ИНФОРМАТИКА, ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И УПРАВЛЕНИЕ

COMPUTER SCIENCE, COMPUTER ENGINEERING AND CONTROL

УДК 004.9
doi:10.21685/2072-3059-2022-2-1

Алгоритмика, логика и моделирование агентно-базированных метакомпьютерных систем с повышенным уровнем параллельности

В. И. Волчихин[1], Н. С. Карамышева[2], С. А. Зинкин[3], Е. И. Гурин[4]

[1,2,3,4]Пензенский государственный университет, Пенза, Россия

[1]cnit@pnzgu.ru, [2]vt@pnzgu.ru, [3]zsa49@yandex.ru, [4]gurin2@yandex.ru

Аннотация. *Актуальность и цели*. Предложено проводить разработку агентно-базированных сетевых метакомпьютерных систем и приложений на основе логических методов и связанных с ними концептуальных графических моделей, что позволяет сочетать императивные и декларативные методы при проектировании функциональной архитектуры и программного обеспечения метакомпьютера. Предложены формализованные спецификации для создания агентно-базированных сетевых приложений на основе концептуальных и логических моделей искусственного интеллекта. Для обозначения сетевой среды, в которой разворачивается сценарий действий, выбран термин «метакомпьютер». Другое название – облачно-сетевое приложение – в принципе означает то же самое, но отличается дополнительным учетом терминологии из области современных сетевых технологий в явной форме. В связи с ростом значения глобальных вычислительных сетей в науке и образовании проблема создания крупномасштабных приложений является актуальной. Предложена функциональная организация метакомпьютерных агентно-базированных сетевых распределенных вычислений, реализующих основные конструкции распределенного программирования, где сеть рассматривается реально как компьютер с распределенным программным управлением на основе парадигмы передачи сообщений (*message-driven computing*), а не как средство реализации простейших клиент-серверных или *master-slave* (мастер-слуга) приложений. Целью работы является повышение уровня параллельности при обработке данных в метакомпьютерных системах за счет организации конвейерного перемещения сообщений по сети. *Материалы и методы*. В качестве основных методов применяются концептуальные модели, логико-алгебраические операционные модели, логические сети Петри. *Результаты*. Предложены концептуальные графы распределенных алгоритмов и логико-алгебраические операционные выражения, пригодные для использования в качестве непосредственно исполнимых спецификаций, разработан метод перехода от концептуальных графов к исполнимым спецификациям, определяющим функциональную архитектуру метаком-

пьютера. Разработаны имитационные модели для распределенных алгоритмов. *Выводы*. Практическая реализация изложенных концепций и моделей позволит повысить уровень параллельности при работе агентно-базированных виртуальных метакомпьютерных вычислительных систем за счет конвейерной организации прохождения сообщений.

Ключевые слова: метакомпьютеры, облачно-сетевые приложения, виртуальные архитектуры распределенных вычислительных систем, формальные модели, логико-алгебраические операционные выражения, формализованные спецификации, программные агенты, логические сетевые модели

Для цитирования: Волчихин В. И., Карамышева Н. С., Зинкин С. А., Гурин Е. И. Алгоритмика, логика и моделирование агентно-базированных метакомпьютерных систем с повышенным уровнем параллельности // Известия высших учебных заведений. Поволжский регион. Технические науки. 2022. № 2. С. 5–25. doi:10.21685/2072-3059-2022-2-1

Algorithms, logic and modeling of agent-based metacomputer systems with a high level of parallelism

V.I. Volchikhin[1], N.S. Karamysheva[2], S.A. Zinkin[3], E.I. Gurin[4]

[1,2,3,4]Penza State University, Penza, Russia

[1]cnit@pnzgu.ru, [2]vt@pnzgu.ru, [3]zsa49@yandex.ru, [4]gurin2@yandex.ru

Abstract. *Background.* It is proposed to develop agent-based network metacomputer systems and applications based on logical methods and related conceptual graphical models, which allows combining imperative and declarative methods when designing the functional architecture and software of a metacomputer. Formalized specifications for creating agent-based network applications based on conceptual and logical models of artificial intelligence are proposed. The term “metacomputer” is chosen to denote the network environment in which the action script is deployed. Another name is a cloud-network application, in principle it means the same thing, but it differs in the additional consideration of terminology from the field of modern network technologies in an explicit form. In connection with the growing importance of global computer networks in science and education, the problem of creating large-scale applications is relevant. A functional organization of metacomputer agent-based network distributed computing is proposed, which implements the main structures of distributed programming, where the network is actually considered as a computer with distributed program control based on the message-driven computing paradigm, and not as a means of implementing the simple client-server or master-slave applications. The aim of the work is to increase the level of parallelism in data processing in metacomputer systems by organizing the pipeline movement of messages over the network. *Materials and methods.* Conceptual models, logical-algebraic operating models, logical Petri nets are used as the main methods. *Results.* Conceptual graphs of distributed algorithms and logical-algebraic operational expressions suitable for use as directly executable specifications are proposed, a method is developed for moving from conceptual graphs to executable specifications that define the functional architecture of a metacomputer. Simulation models for distributed algorithms have been developed. *Conclusions*. The practical implementation of the above concepts and models will increase the level of parallelism in the operation of agent-based virtual metacomputer systems due to the pipeline organization of message passing.

Keywords: metacomputers, cloud network applications, virtual architectures of distributed computing systems, formal models, logical-algebraic operational expressions, formalized specifications, software agents, logical network models

For citation: Volchikhin V.I., Karamysheva N.S., Zinkin S.A., Gurin E.I. Algorithms, logic and modeling of agent-based metacomputer systems with a high level of parallelism. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences.* 2022;(2):5–25. (In Russ.). doi: 10.21685/2072-3059-2022-2-1

Введение

В последнее время появился ряд технологий, облегчающих создание мультиагентных систем, но проблема разработки языковых средств высокого уровня остается актуальной. В работах [1–4] ранее была предложена организация метакомпьютерных агентно-базированных сетевых распределенных вычислений, реализующих основные конструкции распределенного программирования, где сеть рассматривается реально как компьютер с распределенным программным управлением на основе парадигмы передачи сообщений (*message-driven computing*), а не как средство реализации простейших клиент-серверных или *master-slave* (мастер-слуга) приложений. Модули, или агенты, распределенного приложения способны работать как в реактивном режиме, ожидая прием данных и передачу управления, так и в проактивном режиме, запрашивая данные и управление от предшествующих модулей (агентов).

Традиционные коммуникации в архитектуре клиент-сервер обладали рядом недостатков, для устранения которых потребовалось найти альтернативное решение, такое как *peer-to-peer* (P2P) – пиринговые, или бессерверные, приложения, привлекшие большое внимание разработчиков и пользователей. Этому во многом способствовала успешная реализация таких приложений, как Napster, Gnutella, FastTrack, BitTorrent, Skype, Chord, Pastry и др. [5, 6].

Для доставки контента через Интернет была предложена также гибридная архитектура P2PWeb, при создании которой был найден компромисс между централизованной реализацией взаимодействия клиента с сервером и нецентрализованной архитектурой для хранения и доставки контента [7].

В работе [8] представлена новая модель и реализация виртуальных машин с массовым параллелизмом, предназначенная для параллельных и высокопроизводительных вычислений в распределенных системах. Эта модель построена с использованием динамически распределенных виртуальных процессоров, каждый из которых реализован при помощи мобильного агента, развернутого в физическом процессоре. Распределенная система в целом предназначена для реализации вычислений как в режиме SPMD (англ. *Single Program Multiple Data* – одна программа, множественные данные, т.е. одна и та же программа загружается на несколько компьютеров для решения одной и той же задачи на различных наборах данных), так и в режиме MPMD (англ. *Multiple Programs*, *Multiple Data* – множество программ, множество данных).

В книге [9] рассмотрена ситуация, когда вычислительная модель MPMD представляется в форме модели MTMD (*Multiple Threads*, *Multiple Data* – множество потоков выполнения, множество потоков данных), описывающей систему с различными потоками выполнения (*threads*), которые обрабатывают различные наборы данных, или потоки данных (*streams*). Вычислительная модель SPMD представлена в форме модели STMD (*Single Thread*, *MultipleData* – один поток выполнения, несколько потоков данных), что означает обработку различных наборов данных несколькими идентичными потоками выполнения, которые вызывают одну и ту же подпрограмму.

В настоящей статье рассмотрены вопросы создания агентно-базированных сетевых приложений на основе концептуальных и логических моделей искусственного интеллекта. Для обозначения сетевой среды, в которой разворачивается сценарий действий, выбран известный термин «метакомпьютер», который, на взгляд авторов, более соответствует назначению и реализации создаваемого приложения и виртуальной среды. Другой вариант названия – облачно-сетевое приложение, приблизительно может именовать тот же самый объект – распределенное приложение, развернутое в сетевой среде, в терминах современных сетевых технологий в явной форме.

