УДК 681.61

МОДЕЛИРОВАНИЕ GRID-УЗЛА НА ОСНОВЕ СЕТЕЙ ПЕТРИ

А.Ю. ШЕЛЕСТОВ

Исследуется узел Grid-системы с применением аппарата сетей Петри. Построена модель работы узла и рассмотрены ее структурные свойства. Показано, что полученная сеть является ограниченной, живой и не содержит недостижимых позиций. Проведен анализ выполнимости свойств взаимного исключения и равноправия.

ВВЕДЕНИЕ

Бурное развитие вычислительной техники и систем телекоммуникаций, сопровождаемое усложнением инфраструктуры распределенных информационно-аналитических систем, привело к необходимости развития средств и методов их моделирования. Одной из наиболее активно развиваемых технологий создания распределенных систем является Grid-технология, направленная на обеспечение работы виртуальных организаций, решающих вычислительно сложные задачи с использованием распределенных хранилищ данных и мощных вычислительных ресурсов [1].

Для моделирования подобных систем используются различные подходы, ни один из которых не может претендовать на исчерпывающее описание, а представляет лишь один из аспектов функционирования или структуры системы. Достаточно полный обзор существующих моделей разного уровня абстракции содержится в [1–3]. В данной работе для моделирования динамики таких систем предлагается использовать математический аппарат сетей Петри [4], поскольку Grid-система представляет собой набор взаимодействующих между собой компонентов (вычислительных узлов и хранилищ данных), которые могут функционировать параллельно и в работе которых должна быть обеспечена синхронизация.

Большое значение такие модели приобретают при исследовании Grid-систем наблюдения Земли, поскольку в этих системах особое внимание уделяется синхронизации доступа к общим сегментам данных и распараллеливанию вычислений.

ПРЕДМЕТНАЯ ОБЛАСТЬ И ФОРМУЛИРОВКА ТРЕБОВАНИЙ К МОДЕЛИ

Распределенные системы обработки данных наблюдения Земли имеют свои особенности. В частности, для таких систем характерно использование как распределенных вычислительных, так и информационных ресурсов (совместно используемых баз данных или хранилищ). Как правило, прикладные задачи, решаемые в системах обработки спутниковых данных (в том числе Grid-системах), предполагают обмен данными большого объема, использование информации из различных географически удаленных друг от друга источников, синхронизацию потоков выполнения задач и доступ к общим

хранилищам. Именно поэтому Grid-инфраструктуры, связанные с обработкой данных наблюдения Земли, выделяют в отдельный класс Grid-систем, исследованию которых посвящен проект DEGREE [5] в рамках программы EGEE. Подобные системы содержат следующие функциональные компоненты.

1. Метапланировщики (управляющие узлы) уровня всей системы в целом или ее отдельной, архитектурно значимой, части, которые обеспечивают передачу заданий (т.е. исполняемого программного кода и данных) на требуемый вычислительный ресурс (в терминах пакета gLite [6] — Computational Element, или вычислительный элемент) или ресурс хранения (Storage Element). В состав метапланировщика входят алгоритмы планирования выполнения задач, которые могут отличаться для разных систем.

Примеры метапланировщиков: GridWay [7] или WMS (gLite сегодня не находит широкого применения в исследовании Земли из космоса), GrAS [8]. Однако наиболее востребован GridWay, включенный в состав последних версий Globus Toolkit [9] и активно применяемый в гетерогенных системах.

Метапланировщики являются надстройкой над базовой Grid-инфраструктурой (работающей под управлением программного обеспечения промежуточного уровня) и взаимодействуют не с конкретными аппаратными ресурсами, а с представляющими эти ресурсы Grid-сервисами. Для выполнения своих функций метапланировщики используют данные, предоставляемые информационными сервисами Grid-инфраструктуры, что позволяет им учитывать общее состояние системы при принятии решений о распределении задач между Grid-ресурсами. Для отправки задач на конкретные ресурсы системы используются интерфейсы, предоставляемые сервисами выполнения задач.

2. Локальные планировщики, например, Torque [10], PBS Pro [11], Sun Grid Engine [12], LSF [13], которые обеспечивают управление выполнением задач на локальном ресурсе и взаимодействуют с метапланировщиком.

3. Распределенные вычислительные узлы и ресурсы хранения данных. Доступ к ним и их использование должны быть синхронизированы, а на узлах с этими ресурсами установлены также Grid-сервисы программной инфраструктуры (MDS, GRAM, GridFTP, RFT и т.д.), которые могут предоставлять на верхний уровень информацию о состоянии данного ресурса и тем самым позволять реализацию эффективного планирования выполнения задач.

