МОДЕЛЬ РАСПРЕДЕЛЕННОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ НА СЕТИ ПЕТРИ

С.К. Дорожкин

В статье рассматриваются вопросы построения моделей распределенных вычислительных систем с помощью раскрашенных или цветных сетей Петри. В работе приводится пример построения модели распределенной вычислительной системы.

В настоящее время наблюдается рост числа проектов по разработке распределенных и параллельных систем для многих прикладных областей – от больших масштабируемых систем в области телекоммуникации до систем среднего и начального уровня в области web-приложений. Разработка вычислительных систем этого класса является сложным процессом. И наиболее эффективным подходом к проектированию такого рода систем является моделирование.

Цветные или *раскрашенные* сети Петри (Colored Petri Nets) обеспечивают среду для создания, тестирования и анализа моделей распределенных вычислительных систем. Модель системы, построенная на цветной сети Петри, описывает состояния, в которых может находиться моделируемая система, и переходы между этими состояниями. Эти состояния системы в терминах сетей Петри принято называть *местами*. Сети Петри применяются для моделирования различных вычислительных систем и процессов, начиная от телекоммуникационных протоколов и системного программного обеспечения и заканчивая моделированием аппаратуры больших вычислительных систем и встроенных управляющих систем специального назначения.

Разработка сетей Петри была инициирована потребностью в создании промышленного языка моделирования, имеющего сильную теоретическую базу и являющегося достаточно гибким для использования на практике при разработке промышленных вычислительных систем. Для достижения этой цели и были созданы *цветные* или *раскрашенные* сети Петри. Они сочетают в себе свойства сетей Петри по моделированию параллельных и синхронных процессов, а примитивы языка программирования позволяют использовать данные разных типов, которые называются, в терминах сети Петри, *цветными наборами* или *мультимножествами*.

Модели, построенные на цветных сетях Петри, могут быть построены из составных модулей, которые могут представлять собой также сети Петри. Это особенно важно, когда моделируются большие промышленные системы, к которым можно отнести распределенные вычислительные системы. Модульная концепция цветных сетей Петри основывается на иерархическом механизме, который поддерживает разработку модели вычислительной системы как снизу вверх, так и сверху вниз. Новые модели могут быть созданы из уже готовых модулей, что позволяет значительно упростить процесс построения модели вычислительной модели, а также одни и те же модули могут использоваться для моделирования различных узлов вычислительной системы, выполняющих одинаковые функции.

Модели вычислительных систем, построенные на сетях Петри – исполняемые, т.е. они позволяют исследовать поведение вычислительной системы, симулируя процессы, которые протекают в реальной вычислительной системе. Симуляция работы вычислительной системы на модели используется для исследования поведения системы в той или иной ситуации, для проверки корректности функционирования вычислительной системы или для исследования ее производительности.

Время выполнения задачи или время ответа играет серьезную роль в работе распределенной вычислительной системы. Корректное функционирование большинства систем зависит от времени, затрачиваемого на выполнения определенных действий. Различные варианты реализации вычислительной системы могут серьезно влиять на время выполнения одних и тех же операций. Моделирование призвано помочь решить подобную задачу. В цветных сетях Петри имеется возможность моделировать время,

затраченное на выполнение операций в системе. Такие модели позволяют анализировать производительность вычислительных систем.

В качестве примера рассмотрим модель распределенной вычислительной системы, обеспечивающей работу с распределенной базой данных. Моделируемая вычислительная система включает в себя несколько серверов. Каждый сервер хранит свою копию базы данных, которая управляется локальным менеджером базы данных. Модель, построенная с помощью цветной сети Петри, показывает процесс синхронизации контента базы данных в случае изменения данных на одном из вычислительных узлов. Таким образом, имеется набор менеджеров:

$DBM=\{d1, d2, ..., dn\}$.