1. Инфраструктура для развертывания метакомпьютерных приложений

Инфраструктура, представленная на рис. 1, состоит из связанных системой коммуникаций серверов (*servers*) и пользовательских компьютеров (*users*). Система коммуникаций содержит каналы связи, коммутаторы локальных сетей (*switches*) и маршрутизаторы глобальной сети (*routers*). Это типовая инфраструктура глобальной компьютерной TCP/IP сети [10], используемой для развертывания метакомпьютерных приложений. Обработка данных в сети осуществляется программными компонентами, в качестве которых выбраны мобильные или стационарные агенты, размещенные на узлах сети. Кружками обозначены вводимые и выводимые данные, т.е. исходные аргументы и результаты их обработки.

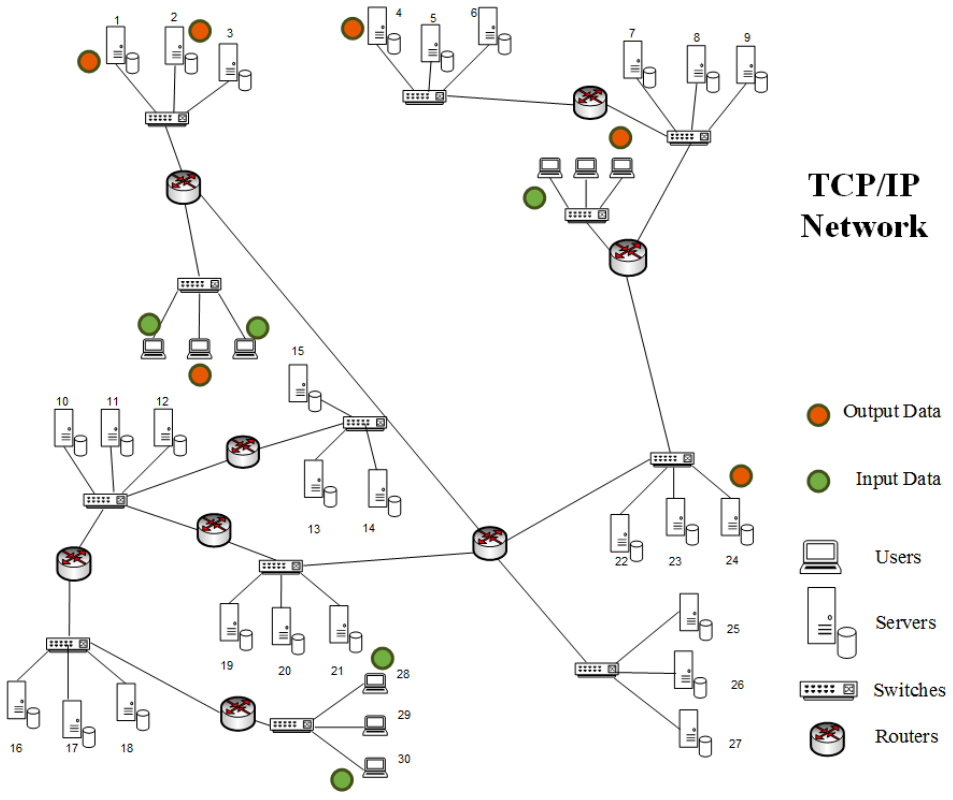

Рис. 1. Типовая инфраструктура глобальной компьютерной сети, используемой для развертывания метакомпьютерных приложений

Схема метакомпьютерного приложения соответствует сценарию распределенной обработки данных в среде вычислительной сети. Этот сценарий соответствуют обработке данных в распределенной среде типа «грид» с возможной организацией в виде частного облачного сервиса. В настоящей работе предлагается технология создания распределенных приложений на основе использования некоторых моделей искусственного интеллекта и, в частности, на основе описания компонент моделей в виде концептуальных графов. Использование концептуальных моделей позволяет визуализировать процесс создания проектов распределенных систем.

2. Концептуальная модель агентно-базированной реализации распределенного метакомпьютерного приложения

Пример концептуального графа распределенного приложения A, описывающего декларативные и процедурные знания о распределенном приложении, выполняемом в сетевой среде, представлен на рис. 2. Этот граф определяет ***функциональную*** архитектуру метакомпьютера. Формульная запись распределенного приложения A в виде сценария имеет следующий вид:

$$A = A_0/Y_0,\ A_1/Y_0,\ A_2/Y_0\ ||F(A_3/Y_1;\ A_4/Y_2;\ A_5/Y_3),\ J(A_6/Y_3)||,\ A_7/Y_4,\ A_8/Y_4,$$
$$S[A_9|k/Y_4]((k = 1)\colon A_{10}/Y_5 \oplus (k = 2)\colon$$
$$A_{11}/Y_6 \oplus (k = 3)\colon A_{12}/Y_7)\ Or(A_{13}/Y_8),\ A_{14}/Y_8, \quad (1)$$

где запись A_i/Y_j означает, что агент A_i расположен на узле Y_j, а запись $A_i/Y_j|k$ означает получение значения переменной k в результате выполнения агента A_i, расположенного на узле Y_j. Выражение (1) содержит оператор ***fork-join*** (сокр. в формулах – $||(F\ldots J)||$), реализующий в приложении параллельную передачу и обработку сообщений по ветвям распределенного алгоритма. Формальная запись данного оператора имеет следующий вид:

$$(A_{begin}/Y_{begin})\ ||F(A_i/Y_k;\ A_j/Y_m;\ \ldots;\ A_k/Y_n),\ J(A_{end}/Y_{end})||, \quad (2)$$

Оператор ***switch-or*** (сокр. $S\ldots Or$), также включенный в выражение (1), реализует обработку и передачу сообщения по одной из ветвей распределенного алгоритма в соответствии со значением переменной k и формально описывается следующим выражением:

$$S[(A_i|k)/Y_s]((k = 1)\colon A_q/Y_p \oplus (k = 2)\colon A_j/Y_m \oplus \ldots \oplus (k = n)\colon A_h/Y_r),\ Or(A_{end}/Y_{end}). \quad (3)$$

Символ «⊕» в выражениях (1) и (3) обозначает операцию выбора одной из нескольких альтернатив (все значения переменной k должны быть различны); лишь для случая двух операндов – это символ логической операции «исключающее или».

Абстрактный характер данных выражений и их ориентация на парадигму передачи сообщений позволяет программисту рассматривать их в качестве спецификаций начального уровня для агентно-базированных сетевых приложений.

Графическое представление концептуальных моделей выполнено при помощи известного в международной практике открыто распространяемого редактора CharGer – Conceptual Graph Editor 3.6, URL: https://charger-conceptual-graph-editor.soft112.com.

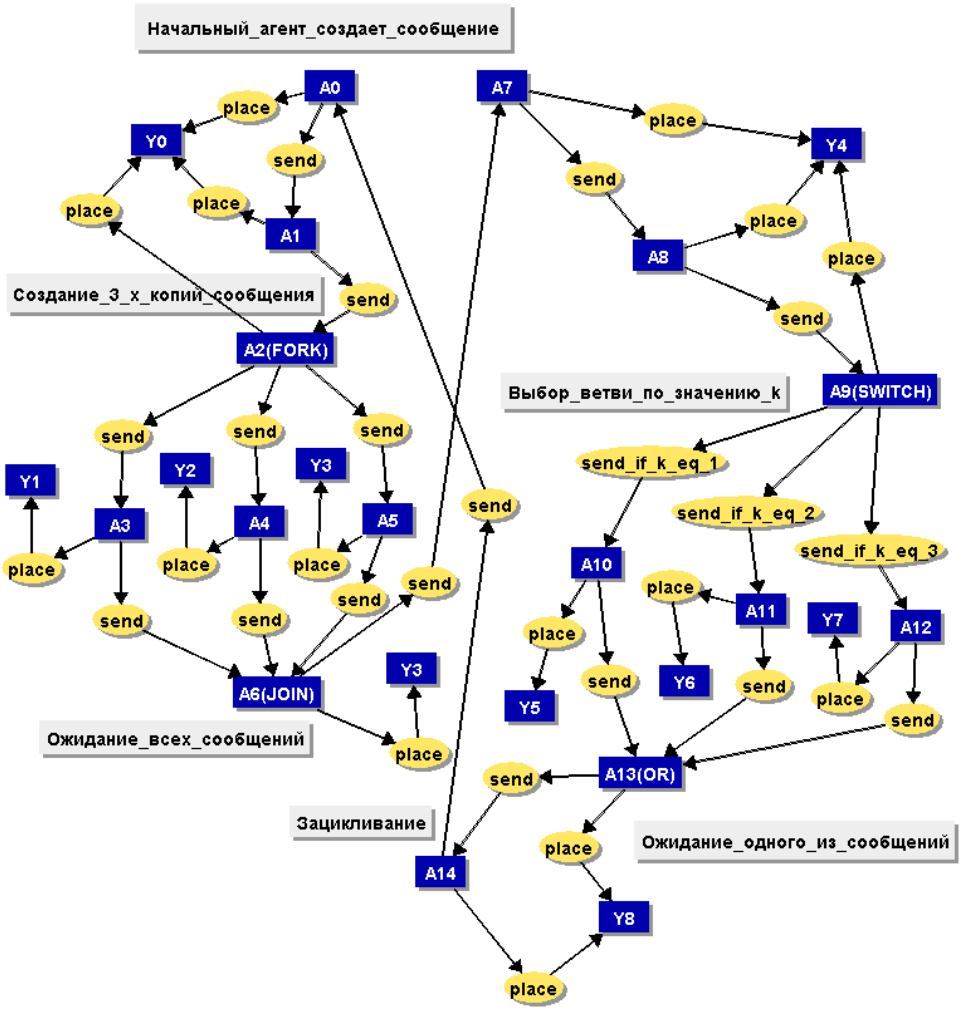

Рис. 2. Концептуальный граф распределенного алгоритма, описывающий процедурные и декларативные знания о распределенном приложении, выполняемом в метакомпьютерной среде