Таким образом, выполнение задачи в Grid-среде является трехуровневым иерархическим процессом, на верхнем уровне которого находится метапланировщик, на среднем — локальные планировщики, а на нижнем — физические вычислительные ресурсы. В частности, такая иерархия поддерживается в системах на основе метапланировщика GridWay платформы Globus Toolkit или брокера ресурсов Resource Broker платформы gLite. Задачи пользователей поступают в очередь метапланировщика заданий. Этот компонент, используя информацию от информационных сервисов Grid-системы и отдавая команды сервисам передачи данных и управления задачами, распределяет задачи по ресурсам Grid-системы, реализуя некоторый алгоритм планирования с учетом статистики, предоставляемой информационным сервисом MDS.

Все перечисленные выше компоненты и взаимосвязи между ними показаны на рис. 1.

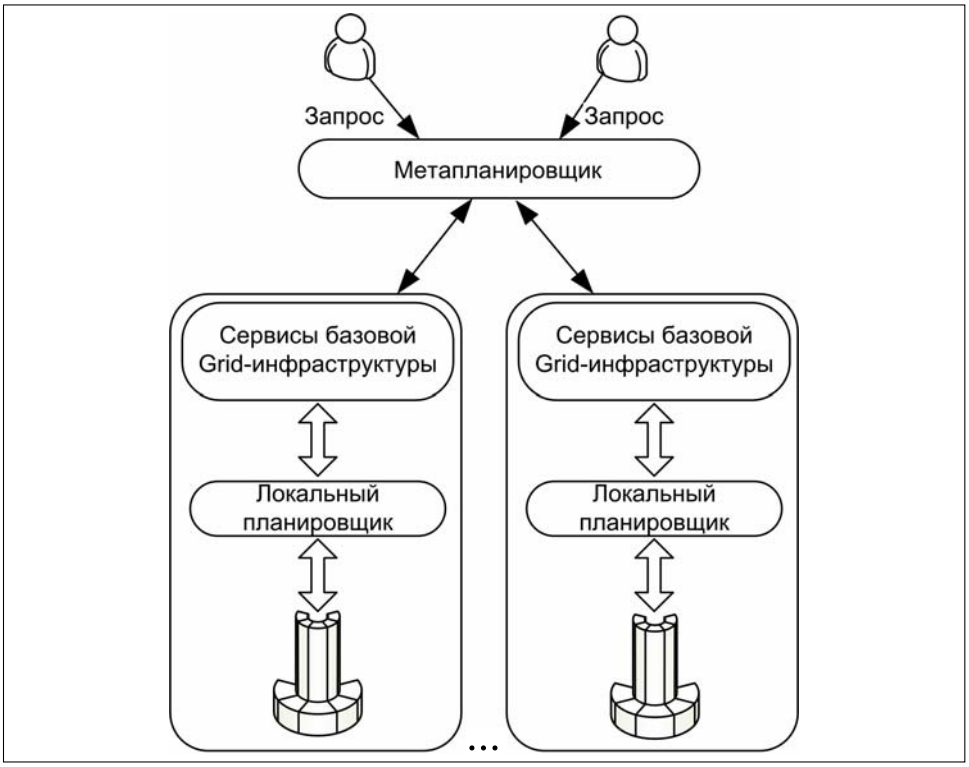

Рис. 1. Роль метапланировщика в Grid-инфраструктуре

Поскольку задачи обработки данных наблюдения Земли характеризуются высокой вычислительной сложностью и большим объемом используемых данных, то очень важно обеспечить синхронизацию как отдельных потоков выполнения, так и доступа к общим ресурсам, в частности хранилищам данных.

В данной статье исследуется синхронизация доступа к общей памяти. При этом учитывается то, что в структуре Grid-системы наблюдается фрактальность, т.е. синхронизацию доступа необходимо обеспечивать как на уровне метапланировщика и доступа к общему хранилищу (одному из общих ресурсов системы), так и на уровне локального планировщика и доступа к общей памяти вычислительных элементов.

Ситуация, когда группа задач или потоков выполнения одновременно использует общее сетевое хранилище или общую память, является достаточно распространенной в области обработки спутниковых данных и моделирования окружающей среды. В частности, разовый запуск региональной метеорологической модели WRF для территории Украины требует более 10 Гбайт входных данных.

Как правило, в современных сетевых системах хранения данных в качестве носителей информации используются жесткие диски, одним из основных свойств которых является нелинейное снижение производительности при увеличении числа параллельных запросов. Например, один жесткий диск позволяет выполнять однопоточное линейное чтение со скоростью

около 80 МБ/с, однако при считывании двух линейных потоков скорость каждого из них будет не более 30 МБ/с. С точки зрения суммарного времени выполнения задач предпочтительней последовательное считывание данных. Поэтому актуальна задача оптимизации доступа к общим ресурсам хранилища, в том числе путем блокировки параллельных процессов чтения/записи.

Таким образом, задача состоит в построении Grid-системы, обеспечивающей прозрачную для пользователя обработку его запросов на поиск и обработку данных, в том числе синхронизацию доступа к необходимым данным, обработку этих данных и предоставление пользователю результатов обработки. Такая Grid-система относится к Grid-системам смешанного типа (одновременно вычислительная и информационная [1]), поскольку содержит высокопроизводительные вычислительные узлы и обеспечивает доступ к данным распределенных хранилищ [3].

