Библиотека мониторинга для многопоточных программ

А.И. Грюнталь[1], К.Г. Нархов[2], А.М. Щегольков[3]

ФГУ «ФНЦ Научно-исследовательский институт системных исследований РАН».
E-mail's: [1]grntl@niisi.ras.ru, [2]kostas@niisi.ras.ru, [3]ashch@niisi.ras.ru

Аннотация: Статья посвящена вопросам реализации средств и механизмов контролируемого выполнения многопоточной прикладной программы в среде операционной системы реального времени, функционирующей на многопроцессорной вычислительной системе. Рассматриваются архитектура библиотеки мониторинга, технологические и архитектурные аспекты ее применения.

Ключевые слова: контролируемое выполнение, библиотека мониторинга, многопоточная программа, поток управления, очередь сообщений, сигнал, операционная система реального времени, многопроцессорная система.

Введение

В статье излагается назначение, структура и принципы реализации многопоточных прикладных программ (далее, приложений). Многопоточные программы, реализованные по принципу функциональной декомпозиции задачи, характеризуются модульностью и масштабируемостью [1]. Функционально независимые (или слабо зависимые) друг от друга части программы, реализованные с использованием отдельных потоков управления, могут свободно исключаться, заменяться и дополняться без внесения существенных (а зачастую и вообще каких-либо) изменений в другие части программы. Многопоточная программа более понятна, структурирована и, как следствие, более надежна по сравнению с программами, в которых средства декомпозиции не применяются.

При проектировании приложения в многопроцессорных системах сначала выполняется развёртывание программного обеспечения на процессорных модулях в соответствии с требованиями к аппаратуре, быстродействию, конвейеризации и объему передаваемых между процессорными модулями данных. В результате развёртывания получается «топология задачи» (термин введен по аналогии с «топологией сети») – схема расположения и взаимодействия компонентов программного обеспечения в многопроцессорной системе.

На следующем этапе проектирования выполняется функциональная декомпозиция программного обеспечения на каждом из процессорных модулей (т. е. на узлах в смысле топологии) многопроцессорной системы. В соответствии с [2] результатом декомпозиции, в типовом случае, должно быть выделение в отдельные группы потоков управления, наряду с фрагментами, реализующими основную функциональность приложения, следующих функциональных фрагментов программного обеспечения:

– фрагментов, обеспечивающих асинхронный ввод-вывод; программирование конкретных особенностей ввода-вывода может выполняться в специализированных потоках управления, что снижает логическую сложность программы;

– фрагментов, обеспечивающих диалог с оператором;

– фрагментов, обеспечивающих связь с другими программами, например, графический сервер, геоинформационная система, база данных.

Библиотека мониторинга позволяет реализовать функции, которыми должна обладать система, спроектированная в соответствии с принципами контролируемого выполнения [2, 3, 4]. В соответствии с [2], программа с контролируемым выполнением – это программа со встроенными механизмами, обеспечивающими выполнение программы в критических условиях (например, при поступлении в программу некорректных

данных, ошибок в среде исполнения или в самой прикладной программе).

Средства контролируемого выполнения прикладной программы содержат механизмы, обеспечивающие контроль состояния выполняемой программы, сравнение параметров состояния прикладной программы с эталоном, а также генерацию управляющих воздействий, в случае отклонения поведения программы от эталона.

Реализация средств контролируемого выполнения требует наличия в прикладной программе агентов, аккумулирующих данные о выполнении (контрольные параметры состояния), монитора, анализирующего в реальном масштабе времени получаемые от агентов контрольные параметры состояния и генерирующего управляющие воздействия (команды управления приложением), а также средств выполнения сгенерированных монитором команд управления приложением. Кроме того, в прикладной программе должны присутствовать средства передачи контрольных параметров состояния от агентов к монитору и средства, передающие команды управления прикладной программой.

1 Архитектура библиотеки мониторинга

Архитектура библиотеки мониторинга, однопроцессорной системы и структура прикладной программы, работающей совместно с библиотекой мониторинга на одном процессоре, рассмотрена в [2].

Структура прикладной программы, работающей совместно с библиотекой мониторинга на многопроцессорной системе, представлена на рисунке 1.