Более детализированные формализованные спецификации основаны на предлагаемой в настоящей работе модели искусственного интеллекта, описываемой далее логико-алгебраическими операционными выражениями.

3. Логико-алгебраические модели представления знаний о функционировании агентно-базированных метакомпьютерных приложений

Знания о работе распределенного приложения возможно представить в виде фактов и композиций правил модификации бинарных предикатов. Приведенный на рис. 2 пример соответствует случаю, когда во фрагменте компьютерной сети, используемом для реализации метакомпьютера, циркулирует только одно сообщение. В модели представления знаний используются следующие понятия: $send(A_i, A_j)$ – «Агент A_i отправляет по сети сообщение агенту A_j», т.е. таким образом представлена передача управления вычисли-

тельным процессом и данных; *place*(A_i, Y_k) – «Агент A_i размещен на узле Y_k»; *link*(Y_k, Y_m) – «Между узлами Y_k и Y_m установлена связь, используемая для обмена сообщениями между агентами». Правила модификации предикатов представлены выражениями вида *send*(A_i, A_j) ← T, где $T \in$ {**true**, **false**}, а также выражениями следующего вида:

[*условие*] ((*блок 1 элементарных правил модификации предикатов*) ⊕
⊕ (*блок 2 элементарных правил модификации предикатов*)),

где «⊕» – символ операции «исключающее или» (*XOR*), или строгой дизъюнкции, обозначающий выбор одной из двух альтернатив по условию. Если условие истинно, то выбирается для выполнения первая альтернатива, а если ложно – то вторая. В формуле (3) данный символ обозначает выбор единственной альтернативы из нескольких блоков правил. Частная форма последнего выражения имеет следующий вид продукционного правила «если *условие*, то *действия*»:

[*условие*] ((*блок элементарных правил модификации предикатов*) ⊕ E),

где E – символ пустой операции.

Блоки содержат элементарные правила модификации предикатов, разделенные символом запятой «,» – последовательное выполнение правил.

4. Система Σ_1 формализованных спецификаций для агентно-базированного метакомпьютерного приложения, в котором циркулирует одно сообщение

Детальные спецификации, описывающие работу распределенного приложения и соответствующие рис. 2, содержат факты и правила. Факты базы знаний о передаче сообщений имеют следующий вид:

$send(A_0, A_1)$	$send(A_3, A_6)$	$send(A_8, A_9)$	$send(A_{11}, A_{13})$
$send(A_1, A_2)$	$send(A_4, A_6)$	$send(A_9, A_{10})$	$send(A_{12}, A_{13})$
$send(A_2, A_3)$	$send(A_5, A_6)$	$send(A_9, A_{11})$	$send(A_{13}, A_{14})$
$send(A_2, A_4)$	$send(A_6, A_7)$	$send(A_9, A_{12})$	$send(A_{14}, A_0)$
$send(A_2, A_5)$	$send(A_7, A_8)$	$send(A_{10}, A_{13})$	

Правила вывода в форме логико-алгебраических операционных выражений соответствуют различным этапам выполнения сценария Σ_1:

– начальная фаза:

$$send(A_0, A_1)\ \textbf{true};$$

$$\left[send(A_0, A_1)\right]\left(send(A_1, A_2)\ \textbf{true},\ send(A_0, A_1)\ \textbf{false}\ E\right); \qquad (4)$$

– выполнение распределенной операции ***fork***:

$$\left[send(A_1, A_2)\right](send(A_2, A_3)\ \textbf{true},\ send(A_2, A_4)\ \textbf{true},$$

$$send(A_2, A_5)\ \textbf{true},\ send(A_1, A_2)\ \textbf{false}\ E);$$

$$\left[send(A_2, A_3)\right]\left(send(A_3, A_6)\ \textbf{true},\ send(A_2, A_3)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_2,A_4\right)\right]\left(send\left(A_4,A_6\right)\ \textbf{true},\ send\left(A_2,A_4\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_2,A_5\right)\right]\left(send\left(A_5,A_6\right)\ \textbf{true},\ send\left(A_2,A_5\right)\ \textbf{false}\ E\right); \quad (5)$$

– выполнение распределенной операции ***join***:

$$\left[send\left(A_3,A_6\right)\ \&send\left(A_4,A_6\right)\ \&send\left(A_5,A_6\right)\right](send\left(A_6,A_7\right)\ \textbf{true},$$

$$send\left(A_3,A_6\right)\ \textbf{false},\ send\left(A_4,A_6\right)\ \textbf{false},\ send\left(A_5,A_6\right)\ \textbf{false}\ E); \quad (6)$$

– выполнение распределенной последовательности операторов:

$$\left[send\left(A_6,A_7\right)\right]\left(send\left(A_7,A_8\right)\ \textbf{true},\ send\left(A_6,A_7\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_7,A_8\right)\right]\left(send\left(A_8,A_9\right)\ \textbf{true},\ send\left(A_7,A_8\right)\ \textbf{false}\ E\right); \quad (7)$$

– выполнение распределенной операции ***switch***:

$$\left[send\left(A_8,A_9\right)\&\left(k=1\right)\right]\left(send\left(A_9,A_{10}\right)\ \textbf{true},\ send\left(A_8,A_9\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_8,A_9\right)\&\left(k=2\right)\right]\left(send\left(A_9,A_{11}\right)\ \textbf{true},\ send\left(A_8,A_9\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_8,A_9\right)\&\left(k=3\right)\right]\left(send\left(A_9,A_{12}\right)\ \textbf{true},\ send\left(A_8,A_9\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_9,A_{10}\right)\right]\left(send\left(A_{10},A_{13}\right)\ \textbf{true},\ send\left(A_9,A_{10}\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_9,A_{11}\right)\right]\left(send\left(A_{11},A_{13}\right)\ \textbf{true},\ send\left(A_9,A_{11}\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_9,A_{12}\right)\right]\left(send\left(A_{12},A_{13}\right)\ \textbf{true},\ send\left(A_9,A_{12}\right)\ \textbf{false}\ E\right); \quad (8)$$

– выполнение распределенной операции ***or***:

$$\left[send\left(A_{10},A_{13}\right)\ \ send\left(A_{11},A_{13}\right)\ \ send\left(A_{12},A_{13}\right)\right](send\left(A_{13},A_{14}\right)\ \textbf{true},$$

$$send\left(A_{10},A_{13}\right)\ \textbf{false},\ send\left(A_{11},A_{13}\right)\ \textbf{false},\ send\left(A_{12},A_{13}\right)\ \textbf{false}\ E); \quad (9)$$

– переход к выполнению начального оператора A_0:

$$\left[send\left(A_{13},A_{14}\right)\right]\left(send\left(A_{14},A_0\right)\ \textbf{true},\ send\left(A_{13},A_{14}\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_{14},A_0\right)\right]\left(send\left(A_0,A_1\right)\ \textbf{true},\ send\left(A_{14},A_0\right)\ \textbf{false}\ E\right). \quad (10)$$

Правила формирования связей для передачи сообщений, т.е. правила формирования концептуального графа виртуальной системной архитектуры агентно-базированного метакомпьютера, имеют следующий вид [4]:

$$\left(aA,bA,yY,zZ\right)\left[send\left(a,b\right)\&\ place\left(a,y\right)\&\ place\left(b,z\right)\ \ link\left(y,z\right)\right]; \quad (11)$$

$$\left(aA,bA,yY,zZ\right)\left[send\left(a,b\right)\&\ place\left(a,y\right)\&\ place\left(b,z\right)\ \ link\left(z,y\right)\right]. \quad (12)$$

При помощи правила (11) формируются связи для передачи сообщений *Contr*, управляющих вычислительным процессом. Правило (12) используется для формирования связей с целью передачи квитирующих сообщений *Ack*.

