Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

"Национальный исследовательский университет "Высшая школа экономики"

Факультет бизнес информатики отделение ПМиИ

Кафедра технологий моделирования сложных систем

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

На тему Формализация и анализ гибридных систем потока работ для решения научных задач.

Студент группы №271

Козлов Алексей Сергеевич

Научный руководитель

_____
(должность, звание, Ф.И.О.)

Москва, 2013 г.

Содержание

1 Введение

За последних два десятилетия в научном сообществе компьютерное моделирование, названное e-Sience, стало незаменимой частью исследовательского процесса наравне с традиционными инструментами, такими как эмпирические, основанное на экспериментальных наблюдениях, теоретическое моделирование. Современная парадигма (e-Science) связана с развитием инструментария распределенных вычислений для научных исследований, позволяющего консолидировать вычислительные и программные ресурсы для решения сложных мультидисциплинарных задач в форме так называемых композитных приложений.

Термин e-Sience, возникший первоначально в Великобритании, где крупные исследовательские проекты в этой области начались в 2001г. Именно там было дано первое определение e-Science, получившее в дальнейшем широкое распространение: "научно-технологическая область, в которой всевозрастающую роль играет распределенное глобальное взаимодействие посредством сети интернет, с использованием очень больших коллекций данных, компьютерных ресурсов терауровня и высококачественной визуализации, доступных индивидуальному пользователю". В русском языке термин e-Science существует пока преимущественно в англоязычном варианте.
Необходимость в ,что кроме "обычной"информации, размещенной в интернете, специалисты, работающие в сфере науки и образования, нуждаются в доступе к крупномасштабным информационным массивам, базам данных, имеющим объемы памяти, измеряемые терабайтами. Работа с такими массивами требует вычислительных мощностей с производительностью уровня терафлоп. Задача e-Science, таким образом, - создание организационных и технологических структур, разработка соответствующего программного обеспечения для функционирования новой информационной среды с распределенными ресурсами (информационными и вычислительными), обеспечивающими доступ к ним индивидуальных пользователей, исследовательских групп, лабораторий и институтов. Основное русло реализации задач e-Science прокладывают грид-технологии. Эта концепция (нередко ее называют Grid Computing - распределенные сети вычислительных ресурсов) соответствует одному из ведущих и перспективных направлений развития инфор-

мационно-коммуникационные технологий. Наиболее распространенным подходом к представлению композитных приложений является формализм workflow.

Термин "workflow"используется в двух аспектах - как формальное представление некоторого процесса и как некоторый подход к автоматизации процессов, основанный на подобном представлении.

Начнем с первого аспекта. Буквальный перевод термина "workflow"как "поток работ"плохо раскрывает содержание данного понятия, поэтому зачастую используется термин "сценарий". Под workflow подразумевается формальное представление (модель) некоторого процесса, включающее в себя:

- Описание элементарных операций, из которых состоит процесс.
- Описание исполнителей, которые выполняют указанные операции.
- Описание зависимостей между операциями, а именно - потоков управления, которые определяют последовательность выполнения операций и синхронизацию между ними, и потоков данных, которые определяют передачу информации между операциями.
- Описание внешних событий, которые могут влиять на ход процесса, и правил их обработки.

Второй аспект термина "workflow"нашел отражение в определении, данном в Workflow Management Coalition [?], - "это автоматизация, полностью или частично, бизнес-процесса, при которой документы, информация или задания передаются для выполнения необходимых действий от одного участника к другому в соответствии с набором процедурных правил". Данное определение, несмотря на привязку к бизнес-процессам, хорошо отражает суть workflow-методологии как некоторого подхода к автоматизации вообще говоря различных процессов. Системой управления workflow (workflow management system, WFMS) будем называть систему, позволяющую создавать сценарии, запускать и управлять их выполнением [?]. WFMS состоит из набора программных компонентов, предназначенных для хранения и интерпретации описаний процессов (сценариев), создания и управления экземплярами запущенных процессов, а также организации их взаимодействия с участниками процесса и внешними приложениями. Программное приложение, непосредственно выполняющее интерпретацию и запуск сценария, а

также управляющее экземплярами запущенных процессов, будем называть средой выполнения сценариев (workflow engine).

