ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ № 4(22), 2014, с. 183–194

УДК 004.75

С. Ю. Волков, О. В. Сухорослов

Реализация запуска многовариантных расчетов на платформе Everest

Аннотация. Многовариантные расчеты являются чрезвычайно важным классом приложений, обычно определяемых как набор вычислительных задач, определенных на множестве входных параметров и запускаемых с различными значениями данных параметров. Необходимость такого рода вычислений возникает во многих научных областях. Данная статья рассматривает веб-сервис, реализующий запуск данных приложений в распределенной вычислительной среде, а также облачную платформу Everest, на базе которой реализован данный сервис.

Ключевые слова и фразы: grid-технологии, многовариантные расчеты, вычислительные веб-сервисы.

Введение

Многовариантные расчеты играют важную роль в науке и инженерии. В качестве примера можно привести изучение поведения крыла самолета путем прогона ее модели несколько раз в зависимости от таких свойств крыла, как его скорость, угол атаки, форма и т.д. Многовариантные расчеты как раз предназначены для такого рода вычислений. Они включают в себя множество вычислительных параметров (таких как угол атаки в приведенном выше примере) и файлов. У каждого параметра есть множество значений, для каждой комбинации которых запускаются вычислительные задачи. Каждый файл может принадлежать двум или более задачам. У каждой задачи должен быть некий результат, обычно в виде выходных файлов и/или выходных параметров, описывающих вычисленные характеристики модели в зависимости от входных параметров. Итоговое множество

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 14-07-00309 а.

результатов всех вычислительных задач является результатом всего многовариантного расчета.

Представленный далее веб-сервис реализован под влиянием инструментария Nimrod [1, 2]. У данной системы есть так называемый план-файл, описывающий весь эксперимент, включая входные параметры, входные и выходные файлы и команду, выполняемую для каждой вычислительной задачи. Данные задачи генерируются для каждой комбинации значений параметров, используя их декартово произведение. Представленный в данной работе сервис также использует декартово произведение, но позволяет пользователям налагать ограничения на комбинации параметров в виде *директивы ограничений* (см. раздел 3). Он также позволяет фильтровать результаты вычислительных задач и ввести вычислительный критерий для определения 'лучших' (с точки зрения критерия) значений выходных параметров. Данный сервис реализован на платформе Everest. Прежде чем перейти к описанию сервиса, рассмотрим кратко саму платформу.

1. Платформа Everest

Everest [3, 4] представляет собой облачную платформу, поддерживающую публикацию, разделение и многократное использование приложений в качестве веб сервисов. Использованный подход основывается на унифицированном представлении вычислительных вебсервисов и его реализации с использованием стиля REST.

В отличие от традиционных средств разработки сервисов, Everest использует модель 'Platform as a Service' (PaaS) путем предоставления всей функциональности через удаленные интерфейсы. Платформа доступна пользователям для создания, запуска и разделения сервисов без необходимости устанавливать дополнительное программное обеспечение на своих компьютерах.

Другой отличительной чертой Everest является возможность подключать к сервисам внешние вычислительные ресурсы. Это означает, что разработчик сервиса может предоставить вычислительные ресурсы для запуска задач. Пользователь сервиса также может переопределить ресурс, стоящий по умолчанию, путем предоставления другого ресурса, предназначенного для запуска своих задач.

Хотя платформа не предоставляет собственной инфраструктуры для запуска вычислительных заданий как классические примеры PaaS, она может справляться с проблемами назначения ресурсов,

управлением заданий, передачей данных и т.д. без вмешательства пользователей.

