БИБЛИОТЕКА ДЛЯ УПРАВЛЕНИЯ ЗАДАНИЯМИ НА УДАЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСАХ И РАСПРЕДЕЛЕННЫХ СИСТЕМАХ

И.А. Валуев, И.В. Морозов

Введение

В работе представлены архитектура, интерфейс программирования и примеры использования высокоуровневой библиотеки GridMD, предназначенной для выполнения отдельных заданий или сложных сценариев на удаленных вычислительных системах.

В настоящее время существует и разрабатывается много программных систем и технологий, предназначенных для формулировки сценариев выполнения программ в распределенных (Грид) или облачных средах, таких как UNICORE [1], Kepler [2], Pegasus [3], MathCloud [4], Nimrod [5] и др. Некоторые системы (Globus [6], DRMAA [7], SAGA [8]) обеспечивают универсальную по отношению к вычислительному ресурсу спецификацию отдельной задачи, в то время как другие позволяют организовать поток исполнения (workflow) из задач, связанных друг с другом по входным и выходным данным. Например, система gUSE [9] предоставляет возможность организации сценариев исполнения, используя различные типы вычислительных ресурсов, среди которых кластеры на основе PBS, сайты под управлением Globus Toolkit, облачные сервисы Google и другие. Пользователи-разработчики имеют возможность регистрировать в системе новые специфические шаблоны распределенных сценариев, в то время как простые пользователи могут выбирать их из базы данных и вставлять в свои сценарии. Возможно создание адаптеров к новым вычислительным ресурсам на уровне веб-сервисов. Система требует установки WEB-портала для регистрации пользователей и ресурсов и управляется с помощью графического интерфейса. Это делает ее неудобной для реализации проектов, требующих других архитектурных решений и программного управления компонентами.

Целью разработки GridMD являлось создание компактной библиотеки для удаленного запуска задач и организации потоков распределенных вычислений, полностью управляемой функциями программного интерфейса C++, в отличие от систем визуального конструирования сценария [2-4]. Библиотека не требует установки и администрирования никаких сторонних компонент и сервисов, управление исполнением задач происходит со стороны программы-клиента от имени авторизованного пользователя вычислительного ресурса. Организация порталов совместного пользования, поддержка базы данных пользователей, приложений и вычислительных ресурсов, и другие задачи высокоуровневого администрирования выведены за рамки GridMD и могут решаться разработчиками клиентского программного обеспечения в соответствии с архитектурой и потребностями их программ.

Библиотека в наибольшей степени ориентирована на разработчиков распределенных и облачных вычислительных приложений. Она представляет собой компактное кросс-платформенное средство для программирования сложных вычислительных сценариев, в которых часть процедур переносится на удаленные вычислительные ресурсы. Под вычислительными приложениями понимаются такие, в которых основная нагрузка связана с расчетом, а не с хранением данных. Основная задача библиотеки – избавить разработчика от необходимости изучать детали функционирования конкретного ресурса (кластера, Грид-системы, виртуальной среды облачных вычислений) и унифицировать алгоритм взаимодействия с ними.

В рамках средств одного высокоуровневого языка программирования библиотека предоставляет набор функций для спецификации независимых (распределяемых) стадий вычисления и развертывания процедур, программно реализующих эти стадии, на удаленных вычислительных ресурсах различных типов. Кроме того, в функции библиотеки входит контроль за исполнением распределенного приложения: запуск и мониторинг состояния задач, исполняемых удаленными процедурами, балансировка нагрузки, организация контрольных точек. В настоящее время библиотека GridMD реализована на языке C++, однако допускает создание интерфейсов и для других языков программирования.

Библиотека является удобным средством быстрого создания распределенных расчетных приложений с нуля; организации сложных сценариев вычислений с использованием сторонних программ, доступных удаленно, например, в качестве облачных сервисов; разработки средств мониторинга исполнения задач для порталов совместного пользования.

Пользователи конечных приложений, созданных на базе рассматриваемой библиотеки, получают возможность использовать доступные им вычислительные ресурсы как серверы для удаленного проведения расчетов. Для этого от пользователя потребуются минимальные действия по конфигурации приложения.