Синхронизация доступа к данным и их корректная обработка предполагает выполнение некоторых базовых условий.

Взаимное исключение (mutex) — синхронизация событий при обработке данных: два или более вычислительных узла не могут одновременно иметь доступ к общей области данных (общему хранилищу или общей памяти). Здесь и ниже в качестве вычислительного узла будем понимать либо отдельный вычислительный элемент, либо один из процессоров, входящих в его состав.

Равноправие (fairness) — отсутствие дискриминации заданий. Если пользователь сформировал запрос (задание) и отправил его в систему, то обязательно наступит момент, когда это задание начнет выполняться и будет выполнено.

Отсутствие тупиков (deadlock free) — в системе не может возникнуть ситуация взаимной блокировки вычислений или доступа к данным (общим или распределенным).

На рис. 2 показана работа Grid-системы смешанного типа без детализации, где видно, что при отправке запроса пользователя в систему он попадает на управляющий узел (УУ) (планировщик), который выполняет распределение этого задания между ресурсами системы (вычислительными узлами) и определяет местонахождение необходимых данных. На рис. 2 вычислительные узлы и хранилища данных изображены в виде элемента Узел. Этот же УУ обеспечивает синхронизацию при распараллеливании задания между ресурсами или при выполнении операций доступа к общей памяти.

Учитывая высокую стоимость компонентов Grid-систем, прежде чем приступать к построению системы с указанными свойствами, необходимо построить ее модель и исследовать свойства этой модели.

Построим модель функционирования вычислительного узла, работающего над общей памятью, и исследуем ее свойства. Учитывая специфику работы Grid-системы и необходимость синхронизации доступа к общим сегментам данных при организации параллельных вычислений, в качестве математического аппарата для моделирования воспользуемся аппаратом сетей Петри.

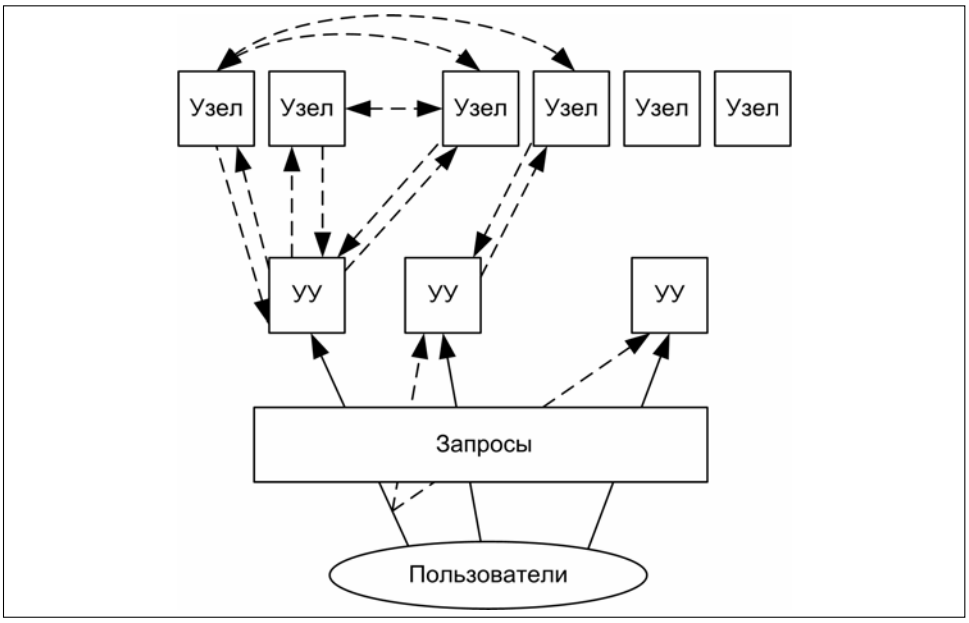

Рис. 2. Модель информационных потоков в Grid-системе

СЕТИ ПЕТРИ. ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Введем основные понятия, связанные с моделированием распределенных систем с разделением ресурсов с помощью сетей Петри.

Определение 1. Сетью Петри (согласно [4]) называется четверка элементов

$$C = (P, T, I, O), \tag{1}$$

где

$$P = \{p_1, p_2, \ldots, p_n\}, \ n > 0 \tag{2}$$

— конечное множество позиций;

$$T = \{t_1, t_2, \ldots, t_m\}, \ m > 0 \tag{3}$$

— конечное множество переходов;

$$I : T > P \tag{4}$$

— функция входов (отображение множества переходов во входные позиции);

$$O : T > P \tag{5}$$

— функция выходов (отображение множества переходов в выходные позиции).

Если $p_i = I(t_j)$, то p_i — входная позиция j-го перехода. Если $p_i = O(t_j)$, то p_i — выходная позиция j-го перехода.

Для наглядного представления сетей Петри используются графы.