2. Веб-сервис, реализующий запуск многовариантных расчетов

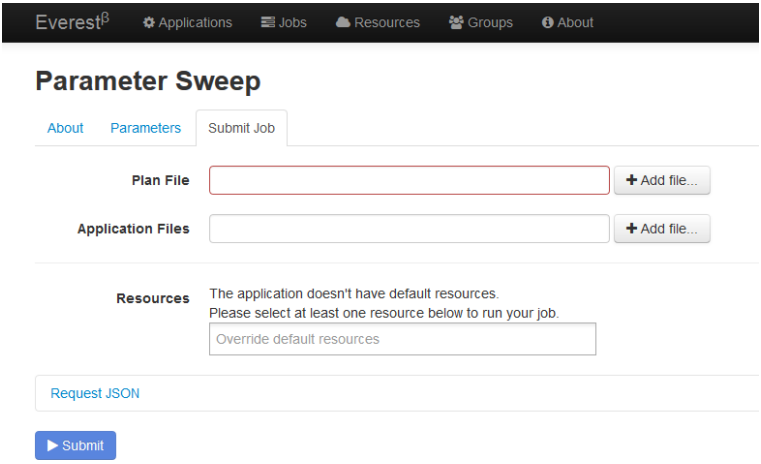

Рис. 1. Веб-сервис многовариантных расчетов

На рис. 1 показано, как данный сервис выглядит в веб-браузере. Для его работы пользователю необходимо предоставить два файла. Первым является план-файл, а вторым архив, содержащий входные файлы расчета (в данный момент поддерживаются архивы в форматах `tar.gz` и `zip`). По завершении расчета пользователь может скачать с сервера архив с результатами вычислительных задач, удовлетворяющих всем фильтрам и критерию, если таковые имеются, или всех задач в противном случае. Результат одной вычислительной задачи представляет собой папку, содержащую выходные файлы задачи и файл под названием `Parameters`, содержащий соответствующий задаче набор значений входных параметров.

Рассмотрим более подробно, как этот сервис работает. На рис. 2 изображена архитектура сервиса. Она включает в себя пакет под названием parametric, написанный на языке программирования Scala, и так называемое параметрическое приложение, реализованное в виде нового расширения платформы Everest и имеющее три функции:

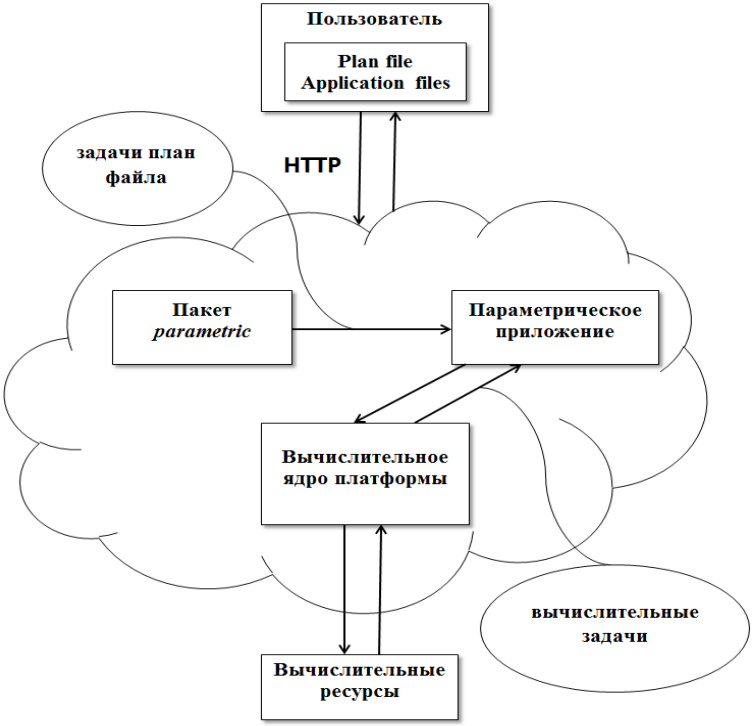

Рис. 2. Архитектура веб-сервиса

prolog, epilog и taskStateChanged. Функция prolog генерирует вычислительные задачи согласно план-файлу, переводит их во внутреннее представление платформы и передает их самой платформе для последующего выполнения на распределенных вычислительных ресурсах.

Функция taskStateChanged вызывается при изменении состояния вычислительной задачи, например, если задача завершилась успешно или с ошибкой. Также в ней происходит обработка результатов задачи, а именно применение с ним фильтров и вычисление значения критерия (см. раздел 3). Наконец, функция epilog вызывается в момент завершения последней задачи. В случае наличия критерия она выбирает оптимальные с его точки зрения результаты и загружает их на сервер для последующего скачивания пользователем.