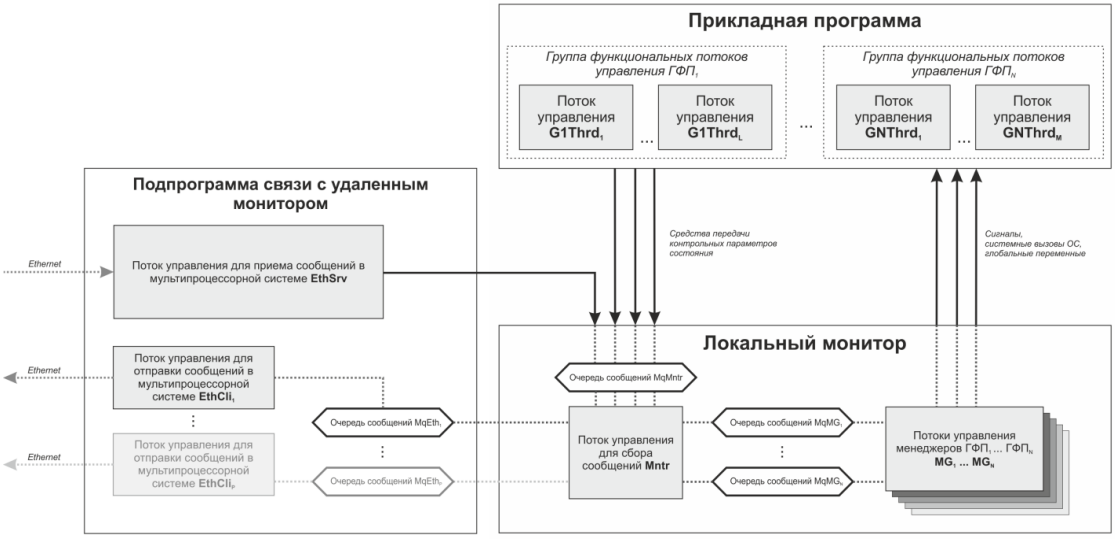

Рисунок 1 – Библиотека мониторинга в многопроцессорной системе

При работе библиотеки мониторинга на многопроцессорной системе используются удалённый (главный) и локальный (подчиненный) мониторы. Главный монитор назначается при конфигурировании библиотеки мониторинга, при этом прочие мониторы считаются подчиненными.

В многопроцессорной системе соответствие между ГФП прикладной программы к процессору осуществляется посредством идентификаторов коммуникационной среды RapidIO (идентификаторы процессоров назначаются при конфигурировании многопроцессорной системы).

Связь между главным и локальными мониторами выполняется с использование эмуляции Ethernet его эмуляции в коммуникационной среде RapidIO. Запись протокола работы (лога) главного монитора выполняется по протоколу **`nfs`**.

Для организации связи удалённого монитора с главным монитором используется дополнительные потоки управления, конвертирующие поступающие от главного потока локального монитора сообщения, приходящие через очередь сообщений, в пакеты, передаваемые по Ethernet. Соответственно, данные, поступающие от главного монитора, перенаправляются через очередь сообщений

главному потоку локального монитора. На рисунке 1 поток управления **EthSrv** используется для приема сообщений от главного монитора, а потоки **EthCli1** выполняет передачу сообщений главному монитору.

2 Технологические аспекты применения библиотеки мониторинга

Применение библиотеки мониторинга предполагает использование следующей технологии проектирования. На первом этапе определяется структура многопоточного приложения: количество групп потоков управления, количество потоков управления в группе, выполняет распределение групп потоков управления по процессорным элементам многопроцессорной системы. Затем спроектированная структура многопоточного приложения переносится в файл конфигурации БМ.

Перенос проекта в файл конфигурации БМ может быть выполнен двумя способами. Первый способ состоит в ручном порождении это файла. Файл конфигурации БМ – это информационный модуль на языке Си, содержащий определения типов данных и переменных, также в этом файле выполняется инициализация заданных переменных. Таким образом, данных способ конфигурирования состоит в написании кода на Си, компиляции и сборки.

При втором способе конфигурирования используются графические средства программирования. Пользователь (разработчик системы) заполняется в ходе интерактивного сеанса значения конфигурационных полей. В редакторе многопоточного приложения программист, используя визуальные средства, составляет структуру приложения и задает свойства его объектов (групп потоков управления, непосредственно потоков управления).