Возможно также использование библиотеки в учебном процессе как наглядного и компактного средства для изучения возможностей Грид- и облачных технологий в применении к высокопроизводительным вычислениям.

В функции библиотеки в настоящее время входит запуск заданий на локальной или удаленной системе с использованием наиболее распространенных систем очередей (PBS, SLURM, ССРВ МСЦ РАН) или Грид-систем (Globus, Unicore), передача входных и выходных данных, мониторинг и управление выполнением

задания. При этом установка и запуск каких-либо резидентных компонент библиотеки (служб) на целевой системе не требуется.

Отдельные задания могут формироваться пользователем или генерироваться автоматически в результате выполнения сценария, состоящего из набора задач, связанных по данным (workflow). Поддерживается упрощенное создание типичных сценариев, таких как расчеты с вариацией или оптимизацией входных параметров. Для удаленного выполнения команд и передачи данных используется команды операционной системы или “внешние” утилиты и библиотеки, такие как PuTTY [10] и LibSSH [11].

Эта часть библиотеки во многом реализует функциональность, описанную в стандарте SAGA [8], хотя явно не следует этому стандарту. В отличие от существующих реализаций SAGA на C++ библиотека GridMD в большей степени оптимизирована для выполнения простых заданий на указанных выше системах очередей, поддерживает компиляцию под Windows и Linux.

Название GridMD возникло из первоначального замысла адаптировать библиотеку к задачам молекулярно-динамического моделирования [12], однако, в текущем варианте библиотека является более универсальной. В частности, она может быть рекомендована для создания программ многоуровневого (multiscale) моделирования, в котором при расчетах на каждом уровне детализации (квантовые вычисления, молекулярная динамика, уравнения сплошной среды и т.п.) применяются специализированные программные пакеты, предустановленные на целевой вычислительной системе или загружаемые пользователем.

Библиотека апробирована и используется как средство удаленного запуска заданий и управления сценариями в ряде программных комплексов по многоуровневому моделированию, в частности, в российско-европейском проекте по моделированию органических светодиодов IM3OLED [13].

Функциональность и структура библиотеки

Внутри библиотеки GridMD выделяются два уровня виртуализации ресурсов, с которыми может взаимодействовать программа пользователя: «менеджер сценариев» и «менеджер заданий». В обычном режиме менеджер сценариев, обрабатывая граф исполнения, отправляет менеджеру заданий запросы на выполнение отдельных наборов команд и пересылку данных. При этом менеджер заданий имеет унифицированный интерфейс для различных способов доступа к удаленной системе (local shell, SSH, Globus), а также для различных способов выполнения сформированных по общему образцу заданий, начиная от простейшего выполнения скрипта в командном интерпретаторе, до постановки параллельной задачи в очередь с использованием различных систем очередей. Общая схема взаимодействия программы пользователя с вычислительными ресурсами показана на рис. 1.