Приведем основные отличия научных вычислительных процессов от бизнес-процессов:

- Ориентация на сложные вычислительные ресурсы.
- Количество ресурсов, которые потребуются для выполнения сценария (решения задачи), может быть неизвестно априори, так как для некоторых классов задач трудно оценить предстоящий объем вычислений.
- Необходимость работы в динамичной распределенной среде, в которой ресурсы не известны априори и могут быть подвержены отказам.
- Работа с большими объемами данных.
- Необходимость выполнять большое количество идентичных заданий с переменными параметрами.
- Необходимость следить за выполнением процесса и контролировать его, в том числе внося специальные для конкретных случаев изменения.
- Для многих научных workflow характерны иерархии вложенных workflow, создаваемых и уничтожаемых по необходимости.

Указанные особенности научных вычислительных процессов определяют требования, которым должна удовлетворять WFMS, подходящая для описания и выполнения данных процессов в виде сценариев. Традиционные WFMS, рассчитанные на работу с бизнес-процессами, в подавляющем большинстве не подходят для решения научных задач. Поэтому, требуется разработка новых, научных WFMS, с одной стороны опирающихся на сформировавшуюся workflow-методологию, а с другой стороны - специально рассчитанных на требования научных приложений. Пожалуй, главное, что могут почерпнуть научные WFMS из накопленного в данной области опыта, это способы формального представления workflow. В следующем разделе мы рассмотрим основные подходы к представлению сценариев, ссылаясь на уже созданных научных WFMS для того, чтобы одновременно дать обзор существующих решений в области научных вычислительных workflow.

Мотивация

Целью работы является построение модели workflow, ориентированного на решение научно-инженерных задач в распределенной вычислительной среде. Модель разрабатывается с целью получения более естественного способа описания реальных инженерных и научных задач. К целям работы так же относится разработка методов анализа построенной модели. Средства анализа разработанные в ходе работы:

- проверка построенных сценариев на ошибки.
- методология выработки наилучшей стратегии запуска сценария в распределённой вычислении среде на основе эвристик.

Программная реализация модели и средств её анализа будет представлена в виде модуля на языке программирования Python.

2 Обзор существующих моделей workflow

2.1 Модели управления запуском workflow

Большинство моделей управления workflow можно разделить на два класса: модели ориентированные на потоки данных(data-flows) и модели ориентированные на потоки управления(control-flows). Оба класса определяют взаимодействие между отдельными задачами-компонентами workflow, но различаются принципами реализации этого взаимодействия.
В workflow, основанных на принципе control-flow, связи между элементами workflow представляет передачу управления от одного задания следующему. Это позволяет формировать внутри workflow такие типы структур, как: последовательное выполнение, параллельное выполнение, циклическое выполнение и условные переходы.
в workflow, основанные на принципе data-flow, зависимости между элементами workflow, определяют направление потоков данных от

Так же существуют гибридные системы управления workflow, сочетающие в себе принципы обоих приведённых выше классов. Гибридные системы поддерживают оба типа зависимостей между компонентами workflow, но один из типов, как правило, является доминирующим, а другой используется при необходимости в особых случаях. Например, в data-flows системе такой как Triana, возникают ситуации, когда необходимо последовательно связать задачу, не производящую никакие данные, с задачей, не требующей данных на вход. В таком случае, на этом участке будет использован переход к control-flow зависимости.

Существует много техник построения расписаний запуска workflow. Но почти все они применимы, только если worklow представим в виде направленного ациклического графа (DAG) заданий. В случае, если workflow содержит циклы, параллельные циклы или условные переходы, применяются методы приведения графа заданий workflow к требуемому виду. С подобной проблемой преобразования графа задач уже сталкивались при разработке компиляторов, поддерживающих автоматическую параллелизацию, и были выработаны техники развёртывания параллельных циклов, устранения обычных циклов, предсказаний при условных переходов.

