Math-Net.Ru

Общероссийский математический портал

С. В. Востокин, Применение метода парного взаимодействия объектов для построения сред разработки распределенных приложений, *Вестн. Сам. гос. техн. ун-та. Сер. Физ.-мат. науки*, 2005, выпуск 38, 26–28

DOI: <https://doi.org/10.14498/vsgtu366>

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
<http://www.mathnet.ru/rus/agreement>

Параметры загрузки:

IP: 3.85.7.115

26 апреля 2023 г., 09:53:44

ПРИМЕНЕНИЕ МЕТОДА ПАРНОГО ВЗАИМОДЕЙСТВИЯ ОБЪЕКТОВ ДЛЯ ПОСТРОЕНИЯ СРЕД РАЗРАБОТКИ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ

Предлагается метод построения среды разработки распределенных вычислительных приложений, основанный на парном взаимодействии объектов. Его целью является отделение механизма управления вычислениями от логики приложения и обеспечение масштабируемости за счет структурной избыточности кода. Приводятся достоинства и недостатки метода, а также данные экспериментального исследования эффективности тестового приложения.

В распределенных приложениях традиционного типа, построенных с использованием библиотек стандарта MPI [1], конкретная задача определяет поток вычислений. При этом код для управления вычислениями интегрирован в код задачи, что делает его трудным для понимания. Дополнительные сложности возникают при написании метакомпьютерных приложений [2], где требуется обеспечивать отказоустойчивость, возможность динамической миграции и балансировки загрузки.

В статье предлагается альтернативный способ управления распределенными вычислениями, основанный на использовании сред разработки [3]. Он заключается в такой организации кода, когда поток управления формируется средой разработки и не меняется от приложения к приложению. Для каждой новой задачи переопределяются подпрограммы, вызываемые средой, но общая структура остается универсальной. Данный подход позволяет отделить сервисный код распределенного приложения от кода задачи, обеспечивает повторное использование сервисного кода, снижает сложность разработки приложения.

Структура классов предлагаемой среды разработки показана на рис. 1. Основу среды разработки составляют три класса: класс *Постояльцев* (**Resident**), класс *Посетителей* (**Visitor**) и класс *Комнат* (**Room**). Данные классы инкапсулируют механизм синхронизации и логику сервисов. Классы *Конкретный Постоялец* (**ConcreteResident**) и *Конкретный Посетитель* (**ConcreteVisitor**) определяют процедуры и внутренние данные приложения.

Р и с. 1. Структура классов среды разработки распределенных приложений

Принцип управления вычислениями в среде разработки (см. рис. 1) состоит в следующем. Имеется несколько объектов-комнат. В каждой из комнат может находиться один или несколько объектов-постояльцев (отношение `live_in` на рис. 1). Объект-постоялец может быть посещен объектом-посетителем (отношение `visited` на рис. 1). При посещении происходит изменение внутреннего состояния пары объектов постоялец-посетитель. Также возможна активация локальных по отношению к текущему постояльцу посетителей (отношение `local` на рис. 1). Когда посетитель заканчивает обход постояльцев, определяемый логикой приложения, он возвращается к своему постояльцу и ждет следующей активации. Объекты-комнаты обеспечивают взаимоисключающий доступ посетителей к постояльцам,

Произвольный вариант развития вычислений, иллюстрирующий работу механизма синхронизации предлагаемой среды разработки, показан на рис. 2. Комната **aRoom1** является активным объектом, имеющим собственный поток исполнения. В начальный момент комната **aRoom1** извлекает посетителя **aVisitor1** из своей очереди и запускает его метод **run()**. Посетитель **aVisitor1** начинает обход постояльцев комнаты **aRoom1**. На диаграмме рис. 2. показано посещение двух постояльцев комнаты: постояльца **aResident1** и постояльца **aResident2**. Как видно из рис. 2, посещение заключается в вызове метода **accept()** соответствующего постояльца и передаче самого себя в качестве параметра вызова. Метод **accept()** переопределяется в приложении. Он возвращает адрес следующего постояльца и тем самым указывает среде исполнения, куда направить текущего посетителя. Если следующий посещаемый постоялец находится в другой комнате **aRoom2**, то происходит синхронизируемое примитивом «защелка» добавление посетителя в очередь данной комнаты. То же самое происходит при активации новых посетителей. Когда очередной посетитель заканчивает обход объектов внутри комнаты, комната извлекает следующего посетителя из очереди. При этом, если очередь пуста, то для предотвращения активного ожидания поток комнаты использует примитив «событие».

