КОМПОЗИТНЫЕ ПРИЛОЖЕНИЯ В РАСПРЕДЕЛЕННЫХ ПРЕДМЕТНО-ОРИЕНТИРОВАННЫХ СРЕДАХ

К.В. Князьков, А.В. Ларченко

СПбГУ ИТМО

E-mail: kostantin.knyazkov@gmail.com, aleksey.larchenko@gmail.com

Введение

Возрастающая сложность современных систем распределенных вычислений ограничивает возможности их широкого использования специалистами предметных областей. Как следствие, это определяет необходимость развития специализированных подходов к разработке распределенных приложений с максимальным уровнем абстрагирования от специфики их технической реализации и исполнения на вычислительных ресурсах.

В настоящее время хорошо зарекомендовал себя подход, основанный на описании сложных вычислительных процессов в форме потоков заданий (workflow, далее – WF). Под WF понимается ориентированный граф, узлами которого, как правило, являются запуски отдельных задач, а ребрами - зависимости между ними, характеризующие обмен данными и управляющими параметрами в распределенной вычислительной среде. В отличие от алгоритма, формализм WF не определяет последовательность выполнения операций; он регламентирует только их состав и зависимости.

На текущий момент существует ряд систем распределенных вычислений, построенных на технологиях работы с WF [1]. Однако возможности их эффективного использования и расширения в целом ограничены, поскольку в настоящее время не существует унифицированных способов их описания, интерпретации и исполнения в терминах соответствующей предметной области [2].

В работе рассматриваются вопросы, касающиеся всего жизненного цикла WF, и предлагается подход к построению и реализации композитных приложений в распределенных вычислительных средах на основе предметно-ориентированных описаний. Подход основывается на концепции iPSE [3]; как следствие, это приводит к необходимости обеспечить максимальное абстрагирование пользователя от технических аспектов исполнения приложения. Пользователь должен описать в форме WF только содержательную составляющую своего приложения, при этом выбор условий выполнения (вычислительных ресурсов, необходимого количества процессов и пр.) осуществляется автоматически [4]. Потому принципиальным аспектом является разработка формы человеко-компьютерного взаимодействия, удобной для пользователя и достаточной для однозначного восприятия системой. Ниже она реализуется посредством двух взаимосвязанных языков: описаний прикладных пакетов (EasyPackage) и описаний структуры WF (EasyFlow).

Описание прикладных пакетов в распределенной среде

Описание вычислительных пакетов, входящих в состав композитного приложения, осложнено тем, что разные пакеты используют свою стратегию работы с данными (конфигурационный файл, командная строка аргументов, использование переменных окружения, использование проектов, хранящихся в структуре директорий и файлов), к

тому же при запуске на гетерогенных ресурсах необходимо учитывать влияние различных сред исполнения на конфигурацию пакета. Во избежание этих проблем предлагается специальный предметно-ориентированный язык (Domain Specific Language, DSL) EasyPackage, позволяющий описывать пакеты в наглядной форме, понятной специалистам-предметникам, и легко разбираемой автоматически. Язык разработан на основе реализации языка Ruby (IronRuby) и является интерпретируемым языком со строгой динамической типизацией с явным приведением типов. Базовые элементы данного языка являются идентичными элементам в Ruby.

Описание пакета представляет собой один или несколько текстовых файлов. Оно использует следующие понятия: пакет, входной/выходной параметр, входной/выходной файл, режим запуска. *Пакет* – исполняемое приложение, запускаемое в пакетном режиме (модель IPO – Input-Process-Output), которое на вход принимает определенный набор входных файлов, параметров командной строки, переменных окружения и других источников данных, а на выходе генерирует набор выходных файлов. *Параметр пакета* – элемент данных, имеющий имя, тип и значение. Параметр может быть входным или выходным, а также может быть параметром исполнения. Тип параметра может быть одним из базовых: строка, логический тип, число с плавающей точкой, перечислимый тип, целое число, список. Режим запуска характеризуется набором используемых в нем параметров.