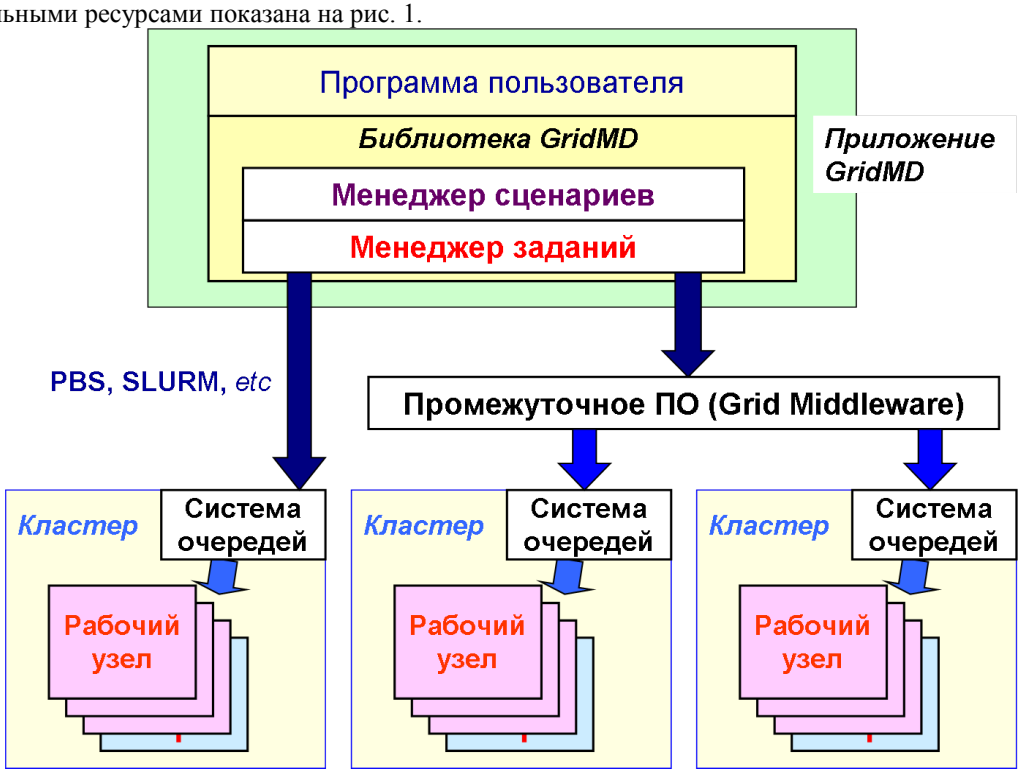

Рис. 1. Запуск задач с использованием программы, созданной на основе библиотеки GridMD

В последующих разделах описывается функциональность каждого из указанных компонент.

Менеджер заданий

В функции менеджера заданий входит подготовка удаленной системы к выполнению сценария, копирование входных данных для каждой отдельной задачи, запуск задачи с помощью выбранной системы очередей, контроль за ее выполнением и копирование выходных данных.

Базовой подсистемой менеджера заданий является набор функций для выполнения удаленных команд и копирования файлов между локальной и удаленной системой. Пользователь может применять эти функции отдельно от оставшейся части библиотеки, получая таким образом, высокоуровневый стандартизированный интерфейс для локального выполнения команд, утилит ssh (в ОС Linux), PuTTY (в ОС Windows) или кросс-платформенной библиотеки LibSSH. В режиме работы со сценариями пользователь может обращаться к этим функциям, только если есть необходимость некоторой подготовки удаленной системы, т.е. создание необходимых каталогов, установки пакетов программ, необходимых для всей серии вычислений.

Как и другие компоненты GridMD интерфейс менеджера заданий реализован в виде набора классов (рис. 2). Классы с описанной выше базовой функциональностью, реализующие различные протоколы доступа к целевой системе, наследуются от класса gmShell.