Определение 2. Сеть Петри — это двудольный ориентированный мультиграф

$$G = (V, A), \tag{6}$$

где $V = \{u_1, u_2, \ldots, u_{m+n}\}$ — множество вершин; $A = \{a_1, a_2, \ldots, a_r\}$ — комплект направленных дуг. (Комплектом называется множество, содержащее многократно повторяющиеся элементы [4].) Вершинами графа являются непересекающиеся множества позиций P и переходов T сети Петри.

$$V = P \cup T, \text{ причем } P \cap T = \varnothing .$$

Направленные дуги соединяют связанные между собой позиции и переходы.

Определение 2 обеспечивает наглядное графическое представление сети Петри, на основе которого можно сформулировать еще одно определение сети Петри, которое и будет использоваться в дальнейшем при исследовании свойств сети.

Определение 3. Сетью Петри называется тройка элементов

$$C = (P, T, A), \tag{7}$$

где P и T множества позиций и переходов сети Петри соответственно; **A** — матрица инцидентности графа сети, связывающая позиции с переходами сети.

Если элемент $A_{ij} = 0$, значит i-я позиция и j-й переход не связаны между собой. Если $A_{ij} = k > 0$, значит в результате запуска j-го перехода в i-й позиции добавляется k фишек. Если $A_{ij} = k < 0$, значит в результате запуска j-го перехода из i-й позиции убирается k фишек.

Разметкой сети Петри μ называется присвоение фишек позициям сети [4].

Определение 4. Разметка μ сети Петри $C = (P, T, I, O)$ есть функция, отображающая множество позиций на множество натуральных чисел.

$$\mu : P \to N . \tag{8}$$

Разметку можно также определить как вектор

$$\mu = \{\mu_1, \mu_2, \ldots, \mu_n\}, \tag{9}$$

где n — мощность множества позиций, а $\mu_i \in N$, $i = 1, \ldots, n$. Вектор μ определяет для каждой позиции p_i сети количество фишек в этой позиции.

На графе разметка отображается соответствующим числом точек в каждой позиции. Точки называются маркерами или фишками. Если фишек много (больше трех), то их количество отображается числом.

МОДЕЛЬ РАБОТЫ УЗЛА В ВИДЕ СЕТИ ПЕТРИ

Модель работы вычислительного узла с четырьмя процессорами, обрабатывающего задания пользователя под управлением планировщика (глобального или локального уровня) в соответствии с общей схемой (рис. 2) показана на рис. 3. Эта модель представляет собой одноцветную сеть Петри вида (7), начальное состояние которой описывается разметкой

$$M_0 = (\text{proc, want, query, block, free, busy, wait, work}) = (4, 0, 0, 1, 1, 0, 0, 0) .$$

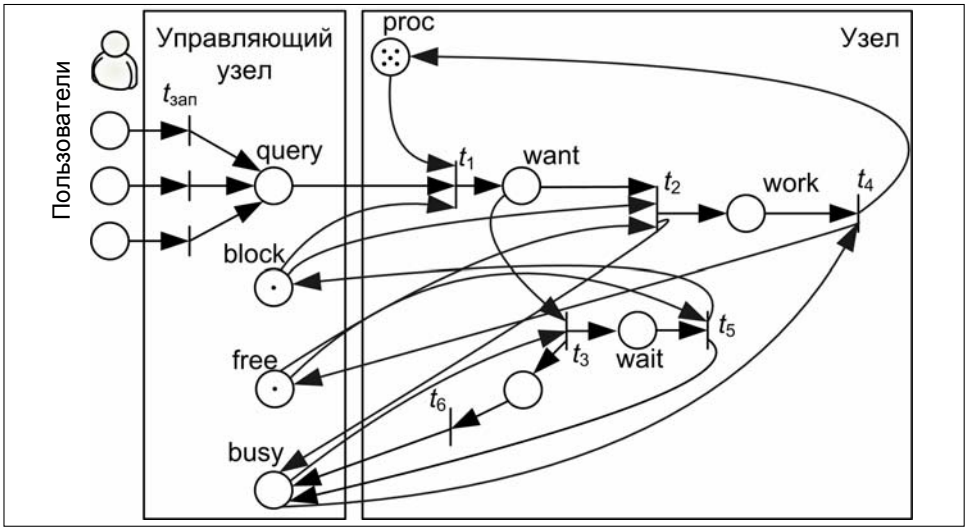

Рис. 3. Модель вычислительного узла, работающего под управлением планировщика над общей памятью с глубиной очереди 1

Позиции и переходы в данной сети Петри имеют следующую семантику:

`proc` — свободные в данный момент времени процессоры;

`query` — запрос на выполнение задания от пользователя;

`want` — процессор активен и желает начать работу;

`work` — процессор получил доступ к общей памяти и выполняет вычисления;

`wait` — активный процессор ожидает доступа к общей памяти;

`block` — блокируется доступ к общей памяти и очереди ожидания, если они заняты другими процессорами;

`free` — управляющий узел сигнализирует о снятии блокировки общей памяти;

`busy` — управляющий узел сигнализирует о том, что общая память заблокирована для доступа.

