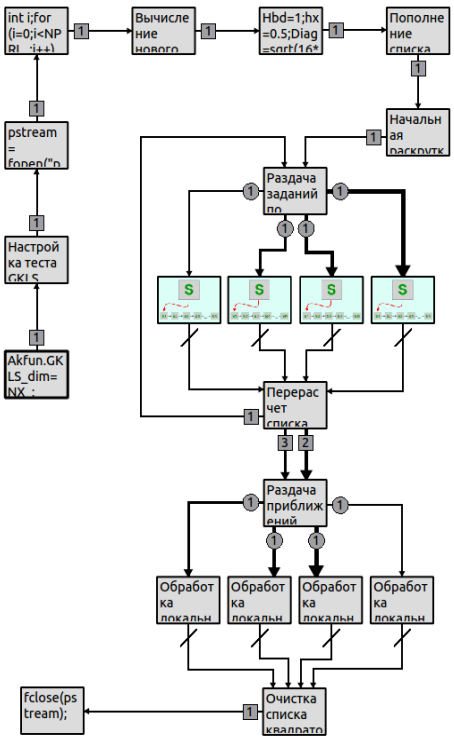

Рис. 1. Граф-модель параллельного алгоритма нахождения глобального экстремума

Под предметной областью программирования (ПО) будем понимать некоторую среду программирования, имеющую общую цель – разработку программного обеспечения автоматизации расчетов в некоторой области практических интересов (авиационные двигатели, бизнес и т.д.), общую область данных и общую область знаний.

Развитие вычислительного процесса, описываемого моделью, ассоциируется с переходами из вершины в вершину по дугам управления. При этом переход по дуге управления возможен лишь в случае истинности предиката, которым она помечена. Если несколько предикатов, помечающих исходящие из вершины дуги, одновременно становятся истинными, переход осуществляется по наиболее приоритетной дуге. Функционирование модели начинается с выполнения оператора f0(D), помечающего начальную вершину A0.

Граф-модель в данном случае заменяет текстовую (вербальную) форму описания алгоритма программы.

Этапы разработки граф-программ в технологии ГСП

1. Создание словаря данных ПО. На данном этапе создаются новые типы и структуры данных, а также происходит накопление словаря данных, где хранится информация обо всех переменных программы.
2. Разработка базовых модулей. Это этап традиционного текстового программирования, на котором программист создает исходные тексты небольших подпрограмм – базовых модулей, учитывая требования стандарта ГСП к оформлению этих текстов.

3. Создание объектов ПО. Этот этап производится автоматически после привязки формальных параметров базовых модулей к фактическим данным предметной области. Объекты, формируемые на основе базовых модулей, называются акторами и формируют множество операторов F модели ГСП.
4. Конструирование агрегатов. На этапе графического программирования пользователь может создать графовый образ новой программы.

Разработанный и отлаженный агрегат, в свою очередь, может быть использован в качестве исходного материала при конструировании следующих агрегатов. Таким образом, в общем случае агрегат имеет иерархическую структуру.

Эффективность программирования в технологии ГСП возрастает по мере развития пользователем своей среды программирования. Доля текстового программирования с традиционной трудоемкой отладкой постепенно снижается, и программирование перерастает в конструирование агрегатов из надежных программных модулей. Отладка при этом заключается только в корректировке структуры графа.

Стандарт хранения и использования данных в технологии ГСП

В технологии ГСП используется внутренний стандарт при организации межмодульного информационного интерфейса [2]. Стандарт обеспечивается выполнением пяти основных правил:

1. Вводится единое для всей предметной области программирования хранилище данных, актуальных для всей области, – словарь данных ПО.
2. В пределах ГСП описание типов данных размещается централизованно в архиве типов данных.
3. В базовых модулях в качестве механизма доступа к данным допускается только передача параметров по адресам данных.
4. Привязка данных объектов ПО реализована в паспортах объектов ПО.
5. В технологии ГСП не рекомендуется использовать иные способы организации межпрограммных связей по данным.

