Модел. и анализ информ. систем. Т.**15**, №**1** (2008) 37–40

УДК 519.68

Библиотека параллельного исполнения rpC-программ для Win32

Васильчиков В.В., Шубин А.В.
Ярославский государственный университет
e-mail: vasilch@uniyar.ac.ru

получена 17 января 2008

Аннотация

Рассматривается новая версия библиотеки поддержки параллельного режима исполнения для программ, написанных на языке rpC, предназначенная для использования в операционных системах Win32. Эта работа является существенной частью проекта по созданию и развитию программных средств поддержки рекурсивно-параллельного (РП-) стиля программирования, объединенных в рамках интегрированной среды РП-программирования RpmShell.

1. Основные отличия от предыдущих версий

Предыдущая полнофункциональная версия библиотеки [1] была разработана под MS DOS и морально устарела. Работу по ее переносу под операционные системы Win32 нельзя назвать просто портированием ранее написанного кода, поскольку в данном случае используются совершенно иные программные средства и механизмы. Это относится как к организации параллельных ветвей программы и управления ими, так и к передаче информации по сети (в среде Win32 более естественным было использовать для передачи данных протоколы на базе IP, а не IPX/SPX, как в предыдущей версии). При этом, разумеется, алгоритмы функционирования основных компонентов библиотеки, разработанные ранее [1], остались неизменными.

Значительная часть работы по созданию библиотеки была сделана ранее [2], [3], однако эту работу нельзя было назвать вполне завершенной. Доработки требовали следующие основные моменты:

- В прежней версии не были реализованы операции для работы с распределенной общедоступной памятью, а также со статической (в смысле языка rpC) памятью. Это очень ограничивало возможности использования библиотеки.

- Тестирование старой версии библиотеки продемонстрировало существенную вероятность сбоев в ее работе. Одной из основных причин было использование для передачи части информации (некритичной с точки зрения разработчиков) ненадежного протокола UDP. Однако в условиях довольно высокой загрузки сети терялась значительная часть таких сообщений, что временами приводило к остановке вычислительного процесса.

- Не были реализованы некоторые операторы, предусмотренные в языке rpC, например, механизм синхронизации посредством замков.

- Ряд функций был реализован не вполне корректно (с отступлением от описанных в [1] алгоритмов), что приводило к нежелательным последствиям. Например, некорректная реализация операции временного запрета на выборку из дека Set_M(), приводила временами к неравномерному распределению работы по системе. Впрочем, полный список исправленных недочетов в данном случае навряд ли уместен, и поэтому мы его опускаем.

- Потребовалось внести несколько изменений в механизм порождения и синхронизации запускаемых потоков с целью сделать библиотеку совместимой с использованием библиотеки Microsoft Foundation Classes, часто используемой при разработке Win32-программ.

2. Компоненты библиотеки и используемые технологии

В соответствии с базовой архитектурой процессорного модуля (ПМ), описанной в [1], его основными компонентами являются арифметический (АП), управляющий (УП) и коммуникационный (КП) процессоры. Важными компонентами также являются специальные структуры данных для поддержки основных алгоритмов функционирования системы, управления параллельными активациями, динамической балансировки загрузки и обмена данными. К таковым следует отнести дек, в котором хранятся потенциально мигрирующие процессы (ПМП), очередь готовых к выполнению процессов, очередь процессов, порожденных посредством оператора явного направления на модуль P_Send(), а также иерархический набор очередей для передачи служебных сообщений. Здесь и далее под процессом мы понимаем не программу, как это обычно принято, а активацию параллельной процедуры.

Соответственно основными компонентами библиотеки поддержки параллельного режима работы и являются АП, УП и КП, реализованные как отдельные Win32 потоки (thread). Они являются слабосвязанными и обмениваются данными посредством передачи сообщений. Подобная схема организации вычислительного процесса не доставляет трудностей при реализации, поскольку большинство стандартных библиотечных функций языка C++, использованного при разработке, поддерживают многопоточную реализацию и не требуют дополнительных мер по синхронизации. Исключение составили лишь блоки доступа к очередям сообщений, посредством которых ведется обмен данными между компонентами системы. Данные очереди были реализованы посредством стандартных классов-шаблонов STL, которые не имеют встроенного механизма защиты от параллельного обращения. Синхронизация работы этих объектов осуществлялась с использованием критических секций – простейшего объекта синхронизации, предоставляемого ядром Win32.