Показанная на рис. 3 модель достаточно точно описывает процесс выполнения задач с большими потребностями в данных в Grid-окружении. При этом элементам модели соответствуют такие компоненты реальной системы:

1. Позиция `query` является представлением очереди задач метапланировщика (или планировщика) Grid-системы, а метки в ней — представлением задач в очереди (глобальной или локальной).

2. Метки в позиции `proc` соответствуют свободным вычислительным узлам Grid-системы (вычислительным элементам или процессорам).

При начальной разметке $M_0 = (4,0,0,1,1,0,0,0)$ система находится в состоянии покоя (который иногда называют состоянием сна). Из семантики позиций следует и смысл переходов:

$t_{зап}$ — наличие запроса на обработку данных;

t_1 — один из свободных процессоров хочет работать;

t_2 — процессор получает доступ к общей памяти;

t_3 — активный процессор переходит в состояние ожидания;

t_4 — процессор завершил работу с общей памятью и она разблокирована;

t_5 — активный процессор из состояния ожидания переходит в состояние работы с общей памятью, при этом одновременно разблокируется переход активного процессора в состояние ожидания.

Переход t_6 и позиция без наименования между переходами t_3 и t_6 введены в результате модификации структуры сети Петри и преобразования полуцикла в цикл. Это сделано во избежание неоднозначностей в процессе последующего анализа сети.

При появлении запроса пользователя на выполнение задания срабатывает сначала один из переходов $t_{\text{зап}}$, а затем переход t_1. В результате этого один из свободных процессоров переходит в состояние активности (готовности к работе), т.е. переходит в позицию `want`. Если одновременно поступают два запроса, то выполнение одного из них будет задержано из-за отсутствия маркера в позиции `block`. Дальнейшая работа модели очевидна.

ИССЛЕДОВАНИЕ СТРУКТУРНЫХ СВОЙСТВ МОДЕЛИ

Рассмотрим определения некоторых основных свойств сетей Петри.

Определение 5. Переход t *живой* (активный), если он срабатывает при запуске сети из любой начальной разметки.

$$\forall\, \mu \in R(N): \mu \to t\ .$$

Сеть Петри жива, если живым является каждый ее переход.

$$\forall t \forall\, \mu\ \ (t \in T \,\&\, \mu \in R(N)):\ \mu(p) \to t\ .$$

Определение 6. Если существует разметка (маркировка) сети, из которой переход t_i никогда не запускается,то он называется мертвым.

Определение 7. Для сети $C = (P, T, I, O)$ разметка μ' называется непосредственно достижимой из μ, если существует такой переход $t_j \in T$, при котором результатом запуска перехода t_j является разметка μ'.

$$\delta(\mu, t_j) = \mu'\ .$$

На основе определения 7 можно сформулировать определения достижимой и недостижимой позиций.

Определение 8. Позиция p_i называется достижимой, если для любой начальной разметки существует последовательность переходов, приводящая к $\mu(p_i)>0$.

Определение 9. Если $\exists\, p_i$ такая, что

$$\forall t \forall\, \mu\ \ (t \in T \,\&\, \mu \in R(N)):\ \mu(p_i) = 0\,,$$

то позиция p_i называется недостижимой.

Сначала исследуем структурные свойства построенной на рис. 1 сети Петри: проверим выполнение свойства ограниченности, удостоверимся в отсутствии мертвых (лишних) переходов и недостижимых позиций. Для подтверждения достижимости всех позиций сети и доказательства живости сети необходимо построить S- и T-инварианты сети.

S-инвариант — это линейное отношение на разметке подмножества позиций, выражающееся в том, что сумма различных меток в позициях положительна. Если в S-инвариант входят все позиции сети Петри, то в построенной модели все позиции являются достижимыми. T-инвариант соответствует последовательности срабатывания переходов, переводящей сеть из разметки M в ту же самую разметку M [14]. Если T-инвариант содержит все переходы сети, то она жива.

Для нахождения S- и T-инвариантов сети построим целочисленную $n \times m$ матрицу инцидентности сети Петри $\mathbf{A}$ (табл. 1), где n и m — мощности множеств P и T соответственно. В матрице $\mathbf{A}$ элемент $A_{ij} = 1$, если при выполнении перехода j фишка добавляется в позицию i. $A_{ij} = -1$, если при выполнении перехода j фишка убирается из позиции i. $A_{ij} = 0$, если при выполнении перехода j число фишек в позиции i не изменяется.