При разработке описанной схемы проектирования ставилась цель: обеспечить готовность приложения динамически занимать подключаемые вычислительные ресурсы или исполняться на ограниченных ресурсах без значительных накладных расходов.

Результаты вычислительных экспериментов показаны в таблице. Анализ данных эксперимента позволяет заключить, что для однопроцессорной машины, даже в случае значительной структурной избыточности (пятьдесят «лишних» ступеней конвейера), не происходит заметной потери эффективности. При использовании двух процессоров путем подбора соответствующей

щего числа комнат можно достигнуть приемлемой эффективности. Так конфигурация из двух процессоров и четырех комнат обеспечила для пятидесяти ступеней эффективности 98,89%, что лишь на 0,49% хуже, чем у оптимизированной параллельной программы.

Число сегментов	1 процессор, 1 комната		2 процессора, 2 комнаты		2 процессора, 4 комнаты	
	Время, секунды	Эффект., %	Время, секунды	Эффект., %	Время, секунды	Эффект., %
2	44,75	99,93	22,515	99,91	44,718	50
10	44,75	99,99	22,594	98,98	26,254	85,23
20	44,734	99,96	22,671	98,63	22,5	99,38
30	44,75	99,99	22,765	98,22	22,844	97,88
40	44,766	99,9	22,859	97,81	22,547	99,17
50	44,75	99,9	22,853	97,41	22,61	98,89

Примечание: Время счета последовательной программы – 44,719 секунд

Время счета оптимизированной параллельной программы – 22,5 секунд

Эффективность оптимизированной параллельной программы – 99,38%

Система: Pentium IIIx2 1GHz 512 MB, MS Windows 2000, SP4

Использование предложенной структуры объектов для построения конкретных сред разработки распределенных приложений имеет как достоинства, так и недостатки.

Достоинствами предложенной схемы проектирования являются: независимость функциональности от физического размещения компонентов приложения; удобная декомпозиция логики сервисов и логики приложения; хорошая масштабируемость за счет использования структурной избыточности; возможность совмещения во времени вычислений и обменов; возможность проведения имитационного моделирования хода распределенных вычислений на однопроцессорном компьютере.

При использовании схемы проектирования нужно учитывать и некоторые ее недостатки: код приложения может выглядеть более сложным по сравнению с кодом MPI; дисбаланс в загрузке комнат приводит к потере эффективности; обращение к данным посетителя, который не является инициатором текущего вызова **accept()** и не находится в пассивном состоянии, вызывает тонкие ошибки синхронизации.

Таким образом, использование сред разработки на основе парного взаимодействия объектов-постояльцев и объектов-посетителей упрощает код и повышает его надежность. Динамическая масштабируемость приложений, построенных по предложенному методу, обеспечивается за счет структурной избыточности. Описанная схема построения сред разработки используется в качестве модели исполнения визуального языка распределенного программирования GraphPlus [4].

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Антонов А. С. Параллельное программирование с использованием технологии MPI: Учебное пособие. М.: Изд-во МГУ, 2004. 71 с.
2. Воеводин В. В., Воеводин Вл. В. Параллельные вычисления. СПб: БХВ-Петербург, 2002. 608 с.
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. СПб: Питер, 2003. 368 с.
4. Востокин С. В. Технология моделирования распределенных систем, основанная на визуальном языке и ее приложения // Известия ШНЦ РАН, 2004. Т. 6, №1(10). С. 185-193.

Поступила 29.06.05 г.