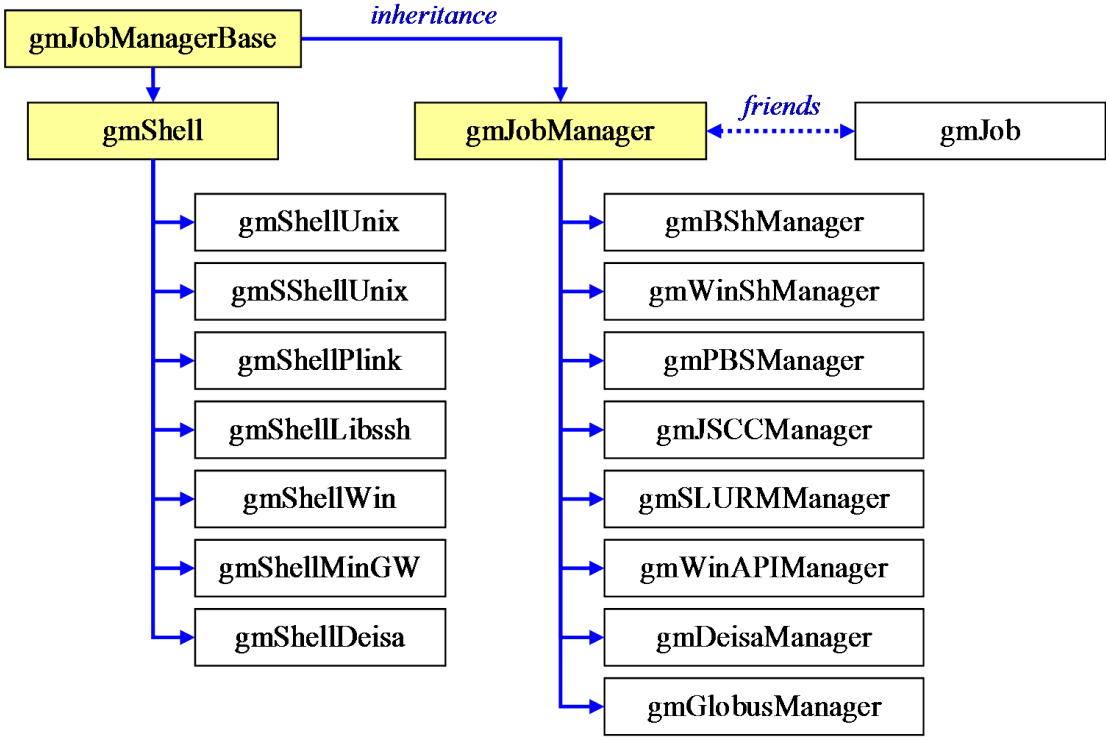

Рис. 2. Структура классов в объектно-ориентированной модели менеджера заданий GridMD. Заливкой внутри прямоугольников показаны абстрактные классы.

Функциональность классов протокола доступа включает:

- Синхронное или асинхронное выполнение набора команд. Если целевая система работает под управлением ОС Linux, используется интерпретатор Bash, для ОС Windows - интерпретатор cmd.exe. При синхронном выполнении команд функции возвращают данные, записанные при выполнении команд в потоки stdout и stderr.
- Копирование файлов или каталогов между локальной и удаленной системой. Поддерживается рекурсивное копирование каталогов, использование масок с символами “*” и “?” для задания группы файлов, преобразование окончаний строк текстовых файлов из формата Linux в формат Windows и обратно.
- Создание каталогов на удаленной системе.
- Удаление файлов или каталогов на удаленной системе.
- Измерение временных задержек при операциях с удаленной системой.

Некоторые из указанных классов предназначены для компиляции только под ОС Windows (gmShellWin, gmShellMinGW, gmShellPlink) или Linux (gmShellUnix, gmSShellUnix), другие являются кросс-платформенными (gmShellLibssh, gmShellDeisa). Классы gmShellWin, gmShellMinGW, gmShellUnix обеспечивают выполнение команд на локальной системе средствами Win32 API, пакета MinGW для адаптации GNU-приложений под Windows и интерпретатора Bash, соответственно. Классы gmShellPlink, gmSShellUnix, gmShellLibssh обеспечивают выполнение удаленных команд и передачу данных по протоколу SSH средствами пакета PuTTY, утилиты ssh в ОС Linux и библиотеки LibSSH.

На более высоком уровне менеджер заданий предоставляет интерфейс для работы с различными системами очередей. Классы, реализующие эту функциональность, наследуются от базового класса

gmJobManager (рис. 2). После создания и инициализации объекта класса менеджера заданий пользовать может создавать объекты класса gmJob, описывающие отдельную задачу. Объект gmJob хранит информацию о входных и выходных файлах, командах, исполняемых на удаленной системе, требуемом числе процессоров и других ресурсов. Описание задачи не зависит от типа менеджера заданий, таким образом задача может быть передана на выполнение любому менеджеру, однако, после запуска в объекте gmJob сохраняется информация о том, где задача была запущена.