В работе [4] рассмотрен также случай передачи сообщений одному из двух агентов-операторов по некоторому условию *w*, для чего введены бинарные предикаты *y_send*(*x*, *y*) (передача сообщения от агента *x* к агенту *y* при *w* = **true**) и *n_send*(*x*, *z*) (передача сообщения от агента *x* к агенту *z* при *w* = **false**). В примере на рис. 2 эта конструкция не используется.

Факты базы знаний о размещении агентов-операторов имеют для любых алгоритмов следующий типовой вид:

$place(A_0,Y_0).$	$place(A_4,Y_2).$	$place(A_8,Y_4).$	$place(A_{12},Y_7).$
$place(A_1,Y_0).$	$place(A_5,Y_3)..$	$place(A_9,Y_4).$	$place(A_{13},Y_8).$
$place(A_2,Y_0).$	$place(A_6,Y_3).$	$place(A_{10},Y_5).$	$place(A_{14},Y_8).$
$place(A_3,Y_1).$	$place(A_7,Y_4).$	$place(A_{11},Y_6).$	

Концептуальный граф виртуальной ***системной*** архитектуры метакомпьютера, т.е. граф развертывания мультиагентной системы в компьютерной сети, получен на основании правил вывода (11), (12) и представлен на рис. 3. Приведенные на рис. 2 и 3 концептуальные графы и формулы (1), (4)–(12) представляют декларативные и процедурные знания о распределенном приложении, выполняемом в сетевой среде.

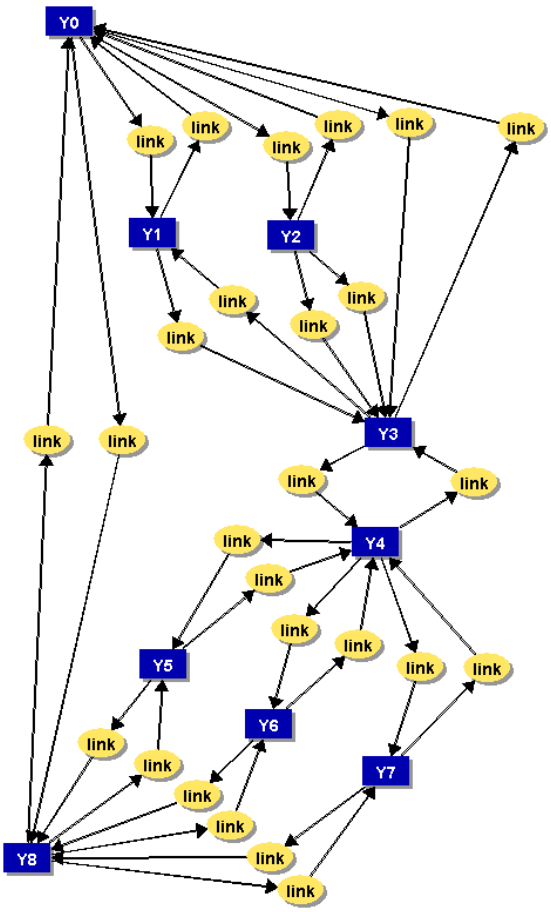

Рис. 3. Концептуальный граф виртуальной системной архитектуры метакомпьютера

5. Система Σ_2 формализованных спецификаций для агентно-базированного метакомпьютерного приложения с конвейерным параллелизмом

Уровень параллельности в агентно-базированных системах определяется не только использованием операции ***fork*** для размножения сообщений, но и числом одновременно запускаемых в систему управляющих сообщений. Новая система Σ_2 правил вывода в форме логико-алгебраических операционных выражений отличается от системы Σ_1 включением правил, используемых для организации очереди сообщений с проверкой готовности агентов-операторов к приему сообщений от предыдущих агентов-операторов:

– запуск сообщений в очередь B:

$$send\left(B_0, B_1\right)\ \textbf{true};$$

$$send\left(B_1, B_2\right)\ \textbf{true};$$

$$send\left(B_2, B_3\right)\ \textbf{true};$$

$$send\left(B_3, B_4\right)\ \textbf{true};$$

$$send\left(B_4, B_5\right)\ \textbf{true}; \qquad (13)$$

– перемещение сообщений по входной очереди B:

$$\left[send\left(B_0, B_1\right) \,\&\, send\left(B_1, B_2\right)\right]$$

$$\left(send\left(B_1, B_2\right)\ \textbf{true},\ send\left(B_0, B_1\right)\ \textbf{false}\ \ E\right);$$

$$\left[send\left(B_1, B_2\right) \,\&\, send\left(B_2, B_3\right)\right]$$

$$\left(send\left(B_2, B_3\right)\ \textbf{true},\ send\left(B_1, B_2\right)\ \textbf{false}\ \ E\right);$$

$$\left[send\left(B_2, B_3\right) \,\&\, send\left(B_3, B_4\right)\right]$$

$$\left(send\left(B_3, B_4\right)\ \textbf{true},\ send\left(B_2, B_3\right)\ \textbf{false}\ \ E\right);$$

$$\left[send\left(B_3, B_4\right) \,\&\, send\left(B_4, B_5\right)\right]$$

$$\left(send\left(B_4, B_5\right)\ \textbf{true},\ send\left(B_3, B_4\right)\ \textbf{false}\ \ E\right);$$

$$\left[send\left(B_4, B_5\right) \,\&\, send\left(A_0, A_1\right)\right]$$

$$\left(send\left(A_0, A_1\right)\ \textbf{true},\ send\left(B_4, B_5\right)\ \textbf{false}\ \ E\right);$$

$$\left[send\left(A_0, A_1\right) \,\&\, send\left(A_1, A_2\right)\right]$$

$$\left(send\left(A_1, A_2\right)\ \textbf{true},\ send\left(A_0, A_1\right)\ \textbf{false}\ \ E\right); \qquad (14)$$

– выполнение распределенной операции ***fork***:

$$\left[send\left(A_1,A_2\right)\&\,send\left(A_2,A_3\right)\&\,send\left(A_2,A_4\right)\&\,send\left(A_2,A_5\right)\right]$$

$$(send\left(A_2,A_3\right)\ \textbf{true},\ send\left(A_2,A_4\right)\ \textbf{true},\ send\left(A_2,A_5\right)\ \textbf{true},$$

$$send\left(A_1,A_2\right)\ \textbf{false}\ E);$$

$$\left[send\left(A_2,A_3\right)\&\,send\left(A_3,A_6\right)\right]$$

$$\left(send\left(A_3,A_6\right)\ \textbf{true},\ send\left(A_2,A_3\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_2,A_4\right)\&\,send\left(A_4,A_6\right)\right]$$

$$\left(send\left(A_4,A_6\right)\ \textbf{true},\ send\left(A_2,A_4\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_2,A_5\right)\&\,send\left(A_5,A_6\right)\right]$$

$$\left(send\left(A_5,A_6\right)\ \textbf{true},\ send\left(A_2,A_5\right)\ \textbf{false}\ E\right);\qquad(15)$$

– выполнение распределенной операции ***join***:

$$\left[send\left(A_3,A_6\right)\&\,send\left(A_4,A_6\right)\&\,send\left(A_5,A_6\right)\&\,send\left(A_6,A_7\right)\right]$$

$$(send\left(A_6,A_7\right)\ \textbf{true},\ send\left(A_3,A_6\right)\ \textbf{false},\ send\left(A_4,A_6\right)\ \textbf{false},$$

$$send\left(A_5,A_6\right)\ \textbf{false}\ E);\qquad(16)$$

– выполнение распределенной последовательности операторов:

$$\left[send\left(A_6,A_7\right)\&\,send\left(A_7,A_8\right)\right]$$

$$\left(send\left(A_7,A_8\right)\ \textbf{true},\ send\left(A_6,A_7\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_7,A_8\right)\&\,send\left(A_8,A_9\right)\right]$$

$$\left(send\left(A_8,A_9\right)\ \textbf{true},\ send\left(A_7,A_8\right)\ \textbf{false}\ E\right);\qquad(17)$$