Особо хочется обратить внимание на реализацию функционирования арифметического процессора. АП предназначен для выполнения пользовательского кода, т.е. активаций параллельных процедур. Активация параллельной процедуры является особым объектом, выполнение которого, единожды начавшись, может быть прервано (при необходимости дождаться завершения каких-либо параллельных операций). Затем выполнение программы должно быть продолжено с того места, где произошла остановка. Поскольку переключение контекста (переход с выполнения одного потока на другой) осуществляется операционной системой, и возможности повлиять на него весьма ограничены, необходимо было реализовать особый механизм управления активациями.

Такой механизм был реализован с помощью волокон Win32 (fiber). Волокно, являясь частью одного потока и выполняясь в его контексте, имеет механизмы переключения управления на другие волокна в данном потоке, что позволяет оперативно прерывать выполнение активаций при вызове некоторых операторов rpC и передавать управление главному волокну потока. Главное волокно потока служит диспетчером сообщений, приходящих извне данному компоненту системы, и производит переключение управления на контекст волокна конкретной активации.

3. Основные функции библиотеки

Библиотека поддержки параллельного режима выполнения РП-программ представляет собой статический модуль, компонуемый с прикладной программой для создания исполняемого файла (возможно, несколько версий для различных инструментальных сред, используемых при сборке). Она обеспечивает возможность параллельного исполнения программы в локальной сети с поддержкой протоколов TCP и UDP. Перечислим основную функциональность, которую данная библиотека добавляет к прикладной программе на языке rpC.

- *Установление связи на начальном этапе.* При запуске программы посредством широковещательной рассылки UDP-сообщений она оповещает уже запущенные на других компьютерах сети свои копии об этом событии. Те посылают ответный пакет. Таким образом на каждом модуле формируется своя таблица IP-адресов, участвующих в работе компьютеров (их нумерация может быть различной). Собственно выполнение прикладной программы начинается после того, как оператор даст на это команду, в нашем случае нажмет на одном из компьютеров пробел. Его можно условно назвать главным модулем (нулевым), поскольку именно на нем будет вызвана функция main() РП-программы. Однако до этого главный модуль установит TCP-соединение со всеми остальными и перешлет им свою таблицу IP-адресов, чтобы нумерация у всех была одинаковой. После этого он породит потоки для работы АП, УП и КП, создаст в потоке АП дополнительное волокно для вызова функции main() и передаст ей управление. Все остальные компьютеры, получив таблицу адресов

от главного, по специальному алгоритму (чтобы исключить взаимные блокировки) устанавливают попарно каналы связи для обмена информацией по протоколу TCP. Они также создают потоки для работы АП, УП и КП и запускают цикл обслуживания входящих сообщений.

- *Создание дополнительных волокон для выполнения активаций параллельных процедур.* Как было отмечено выше, каждая активация (т.е. вызов процедуры с конкретными значениями параметров) должна быть полностью управляема. Мы должны иметь возможность ее останавливать в произвольный момент, передавать управление другой активации, а впоследствии запускать с точки остановки, полностью восстановив ее контекст исполнения. Для этой цели в Win32 появился такой удобный механизм, как волокно, который можно считать облегченным и полностью управляемым потоком.

- *Обеспечение синхронизации параллельных операций.* Под параллельными операциями здесь понимаются те, которые требуют значительного времени для завершения (например, нужно получить данные по сети), однако могут выполняться на фоне других действий. Для их синхронизации используется rpC-конструкция Wait(). Ее реализация основана на использовании специальных счетчиков запущенных параллельных операций и механизма работы очереди готовых к выполнению процессов (ГВП). Для дополнительной синхронизации прикладной программист может использовать rpC-конструкции Lock()/Unlock() (глобальная синхронизация) или любые средства, предоставляемые Win32 API: семафоры, события, мьютексы, критические секции (в пределах данного ПМ).

- *Обеспечение динамической балансировки загрузки.* В данном случае речь идет как о первоначальном распределении работы, так и о перераспределении работы между модулями с целью их равномерной загрузки. В библиотеке реализованы два алгоритма динамической балансировки загрузки, рассмотренные в [1], выбор между ними осуществляется через установки в файле конфигурации. Там же можно сделать установки, определяющие такой важный для распределения работы параметр, как "жадность" дека. Другим инструментом управления (скорректированным в данной версии) является реализация rpC-конструкции Set_M(), временно запрещающей выборку из дека с целью исключить негативный эффект "надкусывания" активаций.

- *Поддержка механизма управления активациями.* Сюда относится переключение контекста и обеспечение должной последовательности выполнения параллельных ветвей задачи, поддержка дека и других структур данных, участвующих в этом процессе.