3. Структура план-файла

План-файл представляет собой обычный текстовый файл, описывающий вычислительное задание. Он содержит следующие директивы: параметры, ограничения, входные файлы, команда, выходные файлы, фильтры и критерий. Директивы параметров, входных файлов, команды и выходных файлов являются обязательными. Директивы ограничений, фильтров и критерия опциональны. В план-файле каждая директива, кроме директивы команды, может занимать несколько строк. Иными словами, пользователь может сделать перенос директивы на новую строку, либо начать новую строку той же самой директивой. Директивы должны следовать в том порядке, в каком они будут описаны далее.

Для ссылки на значения параметров внутри директив используется стандартный синтаксис замены $var или ${var}. В нашем случае фигурные скобки '{' и '}' опциональны с одним единственным исключением. Если одна из заменяемых переменных является префиксом другой, то та, которая является префиксом, должна быть заключена в фигурные скобки. В качестве иллюстрации предположим, что у нас есть две переменные, var и var1. Если мы не заключим вторую в фигурные скобки (т.е. оставим ее как $var1), то будет заменена только ее часть $var, так как у нас уже есть переменная с именем var.

3.1. Директива параметров

Эта директива описывает параметры расчета. Она имеет следующий синтаксис:

```
Everest
parameter {имя параметра} {значения параметра}
```

Значения параметра могут принимать следующие две формы:

```
Everest
from {значение} to {значение} step {значение}
```

или

```
Everest
{список значений, разделенных пробелами}
```

Первый вариант может применяться только для численных значений параметров. В противном случае необходимо привести список

значений, разделенных пробелами. Если в самом значении есть пробелы, его необходимо заключить в кавычки.

Приведем несколько примеров использования директивы:

Everest

```
parameter i from 1 to 13 step 3
parameter d -12 0 0.12 36.01 125
parameter f file1 file2 "file 3"
```

3.2. Директива ограничений

Данная директива является опциональной. Она налагает ограничения на значения параметров, определенных предыдущей директивой. Вот ее синтаксис:

Everest

```
constraint {тип ограничения} {список ограничений, разделенных запятыми}
```

Ограничения представляют собой математические выражения, которые могут содержать круглые скобки и стандартные математические функции. Имена параметров используют синтаксис замены $var или ${var}, упомянутый выше.

Тип ограничения определяет, будут ли подставлены в ограничения сами значения параметров или индексы следования этих значений в директиве параметров. Соответственно тип имеет значения ‘value’ или ‘index’. Второй вариант может быть полезным, если пользователь хочет использовать не все возможные комбинации значений параметров (т.е. их декартово произведение), а, например, брать их попарно: первое значений одного с первым значением другого, второе со вторым и т.д.

Приведем несколько примеров использования директивы (имена параметров взяты из примеров предыдущей директивы):

Everest

```
constraint value $i + $d <= 100, 10*sqrt($i) - sin($i + $d) > 0.56
constraint index $i = $d
```

3.3. Директива входных файлов

Эта директива перечисляет входные файлы каждой задачи расчета. Она имеет следующий синтаксис:

Everest

```
input_files {имена или пути файлов, разделенные пробелами}
```

Также, как и в случае директивы параметров, имена или пути файлов должны быть заключены в кавычки, если они содержат пробелы. Они также могут быть параметризованы, т.е. содержать имена параметров с использованием синтаксиса замены. Пути к файлам задают директории внутри архива, принимаемого параметрическим заданием. Они поддерживают синтаксис glob.

Некоторым именам файлов или путям к ним может предшествовать символ '@'. Такие файлы называются *шаблонными файлами* и должны содержать в себе имена параметров (определенных в директиве параметров) вместе с синтаксисом замены $ или ${}. В качестве примера предположим, что таким файлом является скрипт, написанный на некотором языке программирования (в нашем примере на Scala). Предположим, что данный скрипт содержит следующие строки:

Everest

```
val v1 = $i
val v2 = $d
val result = someFunction(v1, v2)
```

Здесь i и d определены в примерах директивы параметров. Данный файл будет просмотрен на предмет содержания синтаксиса $var или ${var}. В нашем примере будут найдены выражения $i и $d. Они будут заменены на значения параметров i и d соответствующей вычислительной задачи. Например, если одна из задач имеет значения параметров i=4 и d=125, то представленная выше часть скрипта будет заменена на следующую:

Everest

```
val v1 = 4
val v2 = 125
val result = someFunction(v1, v2)
```

Приведем пример использования директивы:

Everest

```
input_files file1 @file2 /myDir1/* @/myDir2/$f
```

3.4. Директива команды

Данная директива определяет команду, выполняемую задачами на вычислительных узлах. Вот ее синтаксис:

Everest

```
command {команда}
```

Необходимо отметить, что команда может быть только одна. Данная директива также может быть параметризована.