– выполнение распределенной операции ***switch***:

$$\left[send\left(A_8,A_9\right)\&\left(k=1\right)\&\,send\left(A_9,A_{10}\right)\right]$$

$$\left(send\left(A_9,A_{10}\right)\ \textbf{true},\ send\left(A_8,A_9\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_8,A_9\right)\&\left(k=2\right)\&\,send\left(A_9,A_{11}\right)\right]$$

$$\left(send\left(A_9,A_{11}\right)\ \textbf{true},\ send\left(A_8,A_9\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_8,A_9\right)\&\left(k=3\right)\&\,send\left(A_9,A_{12}\right)\right]$$

$$\left(send\left(A_9, A_{12}\right)\ \textbf{true},\ send\left(A_8, A_9\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_9, A_{10}\right) \& send\left(A_{10}, A_{13}\right)\right]$$

$$\left(send\left(A_{10}, A_{13}\right)\ \textbf{true},\ send\left(A_9, A_{10}\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_9, A_{11}\right) \& send\left(A_{11}, A_{13}\right)\right]$$

$$\left(send\left(A_{11}, A_{13}\right)\ \textbf{true},\ send\left(A_9, A_{11}\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_9, A_{12}\right) \& send\left(A_{12}, A_{13}\right)\right]$$

$$\left(send\left(A_{12}, A_{13}\right)\ \textbf{true},\ send\left(A_9, A_{12}\right)\ \textbf{false}\ E\right); \quad (18)$$

– выполнение распределенной операции ***or***:

$$\left[\left(send\left(A_{10}, A_{13}\right)\ send\left(A_{11}, A_{13}\right)\ send\left(A_{12}, A_{13}\right)\right) \& send\left(A_{13}, A_{14}\right)\right]$$

$$(send\left(A_{13}, A_{14}\right)\ \textbf{true},\ send\left(A_{10}, A_{13}\right)\ \textbf{false},$$

$$send\left(A_{11}, A_{13}\right)\ \textbf{false},\ send\left(A_{12}, A_{13}\right)\ \textbf{false}\ E); \quad (19)$$

– переход к выполнению начального оператора B_0 формирования входной очереди сообщений:

$$\left[send\left(A_{13}, A_{14}\right) \& send\left(A_{14}, B_0\right)\right]$$

$$\left(send\left(A_{14}, B_0\right)\ \textbf{true},\ send\left(A_{13}, A_{14}\right)\ \textbf{false}\ E\right);$$

$$\left[send\left(A_{14}, B_0\right) \& send\left(B_0, B_1\right)\right]$$

$$\left(send\left(B_0, B_1\right)\ \textbf{true},\ send\left(A_{14}, B_0\right)\ \textbf{false}\ E\right). \quad (20)$$

Выражения (13)–(20) системы Σ_2 логико-алгебраических операционных выражений относятся к моделям промежуточного уровня и предназначены для использования в качестве формализованных спецификаций при разработке распределенного приложения. В этих выражениях учтена проверка очередности прохождения сообщений без взаимных помех. Однако построение имитационной модели непосредственно по выражениям (13)–(20) связано с трудозатратами, сравнимыми с разработкой рабочего программного обеспечения. Поэтому для предсказания правильности реализации конвейерного параллелизма распределенным приложением потребуется использование метода имитационного моделирования, реализуемого на основе более простой по реализации исполнимой модели типа конечного автомата или сети Петри при наличии соответствующих инструментальных средств.

Специально для имитационного моделирования систем продукционных правил, к которым сводятся логико-алгебраические операционные выражения типа (4)–(12) и (13)–(20), предназначена, например, программа PredNet (предикатные сети) [11], однако ее рекомендуется использовать только после

получения положительного результата предварительной проверки исходной ординарной ингибиторной сети Петри на живость и безопасность (1-ограниченность). Далее в настоящей работе будут кратко определены понятия о живости и безопасности сетей Петри на основании известных работ.

Моделирование распределенных приложений

Распределенные приложения, развернутые на компьютерной сети, могут включать параллельно работающие и конкурирующие компоненты (агенты). Например, возможное одновременное выполнение отдельных компонент ветвей распределенного алгоритма может быть задано распределенной реализацией общеизвестной операции ***fork***, доопределенной ранее. Другой пример касается случая, когда запросы на выполнение задач выдаются источником последовательно, один за другим, и приложение может реализовать конвейерный параллелизм. Одновременно выполняемые процессы должны обладать свойствами живости, безопасности, достижимости результата и др. Признаки ожидаемой правильной работы распределенного приложения, как правило, соответствуют свойствам часто используемого формализма сетей Петри.

Сети Петри, часто используемые для моделирования параллельных и распределенных систем, разработанная К. Петри в работе «Kommunikation mit Automaten», посвященной решению проблем описания и моделирования коммуникаций между автоматами [12]. Широкое распространение аппарата сетей Петри при моделировании различных систем связано с работами [13, 14]. Сети Петри содержат позиции и переходы, которые могут быть соединены дугами. Переходы моделируют действия, а места – состояния или условия. Детальная классификация сетей Петри дана в работе [15]. Вопросы моделирования параллельных процессов на основе сетей Петри в изложении для системных архитекторов, программистов, системных аналитиков и проектировщиков сложных систем управления изложены в работе [16].

Частный вид сетей Петри, так называемые логические, или бинарные, сети Петри, использованы для целей моделирования дискретных процессов в ряде работ. В работах [17, 18] определен ряд свойств логических сетей Петри. Результаты этих работ использованы при исследовании ряда предметных областей. В статье [19] моделируется коммерческая система, а в работе [20] – трафик дорожного движения. В работах [21, 22] предложены концептуальные сети Петри и на их основе построены модели роботизированного производства. Концептуальное представление сетей Петри смешанного типа с логическими и числовыми позициями дано в работе [23]. Обобщая и уточняя понятие о логических сетях Петри из указанных работ, дадим следующее формальное определение.

Логическая, или бинарная, сеть Петри формально определяется как семерка:

$$LPN = (P, T, F, F_{Inh}, F_{Inf}, H, M_0), \tag{21}$$

где $P = \{p_1, p_2, \ldots, p_m\}$ – конечное множество позиций; $T = \{t_1, t_2, \ldots, t_n\}$ – конечное множество переходов, причем $P \cup T \neq \varnothing$, и $P \cap T = \varnothing$;

F: $P \times T \rightarrow \{\mathbf{true}, \mathbf{false}\}$ – «входной» бинарный предикат инцидентности, определяющий наличие «простых» входных дуг к переходам; например,

наличию простой входной дуги, ведущей из позиции p_i в переход t_j соответствует истинность высказывания $F(p_i, t_j)$;

H: $T \times P \rightarrow$ {**true**, **false**} – «выходной» бинарный предикат инцидентности, определяющий наличие «простых» выходных дуг от переходов; например, наличию простой выходной дуги, ведущей из перехода t_j в позицию p_k, соответствует истинность высказывания $H(t_j, p_k)$;

M_0: $P \rightarrow$ {**true**, **false**} – унарный предикат начальной разметки позиций; например, наличию метки в некоторой позиции p_j соответствует истинность высказывания $M_0(p_j)$; динамические свойства сети определяются сменой текущей маркировки M в результате срабатывания переходов;

F_{Inh}: $P \times T \rightarrow$ {**true**, **false**} – «ингибиторный» бинарный предикат, определяющий наличие входящих в переход ингибиторных дуг, запрещающих срабатывание переходов при наличии меток в соответствующих входных позициях, причем метки по данным дугам не передаются;

F_{Inf}: $P \times T \rightarrow$ {**true**, **false**} – «информационный» бинарный предикат, определяющий наличие информационных дуг (например, дугу (p_i, t_j) при $F_{Inf}(p_i, t_j) =$ **true**), участвующих в определении истинности антецедентов продукционных правил срабатывания переходов (например, так определяется дуга (p_i, t_j) при $M(p_i) =$ **true** для определения правила срабатывания соответствующего перехода t_i); как и в предыдущем случае, метки по данным дугам не передаются.

Как и остальные виды сетей Петри, логическая сеть Петри функционирует, переходя от одной маркировки позиций к другой. Например, смена маркировки может быть вызвана выполнением правила срабатывания некоторого перехода t_i с тремя входными дугами (p_{Simp}, t_i), (p_{Inf}, t_i) и (p_{Inh}, t_i) – входными простой, информационной и ингибиторной соответственно и одной выходной дугой (t_i, p_{Out}):

$$t_i: M(p_{Simp}) \,\&\, M(p_{Inf}) \,\&\, \neg M(p_{Inh}) \rightarrow \neg M(p_{Simp}) \,\&\, M(p_{Out}), \qquad (22)$$

т.е., как следует из определения (21), после срабатывания перехода t_i состояние позиций p_{Inf} и p_{Inh} не меняется (позиция p_{Inf} содержит одну метку, позиция p_{Inh} пуста), а метка, которая находилась в позиции p_{Simp}, переходит в позицию p_{Out} . Выражаясь естественным языком, переформулируем правило (22): «Если до срабатывания перехода t_i было истинно составное высказывание $M(p_{Simp})$ & $M(p_{Inf})$ & $\neg M(p_{Inh})$, то после его срабатывания станет истинным составное высказывание $\neg M(p_{Simp})$ & $M(p_{Out})$».

