УДК 510.6
DOI: 10.46548/21vek-2020-0951-0002

АЛГОРИТМЫ ВЫЧИСЛЕНИЯ ПРЕДИКАТОВ НА МНОЖЕСТВЕ ГЛОБАЛЬНЫХ СОСТОЯНИЙ В СИСТЕМАХ ОТЛАДКИ РАСПРЕДЕЛЕННЫХ ПРИЛОЖЕНИЙ

Перекусихина Альбина Николаевна, кандидат технических наук,
доцент кафедры «Математика и математическое моделирование»
Пензенский государственный университет архитектуры и строительства
(440028, Россия, Пенза, ул. Германа Титова, д. 28)
Прошин Иван Александрович, доктор технических наук, доцент, архитектор
АО «Бэлл интегратор»
(440000, Россия, Пенза, ул. Московская, 27, e-mail: proshin.Ivan@inbox.ru)
Мартышкин Алексей Иванович, кандидат технических наук, доцент,
доцент кафедры «Вычислительные машины и системы»
Истомина Татьяна Викторовна, доктор технических наук, профессор,
ведущий научный сотрудник отдела научных исследований
Пензенский государственный технологический университет
(440039, Россия, Пенза, проезд Байдукова/ул. Гагарина, д. 1а/11, e-mails: alexey314@yandex.ru, istom@mail.ru)

Аннотация. Предлагаемая статья посвящена исследованию алгоритмов вычисления предикатов на множестве глобальных состояний в системах отладки распределенных приложений. Важнейшей задачей при отладке приложения является проверка выполнения заданного предиката на множестве переменных, характеризующих состояние этого приложения. Для распределенных систем это является проблемой, так как для них характерна неопределенность в оценке текущего глобального состояния выполнения. Хотя данная проблема хорошо изучена и разработаны алгоритмы ее эффективного разрешения, остается неосвещенным ряд вопросов, связанных с разработкой алгоритмов и программы, способных работать с уже существующими средствами для отладки распределенных систем и при этом не требующих внесения существенных изменений в архитектуру и реализацию таких решений. Настоящая статья посвящена разработке алгоритмов для последующей их реализации в программе, позволяющей проверить выполнимость предикатов, заданных пользователем, основываясь на протоколе событий выполнения распределенной системы. Рассмотрены алгоритмы вычисления выполнимости предиката с модальностью «возможно» и с модальностью «определенно» на множестве глобальных состояний Рассматриваются базовые теоретические вопросы, связанные с распределенными системами, описываются проблемы, приводятся некоторые варианты их решения. В заключение представлены основные выводы по проведенным исследованиям.

Ключевые слова: глобальное состояние, модальность «возможно», модальность «определенно», предикат, программное обеспечение, производительность, пространственно-временная диаграмма выполнения, протокол событий, распределенная система, векторная метка времени.

ALGORITHMS FOR CALCULATING PREDICATES ON A SET OF GLOBAL STATES IN DISTRIBUTED APPLICATION DEBUGGING SYSTEMS

Perekusikhina Albina Nikolaevna, candidate of technical sciences, associate Professor of sub-department
«Mathematics and Mathematical Modeling»
Penza State University of Architecture and Construction
(440028, Russia, Penza, German Titov Street, 28)
Proshin Ivan Aleksandrovich, doctor of technical Sciences, associate Professor, creator
Joint-stock company «Bell integrator»
(440000, Russia, Penza, Moskovskaya street, 27, e-mail: proshin.Ivan@inbox.ru)
Martyshkin Alexey Ivanovich, candidate of technical sciences, docent,
associate Professor of sub-department «Computers and systems»
Istomina Tatyana Viktorovna, doctor of technical Sciences, Professor,
leading researcher of the Department of scientific research
Penza state technological University
(440039, Russia, Penza, Baydukov Proyezd / Gagarin Street, 1a/11, e-mails: alexey314@yandex.ru, istom@mail.ru)

