в сети в контексте использования Self-optimizing network.

Литература

1. Lobinger A., Stefansk, S.; Jansen T., Balan I. Load Balancing in Downlink LTE Self-Optimizing Networks // IEEE 71st Vehicular Technology Conference (VTC 2010-Spring). Taipei, Taiwan, 2010. – P.1-5.
2. Atayero A.A., Luka M.K. A Soft Computing Approach to Dynamic Load Balancing in 3GPP LTE.International // Journal of Computer Applications. Vol.43(19), 2012. – P.35-41.
3. LTE/LTE-Advanced system-level simulator, https://launchpad.net/imtaphy (29.08.2013)
4. Ермолаев С.Ю., Карташевский В.Г., Штовба С.Д. // Электросвязь. №12, 2010. – С. 54-58.

ANALYSIS OF LOAD DISTRIBUTION INDEX IN LTE NETWORKS

Kartashevskaya E.S.

This paper is about estimation the real parameters of LTE network functionality such as load distribution index. Also we presents calculations of index values using fuzzy logic.

Keywords: *LTE, load distribution index, load balancing, neuro-fuzzy logic.*

Карташевская Евгения Сергеевна, ассистент Кафедры информатики и вычислительной техники Поволжского государственного университета телекоммуникаций и информатики. Тел. 8-846-342-34-40. E-mail: ka_es@bk.ru

ТЕХНОЛОГИИ КОМПЬЮТЕРНЫХ СИСТЕМ И СЕТЕЙ

УДК: 681.32

О РЕАЛИЗАЦИИ ИНТЕРФЕЙСА ПЕРЕДАЧИ СООБЩЕНИЙ КАК ОБЛАЧНОГО СЕРВИСА В АГЕНТНО-ОРИЕНТИРОВАННЫХ МЕТАКОМПЬЮТЕРНЫХ СИСТЕМАХ

Вашкевич Н.П., Зинкин С.А., Карамышева Н.С.

Предложены модели логического управления глобальными вычислительными процессами в вычислительных сетях с агентно-ориентированными облачными метакомпьютерными сервисами, основанные на коллективных пересылках данных, что позволяет упростить реализацию массового параллелизма в крупномасштабных прикладных распределенных системах. Предложено для логического управления дискретными процессами использовать формализм асинхронных предикатных сетей, позволяющий провести абстрактный и структурный синтез функциональной архитектуры агентно-ориентированного метакомпьютера с реализацией глобальных коллективных и вычислительных операций для группы агентов – виртуальных узлов метакомпьютера.

Ключевые слова: метакомпьютер, облачные вычисления, логическое управление процессами, коллективные и вычислительные операции.

Введение

Ведущие поставщики облачных сервисов – фирмы Amazon, Sun Microsystems, Microsoft, Google и другие предоставляют различные облачные услуги: хранение данных, в том числе в распределенных хранилищах данных, аренду виртуальных серверов, предоставление вычислительных мощностей, хранение приложений, библиотек и связанных с ними конфигурационных параметров, выбор типа операционной системы, на которой предполагается выполнять приложения, предоставление доступа к высокопроизводительным компьютерам и системам через Internet. Известны также и другие разработчики приложений как сервисов – фирмы Microsoft Dynamics, Salesforce, Taleo, Workday, NetSuite, Oracle, SucsessFactors.

С недавнего времени начали получать применение инфраструктуры и облачные сервисы, созданные упомянутыми фирмами, такие как Amazon Web Services – инфраструктура WebServices платформы в облаке и входящий в нее веб-сервис Amazon Elastic Compute Cloud, который предоставляет вычислительные мощности в облаке, Sun Cloud – сервис облачных высокопроизводительных вычислений, Windows Azure – новая серверная операционная система, предлагаемая в качестве платформы для создания

облачных веб-приложений, известная ранее под названием Windows Cloud, Google App Engine – сервис хостинга сайтов и web-приложений на серверах Google.

Организация метакомпьютеров в виде облачных сервисов отражает современную тенденцию к организации распределенных вычислений, при которой используемое программное обеспечение метакомпьютера (так же, как и облачное программное обеспечение, относящееся к классу middleware) и сами обрабатываемые данные хранятся на облачных серверах, а в распоряжении клиента имеется простой Web-интерфейс.

Однако облачным технологиям присущи и некоторые недостатки, сдерживающие их промышленное и научное применение. Например, требуется постоянное соединение пользователя с Internet, трудоемко настраивание программного обеспечения, затребованного пользователем, под его собственные цели. Последние ограничения частично преодолимы путем интеграции облачных и агентно-ориентированных технологий. Например, концепция мобильных вычислений, поддерживаемая мигрирующими агентами, не требует постоянного соединения клиента с Internet.

Часть работы агент может выполнить автономно и далее передать ее результаты после того момента времени, когда клиент снова подключится к сети. Специальные агенты могут осуществлять поиск, отбор и мониторинг данных. Агенты, осуществляющие обработку данных от различных источников, могут взаимодействовать друг с другом, клонироваться и перемещаться на различные серверы. Благодаря данным функциям агентов возможно преодолеть многие недостатки и развить функциональность облачных сервисов, используя потенциально неограниченные вычислительные ресурсы и ресурсы хранения.

Благодаря агентно-ориентированным технологиям можно увеличить потенциальные возможности облачных технологий, такие как гибкая виртуализация ресурсов, высокая доступность, прозрачный доступ, простое администрирование, масштабирование вычислений и ресурсов хранения.

Метакомпьютеры как облачные сервисы

В промышленности, научных и учебных учреждениях при помощи облачных технологий можно сократить расходы на приобретение, использование и хранение программ и данных, которые могут находиться на удаленных серверах. Состав основных облачных сервисов, рассмотренных в работе [1] и известных также по другим работам [2-4], представлен на рис. 1. Данные сервисы выполняют в основном не вычислительные функции, а функции хранения и обработки данных. Подобно тому, как из «облака» на подключенном к сети компьютере запускаются программы, может быть организован и запуск больших масштабируемых сетевых приложений, например метакомпьютерных сервисов, что не нашло пока отражения в литературе.

Для метакомпьютеров обработки данных могут быть выбраны многие из показанных на рис. 1 сервисов: сервисы хранения данных (возможно, структурированных), сервисы распределенных систем управления базами данных (РСУБД), сервисы управления процессами и ресурсами, сервисы приложений и другие.

Сервисы управления процессами и ресурсами позволяют организовать единое управление многими ресурсами, например, виртуальными узлами метакомпьютера, очередями заданий, метаприложениями. Предоставляемая в качестве сервиса платформа может обеспечить разработку разнообразных метакомпьютерных приложений, в том числе распределенных баз данных.

Управление работой метакомпьютера возможно организовать с помощью сервисов администрирования и управления, позволяющих задавать параметры метакомпьютера как одного из облачных сервисов: его топологию, используемые ресурсы (данные, СУБД, вычислительные узлы и др.), уровень виртуализации и масштабирования задач. Интегрированное программное обеспечение (ПО) как сервис возможно использовать для организации интерфейсов пользователей с программным обеспечением метакомпьютеров.

Технология облачных вычислений в настоящее время находится в стадии интенсивного развития и объединяет в своем составе многие из известных технологий, особенно технологии виртуализации. Ключевым понятием в облачных вычислениях является предоставление пользователям услуг как Internet-сервисов.

Крупные Internet-сервисы, например Amazon, Google и другие, послужили основой развития идеи облачных вычислений. Возможность увеличения объемов хранимой информации, развитие технологий виртуализации привело к созданию сетевого программного обеспечения, обеспечивающего создание вир-

туальной вычислительной инфраструктуры с практически неограниченными возможностями масштабирования (при наращивании аппаратных ресурсов без значительного усложнения коммуникаций) и доступной из любой точки к Internet.