Таблица 1. Построение матрицы инцидентности сети Петри (рис. 2)

Состояние сети Петри	t_0	t_1	t_2	t_3	t_4	t_5	t_6
Query	1	–1	0	0	0	0	0
Proc	0	–1	0	0	1	0	0
Block	0	–1	1	0	0	1	0
Want	0	1	–1	–1	0	0	0
Wait	0	0	0	1	0	–1	0
Work	0	0	1	0	–1	1	0
Free	0	0	–1	0	1	–1	0
Busy	0	0	1	–1	–1	1	1
Users	–1	0	0	0	1	0	0
Dop	0	0	0	1	0	0	–1

Утверждение. Показанная на рис. 2 сеть Петри, описывающая модель функционирования вычислительного узла Grid-системы, является ограниченной, живой и не содержит недостижимых позиций.

Доказательство.

Воспользуемся уравнением состояний

$$\mathbf{A}\,x = 0, \tag{10}$$

где $\mathbf{A}$ — целочисленная $n \times m$ матрица инцидентности сети Петри; n и m — мощности множеств P и T соответственно; $\mathbf{A}\,x$ — m-мерный вектор Париха [15]**.** С помощью уравнения состояния (10) определим S- и T-инварианты сети Петри. Это и даст возможность выявить мертвые переходы, недостижимые позиции и проверить ограниченность сети.

Размерность матрицы инцидентности **A** в (10) для данной сети Петри составляет 10×7 (10 уравнений, 7 неизвестных), а вектора Париха x — 7. В соответствии с табл. 1 матрица инцидентности для данной сети Петри будет иметь вид

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 & -1 & 1 & 1 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}.$$

Для получения множества S- и T-инвариантов сети Петри используется *TSS*-алгоритм [16, 17], который позволяет построить минимальную порождающую систему решений однородной системы линейных диофантовых уравнений над множеством натуральных чисел N.

Согласно [16, 17] *TSS*-алгоритм позволяет сгенерировать множество S- или T-инвариантов. Входными данными алгоритма являются матрица системы S размерности $p \times q$ (p строк и q столбцов) с начальным множеством решений M. Работа *TSS*-алгоритма на языке псевдокода описывается следующим образом.

$TSS(M, p, q, S)$
begin
  if $M = \varnothing$ then $M = \{e \mid e$ is canonical vector of $N^q\}$;
  for $i := 1$ to p do
    $M := TSS1(M, L_i(x))$;
  if $M = \varnothing$ then (print (“NO SOLUTION”); STOP)
  else CLEAN (M)
  print(M)
end

$TSS1(M, L(x))$
begin
  $M^0 = \varnothing$; $M^+ := \varnothing$; $M^- := \varnothing$;
  forall $e \in M$ do
    (if $SCP(e, L(x)) = 0$ then $M^0 := M^0 \cup \{e\}$ else
    if $SCP(e, L(x)) > 0$ then $M^+ := M^+ \cup \{e\}$ else $M^- := M^- \cup \{e\}$);

$M' := M^0$;

if $M^+ \neq \varnothing \wedge M^- \neq \varnothing$ then

(forall $u \in M^+$ do

(forall $v \in M^-$ do

$e := -L(v)u + L(u)v$; $M' := M' \cup \{e\}$))

return (M')

end

В приведенном фрагменте псевдокода функция $L_i(x) = a_{i1}x_1 + + a_{i2}x_2 + \ldots + a_{iq}x_q$ представляется вектором $(a_{i1}, a_{i2}, \ldots, a_{iq})$, $SCP(x, y)$ означает скалярное произведение векторов x и y, а процедура чистки CLEAN (M) удаляет лишние решения.

В соответствии с *TSS*-алгоритмом система диофантовых уравнений (10) имеет два решения.

$$x_1 = \{1,1,1,0,1,0,0\}^T,$$

$$x_2 = \{1,1,0,1,1,1,1\}^T.$$

На основе решений этой системы можно построить T-инварианты (табл. 2).

Таблица 2. *T*-инварианты сети Петри (рис. 2)

t_0	t_1	t_2	t_3	t_4	t_5	t_6
1	1	1	0	1	0	0
1	1	0	1	1	1	1

Поскольку в табл. 2 нет ни одного столбца, содержащего только нулевые элементы, значит все переходы в сети Петри являются живыми при данной начальной разметке, т.е. каждый переход в сети срабатывает хотя бы один раз. Несложно удостовериться, что это свойство выполняется и для других возможных начальных разметок (варьироваться может число фишек во входных позициях сети), следствием чего является *отсутствие тупиков*, когда в системе не может возникнуть ситуация взаимной блокировки вычислений или доступа к данным (общим или распределенным). Действительно, тупик возникает в сети Петри, если нельзя запустить один или несколько переходов. Так как все переходы живые, то сеть Петри обладает третьим требуемым свойством — характеризуется отсутствием тупиков.

Сгенерируем S-инварианты данной сети Петри. Для этого найдем решения системы уравнений

$$\mathbf{A}^m y = 0, \tag{11}$$

где $\mathbf{A}$ — целочисленная $n \times m$ матрица инцидентности сети Петри; m — символ транспонирования матрицы; y — n-мерный вектор.