Функциональность классов gmJobManager и gmJob включает:

- Создание временного каталога на удаленной системе для каждой задачи.
- Задание набора входных и выходных файлов. Допускается копирование файлов как между локальной и удаленной системой, так и между временным и постоянным каталогами удаленной системы.
- Поддержка MPI-задач с возможностью унифицированного запроса полного количества процессов и количества процессов на узел.
- Поддержка задач, требующих специализированных вычислительных узлов, например узлов, включающих GPU.
- Поддержка задач, связанных по данным для минимизации операций копирования (задачи выполняются в одном и том же временном каталоге).
- Запуск задачи с помощью выбранной системы очередей.
- Перезапуск задачи, если система очередей занята или переполнена.
- Получение текущего состояние задачи (см. ниже).
- Загрузка или выгрузка файлов до окончания выполнения задачи, например log-файлов.
- Восстановление состояния задачи после перезапуска программы GridMD.
- Ожидание окончания выполнения задачи.
- Копирование выходных данных, в том числе данных, записанных в потоки stdout, stderr.
- Очистка временного каталога, связанного с задачей.
- Выполнение указанных операций над группой задач, выбранных по маске с использованием их идентификаторов.

Каждая задача может находится в одном из следующих состояний:

JOB_INIT	- инициализация (состояние по умолчанию после создания объекта gmJob);
JOB_PREPARED	- задание подготовлено к выполнению: временный каталог создан, входные файлы в скопированы, осуществлена привязка к менеджеру заданий;
JOB_SUBMITTED	- задание передано системе управления заданиями;
JOB_QUEUED	- задание поставлено в очередь;
JOB_RUNNING	- задание выполняется;
JOB_SUSPENDED	- задание приостановлено;
JOB_EXITING	- выполнение окончено, происходит удаление из очереди;
JOB_COMPLETED	- задание полностью выполнено и выходные данные доступны для копирования;
JOB_HAVERESULT	- выходные данные скопированы на локальную машину;
JOB_FAILED	- при выполнении одной из операций произошла ошибка.

Важно отметить, что после запуска задачи вся информация о ней хранится во временном каталоге на удаленной системе. Таким образом, если программа пользователя на клиентской машине завершена, а затем запущена вновь, вся информация о запущенных ранее задачах может быть восстановлена.

Менеджер сценариев

Проблема формулировки сценариев исполнения для сложных научных приложений обсуждается достаточно давно [14, 15]. Если вычислительная задача может быть разбита на отдельные этапы (элементы сценария), зависящие друг от друга по входным и выходным данным, то приложения, реализующие эти этапы, могут быть запущены на различных узлах распределенной вычислительной системы при условии, что будет обеспечена необходимая передача данных между узлами (workflow). Формулировка сценария обеспечивает корректную последовательность исполнения отдельных элементов и передачу данных между ними в соответствии с логической структурой приложения в целом. Применение грид-систем с относительно медленными и ненадежными соединениями здесь является оправданным, если время выполнения элементов сценария значительно превышает время на передачу данных между ними.

В случае облачных вычислений, наличие сценария обеспечивает оптимальную загрузку выделенных под задачу ресурсов за счет динамического распределения работы между вычислительными узлами. Ограничение на объем передаваемых данных может быть снято при наличии высокоскоростных надежных соединения внутри виртуальной вычислительной машины.

GridMD использует стандартный способ формулировки потока обработки (workflow), который мы также будем называть сценарием приложения. Основными элементами сценария являются узлы и связи, формирующие граф исполнения. С узлами ассоциируются определенные действия программы, а со связями – данные (либо файлы данных), являющиеся их результатами. Входящие и исходящие связи представляют зависимости между узлами. Считается, что система может исполнить все действия узла, если известны результаты всех входящих в узел связей. Связи в GridMD могут классифицироваться по следующим категориям:

- Жесткая логическая связь. Узлы, связанные данным типом связи должны исполняться в рамках одного процесса (иметь общее пространство данных). Данный тип связи предназначен в основном для отражения логических цепочек внутри приложения.
- Связь по данным. Такой тип связи между узлами А и Б подразумевает, что для выполнения узла Б, для которого связь по данным является входящей, требуется либо предварительное выполнения узла А, для которого связь по данным является исходящей, в рамках того же процесса, либо передача данных результата узла А на узел Б в виде файла с данными. Таким образом, появляется возможность выполнения узлов А и Б в рамках различных независимых вычислительных процессов.