Приведем пример использования директивы:

Everest

```
command ./MyScript.sh $i $d $f
```

3.5. Директива выходных файлов

Данная директива определяет выходные файлы вычислительных задач. Эти файлы должны быть результатом выполнения команды из предыдущей директивы. Как и в предыдущем случае, директива может быть параметризована. Вот ее синтаксис:

Everest

```
output_files {имена файлов, разделенных запятыми}
```

Как и в случае входных файлов, выходным файлам может предшествовать символ ‘@’. Такие файлы должны содержать выходные параметры задачи и иметь следующую структуру:

Everest

```
{параметр 1} = {значение параметра 1}
{параметр 2} = {значение параметра 2}
{и т.д.}
```

Важно отметить, что имена выходных параметров должны быть уникальны. Различные выходные файлы должны определять различные параметры.

Приведем пример использования директивы:

Everest

```
output_files f @output1 @"output 2"
```

3.6. Директива фильтров

Данная директива опциональна. Ее цель заключается в фильтрации выходных параметров задачи, определенных в предыдущей директиве. Результатом директивы фильтров являются задачи, выходные параметры которых удовлетворяют заданным пользователем фильтрам. Синтаксис данной директивы имеет вид:

Everest

```
filter {список фильтров, разделенных запятыми}
```

Как и ограничения, фильтры представляют собой математические выражения, в которые вместо значений параметров расчета подставляются значения выходных параметров вычислительной задачи. Так как имена этих параметров уникальны, директива просмотрит выходные файлы, помеченные символом @, и найдет соответствующие параметры.

Приведем пример использования директивы (p1, p2, p3, p4 и p5 являются выходными параметрами):

Everest

```
filter $p1 + $p2 - 10*$p3 >= 10.79, sin($p4) - sqrt($p5) < 4
```

3.7. Директива критерия

Данная директива также опциональна. Она применяется к задачам, прошедшим фильтры предыдущей директивы в случае наличия этой директивы, или ко всем задачам в противном случае. Ее результатом являются задачи, максимизирующие или минимизирующие критериальную функцию. Вот синтаксис данной директивы:

Everest

```
criterion {тип критерия} {функция критерия}
```

Типом критерия являются max или min (значения чувствительны к регистру). Функцией критерия является математическое выражение относительно выходных параметров задачи.

Приведем пример использования директивы (p1, p2, p3, p4 и p5 являются выходными параметрами):

Everest

```
criterion max $p1 - 10*$p2 + sqrt($p3 + cos($p4) - abs($p5))
```

4. Пример многовариантного расчета

Рассмотрим пример использования вышеперечисленных директив с использованием известной программы молекулярного докинга `Autodock Vina` [5]. В нашем расчете мы выполним десять задач докинга с десятью различными лигандами и найдем результаты, минимизирующие выходной параметр affinity (энергия связывания). Вот соответствующий план-файл:

Everest

```
parameter n from 1 to 10 step 1
input_files @run.sh vina write_score.py protein.pdbqt ligand${n}.pdbqt
input_files config.txt
command ./run.sh
output_files ligand${n}_out.pdbqt log.txt @score
criterion min $affinity
```

В данном примере мы используем директиву параметров для определения параметра n, который обозначает номер лиганда и принимает целочисленные значения от 1 до 10.

В директиве входных файлов мы определяем входные файлы каждой вычислительной задачи. Обратите внимание, как синтаксис ${n} используется в имени лиганда для обозначения порядкового номера n. Например, при n=1 соответствующая задача будет использовать входной файл `ligand1.pdbqt`, при n=2 файл `ligand2.pdbqt` и т.д.

Обратите внимание, что файлу `run.sh` предшествует символ '@'. Это означает, что в нем содержатся строки $n или ${n}, которые необходимо заменить соответствующим значением параметра n. Это делается для каждой задачи, так что разные задачи будут запускать разные скрипты.