Данный подход к организации межмодульного информационного интерфейса приводит к тому, что синтезируемые программные коды и информационные связи «пространственно» отделены друг от друга. Модификация любого из объектов (актора, предиката или агрегата) не требует изменения кодов других объектов, входящих в ПО.

В технологии ГСП предполагается, что словарь данных располагается в общей памяти и одинаково доступен всем объектам граф-программы. Вместе с тем большинство современных суперкомпьютеров являются кластерами и имеют распределенную память. Чтобы обеспечить возможность разработки программ для таких систем, не обременяя пользователя технологии ГСП изучением механизмов обмена данными в распределенных системах, была реализована модель общей памяти. Описанный выше стандарт организации межмодульного информационного интерфейса был расширен понятием локальных данных.

Способ реализации общей памяти в технологии ГСП

В среде выполнения программы выбирается вычислительный узел, на котором будет запущен процесс, отвечающий за хранение переменных ПО. Учитывая аппаратные особенности и топологию вычислительной системы, это может быть узел с наибольшим объемом оперативной памяти или центральный узел, имеющий минимальное время доступа от любого из остальных узлов кластера. Преимущество данного подхода в том, что значительно экономится ресурс памяти на вычислительных узлах, т.к. на узлах память выделяется только под те переменные, которые используются процессом, исполняющимся на данном узле.

Предлагаемый способ обмена данными требует введения понятия диспетчера данных – подпрограммы, выполняющей функции хранения, чтения и модификации данных предметной области.

Диспетчер данных

Диспетчер данных, называемый также менеджером памяти, исполняется в отдельном процессе параллельной программы. Он порождает объект, описываемый классом TPOData, который хранит значения данных предметной области. В каждом из процессов, содержащих параллельные ветви граф-модели, также порождается объект класса TPOData. Однако функции доступа к членам-данным у объекта диспетчера данных и у объектов параллельных ветвей различаются. Диспетчер данных хранит все данные в локальной памяти и для обращения к ним использует обычные указатели. На остальных процессах используется ленивая инициализация памяти под переменную при первом доступе.

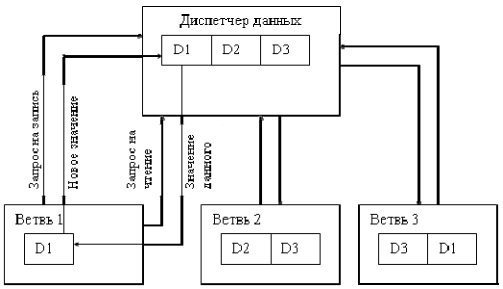

Рис. 2. Модель общих данных в технологии ГСП

В параллельных ветвях граф-модели для чтения или записи некоторого данного осуществляется обращение к диспетчеру памяти с помощью совокупности сообщений. В первом сообщении пересылается запрос на чтение или запись конкретного данного. Каждая переменная из ПО получает уникальный номер, по которому диспетчер памяти может ее идентифицировать. В случае чтения параллельная ветвь переходит к ожиданию ответа от диспетчера данных. При записи во втором сообщении пересылается новое значение данного. Диспетчер данных циклически принимает и обрабатывает запросы параллельных ветвей (рисунок 2).

Обзор класса TPOData

Класс TPOData – это ядро механизма хранения и передачи данных в технологии ГСП. Класс TPOData инкапсулирует все данные ПО и предоставляет доступ к ним через поля-свойства. Полей ровно столько, сколько переменных в ПО, а их названия совпадают с названиями переменных.

Свойство – способ доступа к внутреннему состоянию объекта, имитирующий переменную некоторого типа. Обращение к свойству объекта выглядит так же, как и обращение к полю объекта, но в действительности оно реализовано через вызов функции. При попытке задать значение данного свойства вызывается один метод (*setter*), а при попытке получить значение данного свойства – другой (*getter*).