Система диофантовых уравнений (11), содержащая 7 уравнений и 10 неизвестных, имеет следующие решения:

$$y_1 = \{0,0,0,0,0,1,1,0,0,0\}^T,$$

$$y_2 = \{0,0,1,1,1,0,0,0,0,0\}^T,$$

$$y_3 = \{0,1,0,1,1,1,0,0,0,0\}^T,$$

$$y_4 = \{1,0,0,1,1,1,0,0,1,0\}^T,$$

$$y_5 = \{0,0,0,0,0,0,1,1,0,1\}^T,$$

$$y_6 = \{0,1,0,1,1,0,0,1,0,1\}^T,$$

$$y_7 = \{1,0,0,1,1,1,0,0,1,0\}^T.$$

Решения матрично-векторного уравнения (11) определяют S-инварианты сети Петри (табл. 3).

Таблица 3. *S*-инварианты сети Петри (рис. 2)

query	proc	block	want	wait	work	free	busy	users	dop
0	0	0	0	0	1	1	0	0	0
0	0	1	1	1	0	0	0	0	0
0	1	0	1	1	1	0	0	0	0
1	0	0	1	1	1	0	0	1	0
0	0	0	0	0	0	1	1	0	1
0	1	0	1	1	0	0	1	0	1
1	0	0	1	1	1	0	0	1	0

Из табл. 3 следует, что сеть Петри является *ограниченной*, поскольку все позиции в ней соответствуют положительным инвариантам. Это означает, что ни в одной позиции сети не может скапливаться бесконечное число фишек. Из табл. 3 также следует, что все позиции в сети достижимы, т.е. модель не содержит лишних позиций.

Таким образом, модель вычислительного узла в виде сети Петри живая, ограниченная и все позиции в сети достижимы.

Утверждение доказано.

ИССЛЕДОВАНИЕ ВЫПОЛНИМОСТИ СВОЙСТВ ВЗАИМНОГО ИСКЛЮЧЕНИЯ И РАВНОПРАВИЯ

При доказательстве утверждения было показано, что модель вычислительного узла Grid-системы в виде сети Петри обладает основными структурными свойствами подобных моделей: ограниченностью, достижимостью и активностью. При этом сеть характеризуется *отсутствием тупиков* (deadlock free), т.е. выполняется третье из базовых свойств моделей синхронизации доступа к данным.

Для исследования выполнимости свойств взаимного исключения (`mutex`) и равноправия (`fairness`) необходимо построить *транзиционную систему* [18] или *граф достижимых разметок* сети Петри.

Для начальной разметки, определяемой соотношением $M_0 = (\text{proc}, \text{want}, \text{query}, \text{block}, \text{free}, \text{busy}, \text{wait}, \text{work}) = (4,0,3,1,1,0,0,0)$, транзиционная система сети Петри (рис. 3) показана на рис. 4.

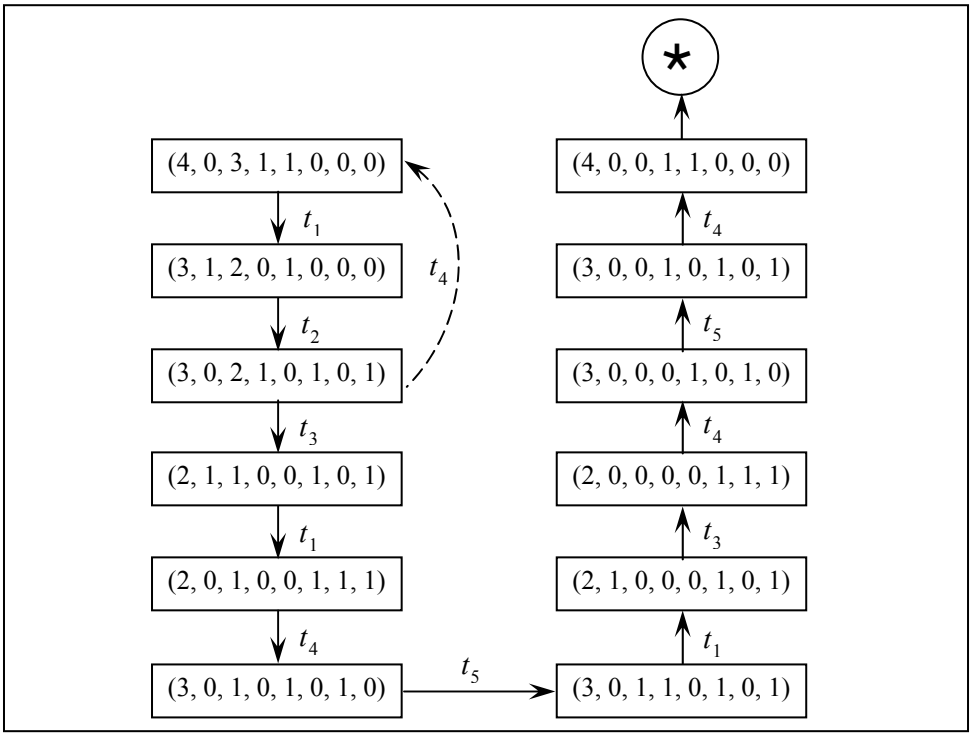

Рис. 4. Транзиционная система сети Петри, моделирующей работу узла Grid-системы

Простой анализ полученной транзиционной системы показывает, что свойство взаимного исключения (`mutex`) выполняется, поскольку в позиции `work` во время функционирования сети Петри появляется не более одной фишки, что справедливо и для позиции `wait`. Значит, в режиме активного ожидания или работы с общей памятью в каждый момент находится не более одного процессора. Число фишек в этих позициях в транзиционной системе на рис. 3 задается двумя последними элементами векторов разметки.