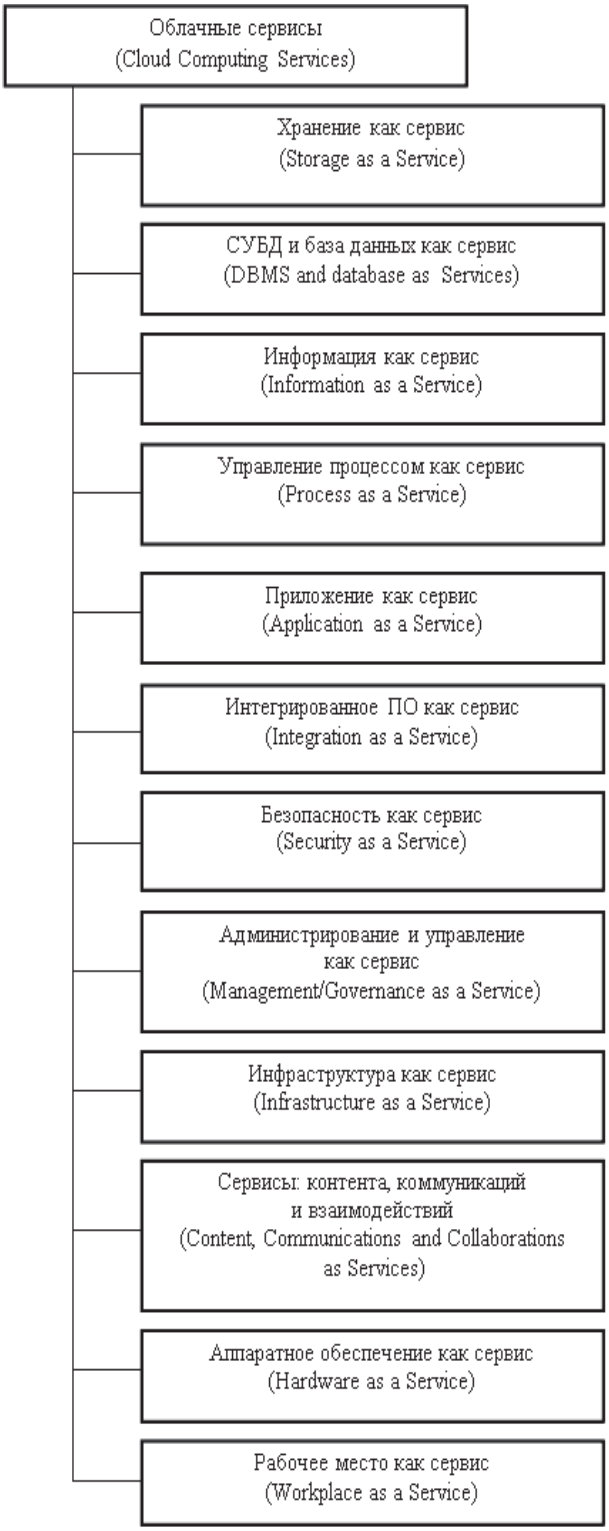

Рис. 1. Основные виды облачных сервисов

Использование предикатных сетей для формального описания интерфейса передачи сообщений

В [5-6] предложено для логического управления дискретными процессами использовать формализм асинхронных предикатных сетей (АПС). Проиллюстрировано применение асинхронных предикатных сетей для формализации основных процессов занятия и освобождения ресурсов в агентно-ориентированных метакомпьютерных системах. Далее в данной среде становится возможной организация метакомпьютерных вычислений. Следующими важными задачами являются организация управления коллективным обменом, коллективными вычислениями и синхронизация процессов. Примеры двух вариантов коллективных пересылок данных между агентами по принципу «каждый с каждым» представлены на рис. 2.

Рассмотрим четыре предикатные сети MCS_1, MCS_2, MCS_3, MCS_4, реализующие при соответствующей программной интерпретации управление коллективным обменом данными и управляющими сообщениями в метакомпьютерных системах. Приводимые формальные спецификации относятся к классу непосредственно интерпретируемых в том смысле, что при создании распределенных программ должны учитываться только эти спецификации. Системой продукций для асинхронной предикатной сети MCS_1 задается процесс логического управления в метакомпьютерной системе при передаче данных от агента A каждому из остальных агентов B, C, D, E с подтверждением приема и передачей результатов от агентов B, C, D, E агенту A.

Данная система MCS_1 соответствует схеме обменов, представленной на рис. 2*а*. Введены предикаты P_x, $x \in \{a,b,c,d,e\}$, характеризующие состояния процессов для мобильных агентов A, B, C, D, E соответственно. Например, если истинно высказывание $P_x(x_i)$, то это означает, что агент хнаходится в состоянии x_i. Каждый агент x может находиться в одном из трех состояний: x_1 (начальная фаза), x_2 (рабочая фаза) и x_3 (заключительная фаза). Для процесса, создаваемого агентом A, добавлено состояние a', соответствующее широковещательной передаче сообщений каждому их остальных агентов. Этим передачам соответствуют модификации предиката P_s. Агенты B, C, D, E затем отвечают отправкой ответных сообщений агенту A, отмечая эти события модификациями предиката P_r. Затем процессы агентов переходят в рабочие состояния, завершающиеся переходами каждого из них

в заключительные состояния и модификациями предиката P_q, что сопровождает факты отправки агентами *B*, *C*, *D*, *E* сообщений об окончании работы или квитанций агенту *A*. Выполнение одноименных процессов *A*, *B*, *C*, *D*, *E* сопровождается согласованными модификациями предикатов $P_x, x \in \{a,b,c,d,e\}$. Инициатором начальной широковещательной рассылки сообщений является агент *A*.

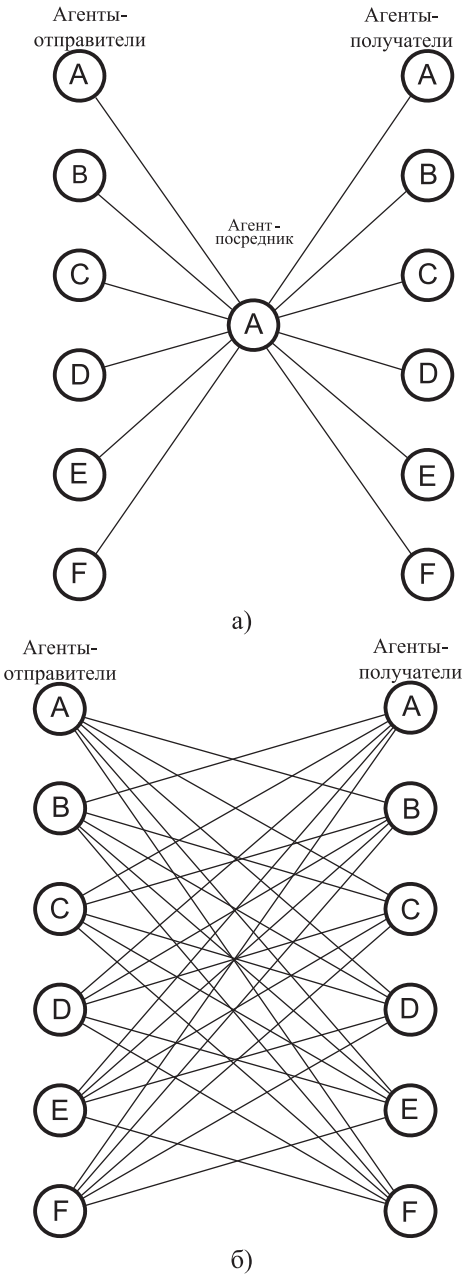

Рис. 2. Коллективные пересылки данных между агентами по принципу «каждый с каждым» через агента-посредника (а) и непосредственно между агентами (б)

Выражения для процедур в правых частях продукций записаны в упрощенной форме. Например, выражение $P_a(a_3) \Rightarrow \neg P_a(a_3) \,\&\, P_a(a_1)$ в полной форме имело бы следующий вид $P_a(a_3) \Rightarrow (P_a(a_3) \leftarrow false) \,\&_{\rightarrow}\, (P_a(a_1) \leftarrow true)$, где $\&_{\rightarrow}$ символ последовательной композиции операторов.