Каждый менеджер может вносить изменения в свою локальную копию базы, а затем должен послать сообщение с требованием провести соответствующие обновления всем остальным менеджерам. В данном примере мы не будем рассматривать содержание сообщения с требованием изменить содержание базы, нас будет интересовать только его заголовок, который описывает *отправителя* (Sender) и *получателя* (Receiver). Таким образом, у нас есть набор сообщений:

$$MS = \{(s,r) \mid s,r \in DBM \wedge s \neq r\},$$

где отправитель s и получатель r должны находится на различных узлах вычислительной системы. Когда менеджер s делает обновление в своей локальной базе, он обязан послать всем остальным менеджерам следующее сообщение:

$$M(s) = \sum_{r \in DBM - \{s\}} 1'(s,r),$$

где сумма обозначает формирование набора из n - 1 элементов сложением мультимножеств $1'\{s,r\} \mid r \in DBM - \{s\}$, каждое из которых содержит один элемент. Результирующий набор содержит по одному сообщению для каждого получателя r, имеющего в заголовке s в качестве отправителя. Все вместе представленные выше определения формируют следующую систему:

constants: n : integer (* $n \geq 3$ *);

colour sets: $DBM = \{d1, d2, ..., dn\}$;

$MS = \{(s,r) \mid s,r \in DBM \wedge s \neq r\}$

E = {e};

functions: $M(s) = \sum_{r \in DBM - \{s\}} 1'(s,r)$

variables: $s, r : DBM$;

Графическое изображение модели вычислительной системы представлено на рис. 1.

Каждый менеджер базы данных может находиться в трех различных состояниях: *Inactive (Неактивен), Waiting (Ожидает)* и *Performing (Выполняет* обновление*)*. В состоянии *Waiting* менеджер ожидает запрос с подтверждением о том, что его запрос на обновлением удаленной копии базы выполнен другим менеджером. В состоянии *Performing* менеджер производит обновление базы по запросу удаленного менеджера.

Каждое сообщение в модели может иметь следующие состояния: *Unused, Send, Received, Acknowledged,* а сама система может находится в состоянии *Active (Активна)* и *Passive (Пассивна).*

Изначально все менеджеры находятся в состоянии *Inactive*, а все сообщения в *Unused*. Такое состояние определяется инициализационным выражением. *DBM.all*() формирует набор, который содержит только один маркер каждого типа (цвета) в наборе *DBM*. Аналогично *MES.all*() формирует набор, который содержит только один маркер каждого типа (цвета) в наборе *MS.*