Если известен граф исполнения определенной части программы, то определить какие из узлов могут быть исполнены в распределенном режиме (одновременно с другими) возможно с помощью специального алгоритма анализа графа исполнения.

С каждым узлом графа исполнения GridMD может быть связано действие, определяемое подпрограммой пользователя (функцией C++) либо последовательностью команд, исполняемых на удаленной вычислительной системе. В эти команды может входить запуск любых вычислительных приложений. Узел может также быть отмечен как «локальный», тогда действия, связанные с узлом, будут исполняться непосредственно управляющим приложением, обрабатывающим граф исполнения. В случае, если действия для узла заданы в виде подпрограммы, и узел не является локальным, требуется, чтобы на удаленной системе была установлена копия приложения GridMD, в котором определена функция обработки данного типа узла. Такая копия, в частности, может быть получена путем компиляции управляющего приложения на удаленном вычислительном ресурсе и указания пути к его исполняемому файлу в конфигурации вычислительного ресурса. В этом случае соответствующая функция узла будет найдена и исполнена на удаленном ресурсе автоматически.

С каждой связью по данным в рамках GridMD может быть ассоциирован либо определенный тип данных, либо один или несколько файлов, заданных своими именами. В случае имен файлов допускается использование шаблонов поиска. Передача файлов является основным механизмом обмена данными, если в качестве узлов сценария используется запуск исполняемых приложений или команд.

Использование типизированных данных предпочтительно для обмена данными между узлами, заданными функциями пользователя, так как такой механизм позволяет контролировать правильность формирования данных. Для типов данных, используемых для передачи между узлами, определяется, в частности, процедура записи и чтения данных в файл.

Граф исполнения задачи сохраняется в виде xml-файла, а также может быть визуализирован средствами пакета Grpaphviz. Библиотека GridMD позволяет динамически отслеживать и изменять состояние графа исполнения по мере исполнения его отдельных узлов. Возможно полное отключение клиентского управляющего приложения и восстановление задачи при его перезапуске. Также допускается динамическое изменение графа исполнения: добавление новых узлов, изменение состояний узлов, не находящихся в обработке (перезапуск, установка статуса «отложенное исполнение», «ошибка»).

Алгоритмические шаблоны

Одна из стратегий развития GridMD – это добавление высокоуровневых средств программирования распределенных сценариев. Алгоритмические шаблоны, или “алгоритмические шаблоны” (algorithmic skeletons) – термин, который применяется для обозначения высокоуровневых моделей программирования для параллельных или распределенных вычислений [16, 17]. Существует стандартная классификация алгоритмических шаблонов, наиболее употребительные среди которых – последовательность (pipeline), ветвление (fork), отображение (map) и др. Преимущество алгоритмического шаблона заключается в том, что для использования реализуемого им алгоритма необходимо только создать его наполнение, т.е. указать те конкретные процедуры, которые будут исполняться на определенных этапах алгоритма. Все служебные действия, связанные с передачей данных, взаимодействием между параллельно запущенными процессами и т.д. выполняются автоматически и скрыты от пользователя алгоритмического шаблона.

Архитектура библиотеки GridMD полностью совместима с использованием алгоритмических шаблонов. Узел GridMD, для которого указываются процедуры получения входных данных, обработки и

выдачи выходных данных является простейшим низкоуровневым алгоритмическим шаблоном типа последовательности (элемент pipeline).

В качестве примера рассмотрим шаблон ветвления gmFork, который предоставляет базовую возможность разбиения исполнения программы на несколько несвязанных между собой конкурентных ветвей. Пример простейшей программы ветвления и соответствующий ей граф представлены на рис. 3 и 4.

Узлы, входящие в ветви шаблона ветвления классифицируются в определенном порядке: начальный узел start (точка расхождения ветвей), узел расщепления split (начальный узел каждой ветви), узел слияния merge (конечный узел каждой ветви), конечный узел шаблона finish (точка слияния ветвей). Между узлами split и merge может быть помещено произвольное число других узлов, в том числе допускается вложение шаблонов ветвления друг в друга. В этом случае шаблон нижнего уровня входит в ветвь шаблона более высокого уровня между узлами типа split и merge.