При втором способе конфигурирования используется независимо компилируемый компонент (плагин) MTA-application для Eclipse. Плагин предоставляет средства верификации структуры приложения и генерации конфигурации БМ в виде объектного файла, пригодного для подключения к ОСРВ.

Программист должен вручную заполнить некоторые свойства объектов многопоточного приложения при составлении конфигурации в плагине MTA-application для Eclipse.

Для объекта многопроцессорных систем должны быть указаны значения следующих атрибутов:

– имя многопроцессорной системы

– идентификатор многопроцессорной системы

– сетевой порт главного монитора многопроцессорной системы

– старшие 3 октета сети многопроцессорной системы с точками;

– идентификатор коммуникационной среды RapidIO главного монитора;

– многопроцессорной системы;

– количество процессоров в многопроцессорной системе.

Для объекта «Поток управления» должны быть заполнены следующие свойства:

имя пользовательской функции (например, «**osUserThread**»);

– **thrd_iters** – количество итерации в цикле потока управления

Прочие свойства объектов «Многопроцессорная система», «Группа потоков управления» и «Поток управления» инициализируются значениями «по умолчанию», однако при необходимости могут быть изменены программистом.

3 Архитектурные аспекты применения библиотеки мониторинга

Библиотека мониторинга не только порождает свою инфраструктуру, но и формирует определённую архитектуру приложения. При использовании библиотеки мониторинга часть ресурсов прикладной программы создаётся автоматически на основе данных, содержащихся в конфигурационном файле. К таким ресурсам относятся потоки управления, семафоры и очереди сообщений. Количество потоков управления, семафоров и очередей сообщений указывается при создании конфигурационного файла. Соответствующие ресурсы генерируются автоматически.

В принципе, запрет на динамическое порождение ресурсов отсутствует, но корректность исполнения такой программы не гарантируется средствами библиотеки мониторинга. Более того, соответствие требованиям библиотеки мониторинга может быть проверено средствами статического анализа. Если, например, вызов функций **pthread_create()** в тексте приложения

отсутствует, то считается, что БМ сконфигурирована корректно и используется в соответствии с правилами.

В библиотеки мониторинга реализована концепция порождения потоков управления при старте – все потоки управления в заданной группе на заданном процессорном элементе порождаются при старте системы. Таким образом, роль головной функции приложения берёт на себя библиотека мониторинга.

В связи этим в прикладной программе отсутствуют в явном виде последовательное порождение потоков управления. Последовательное порождение потоков удобно тем, что некоторый поток управления может инициализировать ресурсы и проконтролировать корректность их инициализации, а затем породить дочерние потоки, которые эти ресурсы будут использовать. Приложение, использующее БМ, должно быть спроектировано таким образом, чтобы связанные потоки управления (например, в части использования ранее порожденных ресурсов) имели явную синхронизацию. Например, если один поток инициализирует ресурсы, и другие потоки используют эти ресурсы, то требуется синхронизировать заданные потоки через контролируемый библиотекой мониторинга семафор (или множество семафоров) таким образом, чтобы потоки, использующие ресурсы, ожидали семафор, устанавливаемый потоком, который инициализирует ресурсы.

4 Особенности проектирования и разработки прикладных программ, работающих совместно с БМ

Для проектирования и разработки прикладных программ, предназначенных для выполнения в многопроцессорных системах совместно с библиотекой мониторинга, может быть использована среда разработки Eclipse с предустановленным плагином ТСАГ СПО [5].

Плагин ТСАГ СПО предоставляет прикладному программисту широкий набор средств для автоматизированного проектирования и разработки прикладных программ, предназначенных для выполнения в многопроцессорных системах совместно с библиотекой мониторинга. Использование этих средств сокращает трудоемкость интеграции библиотеки мониторинга и готовых (написанных ранее) прикладных программ, уменьшает время, затрачиваемое на документирование и конфигурирование прикладных программ с явной многопоточностью для многопроцессорных систем.

Плагин ТСАГ СПО позволяет в автоматизированном режиме создавать модели многопроцессорной системы в визуальном редакторе, заполнять свойства многопроцессорной системы в диалоговом режиме и генерировать файл конфигурации.

Плагин ТСАГ СПО для среды разработки Eclipse позволяет использовать все функциональные преимущества данной среды (например, редакторы файлов с исходным текстом на языке Си, XML и XSD редакторы) при разработке проектов ТСАГ СПО, а также предоставляет прикладному программисту гибкие и настраиваемые средства для компиляции и сборки программных модулей.