Выражение (22) – это пропозициональное продукционное правило срабатывания перехода t_i. В случае, когда выходная позиция p_{Out} пуста, т.е. истинно высказывание $\neg M(p_{Out})$, переход t_i работает правильно. Однако, если выходная позиция p_{Out} не пуста, т.е. истинно высказывание $M(p_{Out})$, эта «истинность» после срабатывания перехода просто подтвердится, что соответствует потере одной метки. Для предотвращения подобной ошибочной ситуации в антецеденте правила (22) следует предусмотреть проверку незанятости позиции p_{Out} до срабатывания перехода:

$$t_i: M(p_{Simp}) \,\&\, M(p_{Inf}) \,\&\, \neg M(p_{Inh}) \,\&\, \neg M(p_{Out}) \rightarrow \neg M(p_{Simp}) \,\&\, M(p_{Out}). \qquad (23)$$

Исследование свойств сети Петри проводится, как правило, на основе анализа графа достижимых маркировок [13, 14]. Разработчику распределен-

ных приложений необходимо определить следующие свойства обычной сети Петри до перехода к ее логическому варианту: безопасность, или 1-ограниченность – это требование, чтобы при любой достижимой маркировке ни в одной из позиций сети число меток не превышало единицы; ограниченность сети в целом – требование, чтобы при любой достижимой маркировке общее число меток в позициях изменялось в пределах от минимального N_{min} до максимального N_{max}; живость – требование, чтобы каждый переход являлся потенциально срабатывающим при любой маркировке.

Графическое представление и моделирование сетей Петри можно выполнить, используя соответствующий инструментарий. В настоящей работе использована кроссплатформенная система PIPE, версия 4.3.0 (Platform Independent Petri Net Editor), представляющая собой свободно распространяемый программный инструментарий, написанный на языке Java, для графического представления, редактирования, моделирования и анализа сетей Петри (https://sourceforge.net/projects/pipe2/files/PIPEv4/PIPEv4.3.0/). На этапе предварительного моделирования агентно-базированного метакомпьютерного приложения возможно воспользоваться и другими программными инструментами, обеспечивающими моделирование и анализ графа достижимых маркировок временны*ы*х, ординарных и ингибиторных сетей Петри и позволяющими проверить проект на живость, безопасность и ограниченность сети в целом. К таким инструментариям следует причислить, например, CPN Tools (Colored Petri Nets, http://cpntools.org/2018/01/09/colored-nets/).

На рис. 4 представлена ординарная сеть Петри, соответствующая концептуальному графу распределенного приложения на рис. 2 и системе логико-алгебраических выражений Σ_1. Конвейерный параллелизм здесь не реализуется.

Возможность реализации распределенным приложением конвейерного параллелизма можно проверить путем организации в модели очереди заданий на выполнение. Модель, представленная на рис. 5, соответствует концептуальному графу распределенного приложения на рис. 2 и системе логико-алгебраических выражений Σ_2, описывающих сетевое приложение, запускаемое на выполнение последовательно несколькими (в данном случае пятью) сообщениями. Исследование данной сети Петри показало, что она обладает всеми необходимыми для работы проектируемого приложения свойствами: живостью, безопасностью и ограниченностью всей сети. Результат исследования позволяет считать определение (21) сети Петри корректным.

Для сетей Петри, не являющихся безопасными, или 1-ограниченными, определение (21) и последующие правила не имеют места. В случае, когда инструментальное средство позволяет моделировать безопасные сети Петри, для предотвращения логических ошибок необходимо проверять незанятость позиций непосредственно перед их занятием.

Последующую реализацию агентно-базированного распределенного приложения в сетевой среде рекомендуется создавать на основе формализованных спецификаций, представленных системами логико-алгебраических операционных выражений Σ_1 и Σ_2. В случае использования при реализации приложения системы Σ_1 следующее сообщение рекомендуется запускать в сеть только после того, как будет получен ответ на предыдущее сообщение,

свидетельствующий о завершении выполнения распределенного алгоритма. В этом случае параллелизм в выполнении распределенного приложения может быть получен только при помощи реализованной операции ***fork-join***. Реализация распределенного приложения на основе системы Σ_2 свободна от недостатка предыдущей реализации ввиду гарантии отсутствия тупиковых ситуаций, и ее ограниченность определена только временем ожидания результата. На это время влияет число сообщений, запущенных в сеть, реализующую функции метакомпьютерного приложения. Новое качество приложения обусловлено возможностью установления параллелизма конвейерного типа.

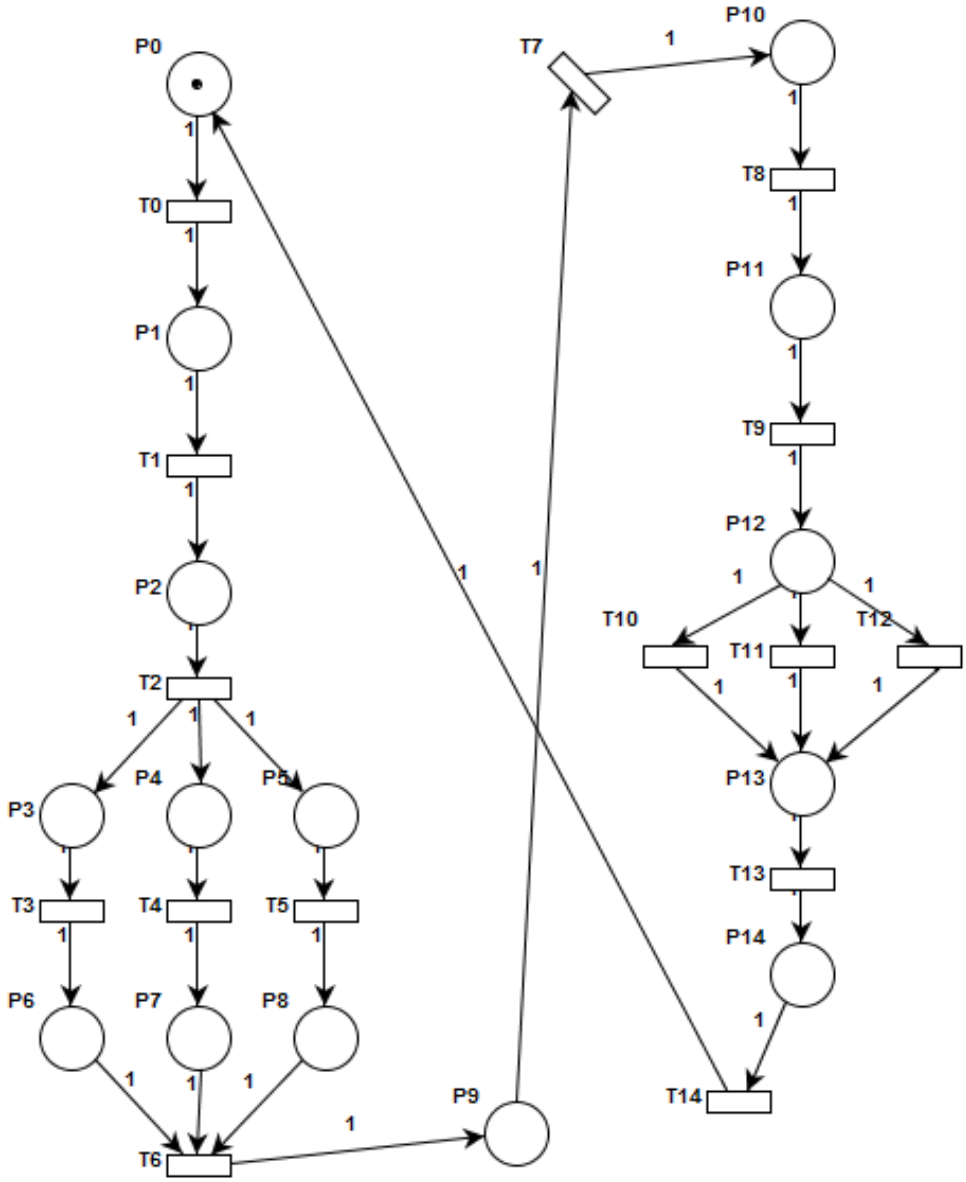

Рис. 4. Сеть Петри, используемая для моделирования сетевого приложения, работа которого происходит под управлением одного сообщения