Система продукций для асинхронной предикатной сети MCS_1 имеет следующий вид:

MCS_1:

– продукции для агента A:

$$T_{a1}: P_a(a_1) \Rightarrow \neg P_a(a_1) \,\&\, P_a(a_2) \,\&\, P_s(s_{ab}) \,\&\, P_s(s_{ac}) \,\&\, P_s(s_{ad}) \,\&\, P_s(s_{ae});$$

$$T_{a'}: P_a(a') \,\&\, P_r(r_{ba}) \,\&\, P_r(r_{ca}) \,\&\, P_r(r_{da}) \,\&\, P_r(r_{ea}) \Rightarrow \neg P_a(a') \,\&\, \neg P_r(r_{ba}) \,\&\, \neg P_r(r_{ca}) \,\&\, \neg P_r(r_{da}) \,\&\, \neg P_r(r_{ea}) \,\&\, P_a(a_2);$$

$$T_{a2}: P_a(a_2) \,\&\, P_q(q_{ba}) \,\&\, P_q(q_{ca}) \,\&\, P_q(q_{da}) \,\&\, P_q(q_{ea}) \Rightarrow \neg P_a(a_2) \,\&\, \neg P_q(q_{ba}) \,\&\, \neg P_q(q_{ca}) \,\&\, \neg P_q(q_{da}) \,\&\, \neg P_q(q_{ea}) \,\&\, P_a(a_3);$$

$$T_{a3}: P_a(a_3) \Rightarrow \neg P_a(a_3) \,\&\, P_a(a_1);$$

– продукции для агента B:

$$T_{b1}: P_b(b_1) \,\&\, P_s(s_{ab}) \Rightarrow \neg P_b(b_1) \,\&\, \neg P_s(s_{ab}) \,\&\, P_b(b_2) \,\&\, P_r(r_{ab});$$

$$T_{b2}: P_b(b_2) \Rightarrow \neg P_b(b_2) \,\&\, P_b(b_3) \,\&\, P_q(q_{ba});$$

$$T_{b3}: P_b(b_3) \Rightarrow \neg P_b(b_3) \,\&\, P_b(b_1);$$

– продукции для агента C:

$$T_{c1}: P_c(c_1) \,\&\, P_s(s_{ac}) \Rightarrow \neg P_c(c_1) \,\&\, \neg P_s(s_{ac}) \,\&\, P_c(c_2) \,\&\, P_r(r_{ca});$$

$$T_{c2}: P_c(c_2) \Rightarrow \neg P_c(c_2) \,\&\, P_c(c_3) \,\&\, P_q(q_{ca});$$

$$T_{c3}: P_c(c_3) \Rightarrow \neg P_c(c_3) \,\&\, P_c(c_1);$$

– продукции для агента D:

$$T_{d1}: P_d(d_1) \,\&\, P_s(s_{ad}) \Rightarrow \neg P_d(d_1) \,\&\, \neg P_s(s_{ad}) \,\&\, P_d(d_2) \,\&\, P_r(r_{da});$$

$$T_{d2}: P_d(d_2) \Rightarrow \neg P_d(d_2) \,\&\, P_d(d_3) \,\&\, P_q(q_{da});$$

$$T_{d3}: P_d(d_3) \Rightarrow \neg P_d(d_3) \,\&\, P_d(d_1);$$

– продукции для агента E:

$$T_{e1}: P_e(e_1) \,\&\, P_s(s_{ae}) \Rightarrow \neg P_e(e_1) \,\&\, \neg P_s(s_{ae}) \,\&\, P_e(e_2) \,\&\, P_r(r_{ea});$$

$$T_{e2}: P_e(e_2) \Rightarrow \neg P_e(e_2) \,\&\, P_e(e_3) \,\&\, P_q(q_{ea});$$

$$T_{e3}: P_e(e_3) \Rightarrow \neg P_e(e_3) \,\&\, P_e(e_1).$$

Начальное состояние сети задают следующие начальные факты (истинные атомарные константные формулы): $P_a(a_1)$, $P_b(b_1)$, $P_c(c_1)$, $P_d(d_1)$ и $P_e(e_1)$. Имена продукционных правил,

описывающих процесс логического управления обменом данных в метакомпьютерной системе, записаны слева от ядер соответствующих продукций. Для индексации предметных констант, предикатных констант и продукционных правил здесь и далее использованы имена констант, содержащие одно- или двухбуквенные сочетания символов (например, *ad*, *e*), либо букву и цифры (например, *d*2, *e*3), имеющие мнемонический смысл и заменяемые в программной реализации продукционных правил числами, выбираемыми из натурального ряда. Здесь и далее все массивы – одномерные.

Следующей системой продукций для асинхронной предикатной сети MCS_2 описывается процесс обмена данными между агентами по принципу «каждый с каждым» (через агент *A*) с возвратом результатов агенту *A*. Данный метод управления обменом данных можно использовать также для синхронизации процессов в метакомпьютерных системах. Введены дополнительные состояния b_0, c_0, d_0, e_0 для агентов *B*, *C*, *D*, *E* и не используется состояние a' для агента *A*. Здесь агенты *B*, *C*, *D*, *E* синхронизируют свои действия, направляя сообщения агенту *A*, что сопровождается соответствующими модификациями предиката P_r. Ответные посылки сообщений от агента *A* сопровождаются модификациями предиката P_s. Завершив согласованные действия, агенты *B*, *C*, *D*, *E* сигнализируют об этом агенту *A*, модифицируя предикат P_q. Начальное состояние сети задают следующие начальные факты (истинные атомарные константные формулы): $P_a(a_1)$, $P_b(b_0)$, $P_c(c_0)$, $P_d(d_0)$ и $P_e(e_0)$.

Система продукций для асинхронной предикатной сети MCS_2 имеет следующий вид:

MCS_2:

– продукции для агента *A*:

$$T_{a1}: P_a(a_1) \,\&\, P_r(r_{ba}) \,\&\, P_r(r_{ca}) \,\&\, P_r(r_{da}) \,\&\, P_r(r_{ea}) \Rightarrow \neg P_a(a_1) \,\&\, P_a(a_2) \,\&\, \neg P_r(r_{ba}) \,\&\, \neg P_r(r_{ca}) \,\&\, \neg P_r(r_{da}) \,\&\, \neg P_r(r_{ea}) \,\&\, P_s(s_{ab}) \,\&\, P_s(s_{ac}) \,\&\, P_s(s_{ad}) \,\&\, P_s(s_{ae});$$

$$T_{a2}: P_a(a_2) \,\&\, P_q(q_{ba}) \,\&\, P_q(q_{ca}) \,\&\, P_q(q_{da}) \,\&\, P_q(q_{ea}) \Rightarrow \neg P_a(a_2) \,\&\, P_a(a_3) \,\&\, \neg P_q(q_{ba}) \,\&\, \neg P_q(q_{ca}) \,\&\, \neg P_q(q_{da}) \,\&\, \neg P_q(q_{ea});$$

$$T_{a3}: P_a(a_3) \Rightarrow \neg P_a(a_3) \,\&\, P_a(a_1);$$

– продукции для агента *B*:

$$T_{b0}: P_b(b_0) \Rightarrow \neg P_b(b_0) \,\&\, P_b(b_1) \,\&\, P_r(r_{ba});$$

$$T_{b1}: P_b(b_1) \,\&\, P_s(s_{ab}) \Rightarrow \neg P_b(b_1) \,\&\, P_b(b_2) \,\&\, \neg P_s(s_{ab});$$