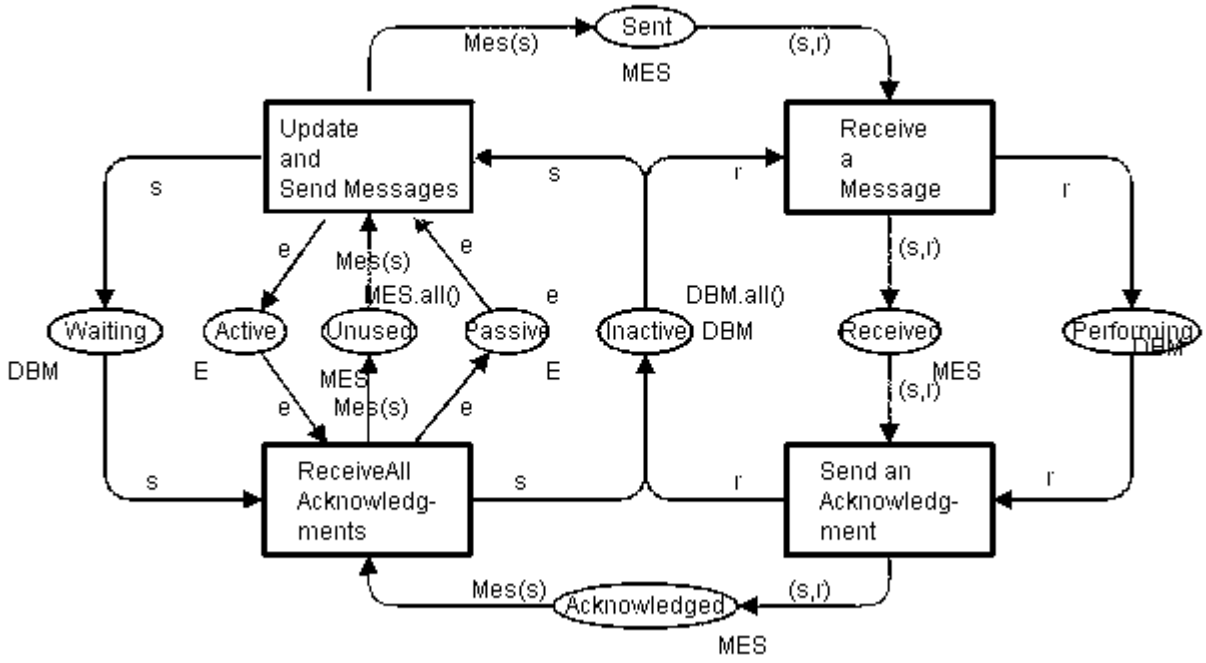

Рис. 1. Графическое изображение модели распределенной вычислительной системы

Когда менеджер s проводит обновление (состояние *Update*) базы, он должен отправить об этом сообщение всем остальным менеджерам. После обновления локальной копии данных состояние менеджера меняется с *Inactive* на *Waiting*, а состояние сообщения меняется с *Unused* на *Sent*. После этого менеджер ожидает, пока все остальные менеджеры пришлют ему оповещение о том, что они произвели обновление своих локальных копий базы данных.

Когда один из этих удаленных менеджеров *r* получает сообщение, он меняет свое состояние с *Inactive* на *Performing*, т.е. производит обновление, а состояние соответствующего сообщения от менеджера *s* меняется с *Sent* на *Received*. После этого менеджер *r* должен отправить уведомление *Acknowledgment*, подтверждая таким образом, что он закончил обновление своей копии данных, а его состояние при этом меняется с *Performing* на *Inactive*. Состояние сообщения меняется с *Received* на *Acknowledged*. Когда все сообщения *M*(*s*), которые были посланы менеджером *s*, получат статус *Acknowledged*, менеджер *s* меняет свое состояние с *Waiting* на *Inactive*, так как это означает, что все удаленные менеджеры произвели обновление базы. Состояние сообщения *M*(*s*) меняется с *Acknowledged* обратно на *Unused*.

Чтобы обеспечить актуальность различных копий на узлах распределенной вычислительной системы, такая простая схема синхронизации позволяет произвести только одно обновление в один момент времени. Другими словами, когда менеджер инициировал обновление, оно должно быть выполнено на всех локальных копиях базы данных, и только после этого новое обновление базы может быть инициировано менеджером. Для этого используется состояние *Passive*. В нем может находиться только один маркер, что соответствует тому, что только один менеджер может послать запрос на обновление базы данных остальным.

Эта модель демонстрирует основные свойства сети Петри: параллельность, конфликтность, зависимость по условию.

В первоначальном состоянии модели распределенной вычислительной системы переход *Update and Send Message* открыт для всех менеджеров, но сработать он может только для одного менеджера базы данных (из-за единственного маркера в месте *Passive*). Такая ситуация называется конфликтом, из-за того что связанные элементы

доступны поодиночке, но одновременно не разрешены. Можно сказать, что переход *Update and Send Message* конфликтен по отношению к себе.

Когда переход *Update and Send Message* выполняется для некоторого менеджера *s*, то переход *Receive a Message* в этот момент параллельно доступен для всех менеджеров, отличных от *s*. Такая ситуация называется параллельностью, и можно сказать что переход *Receive a Message* параллелен сам себе.

Переход *Receive all Acknowledgments* разрешен только, тогда когда переход *Update and Send Message* выполняется для какого-то определенного менеджера *s*, а переходы *Receive a Message* и *Send an Acknowledgment* выполнились для всех менеджеров, отличных от *s*. Такая ситуация называется зависимостью по условию (потому что есть элемент, который может сработать только после срабатывания другого элемента).

Симуляция работы вычислительной системы используется с той же целью, что и тестирование программного обеспечения. К сожалению, получить данные о порядке функционирования вычислительной системы, достаточные для ее анализа, с помощью симуляции можно только для достаточно простой вычислительной системы. Для сложных систем симуляция не даст необходимой информации о порядке функционирования вычислительной системы или отдельных ее узлов, поэтому для получения более детальной информации о вычислительной системе используется метод анализа состояний системы.