Примеры приложений, использующих DAG для представления workflow: Condor [10], Symphony [11], Cactus [12], UNICORE [13].

2.2 Различные подходы к представлению workflow

За время существования методологии workflow возникло несколько различных подходов их формального описания:

- Использование скриптовых языков;
- Использование графов. Наиболее часто используемые типы графов:
    - ориентированные ациклические графы(DAG);
    - сети Петри.

2.2.1 Скриптовые языки

Скрипт(script) представляет собой набор команд, предназначенный для выполнения интерпретатором без вмешательства пользователя. Скриптовые языки в качестве средства представления сценариев могут быть удобны пользователям, имеющим опыт программирования. Но не смотря на высокий уровень и простоту, они не достаточно наглядны и интуитивны для обычных пользователей.

В качестве примера успешного использования скриптового языка для конструирования сценариев стоит упомянуть систему GridAnt [5], Karajan [6].

2.2.2 Представление сценариев в виде графов

Другой способ представления workflow - это изображение в виде графа. Изначально, графы - это чисто математическая абстракция, но тем не менее, они удобны для неподготовленного пользователя, поскольку представляют workflow наглядно. Правда с увеличением сложности workflow графы тоже усложняются и их становится тяжелее просматривать, тем не менее, наглядность можно сохранить, используя иерархическое представление графа, позволяющее скрывать детали отдельных подграфов.

Для представления workflow наиболее часто используются два типа графов: ориентированные ациклические графы(DAG) и сети Петри.

2.2.2.1 Ориентированный ациклический граф

Ориентированным ациклическим графом называется любой ориентированный граф, в котором нет ориентированных циклов [8]. Вершинами графа являются, например, исполняемые программы или выполняемые операции, а ребра устанавливают зависимости между ними. Преимущество таких графов - простота структуры и реализации. Но есть и недостатки: они накладывают ограничения на типы сценариев - например, нельзя явно задать циклы без применения дополнительных конструкций, уже не связанных с графовым представлением. Кроме того, такие графы способны описывать только модель поведения процесса, не фиксируя его состояние во время выполнения.

2.2.2.2 Сети Петри

Сети Петри - особый класс ориентированных графов. Теория сетей Петри является хорошо известным и популярным формализмом, предназначенным для работы с параллельными и асинхронными системами. Основанная в начале 60-х гг. немецким математиком К. А. Петри, в настоящее время она содержит большое количество моделей, методов и средств анализа.

Рассмотрим основные преимущества сетей Петри при представлении workflow.
Графическая природа. Сети Петри - графический язык. Поэтому они интуитивно понятны и легки для изучения. Их графическая природа также удобна для взаимодействия с конечными пользователями.
Формальность описания. Сценарий, описанный в терминах сети Петри, задается строго и точно, потому что семантика классических сетей Петри, как и некоторых дополнений к ним (цвет, время, иерархия) были введены формально.
Выразительность. Сети Петри поддерживают все базисные элементы, необходимые для описания сценария. С их помощью могут быть смоделированы все управляющие конструкции, существующие в современных системах управления сценариями. Более того, точное представление состояний сценария позволяет описывать ситуации неявного выбора и сохранять промежуточные состояния, с возможностью возвращения к ним.
Свойства. В последние десятилетия были подробно изучены основные свойства сетей Петри. Прочные математические основания позволяют делать строгие вы-

воды из этих свойств.

Анализ. Сети Петри отличаются наличием большого числа методов анализа. Это их ценное преимущество с точки зрения использования для описания сценариев - данные методы могут быть использованы для доказательства различных свойств (выполнимости, инвариантности, мертвых переходов и т. д.) и для вычисления характеристик выполнения сценариев (время отклика, время ожидания, степень занятости и пр.). Таким образом, становится возможным оценивать альтернативные сценарии, используя традиционные инструменты анализа сетей Петри.

Подробнее сети Петри, их свойства и средства анализа будут описаны в следующей главе.