$$T_{b2}: P_b(b_2) \Rightarrow \neg P_b(b_2) \,\&\, P_b(b_3) \,\&\, P_q(q_{ba});$$

$$T_{b3}: P_b(b_3) \Rightarrow \neg P_b(b_3) \,\&\, P_b(b_0);$$

– продукции для агента C:

$$T_{c0}: P_c(c_0) \Rightarrow \neg P_c(c_0) \,\&\, P_c(c_1) \,\&\, P_r(r_{ca});$$

$$T_{c1}: P_c(c_1) \,\&\, P_s(s_{ac}) \Rightarrow \neg P_c(c_1) \,\&\, P_c(c_2) \,\&\, \neg P_s(s_{ac});$$

$$T_{c2}: P_c(c_2) \Rightarrow \neg P_c(c_2) \,\&\, P_c(c_3) \,\&\, P_q(q_{ca});$$

$$T_{c3}: P_c(c_3) \Rightarrow \neg P_c(c_3) \,\&\, P_c(c_0);$$

– продукции для агента D:

$$T_{d0}: P_d(d_0) \Rightarrow \neg P_d(d_0) \,\&\, P_d(d_1) \,\&\, P_r(r_{da});$$

$$T_{d1}: P_d(d_1) \,\&\, P_s(s_{ad}) \Rightarrow \neg P_d(d_1) \,\&\, P_d(d_2) \,\&\, \neg P_s(s_{ad});$$

$$T_{d2}: P_d(d_2) \Rightarrow \neg P_d(d_2) \,\&\, P_d(d_3) \,\&\, P_q(q_{da});$$

$$T_{d3}: P_d(d_3) \Rightarrow \neg P_d(d_3) \,\&\, P_d(d_0);$$

– продукции для агента E:

$$T_{e0}: P_e(e_0) \Rightarrow \neg P_e(e_0) \,\&\, P_e(e_1) \,\&\, P_r(r_{ea});$$

$$T_{e1}: P_e(e_1) \,\&\, P_s(s_{ae}) \Rightarrow \neg P_e(e_1) \,\&\, P_e(e_2) \,\&\, \neg P_s(s_{ae});$$

$$T_{e2}: P_e(e_2) \Rightarrow \neg P_e(e_2) \,\&\, P_e(e_3) \,\&\, P_q(q_{ea});$$

$$T_{e3}: P_e(e_3) \Rightarrow \neg P_e(e_3) \,\&\, P_e(e_0).$$

Следующей системой продукций для асинхронной предикатной сети MCS_3 задается процесс логического управления передачей данных от агента *A* по конвейеру последовательно остальным агентам *B*, *C*, *D*, *E* с возвратом результатов последовательно так же по конвейеру от агентов *E*, *D*, *C*, *B* агенту *A*. Введены предикаты P_x, $x \in \{a,b,c,d,e\}$, характеризующие состояния одноименных процессов для мобильных агентов *A*, *B*, *C*, *D*, *E* соответственно. Истинность высказывания $P_x(x_i)$ означает, что агент *x* находится в состоянии x_i. Каждый процесс, реализуемый агентом *x*, может находиться в одном из трех состояний: x_1 (начальная фаза), x_2 (рабочая фаза) и x_3 (заключительная фаза). Посредством предиката P_s организуется конвейерная связь между процессами в направлении от агента *A* последовательно к остальным агентам *B*, *C*, *D* и *E*. Обратная конвейерная связь организована посредством предиката P_r. Такая организация удобна для реализации конвейерного обмена данными при выполнении глобальных вычислительных операций в метакомпьютерах. Начальное состояние сети задают следующие начальные факты (истинные атомарные константные формулы):

$P_a(a_1)$, $P_b(b_1)$, $P_c(c_1)$, $P_d(d_1)$ и $P_e(e_1)$. Система продукционных выражений для асинхронной предикатной сети MCS_3 имеет следующий вид:

MCS_3:

– продукции для агента *A*:

$$T_{a1}: P_a(a_1) \Rightarrow \neg P_a(a_1) \,\&\, P_a(a_2) \,\&\, P_s(s_{ab});$$
$$T_{a2}: P_r(r_{ba}) \,\&\, P_a(a_2) \Rightarrow \neg P_r(r_{ba}) \,\&\, \neg P_a(a_2) \,\&\, P_a(a_3);$$
$$T_{a3}: P_a(a_3) \Rightarrow \neg P_a(a_3) \,\&\, P_a(a_1);$$

– продукции для агента *B*:

$$T_{b1}: P_s(s_{ab}) \,\&\, P_b(b_1) \Rightarrow \neg P_s(s_{ab}) \,\&\, \neg P_b(b_1) \,\&\, P_b(b_2) \,\&\, \neg P_s(s_{bc});$$
$$T_{b2}: P_r(r_{cb}) \,\&\, P_b(b_2) \Rightarrow \neg P_r(r_{cb}) \,\&\, \neg P_b(b_2) \,\&\, P_b(b_3) \,\&\, P_r(r_{ba});$$
$$T_{b3}: P_b(b_3) \Rightarrow \neg P_b(b_3) \,\&\, P_b(b_1);$$

– продукции для агента *C*:

$$T_{c1}: P_s(s_{bc}) \,\&\, P_c(c_1) \Rightarrow \neg P_s(s_{bc}) \,\&\, \neg P_c(c_1) \,\&\, P_c(c_2) \,\&\, \neg P_s(s_{cd});$$
$$T_{c2}: P_r(r_{dc}) \,\&\, P_c(c_2) \Rightarrow \neg P_r(r_{dc}) \,\&\, \neg P_c(c_2) \,\&\, P_c(c_3) \,\&\, P_r(r_{cb});$$
$$T_{c3}: P_c(c_3) \Rightarrow \neg P_c(c_3) \,\&\, P_c(c_1);$$

– продукции для агента *D*:

$$T_{d1}: P_s(s_{cd}) \,\&\, P_d(d_1) \Rightarrow \neg P_s(s_{cd}) \,\&\, \neg P_d(d_1) \,\&\, P_d(d_2) \,\&\, \neg P_s(s_{de});$$
$$T_{d2}: P_r(r_{ed}) \,\&\, P_d(d_2) \Rightarrow \neg P_r(r_{ed}) \,\&\, \neg P_d(d_2) \,\&\, P_d(d_3) \,\&\, P_r(r_{dc});$$
$$T_{d3}: P_d(d_3) \Rightarrow \neg P_d(d_3) \,\&\, P_d(d_1);$$

– продукции для агента *E*:

$$T_{e1}: P_s(s_{de}) \,\&\, P_e(e_1) \Rightarrow \neg P_s(s_{de}) \,\&\, \neg P_e(e_1) \,\&\, P_e(e_2);$$
$$T_{e2}: P_e(e_2) \Rightarrow \neg P_e(e_2) \,\&\, P_e(e_3) \,\&\, P_r(r_{ed});$$
$$T_{e3}: P_e(e_3) \Rightarrow \neg P_e(e_3) \,\&\, P_e(e_1).$$

Системой продукций для асинхронной предикатной сети MCS_4 задается процесс логического управления обменом данных от каждого агента каждому, причем здесь при обмене не выделяется главный агент. Данная система продукций соответствует схеме обменов, представленной на рис. 2*б*. Каждый агент здесь может находиться в одном из четырех состояний: x_0 (фаза широковещательной рассылки сообщений от каждого из агентов всем остальным агентам), x_1 (фаза приема сообщений каждым из агентов от остальных агентов), x_2 (рабочая фаза) и x_3 (заключительная фаза). Введены предикаты P_x, $x \in \{a,b,c,d,e\}$, характеризующие состояния одноименных процессов для мобильных агентов *A*, *B*, *C*, *D*, *E* соответственно. Система продукций для асинхронной предикатной сети MCS_4 имеет следующий вид:

MCS_4:

– продукции для агента *A*:

$$T_{a0}: P_a(a_0) \,\&\, \neg P_a(a_{ab}) \,\&\, \neg P_a(a_{ac}) \,\&\, \neg P_a(a_{ad}) \,\&\, \neg P_a(a_{ae}) \Rightarrow \neg P_a(a_0) \,\&\, P_a(a_1) \,\&\, P_a(a_{ab}) \,\&\, P_a(a_{ac}) \,\&\, P_a(a_{ad}) \,\&\, P_a(a_{ae});$$
$$T_{a1}: P_a(a_1) \,\&\, P_b(b_{ba}) \,\&\, P_c(c_{ca}) \,\&\, P_d(d_{da}) \,\&\, P_e(e_{ea}) \Rightarrow \neg P_a(a_1) \,\&\, P_a(a_2) \,\&\, \neg P_b(b_{ba}) \,\&\, \neg P_c(c_{ca}) \,\&\, \neg P_d(d_{da}) \,\&\, \neg P_e(e_{ea});$$
$$T_{a2}: P_a(a_2) \Rightarrow \neg P_a(a_2) \,\&\, P_a(a_3);$$
$$T_{a3}: P_a(a_3) \Rightarrow \neg P_a(a_3) \,\&\, P_a(a_0);$$

– продукции для агента *B*:

$$T_{b0}: P_b(b_0) \,\&\, \neg P_b(b_{ba}) \,\&\, \neg P_b(b_{bc}) \,\&\, \neg P_b(b_{bd}) \,\&\, \neg P_b(b_{be}) \Rightarrow \neg P_b(b_0) \,\&\, P_b(b_1) \,\&\, P_b(b_{ba}) \,\&\, P_b(b_{bc}) \,\&\, P_b(b_{bd}) \,\&\, P_b(b_{be});$$
$$T_{b1}: P_b(b_1) \,\&\, P_a(a_{ab}) \,\&\, P_c(c_{cb}) \,\&\, P_d(d_{db}) \,\&\, P_e(e_{eb}) \Rightarrow \neg P_b(b_1) \,\&\, P_b(b_2) \,\&\, \neg P_a(a_{ab}) \,\&\, \neg P_c(c_{cb}) \,\&\, \neg P_d(d_{db}) \,\&\, \neg P_e(e_{eb});$$
$$T_{b2}: P_b(b_2) \Rightarrow \neg P_b(b_2) \,\&\, P_b(b_3);$$
$$T_{b3}: P_b(b_3) \Rightarrow \neg P_b(b_3) \,\&\, P_b(b_0);$$

– продукции для агента *C*:

$$T_{c0}: P_c(c_0) \,\&\, \neg P_c(c_{ca}) \,\&\, \neg P_c(c_{cb}) \,\&\, \neg P_c(c_{cd}) \,\&\, \neg P_c(c_{ce}) \Rightarrow \neg P_c(c_0) \,\&\, P_c(c_1) \,\&\, P_c(c_{ca}) \,\&\, P_c(c_{cb}) \,\&\, P_c(c_{cd}) \,\&\, P_c(c_{ce});$$
$$T_{c1}: P_c(c_1) \,\&\, P_a(a_{ac}) \,\&\, P_b(b_{bc}) \,\&\, P_d(d_{dc}) \,\&\, P_e(e_{ec}) \Rightarrow \neg P_c(c_1) \,\&\, P_c(c_2) \,\&\, \neg P_a(a_{ac}) \,\&\, \neg P_b(b_{bc}) \,\&\, \neg P_d(d_{dc}) \,\&\, \neg P_e(e_{ec});$$
$$T_{c2}: P_c(c_2) \Rightarrow \neg P_c(c_2) \,\&\, P_c(c_3);$$
$$T_{c3}: P_c(c_3) \Rightarrow \neg P_c(c_3) \,\&\, P_c(c_0);$$

– продукции для агента *D*:

$$T_{d0}: P_d(d_0) \,\&\, \neg P_d(d_{da}) \,\&\, \neg P_d(d_{db}) \,\&\, \neg P_d(d_{dc}) \,\&\, \neg P_d(d_{de}) \Rightarrow \neg P_d(d_0) \,\&\, P_d(d_1) \,\&\, P_d(d_{da}) \,\&\, P_d(d_{db}) \,\&\, P_d(d_{dc}) \,\&\, P_d(d_{de});$$

$$T_{d1}: P_d(d_1) \,\&\, P_a(a_{ad}) \,\&\, P_b(b_{bd}) \,\&\, P_c(c_{cd}) \,\&\, P_e(e_{ed}) \Rightarrow \neg P_d(d_1) \,\&\, P_d(d_2) \,\&\, \neg P_a(a_{ad}) \,\&\, \neg P_b(b_{bd}) \,\&\, \neg P_c(c_{cd}) \,\&\, \neg P_e(e_{ed});$$

$$T_{d2}: P_d(d_2) \Rightarrow \neg P_d(d_2) \,\&\, P_d(d_3);$$

$$T_{d3}: P_d(d_3) \Rightarrow \neg P_d(d_3) \,\&\, P_d(d_0);$$

– продукции для агента E:

$$T_{e0}: P_e(e_0) \,\&\, \neg P_e(e_{ea}) \,\&\, \neg P_e(e_{eb}) \,\&\, \neg P_e(e_{ec}) \,\&\, \neg P_e(e_{ed}) \Rightarrow \neg P_e(e_0) \,\&\, P_e(e_1) \,\&\, P_e(e_{ea}) \,\&\, P_e(e_{eb}) \,\&\, P_e(e_{ec}) \,\&\, P_e(e_{ed});$$

$$T_{e1}: P_e(e_1) \,\&\, P_a(a_{ae}) \,\&\, P_b(b_{be}) \,\&\, P_c(c_{ce}) \,\&\, P_d(d_{de}) \Rightarrow \neg P_e(e_1) \,\&\, P_e(e_2) \,\&\, \neg P_a(a_{ae}) \,\&\, \neg P_b(b_{be}) \,\&\, \neg P_c(c_{ce}) \,\&\, \neg P_d(d_{de});$$

$$T_{e2}: P_e(e_2) \Rightarrow \neg P_e(e_2) \,\&\, P_e(e_3);$$

$$T_{e3}: P_e(e_3) \Rightarrow \neg P_e(e_3) \,\&\, P_e(e_0).$$

Начальное состояние сети задают следующие начальные факты (истинные атомарные константные формулы): $P_a(a_0)$, $P_b(b_0)$, $P_c(c_0)$, $P_d(d_0)$ и $P_e(e_0)$. Как и остальные асинхронные предикатные сети, данная сеть функционирует, модифицируя предикаты. Каждая модификация сопровождается изменением состояния метакомпьютерной системы и может сопровождаться передачей сообщений между агентами.

О реализации интерфейса передачи сообщений в агентно-ориентированных метакомпьютерных системах

Непосредственным приложением разработанных в предыдущем разделе сетей MCS_1, MCS_2, MCS_3, MCS_4 является их использование в реализациях различного рода интерфейсов передачи сообщений, например типа спецификации MPI (Message Passing Interface).

Спецификация MPI основана на модели передачи сообщений [7-10]. Данная спецификация (обновленная версия – MPI-2) представляет значительный интерес, поскольку она стала первым стандартом систем передачи сообщений, используемых в области параллельного программирования. Модель программирования MPI нередко называют моделью MPMD (Multiple Program – Multiple Data: множественный поток программ – множественный поток данных). Локальные и неструктурированные коммуникации организуются здесь с помощью двухточечных обменов, а при помощи коллективных обменов организуются глобальные операции.

Спецификация MPI, однако, не использовалась в агентно-ориентированном программировании, поскольку здесь применяются другие механизмы реализации параллельной и распределенной обработки данных. В силу того, что спецификация MPI важна для организации метакомпьютерных и облачных вычислений, целесообразно рассмотреть особенности ее асинхронной реализации для агентно-ориентированных систем. Об актуальности решения данной задачи свидетельствуют сведения о реализации интерфейса MPI поверх системы Grid в работах [11-12] на базе других технологий.