Структура описания пакета состоит из следующих разделов: объявление расширений, общее описание пакета, секционное описание входных и выходных данных пакета, описание параметров исполнения. Раздел объявления расширений предназначен для написания процедур для расширения функциональности базовой библиотеки языка. Общее описание пакета включает в себя набор полей, несущих общую информацию о пакете. Раздел секционного описания включает описания параметров и файлов (входных и выходных). Раздел параметров исполнения призван абстрагировать работу с пакетом от неоднородности ресурсов.

Описание на языке EasyPackage позволяет не только задать правила обращения к конкретному пакету в распределенной вычислительной среде, но и корректно интерпретировать его входные и выходные данные. Это обеспечивает совместимость (по данным) пакетов различных разработчиков в составе WF.

Описание композитных приложений в распределенной среде

Описание композитных приложений, состоящих из нескольких прикладных пакетов, требует не только определения правил работы с пакетами, но и структуры взаимодействия между ними. Для реализации удобного способа задания композитных приложений предлагается специализированный язык EasyFlow. Он предоставляет конечному пользователю гибкие возможности по заданию различных форм WF, в рамках которых происходит выполнение различных прикладных пакетов, генерация выходных данных, их получение, конвертация и обработка.

Основной идеей языка является полное абстрагирование от особенностей распределенной вычислительной среды, в которой работает пользователь. Фактически, EasyFlow — это высокоуровневый язык описания абстрактных WF. Такой подход позволяет описывать саму решаемую задачу, а не способ ее исполнения на конкретной вычислительной архитектуре. Описание WF представляет собой скрипт на языке EasyFlow. Тело скрипта состоит из описания вызовов прикладных пакетов — *шагов*, которые задаются с помощью директивы *step* и представляют собой узлы графа WF. Для описания каждого шага необходимо задать его имя, название запускаемого пакета и перечень предметных параметров этого пакета, который описан в базе пакетов, рассмотренной выше.

Так как WF представляет собой ориентированный граф, в EasyFlow введены возможности по организации его структуры, а именно механизмы определения порядка выполнения шагов. Они делятся на два вида: зависимости по управлению и зависимости по данным. *Зависимости по управлению* представляют собой явные указания на то, что один шаг должен начать свое исполнение после другого. Это делается с помощью директивы *after*. *Зависимости по данным* представляют собой неявный указания на зависимости между шагами, которые анализируются при интерпретации скрипта EasyFlow.

Еще одной полезной возможностью EasyFlow является автоматическое варьирование параметров (*parameter sweep*). Такая задача часто возникает, когда необходимо запустить один и тот же вычислительный пакет, закрепив одни и варьируя другие параметры. Для этого в язык введена директива *sweep*, которая принимает список параметров для варьирования, и из одного шага делает N шагов, где N – количество элементов в декартовом произведении списков варьирования для различных параметров.

Таким образом, WF, описанные на языке EasyFlow, оказываются полностью независимыми от конкретной архитектуры вычислений и хранения данных, что позволяет беспрепятственно обмениваться ими между пользователями распределенной среды и запускать их на различных вычислительных ресурсах.

Разработка и запуск композитных приложений

Одним из подходов к разработке пользовательского интерфейса для работы с WF является создание интегрированной среды разработки, обеспечивающей удобство работы пользователя на всех этапах работы с WF: создание WF, формирование запуска WF путем указания конкретных входных данных, запуск WF, наблюдение за исполнением, анализ результатов. Для примера, при редактировании скрипта WF пользователю могут быть предложены следующие функциональные возможности: подсветка синтаксиса скрипта EasyFlow, авто-дополнение кода (используя данные из описаний пакетов), проверка корректности скрипта, визуализация графа WF.

Существенными недостатками данного подхода являются а) ориентация на компетентного в разработке WF пользователя, и б) относительное неудобство работы с готовыми WF. Для преодоления этих недостатков дополнительно предлагается подход построения предметно-ориентированных пользовательских интерфейсов.

При использовании представленных выше языков для описания пакетов и WF становится возможным ввести еще один уровень абстракции представления вычислительных задач для пользователей — предметно-ориентированные интерфейсы (ПОИ). Этот подход основан на разделении конструирования WF и его использования, когда экспертом предметной области заранее задаются некие шаблоны применения пакетов, которые затем автоматически трансформируются в простые динамически генерируемые графические интерфейсы, доступные пользователям посредством Интернет. С помощью них пользователь может задать необходимые параметры расчета, запустить задачу на исполнение и получить результаты. Примером использования такого подхода может явиться выполнение массовых инженерных расчетов, когда от запуска к запуску требуется варьировать только ограниченный набор параметров.