2.2.3 Модели, ориентированные на потоки данных

Помимо уже упомянутых методов, существует еще один подход, в котором для представления сценариев могут использоваться как скриптовые языки, так и графы, но который несколько отличается от обсуждавшихся ранее подходов своей спецификой. Речь идет о системах управления научными вычислительными сценариями, ориентированными на потоки данных. Как уже было сказано, в научных приложениях часто все необходимые действия сводятся к различным операциям над данными, т. е. в подобных процессах потоки управления и потоки данных совпадают. Поэтому нет необходимости вводить специальные элементы языка для описания логических конструкций, а достаточно просто обеспечить средства объединения элементарных модулей обработки данных в сеть. Каждый модуль имеет один или несколько входов и выходов, дуги сети соответствуют соединениям выхода одного модуля с входом другого, по которым осуществляется передача данных между модулями. Как только на вход модуля поступили все необходимые данные, происходит запуск программного кода модуля, который производит обработку входных данных, после чего полученные результаты (выходные данные) помещаются в выходы модуля и передаются по соединениям на вход других модулей.

Пример систем, обеспечивающих описанную функциональность: Triana [14], Kepler [15], Taverna [16].

Этот подход к представлению workflow и будет использоваться в данной работе.

3 Формальное описание разрабатываемой модели workflow

Как уже было сказано выше, за основу разрабатываемой модели будет взято возможность представления workflow в виде сети, состоящей из модулей, имеющих интерфейс в виде входов и выходов(именуемых в нашей работе портами), и каналов связи(именуемых в нашей работе связями), соединяющих входы и выходы модулей. После по того как данные поступили на все необходимые для начала работы модуля входы, модуль начинает работу, по окончанию работы обработанные данные помещаются в входные порты модуля.

Новизна предлагаемого подхода заключается в том, что каждый модуль может иметь несколько состояний,каждое из которых определяет требования к набору необходимых для запуска входов. После завершения работы модуль так же может отправить обработанные данные по одному из характерных для этого состояния набору выходов и изменить своё состояние.

3.1 Представление workflow

В нашей модели схема workflow задаётся графом связей. Граф связей workflow $WFG = (A, D)$ состоит из конечного и непустого набора вершин A, представляющих блоки и набора рёбер D , представляющих связи, соединяющие блоки через порты.
При этом в состав набора блоков всегда A входят два уникальных абстрактных блока Source и Stock, вводимых для задания точек запуска и завершения workflow.

3.1.1 Связи

В нашей модели связи - абстракция передачи данных, считаем, так же будем считать, что данные между блоками передаются мгновенно и без потерь.

Формально связь $d \in D$ представима в виде четвёрки (s, p_s, t, p_t), где s,$t \in A$ - идентификаторы входной и выходных блоков, $p_s \in out(s), p_t \in in(p)$ - выходной и входной порты соответствующих блоков.

Более того в рассматриваемой модели мы абстрагируемся от типизации данных передаваемых между блоками, считая что для любых двух соединённых блоков типы данных для каждой пары соединённых портов совместимы. Поэтому мы сразу можем представлять передаваемые по связям данные как сигналы.

Опишем некоторые свойства связей:

- Связь может быть установлена только между выходным портом одного блока и входным портом другого блока или самого себя.
- Связи могут быть построены как из одного выходного порта ко многим входным, так и из нескольких выходных в один входной порт.
- Если блок испускает сигнал по какому либо порту, то сигнал распространяется по всем связям исходящих из этого порта, что позволяет моделировать параллельный запуск нескольких блоков.
- Входные и выходные порты блоков могут оставаться неподключенными.

3.1.2 Блок

Блок в нашей модели является единицей исполнения workflow . В рассматриваемой модели считаем, что число состояний блока конечно, и набор входных и выходных портов фиксирован. Тогда поведение блока можно формализовать в виде недетерминированного конечного автомата.

Автоматом называется система, выходы которой зависят не только от поступивших входов, то и от текущего состояния системы. Состояние системы может быть обозначено переменной состояния $s \in \Sigma$, где Σ - это набор всех возможных состояний системы. Конечным автоматом называется автомат, для которого число состояний Σ конечно.