Директива команд запускает скрипт `run.sh`.

Директива выходных файлов определяет выходные файлы каждой вычислительной задачи. В нашем примере это файл с результатами докинга, лог и файл `score`, который должен определять выходные параметры задачи, так как ему предшествует символ '@'. В данном случае это единственный параметр affinity. Как уже отмечалось, файл `score` должен иметь структуру affinity = *{значение параметра affinity}*.

Наконец, мы используем директиву критерия, чтобы определить задачи с минимальным значением affinity. Критериальная функция

является математическим выражением относительно помеченной символом '$' переменной affinity. Это означает, что директива будет искать в выходных файлах, помеченных символом '@', параметр с именем affinity и подставит его значение в критериальную функцию.

Заключение

В данной статье описан класс многовариантных расчетов, важный во многих научных областях. Нами был представлен веб-сервис, реализованный на облачной платформе Everest. В сравнении с предыдущими работами он имеет ряд преимуществ, таких как возможность фильтровать результаты вычислительных задач и/или находить оптимальные результаты в соответствии с критериями пользователя. Более того, так как вычисления реализованы в виде веб-сервиса, пользователям не требуется скачивать, устанавливать и запускать какое-либо программное обеспечение. В настоящее время сервис продолжает дорабатываться. Главной целью является сделать его как можно удобнее для использования, а именно в первую очередь сделать веб-интерфейс для создания план-файла. Этот вариант более удобен и устойчив к ошибкам, чем набор директив вручную в текстовых редакторах. Дальнейшая работа также затронет проблему эффективного планирования вычислительных задач на распределенных ресурсах.

Список литературы

[1] Bethwaite B., Abramson D., Bohnert F., Garic S., Enticott C., Peachey T., "Mixing the Grid and Clouds: High-throughput Science using the Nimrod Tool Family", *Cloud Computing: Principles, Systems and Applications*, eds. N. Antonopoulos, L. Gillam, Springer, London, 2010, pp. 219–237, ISBN: 978-1-84996-240-7 ↑ 184.

[2] Buyya R., Abramson D., Giddy J., "Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid", HPC Asia 2000 (May 14–17, 2000, Beijing, China), pp. 283–289 ↑ 184.

[3] Sukhoroslov O., Afanasiev A., "Everest: A Cloud Platform for Computational Web Services", *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, CLOSER 2014, SCITEPRESS — Science and and Technology Publications, 2014, pp. 411–416 ↑ 184.

[4] Everest, http://everest.distcomp.org/ ↑ 184.

[5] Autodock Vina, http://vina.scripps.edu/ ↑ 192.

Рекомендовал к публикации *Программный комитет*
Третьего национального суперкомпьютерного форума ***НСКФ-2014***

Об авторах:

Сергей Юрьевич Волков

Закончил Московский Физико-Технический Институт, в настоящий момент продолжает обучение в аспирантуре. Стажер-исследователь Института Проблем Передачи Информации РАН. Область научных интересов: grid-технологии, вычислительные веб-сервисы.

e-mail: fizteh.volkov@gmail.com

Олег Викторович Сухорослов

Кандидат технических наук, старший научный сотрудник лаборатории распределенных вычислительных систем Института проблем передачи информации РАН.

e-mail: sukhoroslov@iitp.ru

Образец ссылки на эту публикацию:

С. Ю. Волков, О. В. Сухорослов. *Реализация запуска многовариантных расчетов на платформе Everest* // Программные системы: теория и приложения: электрон. научн. журн. 2014. Т. 5, № 4(22), с. 183–194.

URL http://psta.psiras.ru/read/psta2014_4_183-194.pdf

Sergey Volkov, Oleg Sukhoroslov. *Implementation of parameter sweep computations on Everest platform.*

Abstract. Parameter sweep applications are a very important class of applications, which are typically defined as a set of computational experiments over a set of input parameters, each of which is executed with its own parameter combination. These co mputations arise in many scientific contexts. This article introduces the Parameter Sweep web service that runs such applications in distributed computing environment. Also discussed is the Everest cloud platform, on which this service is built. (*in Russian*).

Key Words and Phrases: grid technologies, parameter sweep computations, computational web services.