Известные схемы коллективных обменов и глобальных вычислительных операций [9] иллюстрируют рис. 3 и рис. 4 (вместо имен процессов проставлены имена агентов, реализующих данные процессы). Здесь A, B, C, D, E – имена агентов, представляющих логические узлы метакомпьютерной системы, а A_0, A_1, …, A_4, B_0, B_1, …, B_4, C_0, C_1, …, C_4, D_0, D_1, …, D_4, E_0, E_1, …, E_4 – передаваемые данные. Рассмотренные в предыдущем подразделе спецификации логического управления глобальными вычислительными процессами в системах метакомпьютерного типа позволяют реализовать все схемы коллективных обменов и глобальных вычислительных операций, ранее примененных в интерфейсе MPI. В том числе возможно и выполнение широковещательных передач, при которых один из процессов передает данные всем процессам, обменов с барьерами, когда обмен сообщениями происходит после того, как синхронизирована работа нескольких процессов и операций приведения, при реализации которых происходит выполнение глобальных вычислительных операций.

В данном разделе с учетом предикатных спецификаций предыдущего параграфа предлагается реализовывать коллективные передачи и глобальные вычислительные операции на основе модели согласования процессов через информационное пространство, а не на основе модели передачи сообщений. Подобный подход отличается лучшей согласованностью информационной и процедурной компонент модели представления

знаний, используемой непосредственно для создания управляющих программ.

Системой продукций для асинхронной предикатной сети MCS_1 задается процесс логического управления в метакомпьютерной системе при передаче данных от агента A каждому из остальных агентов B, C, D, E с подтверждением приема и передачей результатов от агентов B, C, D, E агенту A. Очевидно, что все схемы выполнения коллективных операций, представленные на рис. 3, реализуемы на базе схемы согласования процессов MCS_1. Для реализации последних двух коллективных *allgather* и *alltoall* обменов необходимо последовательно инициировать работу независимых сетей MCS_{1A}, MCS_{1B}, MCS_{1C}, MCS_{1D}, MCS_{1E}, в которых главными являются агенты A,B, C, D, E соответственно.

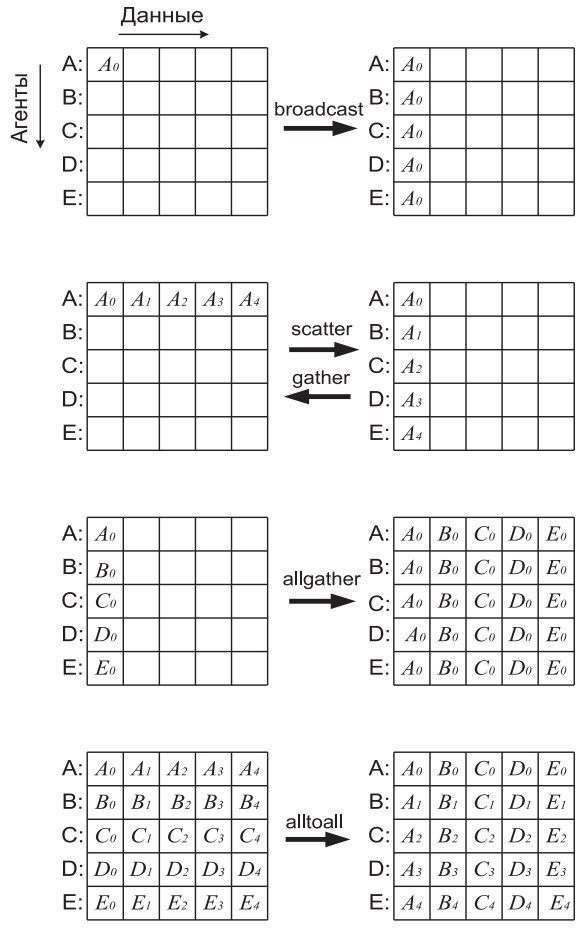

Рис. 3. Коллективные операции для группы агентов (диаграммы построены на основе функций MPI [9])

Системой продукций для асинхронной предикатной сети MCS_2 описывается процесс обмена данными между агентами по принципу «каждый с каждым» (через агент A) с возвратом результатов (или квитанций) агенту A. Для реализации последних двух коллективных *allgather* и *alltoall* обменов необходимо инициировать работу одной сети MCS_2.

Системой продукций для асинхронной предикатной сети MCS_3 задается процесс логического управления передачей данных от агента A по конвейеру последовательно остальным агентам B, C, D, E с возвратом результатов последовательно также по конвейеру от агентов E, D, C, B агенту A. Подобная сеть эффективно реализует логическое управление выполнением глобальных вычислительных операций, схемы выполнения которых представлены на рис. 4.

Системой продукций для асинхронной предикатной сети MCS_4 задается процесс логического управления обменом данных от каждого агента каждому, причем здесь при обмене не выделяется главный агент. Такая организация коллективного обмена данными особенно удобна при реализации операций allgather и alltoall.

Развитие аппарата асинхронных предикатных сетей – асинхронные предикатно-функциональные сети

В основу языка абстрактного описания узлов, или модулей, асинхронных предикатно-функциональных сетей (АПФС) положен язык многосортного исчисления предикатов первого порядка, расширенный правилами выборки и обновления кортежей информационного пространства, а также язык систем алгоритмических алгебр Глушкова [13 - 15]. Необходимость развития формализма сетей АПС до АПФС, например, связана с формализацией обработки данных в агентно-ориентированных метакомпьютерных системах.

При определении АПС И АПФС используется понятие блока совместимых (непротиворечивых) обновлений с последовательным выполнением правил обновлений предикатов и функций. В формальной записи блоки ограничиваются фигурными скобками. При построении сетей АПФС на основе узлов, или модулей, нами будут использоваться обычные для параллельного и распределенного программирования бинарные операции последовательного, параллельного и конкурентного выполнения модулей.

Бинарные темпоральные операции « ; », « , », « : », « || », « |c» предписывают различные способы выполнения модулей. Операция « ; » предписывает последовательное выполнение модулей, из которых второй модуль может зависеть от первого (в продукционных правилах данная операция обозначается составным символом $\&_{\rightarrow}$). Операция « , » предписывает выполнение независимых модулей: последовательное в про-