Помимо организации ветвей, шаблон gmFork позволяет привязать определенный тип данных к связям между узлами определенного типа. Такая привязка обеспечивает строгий контроль типов на уровне компилятора при использовании функций доступа к данным узла. Вместо использования общих функций node_input() и node_output(), в которых проверка соответствия типов передаваемых данных возможна только на этапе исполнения приложения, программисту предлагается пользоваться типизированными шаблонными функциями fork<type1, type2, type3>.vsplit_out(), fork<type1, type2, type3>.vsplit_in(), в которых типы используемых данных определяются шаблонными параметрами класса gmFork<> и известны на этапе компиляции.

```
begin_distributed();
// Define the skeleton and internal data link types
gmFork<void,val_t,void> fork1("loop");
fork1.begin_here();     // marks the loop 'begin' node

int nterms = 3;         // number of terms in the series
val_t sum = 0.;         // 'sum' accumulates the result
for(int i=0; i<nterms; i++){
  // Create a new 'split' node and define its output
  if(fork1.split())
    fork1.vsplit_out() = pow(x, (val_t)i) / i;

  // Define the action of the 'merge' node
  if(fork1.merge())
    sum += fork1.vmerge_in();  // accumulation of the terms
}
if(end_distributed()) // loop 'end' node is optional
  printf("The result is %g\n", sum);
```

Рис. 3. Программа для вычисления членов разложения в ряд функции ln(x) с использованием шаблона ветвления. Контроль типа результатов вычислений (val_t) осуществляется на этапе компиляции

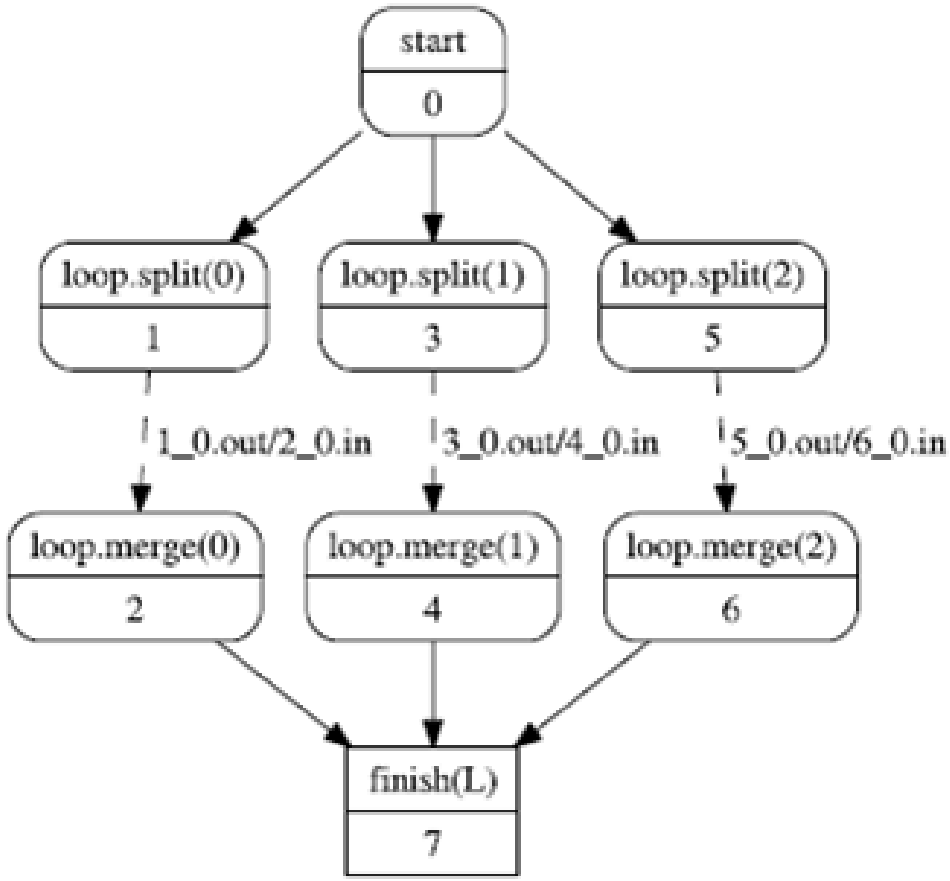

Рис. 4. Граф исполнения, автоматически сгенерированный средствами библиотеки GridMD для программы, представленной на Рис. 3

