Модел. и анализ информ. систем. Т.**15**, **№1** (2008) 41–44

УДК 519.68

Среда рекурсивно-параллельного программирования для Win32

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

получена 17 января 2008

Аннотация

Рассматривается новая версия интегрированной среды рекурсивно-параллельного (РП-) программирования RpmShell, разработанная для использования в операционных системах Win32.

1. Введение

Предыдущая версия [1], разработанная под MS DOS, позволяла осуществить весь цикл разработки, отладки, оптимизации и эксплуатации РП-программы. Она включала в себя набор библиотек для поддержки РП-стиля программирования в режимах последовательного и параллельного исполнения, а также в режиме моделирования с целью сбора статистики и выявления причин снижения эффективности. Кроме того, в состав среды входил набор программ для обработки исходного текста на языке rpC, для осуществления имитационного моделирования, визуализации и анализа собранной при моделировании статистики, а также пользовательскую оболочку для управления всеми этими программными средствами. При параллельном исполнении программ весь обмен информацией основывался на использовании протоколов сетевого взаимодействия IPX/SPX.

Представляемую в данном докладе версию интегрированной среды RpmShell нельзя назвать результатом портирования ранее разработанных программных средств на платформу Win32. Операционная система на базе Win32 не позволяет осуществлять многие из числа использованных под DOS приемов (фактически трюков), например, для создания и осуществления взаимодействия параллельных ветвей программы, но в то же время предоставляет для этого штатные средства: потоки и волокна, необходимые средства синхронизации, реализованные как объекты ядра ОС. Кроме того, для осуществления сетевого взаимодействия в данном случае наиболее естественным представляется выбор протоколов TCP и UDP и штатной реализации средств их поддержки.

Изменился также и выбор инструментов для компиляции и сборки РП-программ. Вместо использовавшихся ранее средств фирмы Borland было решено использовать в таковом качестве программные среды фирмы Microsoft, начиная с Visual Studio 6 версии, а также более свежие.

В результате для переноса системы на платформу Win32 потребовалось заново переписать не менее 90% имевшегося кода (сейчас это около 15 тыс. строк на C и C++). При этом алгоритмы выполнения основных операций языка rpC, механизмы порождения параллельных активаций, их взаимодействия, алгоритмы распределения работы по системе и осуществления динамической балансировки остались в основном неизменными [1], [2], [3].

Представляемая версия среды RpmShell в настоящий момент содержит все необходимое для разработки, отладки и эксплуатации рекурсивно-параллельных программ в последовательном и параллельном режимах. Пока не реализованы библиотеки и программы для исследования РП-программ путем имитационного моделирования, это планируется сделать позже.

2. Основные компоненты и модули системы

Работа начинается с запуска программы-оболочки RpmShell.exe. Она предоставляет удобный интерфейс для осуществления всех дальнейших действий. При разворачивании системы в папке, где размещена эта программа, создаются поддиректории со следующими компонентами:

- *Поддиректория Dll.* Содержит используемые программой-оболочкой динамически подключаемые библиотеки. В настоящий момент это менеджер взаимодействия с другими компьютерами локальной сети CommMan.dll и модуль обработки исходного кода на языке rpC и создания проекта программы Ext_Par.dll.

- *Поддиректория Libraries.* Содержит статические библиотеки, компонуемые с программой пользователя для осуществления программной поддержки операторов грС. В настоящий момент это библиотеки поддержки последовательного и параллельного режима работы. Созданы и протестированы с использованием Microsoft Visual C++ версии 6.0 и Visual Studio 2003 (в дальнейшем упоминаемой как Visual Studio 7).
- *Поддиректория Include.* Содержит заголовочные файлы, необходимые для компиляции и сборки программы в различных режимах работы.
- *Поддиректория Cnf.* Содержит заготовки файлов конфигурации, используемых при выполнении РП-программы.
- *Поддиректория BlankPrj.* Содержит заготовки файлов проекта и файлов рабочего пространства (решения) для поддерживаемых версий Visual Studio.
- *Поддиректория Received.* Создается при необходимости программой-оболочкой на сателлитных компьютерах (при работе оболочки в режиме Slave). Сюда помещается исполняемый модуль, файлы конфигурации и, возможно, дополнительные файлы, присланные с главного компьютера (работающего в режиме Master).

На следующем рисунке приведена общая схема взаимодействия программных модулей среды, их функции, а также процесс создания и запуска РП-программы. Здесь сплошными стрелками обозначен процесс передачи данных, а штрих-пунктирными – информационные зависимости.

Рис. 1. Основные компоненты системы

3. Сценарий работы и основные возможности интегрированной среды RpmShell

Программа-оболочка RpmShell построена на основе MDI-редактора, так что исходный код на языке грС (предполагается, что файлы имеют расширение .grc) можно создавать прямо в ней. Разумеется, можно воспользоваться и более развитым внешним редактором, тем более, что сервис встроенного редактора не

слишком развит. Например, отсутствует подсветка синтаксиса, автоотступы, закладки и тому подобные приятные вещи, которыми нас радуют современные программные среды. Впрочем, в планы автора входит развитие подобных сервисных возможностей текстового редактора среды RpmShell.