Abstract. This article is devoted to the study of algorithms for calculating predicates on a set of global States in debugging systems for distributed applications. The most important task when debugging an application is to check the execution of a given predicate on a set of variables that characterize the state of this application. This is a problem for distributed systems, since they are characterized by uncertainty in the assessment of the current global state of execution. Although this problem has been well studied and algorithms for its effective resolution have been developed, a number of issues related to the development of algorithms and programs that can work with existing tools for debugging

distributed systems and do not require significant changes to the architecture and implementation of such solutions remain unresolved. This article is devoted to the development of algorithms for their subsequent implementation in a program that allows you to check the feasibility of predicates specified by the user, based on the Protocol of execution events of a distributed system. Algorithms for calculating the satisfiability of a predicate with the modality "possible" and with the modality "definite" on a set of global States are considered The basic theoretical questions related to distributed systems are considered, problems are described, and some solutions are given. In conclusion, the main conclusions of the research are presented.

Keywords: global state, "possible" modality, "definitely" modality, predicate, software, performance, space-time execution diagram, event Protocol, distributed system, vector timestamp.

Введение. Локальные сети (*LAN*) связывают множество компьютеров, находящихся в здании, таким образом, что машины в состоянии обмениваться небольшими порциями информации за несколько микросекунд [1]. Важнейшей задачей при отладке приложений является проверка того, что на множестве переменных, характеризующих состояние этого приложения, выполняется заданный предикат [2]. Для распределенных систем (РС) это является проблемой, т.к. для них характерна неопределенность в оценке текущего глобального состояния выполнения [3].

Обозначенная проблема достаточно изучена [4-8], [9-12] и [13-16], разработаны алгоритмы, позволяющие ее эффективно решать [17-19]. Интерес представляет разработка алгоритмов и в дальнейшем программы, которая сможет работать с уже существующими решениями для отладки РС и при этом не потребует внесения существенных изменений в архитектуру и реализацию таких решений. С аппаратной точки зрения РС – совокупность взаимосвязанных автономных компьютеров или процессов. Причем особую важность представляет именно их автономность. С программной точки зрения РС – совокупность независимых процессов, которые взаимодействуют посредством передачи сообщений для обмена данными и координации своих действий.

Из этих определений вытекают следующие отмеченные аспекты задачи:

1. Автономность узлов невозможна без независимого управления для каждого из них. Отсюда следует, что РС не может быть представлена ЭВМ с *SIMD* архитектурой.

2. Каждый процесс имеет свое собственное состояние, которое напрямую не зависит от состояния других процессов и может быть изменено только им самим. Единственный способ для процесса повлиять на внутреннее состояние другого процесса – отправка сообщения. Состояние процесса закрыто для других процессов.

Материалы и результаты исследования. Пример архитектуры конкретной РС приведен на рисунке 1.

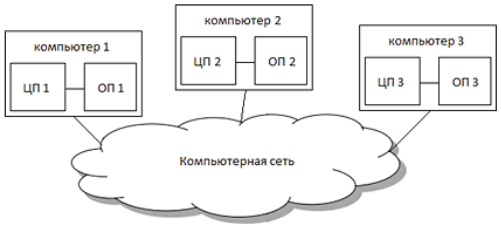

Рисунок 1 – Пример архитектуры распределенной сети

Пространственно-временная диаграмма выполнения РС приведена на рисунке 2.