При автоматическом построении ПОИ могут быть целиком использованы формальные описания пакетов на языке EasyPackage: набор параметров, их возможные значения и ограничения на них, взаимосвязь между параметрами и пр. Исходя их этого можно реализовать следующие возможности: отображение альтернатив при задании параметров, подсказки при выборе значений параметров, проверку значений и полноты состава параметров, отображение результатов вычислений. Таким образом, проблемно-

ориентированный интерфейс приобретает все необходимые функциональности, свойственные графическим интерфейсам самих прикладных пакетов.

Принципиальной особенностью технологии ПОИ, основанной на использовании языков EasyFlow и EasyPackage, является автоматизация процесса построения и поддержки интерфейсов, применимая для пакетов из различных предметных областей. При этом обеспечивается унификация не только по описаниям пакетов, но и по визуальным формам интерфейсов, что является принципиальным для быстрого освоения пользователями новых пакетов.

Исполнение композитных приложений в распределенной среде

Рассмотренные выше подходы к описанию пакетов, композитных приложений и интерфейсов работы с ними реализуются многофункциональной инструментально-технологической платформой CLAVIRE. Композитное приложение в рамках платформы CLAVIRE представляется в виде скрипта описания WF на языке EasyFlow, который может быть параметризован набором входных параметров и файлов, а также параметров исполнения. То есть один и тот же WF может быть исполнен для разного набора входных данных, а также в разных условиях исполнения. За разбор скрипта WF и за исполнение WF в целом отвечает компонент интерпретации (рис. 1). После получения скрипта данный компонент разбирает его и преобразует во внутреннее представление. Обработка представления WF производится непрерывно согласно событийной модели функционирования, то есть обработка WF происходит в рамках цикла обработки поступающих событий. При запуске отдельной задачи происходит интерпретация параметров узла WF и формируется описание запуска задачи, после чего сформированное описание передается в очередь компонента исполнения WF. Для определения параметров исполнения задачи привлекается компонент планирования, который, опираясь на данные моделей производительности пакета, а также, принимая во внимание текущую загруженность вычислительных ресурсов, определяет график для исполнения задач в очереди. Далее компонент исполнения подготавливает данные для пакета, производит запуск пакета на конкретном ресурсе, после чего обрабатывает выходные данные. Для подготовки пакета к запуску и обработки его результатов используется база пакетов, которая позволяет сопоставить абстрактные и фактические правила работы с каждым пакетом в составе WF. База пакетов состоит из двух частей: интерфейсной библиотеки и репозитория пакетов. Описание пакета в такой схеме хранится в виде файла со скриптом EasyPackage в репозитории. При этом компоненты системы для работы с данным описанием получают скрипты и сопутствующие файлы и интерпретируют их на своей стороне, используя при этом только ту информацию, которая им необходима. После получения и обработки результатов данные о завершении работы WF передаются обратно в интерпретатор, откуда они становятся доступны пользователю (через соответствующий интерфейс — графическую оболочку CLAVIRE или ПОИ).

Использование информации об исполнении композитных приложений

После завершения работы WF информация о результатах поступает в компонент хранения профилей исполнения WF. В нем хранится: состояние WF, информации об узлах, информация о полученных файлах и выходных параметрах для каждого узла. Помимо функции хранения данных, указанный компонент позволяет предоставлять пользователям возможность поиска WF из уже исполненных задач по определенным критериям (например по автору, запущенным пакетам). Накопленные данные используются при сборе и выводе статистики использования платформы в целом.