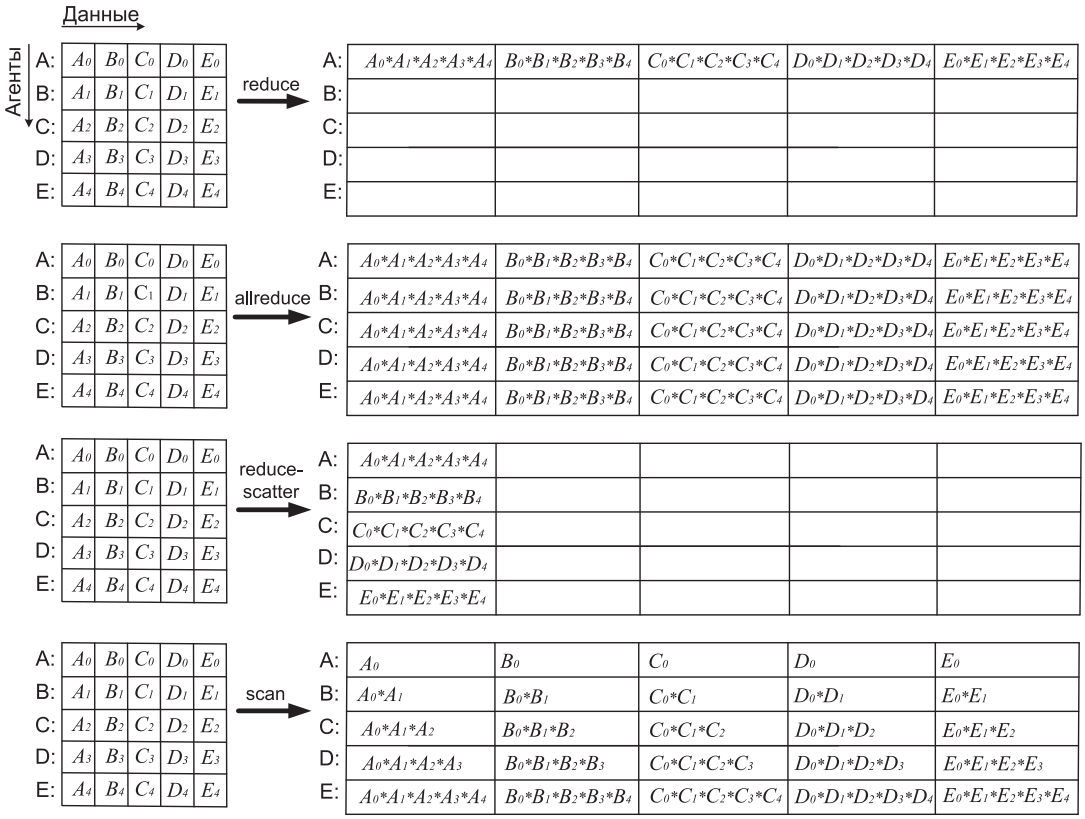

Рис. 4. Глобальные вычислительные операции для группы агентов (диаграммы построены на основе функций MPI [9])

извольном порядке или параллельное (в продукционных правилах данная операция обозначается составным символом $\&_{=}$).

Операция « : » предписывает непараллельное выполнение модулей в произвольном порядке (в продукционных правилах данная операция обозначается составными символами $\&_{-}$ или $\&_{\leftrightarrow}$).

Операция « || » предписывает модулям причинно-следственную связь по крайней мере через единственный предикат либо функцию; при программной реализации применение данной операции требует дальнейшей детализации описания через операции « ; », « , » « : ».

В продукционных правилах вместо символа обновления функции или предиката «←» может использоваться обычный символ присваивания « :=». Операция « $|^{c}$ » указывает на возможное конкурентное выполнение модулей, например использующих разделяемый ресурс; непосредственно при программировании данная операция не используется – ее применение требует дальнейшей детализации описания при помощи других операций, как и в случае операции « || ».

Алгебраические свойства реализуемых операций описаны в работах [16-17]. Кроме того, некоторые алгебраические свойства операций очевидны – например, операции « , » « : » коммутативны и ассоциативны, а операция « ; » некоммутативна и ассоциативна.

В формульной записи имена модулей, сгруппированных в блоки, заключаются в фигурные скобки, а внутри блоков могут использоваться простые скобки для указания на последовательность выполнения операций, например:

$$\{m_1, m_2, m_3\}, \{m_1, m_2\}, \{(m_1, m_2), m_3\},$$
$$\{(m_1, m_2), m_3\}.$$

В случае когда блок содержит лишь один модуль, скобки можно опускать. В настоящей работе алгебра алгоритмов Глушкова используется в основном для записи обычных структурированных (дейкстровских) конструкций – «последовательность операторов», «ветвление», «цикл»; в принципе, здесь могла бы использоваться любая другая нотация, так же пригодная для записи структурированных программ. Все указанные выше операции мы также включаем в состав операций алгебры модулей, что облегчит формирование новых модулей сетей АПФС. Из алгебры операторов мы выбираем тернарную операцию α-дизъюнкцию и бинарную операцию α-итерацию как основы для формирования модулей АПФС (сети, при описании которых не

используется операция α-итерации, в работах [18-19] принято называть сетями абстрактных машин (СеАМ), а для сетей, при описании которых используется данный оператор, выбрано название «расширенных» СеАМ, то есть РСеАМ). Следуя [13-15], напомним правила выполнения данных операций. При выполнении операции α-дизъюнкции $[\alpha](m_1 \vee m_2)$ при $\alpha = true$ выполняется модуль m_1, а при $\alpha = false$ выполняется модуль m_2. При реализации операции α-итерации [α]{m} модуль m выполняется циклически, пока $\alpha = false$, а при $\alpha = true$ происходит выход из цикла (следует отличать итерационные фигурные скобки от блочных). В сетях СеАМ и РСеАМ вместо имен модулей m_1, m_2 и m в указанные выражения могут подставляться подформулы с символами любых из определенных выше операторов, например возможно построение следующих выражений для модулей:

$$m_6 = [\alpha](\{m_1, m_2, m_3\} \vee \{(m_3, m_4), m_5\}),$$
$$m_7 = [\alpha_1](([\alpha_2]\{[\alpha_3](m_1 \vee m_2)\}) \vee$$
$$\{[\alpha_4](m_3 \vee m_4)\}).$$

Элементарный модуль содержит единственное правило обновления предиката или функции. Пустое обновление R^E эквивалентно тождественному оператору E алгебры алгоритмов Глушкова, не выполняющему никаких действий по модификации информационного пространства. Неопределенное обновление R^N соответствует неопределенному оператору N. Продукционному программированию, применяемому в сетях АПС и АПФС, соответствует использование модулей СеАМ (модулей-продукций) следующего ограниченного вида $m = [\alpha](L \vee R^E)$, что эквивалентно записи $m : \alpha \Rightarrow L$, где L – непустая последовательность элементарных или составных модулей. При определенных очевидных условиях данный модуль может выполняться так же, как и модуль РСеАМ, описываемый выражением $[\neg\alpha]\{L\}$.

Поскольку в общем случае не все составные модули непосредственно (без дополнительных преобразований или без введения дополнительных условий) допускают аналитическую запись в виде суперпозиций операций, при составлении сосредоточенных и распределенных программ рекомендуется использовать модули, допускающие аналитическое описание (то есть структурированные модули), а связи между ними организовывать посредством модификации и проверки значений функций и предикатов, составляющих информационное пространство. Такие связи называются причинно-следственными, или каузальными. Подобный подход позволяет использовать при распределенном программировании модули различного вида – от простых модулей-продукций до более сложных структурированных модулей.

Алгебра модулей сетей СеАМ, подобно системам алгоритмических алгебр, имеет систему образующих – элементарные модули и элементарные логические условия. Условиями называются замкнутые (не содержащие свободных вхождений предметных переменных) логические формулы с предикатными символами в качестве логических переменных. Множество используемых предикатных символов включает символы, используемые при формировании информационного пространства, а также символы стационарных, или немодифицируемых, предикатов сравнения. Используемые при сравнениях термы строятся по обычным в многоосновном исчислении предикатов первого порядка правилам (термами, или функциональными выражениями, называют слова, построенные из переменных, функциональных и специальных символов по определенным правилам). Среди элементарных логических условий важное значение имеют условия, формируемые на основе квантифицированных операторов выборки кортежей из отношений.

Определенные сети СеАМ являются в общем случае асинхронными недетерминированными системами ввиду произвольного порядка выбора на исполнение и неопределенного времени работы модулей, модифицирующих общее информационное пространство для получения полезного результата.

При проектировании систем и сетей хранения и обработки данных представляет значительный интерес построение выражений для модулей СеАМ как с использованием квантифицированных операторов выбора $\tilde{\exists}!$, $\tilde{\exists}!!$, $\tilde{\forall}$ и $\tilde{\forall}!!$, так и без них. Применение квантифицированных операторов, а также дополнительных «связывающих» предикатов может привести к существенному уменьшению необходимого числа применяемых модулей СеАМ и упрощению описывающих выражений.