Классическая теория конечных автоматов (Hopcroft and Ullman, 1979) различает два вида автоматов: Автомат Мили и Автомат Мура. В Автомате Мили выходное значения сигнала явно зависит только от входных значений, в отличии от Автомата Мура, выходное значение сигнала в котором зависит лишь от текущего состояния данного автомата. Для полного задания автомата Мили или Мура дополнительно к законам функционирования, необходимо указать начальное состояние и определить внутренний, входной и выходной алфавиты. Между автоматами Мили и Мура существует соответствие, позволяющее преобразовать закон

функционирования одного из них в другой или обратно. Автомат Мура можно рассматривать как частный случай автомата Мили, имея в виду, что последовательность состояний выходов автомата Мили опережает на один такт последовательность состояний выходов автомата Мура, т.е различие между автоматами Мили и Мура состоит в том, что в автоматах Мили состояние выхода возникает одновременно с вызывающим его состоянием входа, а в автоматах Мура - с задержкой на один такт, т.к в автоматах Мура входные сигналы изменяют только состояние автомата.

Определим недетерминированный автомат атомарного блока, который будет использоваться в работе.

Конечный автомат блока

Конечным автоматом блока называется набор $M = (\Sigma, I, O, T, s_0)$, где

- Σ -набор конечных состояния,
- I - непустой набор доступных входных портов блока,
- O - непустой набор доступных выходных портов блока,
- $I \bigcap O = \varnothing$ -
- $s_0 \in \Sigma$ - начальное состояние,
- $T : \Sigma \times (2^I \backslash \{\varnothing\}) \to 2^{\Sigma \times (2^O}$ отображение сопоставляющее каждому состоянию и набору входных портов набор состояний с соответствующим набором выходных портов.

Если для заданного состояния $s' \in \Sigma$ и непустого набора входных портов $in \in 2^I \backslash \{\varnothing\}$ отображение T(s', in) не пусто, то результат отображения T(s', in) является набором возможных вариантов работы блока, с поглощением сигналов(данных) на входных портах, соответствующим переходом в новое состояние и отправлением сигналов по указанным выходным портам. Если же T(s', in) даёт пусто множество, считается, что условия запуска блока из состояния s' не соблюдены и начало работы программного модуля невозможно.

Как уже было определено выше, у блока имеется начальное состояние s_0, т.е. то в котором автомат блока находится перед запуском.

Визуально состояния соединены переходами, рядом с которыми указано, какие порты необходимы для срабатывания перехода и сигналы по каким портам будут испущены.

3.2 Возможные ошибки при построении workflow

Так как в нашей модели мы подразумеваем возможность асинхронной работы блоков , то как и в многопоточных компьютерных программах возможно возникновение состояния неопределённости, известной как состояние гонки(Race condition). В программировании состояние гонки определяется как ситуация, когда несколько потоков одновременно обращаются к одному и тому же ресурсу, причём хотя бы один из потоков выполняет операцию записи, и порядок этих обращений точно не определен.

Неопределенности подобного рода мы будем рассматривать как ошибки, обнаружению которых посвящена часть данной работы. Такого рода неоднозначности, особенно в реальных системах, то есть в условиях неопределенного времени вычислений блоков и их параллельного выполнения, может привести к совершенно иному ходу потока данных, нежели было задумано автором потока данных. Поэтому наличие подобного рода неопределенностей (состояний гонки) говорит скорее о неправильном построении схемы workflow, чем о некой задумке автора.

Рассмотрим некоторые возможные случаи неопределённого поведения контексте нашей модели workflow, которые можно обнаружить ещё на этапе построении сценария workflow:

- На один входной порт могут прийти два сигнала одновременно.
- На входные порты блока одновременно приходит такой набор сигналов, что запуск блока возможен по разным наборам сигналов. Такое состояние гонки потенциально возможно для блоков, для которых выполняется:
  $(S, \Sigma, \Lambda, T, s_0)$ - автомат блока;
  $\exists s \in S, \exists in1, in2 \in 2^{\Sigma} \backslash \{\varnothing\}, in1 \neq in2 :$
  $T(s, in1) \neq \varnothing \bigwedge T(s, in2) \neq \varnothing.$

Более подробно и строго об неоднозначностях в поведении потока данных будет сказано при описании алгоритма, моделирующего работу workflow.