Реализацией свойств в технологии ГСП являются шаблонные классы __property_rw и __indexed_property_rw. В качестве параметра шаблону передается тип переменной, которая будет доступна через данное свойство, и ссылка на класс, в котором это свойство будет использоваться, в нашем случае это TPOData. После инициализации объекта свойства ему необходимо установить методы доступа. В качестве методов доступа используются соответствующие методы из класса TPOData.

Для простых типов и для массивов используются различные классы свойств. Для простых типов данных необходимо указать по одному методу для получения значения и для его установки. Методы должны иметь следующую сигнатуру:

_T (TPOData:: *)(void); //для getter

_T (TPOData:: *)(_T const &); // для setter, где _T – тип переменной.

Для массивов необходимо указать 2 метода получения значения и 2 метода установки значения. Одна пара методов используется для получения всего массива, другая пара используется для получения одного элемента по индексу.

_T* (TPOData:: *)(void); // для получения всего массива

_T* (TPOData:: *)(_T* const &); // для установки всего массива

_T (TPOData:: *)(_I const &); // для получения одного элемента по индексу

_T (TPOData:: *)(_I const &,_T const &); //для установки одного элемента по индексу

Это один из вариантов реализации свойств в языке C++, спроектированный с учетом особенностей технологий ГСП и MPI.

Класс TPOData и экземпляр класса напрямую недоступен для разработчика граф-программ, и разработчик не обязан знать о его существовании. Когда разработчик обращается к переменной ПО, на самом деле он обращается к полям-свойствам объекта класса TPOData.

Использование локальных переменных в параллельных процессах

Описанные выше механизмы позволяют пользователю технологии ГСП создавать параллельные программы для распределенных систем, оставаясь в рамках концепции общей памяти. При определенных условиях такой подход генерирует эффективные параллельные программы без дополнительных усилий со стороны разработчика. Однако в большинстве случаев для повышения производительности вычислений на распределенной системе (например, на кластере), когда скорость доступа к данным в локальной памяти вычислительного узла существенно отличается от скорости доступа к данным другого узла, нужно искусственно увеличивать долю локальных вычислений. Для этого необходим механизм копирования данных из менеджера памяти в локальную память узла, на котором исполняется параллельная ветвь программы. Таким механизмом может служить механизм передачи сообщений между параллельными ветвями, основным назначением которого является синхронизация параллельных ветвей. Вместо передачи единичного флага, разрешающего запуск конкретной вершины, сообщение может содержать данные предметной области. Дуги синхронизации помечаются данными, пересылаемыми в каждом сообщении. При создании сообщения определяется список переменных для отправки и список соответствующих им переменных для приема. Если переменные векторные, то может быть задан диапазон индексов отправляемых и принимаемых элементов вектора. При этом необходимо обеспечить соответствие размера и типов отправляемых и принимаемых данных.

Множество данных предметной области разбивается на два подмножества: общие данные и локальные данные. Общие данные создаются и инициализируются менеджером памяти. Локальные данные создаются и инициализируются в том процессе, в котором они используются. Принадлежность данных конкретному подмножеству определяется установкой флага «Локальное данное» в свойствах каждого данного. По умолчанию все данные – общие. В передаваемых между вершинами сообщениях в качестве передаваемых и/или принимаемых данных должны участвовать локальные данные.

Заключение

Рассмотренные в статье концепции и механизмы технологии ГСП позволяют скрыть от пользователя технологии детали реализации управления и обмена данными в параллельном вычислительном процессе, способствуя упрощению использования современных суперкомпьютерных систем специалистами в различных предметных областях.

Литература

1. Коварцев А.Н. Автоматизация разработки и тестирования программных средств. Самара: Самар. гос. аэрокосм. ун-т, 1999. – 150 с.

2. Коварцев А.Н., Жидченко В.В. Методы и средства визуального параллельного программирования. Автоматизация программирования: электрон. учеб. пособие. Самара, 2010.