Проектирование и разработка задачи в ТСАГ СПО выполняется в следующей последовательности:

– создание модели многопроцессорной системы;

– заполнение свойств объектов модели многопроцессорной системы;

– синхронизация с файловой системой – создание системы каталогов, соответствующей объектам модели многопроцессорной системы;

– создание диаграмм программных модулей объектов модели многопроцессорной системы;

– генерация программных модулей объектов модели многопроцессорной системы;

– создание модели многопоточного приложения;

– связывание объектов модели многопроцессорной системы и объектов модели многопоточного приложения;

– компиляция проекта модели многопроцессорной системы;

– компиляция проекта модели многопоточного приложения;

– загрузка прикладной программы на целевой ЭВМ.

Важной особенностью ТСАГ СПО является наличие встроенных средств коллективной разработки. Эти средства предоставляют прикладному программисту возможность ведения разработки проектов ТСАГ СПО на удаленном сервере. Доступ к удаленной файловой системе организован с помощью RSE (Remote System Explorer). Работа с удаленным сервером выполняется

по защищенному каналу, реализованному с помощью протокола SSH.

Использование RSE является прозрачным для прикладного программиста – удаленный проект выглядит в Eclipse так же, как и локальный. Следует отметить, что использование механизмов удаленной работы с проектами позволяет организовать разработку в режиме ограниченного доступа, когда на локальном компьютере у разработчика нет прав на сохранение данных. В этом случае, он работает с удаленным проектом без ограничений, однако не может сохранить данные на локальный компьютер. Разработка на удаленном сервере может выполняться из сетей общего пользования – уровень безопасности обеспечивается безопасностью протокола SSH.

Плагин ТСАГ СПО совместим со встроенными в Eclipse средствами версионирования git, например, EGit или JGit.

5 Выводы

Приведённая в статье архитектура прикладного программного обеспечения позволяет использовать библиотеку мониторинга для создания приложений, соответствующих принципу контролируемого выполнения.

Разработана технология, обеспечивающая автоматизацию создания программ, использующих библиотеку мониторинга. Применение технологии приводит к созданию структурированных программ, гарантирующих управляемое использование примитивов операционной системы в составе приложения. Технология освобождает разработчика от рутинных действий по использованию типовых конструкций операционной системы.

В статье описывается подход к интеграции библиотеки мониторинга и разработанного ранее технологического пакета ТСАГ СПО. Изложенный в статье подход применим как к однопроцессорным, так и к многопроцессорным системам. Создание библиотеки мониторинга выполнялось в рамках работ ФГУ ФНЦ НИИСИ РАН по технологии разработки систем реального времени.

Monitor Library for Multithread Programs

A. Gryuntal, K. Narkhov, A. Shchegolkov

Abstract: Controlled execution implementation and mechanisms of an application running in a real-time operating system environment on a multiprocessor computer system is under consideration in the paper. Special attention is paid to monitor library architecture, and technological aspects of its application.

Keywords: controlled execution, monitor library, multithread program, thread, message queue, signal, real-time operating system, multiprocessing system.

Литература

1. Безруков В.Л., Годунов А.Н., Назаров П.Е., Солдатов В.А., Хоменков И.И, Введение в ос2000, Вопросы кибернетики. Информационная безопасность, Операционные системы реального времени, Базы данных/Под ред. чл.-корр. РАН В.Б.Бетелина. - Москва; НИИСИ РАН, 1999, – с. 76-106.
2. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Реализация принципа контролируемого выполнения для прикладных программ реального времени. // Труды НИИСИ РАН. – Том 5 № 2. Построение программ реального времени. ISSN 2225-7349, Москва, 2015, с. 113-121.
3. В.А. Галатенко, Контролируемое выполнение / В.А. Галатенко, К.А. Костюхин, К.А., Н.В. Шмырев – М: НИИСИ РАН, 2012. – 157 с.
4. В.Б. Бетелин, Контролируемое выполнение с явной моделью / В.Б. Бетелин, В.А. Галатенко, К.А. Костюхин – ПРОГРАММИРОВАНИЕ, 2014, N- 6, с. 45-55.
5. Генератор текста программ в исходном виде для систем реального времени / К.Г. Нархов // Программные продукты и системы. - 2010. - № 4. с. 24–30.