При анализе состояний моделируемой системы используется так называемый граф состояний. На нем отражаются все возможные состояния моделируемой системы и переходы между ними. Таким образом, появляется возможность проверить корректность функционирования моделируемой системы. Пример графа состояний для рассматриваемой модели приведен на рис. 2.

Узлами на данном графе являются состояния, принимаемые системой в процессе ее функционирования, т.е. все возможные разметки сети Петри, а дуги этого графа отмечают переходы из одного состояния в другое с помощью связующих элементов.

Недостатком такого метода исследования поведения системы является то, что граф состояний будет сильно усложняться с ростом количества узлов, на которых будут работать локальные менеджеры базы данных. Сравнительные данные приведены в таблице.

Приведенные в таблице данные показывают, что в общем случае граф состояний цветной сети Петри растет очень быстро с увеличением количества цветов (различных типов данных) сети Петри. Однако на практике достаточно рассмотреть модель с небольшим объемом различных цветов, чтобы установить корректность работы моделируемой системы. В данном случае достаточно проверить правильность функционирования распределенной базы данных на наборе из 3 или 4 менеджеров (которые кодируются разными цветами). В случае корректной работы с таким количеством менеджеров можно быть уверенными в том, что система будет работать корректно и с большим числом узлов.

К сожалению, такой подход неприменим при исследовании производительности моделируемой системы. В этом случае необходимо вводить параметры, отражающие время обработки запросов в узлах системы. Время в сетях Петри вводится посредством так называемых глобальных часов. Время может отсчитываться дискретно или постоянно. Чтобы ввести время в модель, необходимо к каждому маркеру добавить переменную времени или, как она еще называется, временную метку. Временная метка обозначает момент модельного времени, после наступления которого, маркер может быть использован, то есть, перенесен из одного места в другое.

Выполнение модели, содержащей временные элементы, похоже на выполнение моделей в других языках с дискретным временем, содержащих очереди событий. Система находится в одном состоянии до тех пор, пока есть маркеры, разрешающие переход в данный момент времени. Когда маркеры, содержащие переменную времени со временем исполнения, равным текущему, кончаются, система увеличивает системное

время на величину временной переменной с минимальным значением. Каждая разметка сети существует в ограниченном интервале модельного времени.

Введем для перехода *Update and Send Message* переменную @+5. Это значит, что выполнение операции *Update and Send Message* занимает 5 единиц модельного времени, и, следовательно, переменные времени в маркерах места *Passive* получают значение +5 единиц времени после срабатывания перехода *Update and Send Message*.

Менеджеры базы данных DBM	Узлы	Дуги
2	7	8
3	28	42
4	109	224
5	406	1,090
6	4,459	4,872
7	5,104	20,426
8	17,497	81,664
9	59,050	314,946
10	196,831	1,181,000
15	71,744,536	669,615,690
20	23,245,229,341	249,439,571,680