Отметим, что шаблон ветвления является исключительно интерфейсной конструкцией, упрощающей конструирования сценариев определенного типа. Исполнение узлов, сконструированных с помощью шаблона ветвления, происходит с помощью универсальной процедуры обработки графа исполнения. Возможно ручное создание аналогичного сценария с использованием функций добавления узлов и связей. Такое ручное создание потребовало бы гораздо больше вызовов функций и увеличило бы вероятность программистской ошибки при конструировании сценария. На основе шаблона ветвления возможно создание более специализированных алгоритмических шаблонов, в которых роли отдельных компонент алгоритма (например, число ветвей, функции каждой ветви и т.д.) известны более детально.

Заключение

В работе изложена общая архитектура высокоуровневой библиотеки GridMD с подробным рассмотрением компонент библиотеки, отвечающих за обработку сценариев и управление отдельными заданиями. GridMD является развивающимся проектом с постоянно увеличивающимся набором функций. Потенциальными потребителями библиотеки являются разработчики и пользователи программ численного моделирования, которым приходится регулярно выполнять масштабные численные эксперименты, в том числе с нетривиальным сценарием.

Библиотека разработана в ОИВТ РАН, является свободно распространяемым программным обеспечением и доступна на сайте http://gridmd.sourceforge.net.

Работа выполнена при поддержке по грантам РФФИ №№ 12-02-31783 «мол_а» и 12-02-33170 «мол_а_вед», а также по ПФИ ПРАН №№ 14 и 15.

ЛИТЕРАТУРА:

1. Pytlinski, J., Skorwider, L., Benedyczak, K., et al // LNCS, V. 2658, P. 307 (2003); UNICORE project web site: http://www.unicore.eu
2. Kepler project web site: http://www.kepler-project.org
3. Pegasus project web site: http://pegasus.isi.edu

4. MathCloud project web site: http://mathcloud.org/ru/
5. Abramson, D., Bethwaite, B., Enticott, C., Garic, S. and Peachey, T. // Special issue of IEEE Transactions on Parallel and Distributed Systems on Many-Task Computing. 2011. V. 22(6). P. 960; NIMROD project web site: http://messagelab.monash.edu.au/Nimrod
6. http://www.globus.org/toolkit/
7. P. Troger, H. Rajic, A. Haas, P. Domagalski // Proc. of VII IEEE International Symposium on Cluster Computing and the Grid. 2007. P. 619; DRMAA project web site: http://www.drmaa.org/
8. T. Goodale, S. Jha, H. Kaiser // Computational Methods in Science and Technology. 2006. V. 12(1), P. 7.
9. P. Kacsuk, Z. Farkas, M. Kozlovszky // Journal of Grid Computing. 2012. V. 10(4), P. 601; gUSE project web site: http://www.guse.hu/
10. PuTTY project web site: http://www.chiark.greenend.org.uk/~sgtatham/putty
11. LibSSH project web site: http://www.libssh.org
12. I.V. Morozov, I.A. Valuev. Automatic Distributed Workflow Generation with GridMD Library // Computer Physics Communications. 2011. V. 182. P. 2052–2058.
13. Integrated Multidisciplinary & Multiscale Modeling for Organic Light-Emitting Diodes: http:// www.im3oled.eu
14. W.M.P. van der Aalst // The Journal of Circuits, Systems and Computers, V. 8, No. 1, P. 21 (1998);
15. Horacio González-Vélez and Mario Leyton "A survey of algorithmic skeleton frameworks: high-level structured parallel programming enablers" // Software: Practice and Experience 2010. V. 40. Issue 12. P. 1135.
16. eSkel project web site: http://homepages.inf.ed.ac.uk/abenoit1/eSkel
17. Skandium project web site: http://skandium.niclabs.cl