Учитывая, что формализмы сетей абстрактных машин мы предлагаем использовать в качестве непосредственно интерпретируемых спецификаций при создании нового программного обеспечения систем хранения и обработки данных, следует рассмотреть различные формы записи α-условий в продукционных выражениях

АПФС. При выполнении оператора $\tilde{\exists}!$ из области истинности предиката, описываемого выражением справа, выбирается произвольный кортеж. При выполнении оператора $\tilde{\exists}!!$ выбирается единственный кортеж, находящийся в области истинности стоящего справа предиката. При выполнении оператора $\tilde{\forall}$ выбираются все кортежи, составляющие область истинности соответствующего предиката.

Оператор $\tilde{\forall}!!$ позволяет выбрать все кортежи из области истинности предиката в случае, если его область определения совпадает с его же областью истинности. Во всех случаях подразумевается, что предикат описывается выражением, стоящим справа от символа квантифицированного оператора. Описанные операторы не удаляют выбранные кортежи из областей истинности предикатов. Модификации предикатов и функций могут быть осуществлены далее при помощи описанных выше правил обновления, сгруппированных в блоки модулей абстрактных машин.

Каждому из описанных квантифицированных операторов выборки кортежей из отношений, в случае его использования в условной части выражения для модуля, ставится в соответствие элементарное логическое условие, истинное в случае успешного выполнения оператора и ложное в противном случае. Данный факт в [18-19] отмечался подчеркиванием оператора снизу. Поскольку такой способ образования элементарного логического условия применяется только при формировании условного выражения, заключенного в полном выражении для модуля в квадратные скобки, символ подчеркивания в квадратных скобках можно опускать.

Заключение

1. Рассмотрены области применения концепции агентно-ориентированного продукционно-процедурного программирования, основанного на методах логического управления вычислительными процессами и ресурсами. Решены проблемы декомпозиции, взаимодействия и синхронизации в коллективе агентов на основе согласования процессов и объектов, что позволяет усовершенствовать практические принципы построения агентно-ориентированных систем.

2. Предложены модели логического управления глобальными вычислительными процессами на основе интерфейса передачи сообщений в вычислительных сетях с агентно-ориентированными облачными метакомпьютерными сервисами, основанные на коллективных пересылках данных, что позволяет упростить реализацию массового параллелизма в крупномасштабных прикладных распределенных системах.

3. Даны предложения по организации межагентных взаимодействий в сетевой среде при реализации систем управления распределенными реляционными базами данных, что повышает эффективность метакомпьютерной реализации процессов обработки структурированных данных. Кроме того, продемонстрирована возможность применения модели согласования процессов и агентов при организации обработки структурированных данных.

Литература

1. Шалагинов А. В. CloudComputing – «облачные вычисления» // Технологии и средства связи. № 5, 2010. – С. 15-17.
2. Hewitt C. ORGs for scalable, robust, privacy-friendly client cloud computing // IEEE Internet Computing. Vol. 12, No. 5, 2005. – P. 96-99.
3. Dave P. Introduction to Cloud Computing. http://dotnetslackers.com/articles/sql/introduction-to-cloud-computing.aspx, свободный.
4. Облачные вычисления. http://ru.wikipedia. org/wiki /, свободный.
5. Зинкина Н.С. Методы и модели логического управления дискретными процессами в распределенных вычислительных системах на основе концепции согласования // Известия ВУЗов. Поволжский регион. Технические науки. №1, 2011. – С. 35-47.
6. Механов В.Б., Зинкин С.А., Карамышева Н.С. Формализация управления вычислительными процессами в распределенных системах хранения и обработки данных и знаний // Информационные технологии. №1, 2013. – С. 51-58.
7. Таненбаум Э., ВанСтеен М. Распределенные системы. Принципы и парадигмы. СПб.: Питер, 2003. – 877 с.
8. Хьюз К., Хьюз Т. Параллельное и распределенное программирование на C++. – М.: ИД «Вильямс», 2004. – 672 с.
9. Snir M., Otto S., Huss-Lederman S., Walker D., J. Dongara D. MPI: The complete reference. The MIT Press. Cambridge, Massachusetts, 1996. – 335 p.
10. Немнюгин С.А., Стесик О.Л. Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Петербург, 2002. – 400 с.
11. Karonis N.T., Toonen B., Foster I. MPICH-G2: A Grid-enabled implementation of the Message Passing Interface // Journal of Parallel and

Distributed Computing. Vol. 63, 2003. – P. 551-563.
12. PACX-MPI: The Grid-computing library PACX-MPI extending MPI for computational Grid http://www.hlrs.de/organization/av/amt/research/pacx-mpi/ , свободный.
13. Глушков В.М., Цейтлин Г.Е., Ющенко Е.Л. Методы символьной мультиобработки. Киев: Наукова думка, 1980. – 252 с.
14. Ющенко Е.Л., Цейтлин Г.Е., Грицай В.П., Терзян Т.К. Многоуровневое структурное проектирование программ. Теоретические основы, инструментарий. М.: Финансы и статистика, 1989. – 208 с.
15. Капитонова Ю.В., Летичевский А.А. Математическая теория проектирования вычислительных систем. М.: Наука, 1988. – 296 с.
16. Зинкин С.А. Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (базовый формализм и темпоральные операции // Известия ВУЗов. Поволжский регион. Технические науки. №1, 2007. – С. 3-12.
17. Зинкин С.А. Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (реализация и свойства сценарных моделей) // Известия ВУЗов. Поволжский регион. Технические науки. №2, 2007. – С. 13-21.
18. Зинкин С.А. Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (базовый формализм и его расширения) // Известия ВУЗов. Поволжский регион. Технические науки. №3, 2007. –С. 13-22.
19. Зинкин С.А. Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (механизмы интерпретации и варианты использования) // Известия ВУЗов. Поволжский регион. Технические науки. №4, 2007. – С. 37-51.

TOWARDS THE MESSAGE PASSING INTERFACE IMPLEMENTATION AS A CLOUD SERVICE IN AGENT-ORIENTED METACOMPUTING SYSTEMS

Vashkevich N.P., Zinkin S.A., Karamysheva N.S.

In this paper we propose the models of the logic of managing global computer processes in computer networks with agent-oriented metacomputing cloud services based on the collective data transfers, allowing you to simplify the implementation of massively parallel applications in large-scale distributed systems. Offered for logical control of discrete processes to use the formalism of predicate asynchronous networks, allows abstract and structural synthesis of the functional architecture of agent-based metacomputer with the implementation of global and collective computing operations for a group of agents – the virtual nodes of metacomputer.

Keywords: *metacomputer, cloud computing, the logical process management, collective and computing opera-tions.*

Вашкевич Николай Петрович, Заслуженный деятель науки и техники РФ, д.т.н., профессор Кафедры вычислительной техники (ВТ) Пензенского государственного университета (ПГУ). Тел. (8-412) 36-82-27. E-mail: vt@alice.pnzgu.ru

Зинкин Сергей Александрович, д.т.н., профессор Кафедры ВТ ПГУ. Тел. (8-412) 36-82-27. E-mail: zsa49@yandex.ru

Карамышева Надежда Сергеевна, к.т.н., инженер-программист Кафедры ВТ ПГУ. Тел. (8-412) 36-82-27. E-mail: kagliari@yandex.ru

УДК 004.492.4

ПОСТРОЕНИЕ ОПТИМАЛЬНОГО СПАМ-ФИЛЬТРА НА ОСНОВЕ СОВМЕЩЕНИЯ СТАТИСТИЧЕСКИХ КЛАССИФИКАТОРОВ

Тарасов В.Н., Мезенцева Е.М.

В статье представлены критерии оптимальности для классификации сообщений на основе статистических методов с учетом ошибок первого и второго родов, доли ложных срабатываний и пропущенного спама. Приведены особенности тестирования и обучения спам-фильтра. Предложен подход к организации классификатора, который заключается в совместном использовании методов Байеса и Фишера. Для повышения качества фильтрации предложено использовать механизм анализа подмножеств пере-