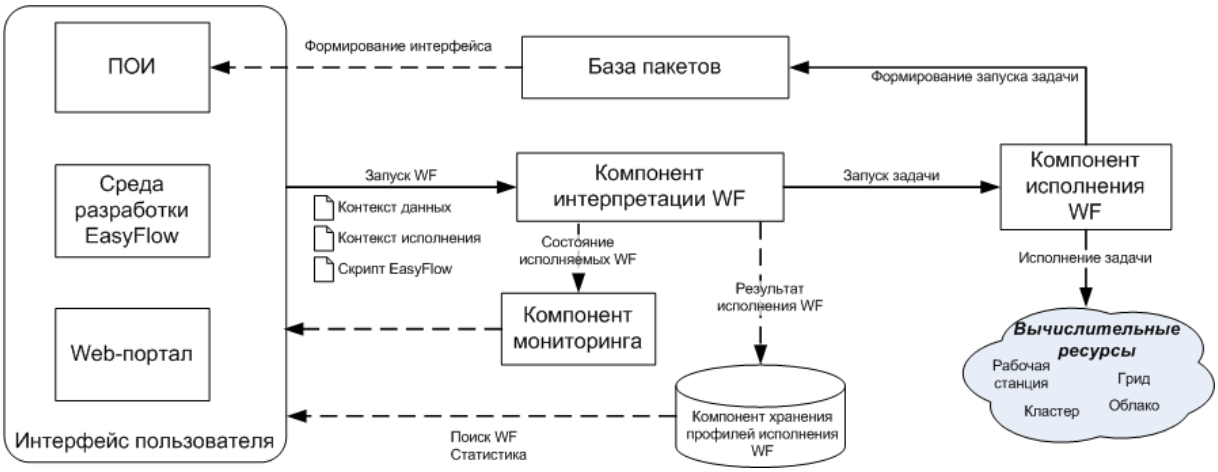

Рис. 1. Схема взаимодействия компонентов платформы при обработке WF

Описательной информации предметного уровня не всегда достаточно для воспроизводимости вычислительного эксперимента. Это связано с изменчивостью свойств гетерогенной вычислительной среды, а также с возможностью изменения компонентов самой платформы (например, при ее обновлении). Решением данной проблемы является хранение дополнительной информации о вычислительном эксперименте и полученных в ходе него результатов – информации о происхождении результатов (*provenance*). Потому описание запусков WF должно быть дополнено временными характеристиками работы WF и отдельных задач, информацией о запуске задачи (например, сформированная командная строка), данными об использованных ресурсах, данных о полученных файлах и их происхождении, информации о событиях.

Помимо обеспечения воспроизводимости результатов, хранение перечисленных типов данных позволит расширить функциональность некоторых компонентов платформы. Во-первых, автоматический анализ хранимых данных позволит включить механизмы корректировки работы платформы (например, уточнение моделей производительности пакетов при планировании). Во-вторых, это позволит более детально анализировать причины отказов при работе системы. К тому же данные о происхождении файлов могут быть использованы при создании надежного хранилища данных: при потере выходных данных их можно воссоздать, повторив соответствующий расчет. История исполнения WF также может быть полезна для оптимизации исполнения задач, если использовать ее качестве кэша: при нахождении идентичного запуска пакета (при совпадении всех параметров пакета) в качестве результатов исполнения можно сразу использовать результаты предыдущего запуска, найденного в истории (для некоторых пакетов).

Заключение

В работе предложены подходы к описанию предметно-ориентированных пакетов и композитных приложений в форме WF. Они обеспечивают эргономичность, гибкость и расширяемость распределенных сред для компьютерного моделирования и обработки данных в рамках парадигмы eScience. Данные подходы реализованы и апробированы в составе многофункциональной инструментально-технологической платформы CLAVIRE.

Литература

1. A Taxonomy of Workflow Management Systems for Grid Computing / J. Yu, R. Buyya // Journal of Grid Computing, Volume 3, Numbers 3-4. 2005. P. 171-200.

2. Seven Bottlenecks to Workflow Reuse and Repurposing / A. Goderis, U. Sattler, P. Lord, C. Goble // The Semantic Web – ISWC 2005. 2005. P. 323-337.
3. Интеллектуальные высокопроизводительные программные комплексы моделирования сложных систем: концепция, архитектура и примеры реализации / А.В. Бухановский, С.В. Ковальчук, С.В. Марьин // Известия высших учебных заведений. Приборостроение. 2009. №10. С. 5-24.
4. Современные программные комплексы компьютерного моделирования e-Science / А.В. Бухановский, В.Н. Васильев // Известия высших учебных заведений. Приборостроение. 2010. №3. С. 60-64.