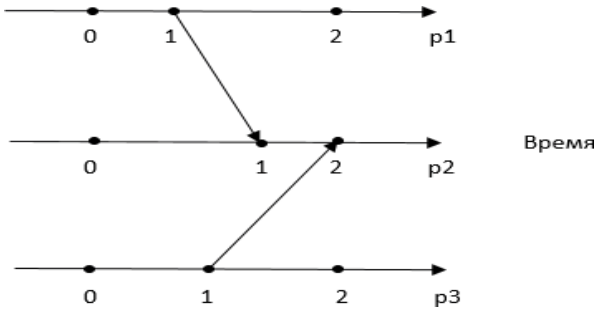

Рисунок 2 – Пространственно-временная диаграмма выполнения РС

Рисунок 2 является пространственно-временной диаграммой выполнения. Прямыми горизонтальными линиями показаны процессы выполнения, здесь отображены три процесса с порядковыми номерами 1, 2 и 3. Для определения времени в РС используется несколько методов. Здесь время определяется с помощью векторных меток времени, под которыми понимается вектор положительных целых чисел длинной n, где n – количество процессов в РС. Компонента вектора с номером i определяет количество событий, которое произошло в процессе с номером i до текущего момента времени. Изначально система находится в состоянии, когда все компоненты вектора времени равны 0. В рамках настоящего представления векторные метки времени используются в двух случаях. В первом случае подразумеваются события выполнения, во втором случае – решетка глобальных состояний.

В первом случае метка времени ставится в соответствие одному событию и, следовательно, можно говорить о процессе, которому соответствует данная метка времени. Компонента вектора, соответствующая процессу, в котором произошло рассматриваемое событие, означает число событий, уже произошедших до наступления этого события. Остальные компоненты вектора времени означают число событий других процессов, о которых данное событие знает, то есть с которыми у него есть причинно-следственная связь.

Во втором случае метка времени ставится в соответствие одному узлу решетки возможных глобальных состояний выполнения РС. Тогда каждая компонента вектора времени означает количество событий, произошедших в соответствующем процессе, чтобы наступило данное глобальное состояние. Также в этом случае имеет смысл ввести термин «уровень»,

как сумму всех компонент вектора времени. Смысл уровня заключается в том, что его значение отображает количество событий, которое должно произойти для того, чтобы РС пришла в соответствующее глобальное состояние.

Рассмотрим выполнение РС, представленное на рисунке 3.

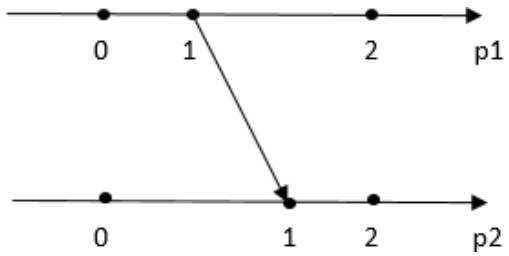

Рисунок 3 – Выполнение РС

Будем говорить, что событие *A* знает о событии *B*, если выполняется одно из следующих условий [20]:

• *A* и *B* – последовательные события одного процесса;

• *A* и *B* – события приема и отправки одного и того же сообщения, причем *A* – событие приема сообщения, а *B* – событие отправки этого сообщения;

Если *A* знает о *B*, значит между ними есть причинно-следственная связь, которую будем обозначать следующим образом: *B->A*.

Причинно-следственная связь обладает двумя важными свойствами.

1. Причинно-следственная связь транзитивна, то есть если *A→B* и *B→C*, то *A→C*.

2. Причинно-следственная связь не коммутативна, то есть если *A→B*, то это не значит, что *B→A*.

Рассмотрим пример выполнения РС, приведенный на рисунке 3.

Событие 1 второго процесса знает о событии 1 первого процесса, т.к. это события отправки и получения одного и того же сообщения.

Событие 2 второго процесса знает о событии 0 первого по свойству транзитивности причинно-следственного порядка, т.к. события 1 первого и второго процессов связаны и все последовательные события внутри процессов тоже.