- *Поддержка распределенной общедоступной памяти (РОП).* Библиотечные функции поддерживают динамическое размещение данных в РОП и их удаление, сетевые операции обмена (всего 10 операций) между данными в РОП и локальной памяти активации, контроль правильности использования этих данных (например, предотвращение выхода за границы).

- *Поддержка статической памяти.* К этой категории функций относится динамическое размещение и удаление данных, а также поддержка операций копирования данных между областями статической памяти на различных ПМ и вспомогательные операции для обмена между статической памятью и локальной памятью активации. Как и в случае РОП, происходит контроль границ выделенных участков памяти.

4. Результаты тестирования

При выборе задачи для тестирования библиотеки параллельного выполнения авторами принимались во внимание следующие пожелания к самой задаче и соответствующему РП-алгоритму:

- Задача должна быть достаточно трудоемкой, чтобы разработка параллельного алгоритма выглядела оправданной, а накладные расходы на РП-организацию вычислительного процесса были относительно невелики.

- Трудоемкость порождаемых параллельных ветвей программы была различной и заранее не предсказуемой. Это необходимо для проверки качества работы алгоритма динамической балансировки загрузки.

- Алгоритм должен предполагать активное использование как распределенной общедоступной, так и статической памяти.

Всем этим требованиям удовлетворяет алгоритм Литтла решения задачи коммивояжера, основанный на методе ветвей и границ. Поскольку решение о том, продолжать или нет вычисления на конкретной ветви программы, зависит от значения текущего рекорда (вычисленного асинхронно), оценить априори трудоемкость данной ветви принципиально невозможно. Для хранения матрицы расстояний разумно использовать статическую память, а для хранения информации о лучшем найденном решении – РОП.

Соответствующий алгоритм был запрограммирован на языке rpC, отлажен в последовательном режиме и использован как тестовый для оценки качества работы библиотеки поддержки параллельного режима выполнения. В качестве средства для компиляции и сборки использовалась среда Microsoft Visual C++ 6 версии, под управлением ОС Windows XP.

Тестирование производилось на сети из компьютеров с процессорами Intel Pentium 4 с тактовой частотой 3 Ггц и объемом ОЗУ 1 Гбт. Эксперименты производились на сети из 1, 2, 4, 6, 8 компьютеров. Решалась задача коммивояжера для 40 городов, в качестве глубины рекурсии использовалось значение 6 (то есть порождалось 64 листьевые активации).

Результаты экспериментов позволяют сделать выводы о надежной работе алгоритмов реализации перечисленных выше основных функций библиотеки. При этом наблюдалось существенное ускорение решения задачи, причем, как правило, в большее число раз, чем количество использованных компьютеров. Это объясняется спецификой метода ветвей и границ. Действительно, если на одной из параллельных ветвей было получено достаточно хорошее решение, объем вычислений на всех остальных заметно уменьшается за счет отсечения неперспективных вариантов. Кстати, это еще один довод в пользу разработки именно параллельных алгоритмов решения такого класса задач.

Этот момент, обусловленная им недетерминированность параллельного вычислительного процесса, а следовательно, и время решения задачи, не позволяют на основании полученных данных произвести количественную оценку накладных расходов на организацию параллельного процесса и обмен информацией по сети. Для того чтобы это сделать, в ближайшее время планируется запрограммировать ряд других трудоемких задач и использовать их в последующих экспериментах.

Список литературы

1. Васильчиков, В.В. Средства параллельного программирования для вычислительных систем с динамической балансировкой загрузки / В.В.Васильчиков. — Ярославль: ЯрГУ, 2001.

2. Васильчиков, В.В. О реализации обмена информацией между компонентами процессорного модуля / В.В.Васильчиков, А.О.Силантьев // Моделирование и анализ информационных систем. — 2002. — Т.9, №2.

3. Васильчиков, В.В. Средства поддержки параллельного выполнения рекурсивно-параллельных программ для платформы Win32 / В.В.Васильчиков, А.О.Силантьев // Актуальные проблемы естественных и гуманитарных наук на пороге XXI века. Информатика: Сб. материалов Всероссийской научной конференции, посвященной 200-летию Ярославского государственного университета им. П.Г.Демидова. – Ярославль: ЯрГУ, 2003.

The library for parallel execution of rpC-programs under Win32

Vasilchikov V.V., Shubin A.V.

The new version of the library for parallel execution of rpC-programs under Win32 is reviewed. This library is a significant part of the RpmShell software for developing recursive parallel programming style.