Когда процесс первоначального набора кода осуществлен, следует задать опции проекта (точнее проектов, поскольку для каждого режима работы будет создаваться свой проект). Для этого предназначен диалог Project Details, вызываемый через выбор пункта Project Settings меню Compile/Run. На вкладках упомянутого диалога следует указать следующие сведения:

- Путь к .rpc-файлам. Предполагается, что они лежат в одной директории.
- Тип проекта (версия Visual Studio).
- Директории и имена проектов для различных режимов работы (достаточно согласиться с предлагаемым по умолчанию вариантом).
- Если предполагается использовать не все .rpc-файлы, находящиеся в указанной директории, на вкладке Source Files можно отметить необходимые.
- На вкладке Run Options можно дополнительно указать аргументы командной строки для запуска и то, какой из исполняемых модулей следует в дальнейшем запускать (Debug/Release). Впрочем, это можно сделать и позже.

После этого, выбрав соответствующие пункты меню (для разных режимов работы – разные), пользователь создает проект для последующей обработки средой Visual Studio.

Процесс создания проекта начинается с обработки исходного кода на языке rpC специальной функцией-препроцессором из библиотеки Ext_Par.dll, которая преобразует его в код на языке C. При этом проверяется синтаксис только rpC-инструкций (но не C), в случае ошибок информация о них, разумеется, выводится в соответствующее окно. Если ошибок не обнаружено, создается соответствующий проект, в котором прописаны необходимые опции, исходные и объектные файлы, а также файл рабочего пространства (или решения – в зависимости от выбранной версии Visual Studio).

Далее, выбрав пункт меню Call Compiler, пользователь может запустить выбранную версию среды Visual Studio, где при необходимости произвести дополнительные установки в созданном проекте, создать исполнимый модуль и, возможно, выполнить программу под управлением отладчика или без такового.

Запускать РП-программу на выполнение можно, разумеется, и из среды RpmShell, причем для запуска программы в параллельном режиме работы на нескольких компьютерах предусмотрен дополнительный сервис. А именно, если запустить среду на всех компьютерах, на которых предполагается параллельное исполнение РП-программы, можно перевести все компьютеры, кроме одного, в режим Slave, выбрав соответствующий пункт меню или нажав кнопку на панели инструментов. Один из компьютеров следует перевести в режим Master. Именно с него будет происходить дальнейшее управление.

С компьютера, находящегося в режиме Master, можно разослать исполняемый модуль программы, настроить конфигурацию выполнения, разослать периферийным компьютерам конфигурационный файл и при необходимости прочие файлы, запустить процесс параллельного исполнения. С него можно оперативно изменять состав периферийных компьютеров, участвующих в процессе. Любой из компьютеров, работающих в режиме Slave, можно перевести в режим Master, при этом старый Master автоматически будет переведен в режим Slave.

Вся информация о подключениях и исполняемых операциях выводится в диалоговых окнах Slave и Master. Последнее окно содержит дополнительные инструменты управления для выполнения следующих действий:

- Выбор режима запуска исполнимого файла: запускать локальную копию на каждом компьютере или один и тот же файл с файл-сервера.
- Формирование списка периферийных компьютеров для рассылки файлов и параллельного исполнения.
- Рассылка файлов, необходимых для параллельной работы.
- Задание аргументов командной строки и выбор варианта исполнимого файла (Debug или Release версия).

- Запуск процесса параллельного исполнения на сети.

Выполнение последнего из перечисленных действий приводит к запуску параллельной программы на всех выбранных для этого модулях. Управление при этом получает главная функция библиотеки параллельного исполнения, она осуществляет установление каналов связи между компьютерами и ожидает команды на старт — на одном из компьютеров (любом) требуется нажать пробел. Именно на этом компьютере начинается работа параллельной программы (разумеется, с функции main).

4. Направления развития

Автор предполагает в дальнейшем следующие основные направления развития Win32- среды РП-программирования:

- Портирование библиотек и программ имитационного моделирования параллельного выполнения программ, а также визуализации и анализа собранной статистики. Для описания основных структур данных и конфигураций моделей предполагается разработать формат представления на базе языка XML.
- Добавление в систему средства для поиска ошибок, специфичных для режима параллельного выполнения программы: некорректное использование общей памяти потенциально параллельными процессами, выходов из активаций без должной синхронизации и т.п.
- Развитие встроенного текстового редактора: подсветка синтаксиса, автоотступы, закладки и т.п.
- Добавление в систему средств для проверки правил использования grC- операторов: требования к использованию тех или иных классов памяти, ограничения на использование тех или иных переменных, правила использования блока параметров и т.п.
- Предполагается также продумать вопрос о предоставлении программисту возможности простого и удобного использования в grC-коде классов MFC, чтобы облегчить разработку РП-приложений с оконным интерфейсом. Это особенно важно, если вспомнить, что MFC накладывает жесткие условия на использование целого ряда важных классов в многопотоковых программах, а в нашем случае программист даже не знает, на каком из компьютеров сети будет выполняться тот или иной участок программы.

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 recursive parallel programming shell for Win32

Vasilchikov V.V.

The new version of the recursive parallel programming shell for Win32 is reviewed.