Табл. Рост числа элементов графа состояний, в зависимости от числа цветов сети

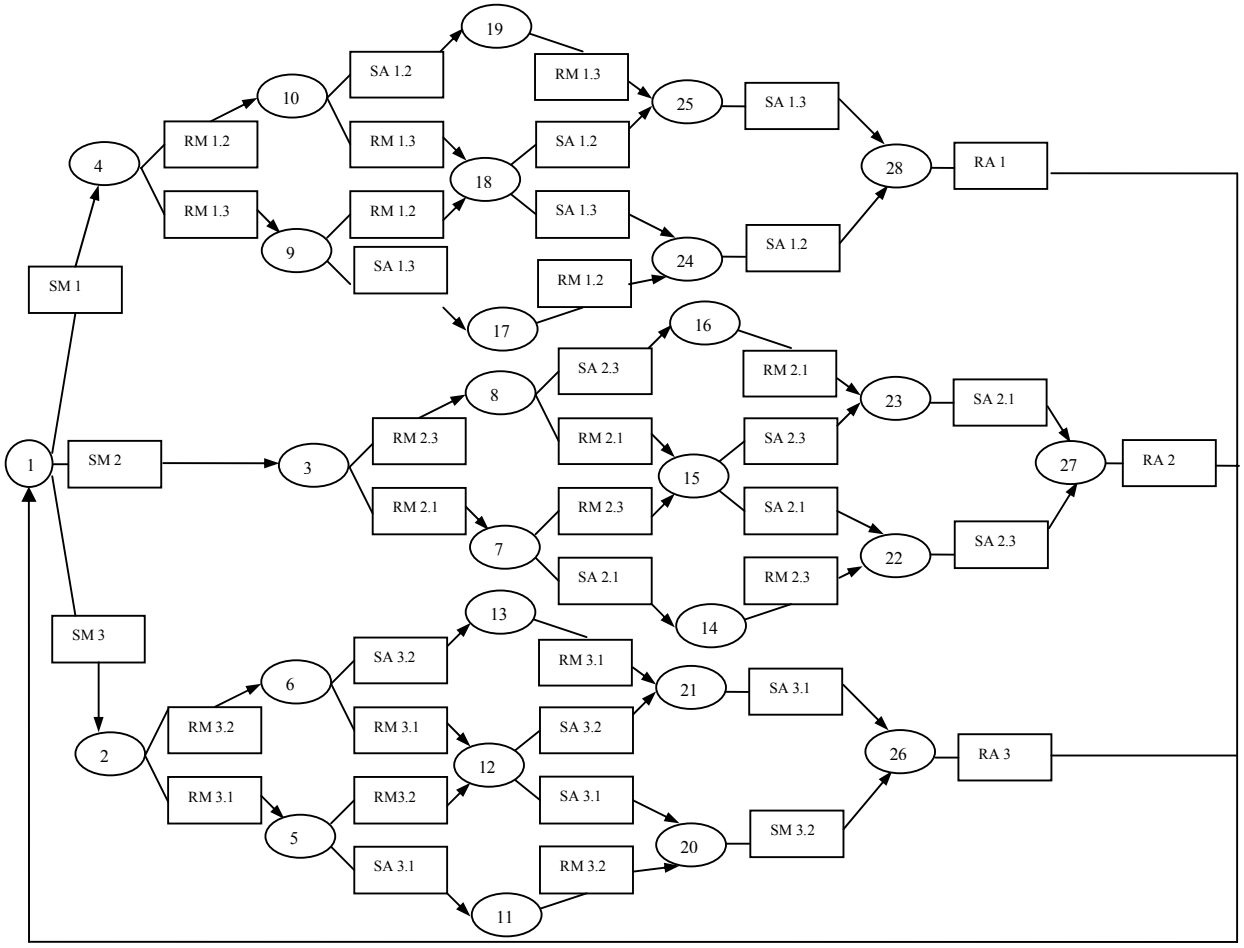

Рис. 2. Граф состояний моделируемой системы

Использование переменных времени в моделях позволяет оценить загрузку от-

дельных узлов вычислительной системы и производительной вычислительной системы в целом.

Модель распределенной вычислительной системы, построенная с помощью сети Петри, является эффективным инструментом, используемым при проектировании вычислительной системы. Аппарат сетей Петри позволяет получить абстрактное описание процессов функционирования вычислительной системы. Результаты моделирования позволяют получить информацию о порядке функционирования как всей вычислительной системы, так и отдельных ее узлов и оценить ее производительность.

В данной работе рассмотрены основные вопросы построения моделей распределенных вычислительных систем, с помощью цветных сетей Петри. В работе приведен пример построения модели распределенной вычислительной системы.

Литература

1. Котов В.Е. Сети Петри. М.: Наука, 1984.
2. Harper Robert Programming in Standard ML. Carnegie Mellon University Spring Semester 2001. Working draft. November, 2004.
3. Kurt Jensen Introduction to the practical use of coloured Petri nets. Springer-Verlag. 1998.
4. Kurt Jensen, Lars M. Kristensen. The practitioner’s guide to coloured Petri nets. Springer-Verlag, 1998.