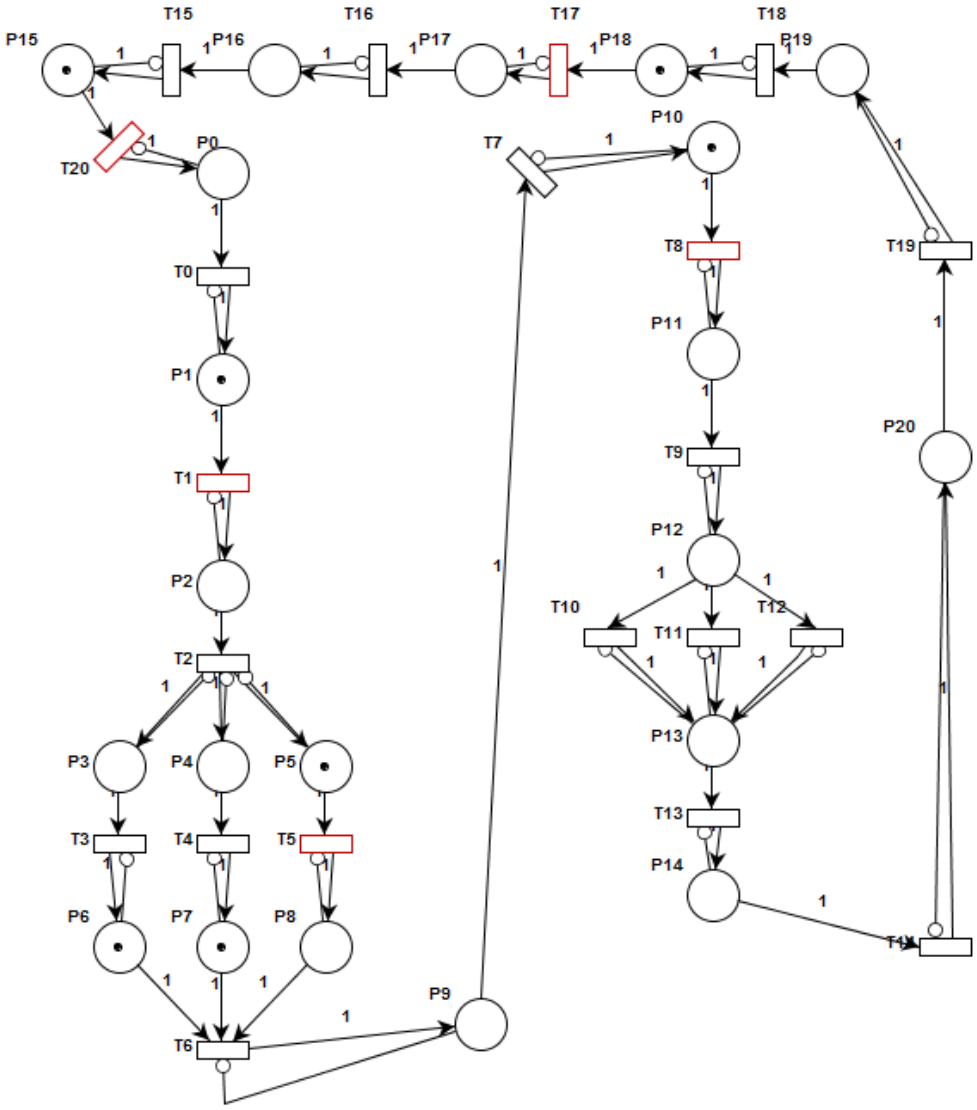

Рис. 5. Модель с очередью сообщений на основе ингибиторной сети Петри для исследования реализуемости конвейерного параллелизма в распределенном приложении

Заключение

Проблема создания распределенных метакомпьютерных или облачно-сетевых приложений требует привлечения новых методов и технологий. Несмотря на то, что в последнее время появилось несколько технологий, облегчающих создание распределенных приложений систем, задача решения алгоритмических и логических проблем, возникающих при создании спецификаций для программного продукта, остается актуальной. В работе предложены элементы технологии создания сложных агентно-базированных сетевых приложений, основанные на логических моделях искусственного интеллекта: концептуальных графов, операторного сценарного метаязыка и исполнимых спецификаций в форме систем логико-алгебраических операционных выра-

жений. Уточнено определение логических информационно-ингибиторных сетей Петри, применяемых для моделирования сетевых приложений, рассматриваемых как системы массового обслуживания. Предложено дальнейшее развитие технологии связать с ее интеграцией с известными мультиагентными технологиями. Отмечено, что очередные шаги в данном направлении были сделаны в работах [1–4, 7, 8, 18].

Список литературы

1. Волчихин В. И., Зинкин С. А., Карамышева Н. С. Организация функционирования облачно-сетевых распределенных вычислительных систем с архитектурой «агенты как сервисы» // Известия высших учебных заведений. Поволжский регион. Технические науки. 2019. № 4. С. 27–50. doi:10.21685/2072-3059-2019-4-3
2. Волчихин В. И., Карамышева Н. С., Горынина А. В., Зинкин С. А. Разработка сетевых агентно-базированных приложений на основе метакомпьютерной технологии // Известия высших учебных заведений. Поволжский регион. Технические науки. 2021. № 4. С. 3–25. doi:10.21685/2072-3059-2021-4-1
3. Zinkin S. A., Volchihin V. I., Karamysheva N. S., Jaafar M. S. Synthesis of the Functional Architecture of Agent-based Computing Systems when Using Conceptual Models of Artificial Intelligence // 2021 International Conference on Engineering Management of Communication and Technology, EMCTECH 2021. Proceedings, Austria, Vienna, 2021. P. 1–6. doi:10.1109/EMCTECH53459.2021.9619174
4. Zinkin S. A., Volchihin V. I., Karamysheva N. S., Jaafar M. S. Dynamic Topology Transformation of Cloud-Network Computer Systems: Conceptual Level // 2020 International Conference on Engineering Management of Communication and Technology, EMCTECH 2020. Proceedings, Austria, Vienna, 2020. P. 1–10. doi:10.1109/EMCTECH49634.2020.9261554
5. Zhang R. P2P System. URL: https://zhangruochi.com/P2P-System/2019/02/14
6. Li J. On peer-to-peer (P2P) content delivery // Peer-to-Peer Networking and Applications. 2008. Vol. 1. P. 45–63. doi:10.1007/s12083-007-0003-1
7. Ghareeb M., Rouibia S., Parrein B., Raad M., Thareau C. P2PWeb: a Client/Server and P2P Hybrid Architecture for Content Delivery over Internet // Third International Conference on Communications and Information Technology ICCIT 2013. Beirut, Lebanon, 2013. P. 1–5. URL: https://hal.archives-ouvertes.fr/hal-00846178
8. Youssfi M., Bouattane O., Bakkoury J., Bensalah M. A New Massively Parallel And Distributed Virtual Machine Model Using Mobile Agents // Conference: IEEE Conference: 4th International Conference on Multimedia Computing and Systems April 14-16, 2014 Marrakech – MOROCCO MSTI Mediterranean Space of Technology and Innovation. Morocco Chapter. Marrakech, Morocco, 2014. P. 407–414. doi:10.1109/ICMCS.2014.6911306
9. Hughes C., Hughes T. Parallel and distributed programming using C++. Addison-Wesley Professional, 2004. 691 p.
10. Hunt C. TCP/IP Network Administration, 3rd Edition. O'Reilly Media. 2022. URL: https://www.oreilly.com. Date of Free (access 01.04.2022).
11. Карамышева Н. С., Голубева Д. В. Система моделирования асинхронных предикатных сетей PREDNET для распределенных систем логического управления // Новые информационные технологии и системы : сб. науч. ст. по материалам XVII Междунар. науч.-техн. конф. НИТиС-2020. Пенза : Изд-во ПГУ, 2020. С. 40–44.
12. Petri C. A. Kommunikation mit Automaten. Communication with Automata, Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York. 1966.
13. Peterson J. L. Petri Net Theory and the Modeling of Systems. New Jersey : Prentice-Hall, 1981. 288 p.
14. Котов В. Е. Сети Петри. М. : Наука. 1984. 160 с.