Два события *A* и *B* называются независимыми если (*!A→B* and *!B→A*). Для примера выполнения, представленного на рисунке 3, независимыми являются события 0 для первого процесса и 0 для второго, а также события 2 для первого процесса и событие 0 для второго. Неопределенность в работе с РС заключается в том, что ни один процесс, входящий в РС, не может определить, в каком взаимном порядке в реальности исполнялись такие события. Более того, при каждом новом запуске РС взаимный порядок таких событий может меняться. Например, для выполнения РС, представленного на рисунке 3, нельзя с определенностью сказать в какой последовательности исполнились события с номером 0 первого и второго процессов. Возможны запуски РС, когда первым произойдет событие из первого процесса, не менее вероятен вариант, когда первым происходит событие второго процесса.

Данная проблема неопределенности исполнения РС подробно описана в работе *«Consistent global states of distributed systems: Fundamental concepts and mechanisms»* [17]. Именно эта конкретная неопределенность и ведет к тому, что в РС нет возможности просто вычислить предикат для выполнения.

Множество глобальных состояний выполнения РС. Неопределенность выполнения РС ведет к тому, что при двух запусках РС может проходить через различные глобальные состояния. Множество всех возможных глобальных состояний, через которые может проходить исполнение РС, удобно представлять в виде решетки, каждый узел которой – одно возможное глобальное состояние, характеризующееся векторной меткой времени. Связь между узлами означает, что возможен переход из вершины с меньшим уровнем в вершину с уровнем, большим на 1. Пример выполнения РС и решетки возможных глобальных состояний для нее приведен на рисунке 4.

В этом примере события 0 первого и второго процессов являются метафорой начального состояния процессов, можно считать, что они уже произошли на старте выполнения РС. Как видно из решетки, после события отправки сообщения из первого процесса возможны два варианта: может произойти событие приема сообщения во втором процессе, а может произойти событие 2 из первого процесса. Это пример неопределенности, которая возникает из-за того, что события 2 первого процесса и 1 второго процесса независимы. Также можно заметить, что в решетке нет глобального состояния, которое бы характеризовалось векторной меткой времени (0, 1). Это происходит потому, что событие 1 второго процесса знает о событии 1 первого процесса, то есть событие принятия сообщения не может произойти раньше события его отправки. Другими словами, наличие причинно-следственных связей уменьшает неопределенность, а соответственно и размер решетки глобальных состояний.

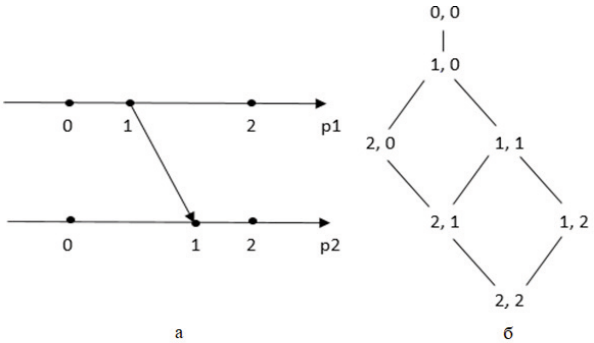

Рисунок 4 – Выполнение РС (а) и соответствующая решетка глобальных состояний (б)

При построении решетки может использоваться следующий алгоритм:

1) *current* = начальное глобальное состояние, векторная метка времени которого содержит нули для всех процессов выполнения РС;

2) *lattice* – пустое множество узловрешетки;

3) пока *current* непусто;

а. добавить в *lattice* все узлы из *current*;

б. *last =current*;

в. очистить *current*;

г. добавить в *current* все возможные допустимые узлы, достижимые из *last*;

4) *returnlattice*.

Узел *A* является достижимым из узла *B*, если:

временные метки *A* и *B* различаются только в одной компоненте, причем данная компонента в метке времени *A* на 1 больше, чем в метке времени *B*. Таким образом моделируется то, что между *A* и *B* произошло ровно одно событие.

Достижимый узел *A* является допустимым для узла *B*, если выполняется одно из следующих условий:

• событие, переводящее РС из состояния *B* в состояние *A*, является внутренним событием одного изпроцессов;

• событие, переводящее РС из состояния *B* в состояние *A*, является событием отправки сообщения;

• событие, переводящее РС из состояния *B* в состояние *A*, является событием приема сообщения, причем событие отправки этого сообщения уже произошло в состоянии *B* или ранее.