3.2.1 Принципы моделирования работы workflow

Для построенной модели workflow, был построен алгоритм, имитирующий работу построенного сценария workflow и позволяющий обнаружить ошибки, допущенные при построении, такие как состояния гонки.

Прежде чем начать описание самого алгоритма, определим условия, характеризующие начало и завершение работы сценария workflow.

- workflow начинает работу, испуская сигналы из всех выходных портов блока Source;
- workflow заканчивает работу, как только на все входные порты блока Stock поступили сигналы, после чего они поглощаются.

Мы считаем, что workflow построен корректно если выполняются все три правила:

- в ходе выполнения этой workflow невозможен случай возникновения состояние гонки;
- workflow всегда завершается;
- при завершении работы workflow, не остаётся непоглощённых сигналов.

Такие workflow мы будем назвать корректными.

Для корректных workflow (при требовании отсутствия состояний гонки) порядок запуска блоков не зависит от времени их работы, а только от распространения сигналов, будем считать, что любой блок в любом состоянии выполняется фиксированное время $T > 0$.

Для описания алгоритма работы workflow потребуется ввести некоторые дополнительные понятия, позволяющие определить его динамические характеристики.

Понятие разбиения потока вводится для формализации понятия конкуренции между сигналами, распространяющимися по связям внутри workflow.

Разбиение потока

Разбиением потока назовём пару (fork, split), где fork - имя блока, которой испустившего набор сигналов, а split - кортеж состоящий из n элементов, где n - число исходящих от блока сигналов. А каждый элемент кортежа может быть либо 1, либо 0, либо другим разбиением.

Нулевое разбиение потока

Под нулевым разбиением понимаем разбиение с пустым параметром fork и кортежем splitting из одного элемента т.е. (, (1))

Операции над разбиением потока

Далее введём некоторый набор операций, доступный над разбиениями, задаваемых в виде рекурсивных функций. В работе функции приведены в виде таблиц, где название столбца - значение первого аргумента функции, название строки - значение второго аргумента функции, а их пересечение - возвращаемое значение.

- Произведение ($\times$) двух разбиений задаётся рекурсивной функцией mult от двух аргументов, возвращающей новое разбиение. Табл.1
- Сложение ($+$) двух разбиений задаётся рекурсивной функцией sum от двух аргументов, возвращающей новое разбиение. Табл.2
- Проверка на конкуренцию ($\perp$) задаётся рекурсивной функцией concurrent от двух аргументов, возвращающей True в случае выявления конкуренции и False в противном случае. Табл.2

В каких случаях используются разбиения и применяются данные операции над ними будет определено в дальнейшем. Для наглядности приведём несколько примеров применения этих операций над разбиениями:

Умножение двух разбиений:

$$(, (1)) \times (b1, (0,1)) = (, (b1, (1,0))); \tag{1}$$

$$(, (b1, (1,0, 1))) \times (b2, (0,1)) = (, (b1, ((b2, (0,1)),0, (b2, (0,1))))); \tag{2}$$

Сложение двух разбиений:

$$(,(b1,(1,0))) + (,(b1,(0,1))) = (,(b1,(1,1)));\tag{3}$$

При сложении двух разбиений так же применяется рекурсивное правило упрощения, которое определяется как:
Пусть дано разбиение WS = (fork, split), у которого кортеж split состоит из одних единиц и WS не является нулевым разбиением, тогда $WS \equiv 1$:

$$(,(b1,(1,0))) + (,(b1,(0,1))) = (,(b1,(1,1))) \equiv (,(1));\tag{4}$$

Проверка двух разбиений на конкуренцию:

$$(,(b1,(1,0))) \perp (,(b1,(0,1))) = True\tag{5}$$

$$(,(1)) \perp (,(b1,(0,1))) = False\tag{6}$$

Заключение

Список использованных источников

1 Грицанов А.А. и др. Новейший философский словарь. Мн.: Книжный Дом., 2003.