Свойство равноправия (`fairness`) тоже, очевидно, выполняется, поскольку при появлении активного процессора (1 в позиции `want`) в разметке μ из нее достижима разметка μ`, при которой в позиции `want` находится 0 фишек. А это означает, что процессор рано или поздно получит доступ к соответствующему ресурсу (памяти, данным и т.п.).

ВЫВОДЫ

Представлена имитационная модель функционирования узла Grid-системы на основе сетей Петри. Выбор математического аппарата для моделирования объясняется необходимостью обеспечить синхронизацию доступа к общим ресурсам (данным) Grid-системы при выполнении параллельных вычислений.

Анализ предлагаемой модели выполняется с использованием матриц инцидентности и *TSS*-алгоритма решения систем линейных диофантовых уравнений над множеством натуральных чисел. Построенная модель обладает необходимыми свойствами сетей Петри — является живой, ограниченной и все ее позиции достижимы.

Доказано выполнение базовых свойств, необходимых для обеспечения синхронизации доступа к данным и их корректной обработки в узлах Grid-системы.

Дальнейшие исследования будут направлены на построение модели узла Grid-системы с учетом очереди возможных запросов, а также на моделирование работы Grid-системы в целом.

Работа выполнена в рамках темы НАНУ «Интеллект» при поддержке гранта INTAS-CNES-NSAU «Data Fusion Grid Infrastructure» (Ref. Nr 06-1000024-9154).

ЛИТЕРАТУРА

1. *Krauter K., Buyya R., Maheswaran M.* A Taxonomy and Survey of GRID Resource Management Systems and Distributed Computing // Software-Practice and Experience. — 2002. — № 32 — P. 135–164.
2. *Менаске Д., Алмейда В.* Производительность Web-служб. Анализ, оценка и планирование. — СПб. — ДиаСофт, 2003. — 480 с.
3. *Куссуль Н.Н., Шелестов А.Ю., Лобунец А.Г.* Применение методов операционного анализа для оценки производительности GRID-систем// Кибернетика и вычислительная техника. — 2004. — Вып. 144. — С. 3–20.
4. *Питерсон Дж.* Теория сетей Петри и моделирование систем. — М.: Мир, 1984. — 264 с.
5. *Dissemination* and Exploitation of Grids in Earth science. — http://www.eu-degree.eu/.
6. *EGEE* gLite middleware. — http://glite.web.cern.ch/.
7. *GridWay* Metascheduler. — http://www.gridway.org/.
8. *Коваленко В.Н., Коваленко Е.И., Шорин О.Н.* Разработка диспетчера заданий грид, основанного на опережающем планировании. — М.: ИПМ РАН. — 2005. — 28 с.
9. *Globus Toolkit*. — http://globus.org/.
10. *TORQUE* resource manager. — http://www.clusterresources.com/pages/products/torque-resource-manager.php.
11. *PBS* Professional. — http://www.pbsgridworks.com/PBSTemp1.3.aspx?top_nav_name=Products&item_name=PBS%20Professional&top_nav_str=1.
12. *Sun* Grid Engine. — http://www.sun.com/software/gridware/.
13. *Platform* LSF. — http://www.platform.com/Products/platform-lsf-family/platform-lsf.
14. *Дубинин В.Н., Зинкин С.А.* Языки логического программирования в проектировании вычислительных систем и сетей: Учеб. пособие. — Пенза: Изд-во Пенз. гос. техн. ун-та, 1997. — 100 с.
15. *Parikh R.* On Context-Free Languages // J. of the ACM. — 1966. — **13**, № 4. — P. 570–581.
16. *Кривой С.Л.* Критерий совместности систем линейных диофантовых уравнений над множеством натуральных чисел// Доп. НАНУ. — 1999. — № 5. — С. 107–112.
17. *Krivoi S.* A criteria of Compatibility Systems of Linear Diophantine Constraints // Lecture Notes in Comp. Science. — 2002. — № 2328. — P. 264–271.
18. *Летичевський О.А.* Сучасні проблеми кібернетики. Нормативний курс. Навчальна електронна бібліотека факультету кібернетики Київського національного університету ім. Тараса Шевченка. — http://www.unicyb.kiev.ua/Library/.

Поступила 12.10.2007