Использование данного алгоритма позволяет построить корректную решетку глобальных состояний для РС с асинхронным обменом сообщениями.

Вычисление предиката с модальностью «возможно». Предикат вычисляется с модальностью «возможно» для выполнения РС, если существует такой вариант исполнения РС, что он проходит хотя бы через одно глобальное состояние, где предикат реализуется.

Для вычисления предиката с модальностью можно использовать следующий алгоритм:

1) *current* = начальное глобальное состояние, все компоненты векторной метки времени которого равны *0: lvl = 0*, равно уровню решетки, который сейчас рассматривается

2) пока ни одно состояние из *current* не удовлетворяет предикату

а. *last =current*

б. *lvl = lvl +1*

в. *current* = все состояния уровня *lvl*, доступные из состояний, содержащихся в *last*

г. если *current* пусто, то *return* предикат не «возможен»

3) *return* предикат «возможен»

Данный алгоритм реализует обход в ширину решетки глобальных состояний выполнения РС и позволяет вычислить предикат с модальностью «возможно» для выполнения РС.

Вычисление предиката с модальностью «определенно». Предикат вычисляется с модальностью «определенно» для выполнения РС, если любое исполнение РС проходит хотя бы через одно глобальное состояние, где предикат реализуется. Вычисление предикатов с такой модальностью полезно, если мы хотим проверить, что для выполнения в принципе невозможно наступление какого-либо условия.

Описание данного алгоритма представлено ниже:

1) *last* = начальное глобальное состояние, все компоненты векторной метки времени которого равны 0. Данная переменная будет содержать все состояния уровня *lvl-1*, которые доступны из начального глобального состояния, по пути, которые не проходит через состояния, удовлетворяющие предикату;

2) удалить из *last* все глобальные состояния, которые удовлетворяют предикату;

3) *lvl* = 1, данная переменная равна уровню решетки, который сейчас рассматривается;

4) пока *last* не пусто;

а. если *last* содержит последнее глобальное состояние, тогда *return* не «определенно»;

б. *current* = все состояния уровня *lvl* доступные из состояний, содержащихся в *last*;

в. удалить все состояния из *current*, которые удовлетворяют предикату;

г. *lvl = lvl +1;*

д. *last =current;*

е. *return* «определенно».

Заключение. Актуальность задачи отладки распределенных приложений не вызывает сомнений. Проблема в том, что стандартные методы отладки в случае с распределенными системами работают плохо из-за возникающих проблем в определении текущего глобального состояния. В общем случае для распределенной системы нельзя однозначно ответить на вопрос, выполняется предикат во время работы программы или нет. Для решения данной задачи приходится рассматривать все множество возможных глобальных состояний системы и вычислять предикаты с различными модальностями. Показана возможность реализации алгоритмов проверки выполнимости предикатов на множестве глобальных состояний.

СПИСОК ЛИТЕРАТУРЫ:

1. Поляков Д.С., Смолко А.С. Проверка выполнимости предикатов на множестве глобальных состояний распределенной системы // Материалы 52-й Международной научной студенческой конференции МНСК-2014: Информационные технологии / Новосиб. гос. ун-т, 2014. – С. 130-131.

2. Таненбаум Э. Распределенные системы. Принципы и парадигмы / Э. Таненбаум, М. ван Стеен. – СПб.: Питер, 2003. – 877 с.

3. Ajay D. Distributed Computing: Principles, Algorithms, and Systems. / Ajay D., Kshemkalyani, Mukesh Singhal. – Cambridge University Press, 2008.

4. Храковский В.С. Семантические типы множества ситуаций (опытклассификации) // ИзвестияАНССР. Серия литературы и языка. Т. 45, №2, 1986. С. 149-158.