15. Ganzhura M. A, Smirnova O. V., Zakharova A. A., Romanov D. L., Dyachenko N. V. Methods of Modeling Objects Using Petri Nets // XV International Scientific-Technical Conference “Dynamic of Technical Systems” (DTS-2019) AIP Conf. Proc. Published by AIP Publishing, 2019. Vol. 2188. P. 050023-1–050023-8. doi:10.1063/1.5138450
16. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. Моделирование параллельных процессов. Сети Петри. Курс для системных архитекторов, программистов, системных аналитиков, проектировщиков сложных систем управления. СПб. : Профессиональная литература, АйТи-Подготовка, 2014. 400 с.
17. Du Y., Jiang C. Formal Representation and Analysis of Batch Stock Trading Systems by Logical Petri Net Workflows. Lecture Notes in Computer Science. 2002. Vol. 2495. P. 221–225.
18. Du Y., Jiang C., Guo Y. Towards a Formal Model for Grid Architecture via Petri Nets // Information Technology Journal. 2006. Vol. 5. P. 833–841. doi:10.3923/itj.2006.833.841
19. Du Y., Qi L., Zhou M. Analysis and Application of Logical Petri Nets to E-Commerce Systems // IEEE Transactions on Systems, Man, and Cybernetics: Systems. 2014. Vol. 44, № 4. P. 468–481. doi:10.1109/TSMC.2013.2277696
20. Yaqub O., Lingxi Li L. Modeling and Analysis of Connected Traffic Intersections Based on Modified Binary Petri Nets // International Journal of Vehicular Technology. 2013. Vol. 2013. P. 1–10. doi:10.1155/2013/192516
21. Зинкин С. А, Пащенко Д. В., Пучкова У. Н., Джафар М. С. Интеграция методов концептуального и поведенческого моделирования дискретно-событийных систем: I. Синтез и анализ концептуальной модели // Кибернетика и программирование. 2016. № 6. С. 83–95.
22. Зинкин С. А, Пащенко Д. В., Пучкова У. Н., Джафар М. С. Интеграция методов концептуального и поведенческого моделирования дискретно-событийных систем: II. Логико-алгебраические операционные модели и инфокоммуникационные технологии // Кибернетика и программирование. 2017. № 1. С. 75–93.
23. Зинкин С. А., Джафар М. С., Карамышева Н. С. Концептуальные представления и модификации сетей Петри для приложений в области синтеза функциональной архитектуры распределенных вычислительных систем с переменной структурой // Известия Юго-Западного государственного университета. 2018. Т. 22, № 6 (81). С. 143–167.

References

1. Volchikhin V.I., Zinkin S.A., Karamysheva N.S. Organization of the functioning of cloud-network distributed computing systems with the architecture “agents as services”. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2019;(4):27–50. (In Russ.). doi:10.21685/2072-3059-2019-4-3
2. Volchikhin V.I., Karamysheva N.S., Gorynina A.V., Zinkin S.A. Development of network agent-based applications based on metacomputer technology. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2021;(4):3–25. (In Russ.). doi:10.21685/2072-3059-2021-4-1
3. Zinkin S.A., Volchihin V.I., Karamysheva N.S., Jaafar M.S. Synthesis of the Functional Architecture of Agent-based Computing Systems when Using Conceptual Models of Artificial Intelligence. *2021 International Conference on Engineering Management of Communication and Technology, EMCTECH 2021*. Proceedings, Austria, Vienna, 2021:1–6. doi:10.1109/EMCTECH53459.2021.9619174
4. Zinkin S.A., Volchihin V.I., Karamysheva N.S., Jaafar M.S. *Dynamic Topology Transformation of Cloud-Network Computer Systems: Conceptual Level.* 2020 *International Conference on Engineering Management of Communication and Technology,*

EMCTECH 2020. Proceedings, Austria, Vienna, 2020:1–10. doi:10.1109/EMCTECH49634.2020.9261554
5. Zhang R. *P2P System*. Available at: https://zhangruochi.com/P2P-System/2019/02/14
6. Li J. On peer-to-peer (P2P) content delivery. *Peer-to-Peer Networking and Applications*. 2008;1:45–63. doi:10.1007/s12083-007-0003-1
7. Ghareeb M., Rouibia S., Parrein B., Raad M., Thareau C. P2PWeb: a Client/Server and P2P Hybrid Architecture for Content Delivery over Internet. *Third International Conference on Communications and Information Technology ICCIT 2013*. Beirut, Lebanon, 2013:1–5. Available at: https://hal.archives-ouvertes.fr/hal-00846178
8. Youssfi M., Bouattane O., Bakkoury J., Bensalah M. A New Massively Parallel And Distributed Virtual Machine Model Using Mobile Agents. *Conference: IEEE Conference: 4th International Conference on Multimedia Computing and Systems April 14-16, 2014 Marrakech – MOROCCO MSTI Mediterranean Space of Technology and Innovation. Morocco Chapter*. Marrakech, Morocco, 2014:407–414. doi:10.1109/ICMCS.2014.6911306
9. Hughes C., Hughes T. *Parallel and distributed programming using C++*. Addison-Wesley Professional, 2004:691.
10. Hunt C. *TCP/IP Network Administration, 3rd Edition. O'Reilly Media*. 2022. Available: https://www.oreilly.com. Date of Free (accessed 01.04.2022).
11. Karamysheva N.S., Golubeva D.V. Modeling system of asynchronous predicate networks PREDNET for distributed logic control systems. *Novye informatsionnye tekhnologii i sistemy: sb. nauch. st. po materialam XVII Mezhdunar nauch.-tekhn. konf. NITiS-2020 = New information technologies and systems: proceedings of the 17th International scientific and engineering conference "New information technologies and systems-2020"*. Penza: Izd-vo PGU, 2020:40–44. (In Russ.)
12. Petri C.A. *Kommunikation mit Automaten. Communication with Automata, Technical Report RADC-TR-65-377, Rome Air Dev*. Center, New York. 1966.
13. Peterson J.L. *Petri Net Theory and the Modeling of Systems*. New Jersey: Prentice-Hall, 1981:288.
14. Kotov V.E. *Seti Petri = Petri nets*. Moscow: Nauka. 1984:160. (In Russ.)
15. Ganzhura M.A, Smirnova O.V., Zakharova A.A., Romanov D.L., Dyachenko N.V. Methods of Modeling Objects Using Petri Nets. *XV International Scientific-Technical Conference "Dynamic of Technical Systems" (DTS-2019) AIP Conf. Proc*. Published by AIP Publishing, 2019;2188:050023-1–050023-8. doi:10.1063/1.5138450
16. Marakhovskiy V.B., Rozenblyum L.Ya., Yakovlev A.V. *Modelirovanie parallel'nykh protsessov. Seti Petri. Kurs dlya sistemnykh arkhitektorov, programmistov, sistemnykh analitikov, proektirovshchikov slozhnykh sistem upravleniya = Simulation of parallel processes. Petri nets. Course for system architects, programmers, system analysts, designers of complex control systems*. Saint Petersburg: Professional'naya literatura, AyTi-Podgotovka, 2014:400.
17. Du Y., Jiang C. *Formal Representation and Analysis of Batch Stock Trading Systems by Logical Petri Net Workflows. Lecture Notes in Computer Science*. 2002;2495:221–225.
18. Du Y., Jiang C., Guo Y. Towards a Formal Model for Grid Architecture via Petri Nets. *Information Technology Journal*. 2006;5:833–841. doi:10.3923/itj.2006.833.841
19. Du Y., Qi L., Zhou M. Analysis and Application of Logical Petri Nets to E-Commerce Systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. 2014;44(4):468–481. doi:10.1109/TSMC.2013.2277696
20. Yaqub O., Lingxi Li L. Modeling and Analysis of Connected Traffic Intersections Based on Modified Binary Petri Nets. *International Journal of Vehicular Technology*. 2013;2013:1–10. doi:10.1155/2013/192516
21. Zinkin C.A, Pashchenko D.V., Puchkova U.N., Dzhafar M.S. Integration of methods of conceptual and behavioral modeling of discrete-event systems: I. Synthesis and analysis of the conceptual model. *Kibernetika i programmirovanie = Cybernetics and programming*. 2016;(6):83–95. (In Russ.)

22. Zinkin C.A, Pashchenko D.V., Puchkova U.N., Dzhafar M.S. Integration of conceptual and behavioral modeling methods for discrete-event systems: II. Logical-algebraic operating models and infocommunication technologies. *Kibernetika i programmirovanie = Cybernetics and programming*. 2017;(1):75–93. (In Russ.)
23. Zinkin S.A., Dzhafar M.S., Karamysheva N.S. Conceptual representations and modifications of Petri nets for applications in the field of synthesis of the functional architecture of distributed computing systems with a variable structure. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of Southwestern State University*. 2018;22(6):143–167. (In Russ.)

Информация об авторах / Information about the authors

Владимир Иванович Волчихин
доктор технических наук, профессор, президент Пензенского государственного университета (Россия, г. Пенза, ул. Красная, 40)

E-mail: cnit@pnzgu.ru

Vladimir I. Volchikhin
Doctor of engineering sciences, professor, president of Penza State University (40 Krasnaya street, Penza, Russia)

Надежда Сергеевна Карамышева
кандидат технических наук, доцент кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: vt@pnzgu.ru

Nadezhda S. Karamysheva
Candidate of engineering sciences, associate professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Сергей Александрович Зинкин
доктор технических наук, доцент, профессор кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: zsa49@yandex.ru

Sergey A. Zinkin
Doctor of engineering sciences, associate professor, professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Евгений Иванович Гурин
доктор технических наук, профессор, профессор кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: zsa49@yandex.ru

Evgeniy I. Gurin
Doctor of engineering sciences, professor, professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflicts of interests.

Поступила в редакцию / Received 10.03.2022

Поступила после рецензирования и доработки / Revised 08.04.2022

Принята к публикации / Accepted 25.04.2022