5. Зинкин, С.А. Самомодифицируемые сценарные модели функционирования систем и сетей хранения и обработки данных (базовый формализм и темпоральные операции) / С.А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 1. – С. 3-12.

6. Зинкин, С.А. Сети абстрактных машин высших порядков в проектировании систем и сетей хранения и обработки данных (базовый формализм и его расширения) / С.А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2007. – № 3. – С. 13-22.

7. Зинкин С.А. Элементы технологии иерархического концептуального моделирования и реализации систем и сетей хранения и обработки данных // Известия высших учебных заведений. Поволжский регион. Технические науки. –2008. –№ 4 (8). – С. 3-15.

8. Зинкин С.А. Согласование и координация объектов и процессов в агентно-ориентированных системах и сетях хранения и обработки данных / С.А. Зинкин // Вопросы радиоэлектроники. – 2009. – Т. 4. – № 4. – С. 83-95.

9. Дубравин А.В., Зинкин С.А. Развитие логико-алгебраического подхода к созданию распределенных приложений для обработки данных в вычислительных сетях // Оптико-электронные приборы и устройства в системах распознавания образов, обработки изображений и символьной информации: сборник материалов XII Международной научно-технической конференции. – 2015. – С. 112-114.

10. Дубравин А.В., Зинкин С.А. Элементы концептуального распределенного программирования в сетях // Университетское образование (МКУО-2015). сборник статей XIX Международной научно-методической конференции, посвященной 70-летию Победы в Великой Отечественной войне: в 2 томах. Под редакцией: А.Д. Гулякова, Р.М. Печерской. – 2015. – С. 222-225.

11. Волчихин В.И., Дубравин А.В., Зинкин С.А. Абстрактный и структурный синтез распределенных систем обработки данных на основе мультипарадигмального подхода // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2015. – № 1 (33). – С. 60-80.

12. Зинкин С.А., Федосин М.Е., Федосин А.С. Разработка программной и аппаратной архитектуры веб-центра с использованием логико-алгебраических моделей представления знаний // Научно-технический вестник Поволжья. –2013. –№ 6.– С. 289-293.

13. Дубинин В.Н., Зинкин С.А. Сетевые модели распределенных систем обработки, хранения и передачи данных: монография. – Пенза: ПриволжскийДомзнаний, 2013. – 452 с.

14. Якимов, А.Н. Предикатная алгебра выбора в моделировании микроволновых антенн сложной конфигурации / А.Н. Якимов // Надежность и качество сложных систем. – 2013. – № 1. – С. 78-83.

15. Волчихин, В.И. Логико-алгебраические модели и методы в проектировании функциональной архитектуры распределенных систем хранения и обработки данных / В.И. Волчихин, С.А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. –2012. – № 2. – С. 3-16.

16. Зинкин, С.А. Элементы технологии иерархического концептуального моделирования и реализации систем и сетей хранения и обработки данных / С.А. Зинкин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2008. –№ 4. – С. 3-15.

17. MarzulloK. Consistent global states of distributed systems: Fundamental concepts and mechanisms / O. Babaoglu and K. Marzullo – S. J. Mullender (ed.). DistributedSystems, Chap. 4, pp 55-96. ACM Press, NewYork, 1993.

18. Marzullo K. Detection of global state predicates. / K. Marzullo, G. Neiger. – In Proc. 5th Int'l. Workshop on Distributed Algorithms (WDAG '91), vol. 579 of Lecture Notes in Computer Science, pages 254–272. Springer, 1991.

19. BabaogluO. Specification and verification of dynamic properties indistributed computations. / O. Babaoglu, M. Raynal – Journal of Paralleland Distributed Systems, 28(2), 1995, pages 173-185.

20. Бежанишвили М.Н. Логическое всеведение и эпистемическое табличное исчисление предикатов// Логические исследования: ежегодник. 2004. № 11. С. 34-45.

Статья поступила в редакцию 11.11.2020
Статья принята к публикации 11.12.2020