УДК 661:532.546

А.Т. Бектемесов, Д.Ж. Ахмед-Заки, Н.Т. Данаев

Казахский национальный университет им. аль-Фараби, Алматы, Казахстан
E-mail: bektemessov@kaznu.kz

Симуляция и верификация распределенных программ при использовании транзакционной памяти

В данной работе исследована модель симуляции распределенных программ с использованием транзакционной памяти. Показаны особенности модели во время симуляции и дальнейшим этапом верификации. Продемонстрирована модель распределенной системы с использованием WSTM, написанной на языке Promela. Проведен анализ результатов выполнения асинхронных компонентов вычислений. Сделано модельное сравнение распределительной системой TORQUE. При выполнении вычисления задач на кластерных компьютерах проанализирована модель распределения ресурсов между узлами и ее верификация с помощью Spin.
Ключевые слова: верификация, кластер, параллельные вычисления, распределенные системы, Spin, Promela, TORQUE.

A.T. Bektemessov, D.Zh. Akhmed-Zaki, N.T. Danaev

Simulation and verification distributed programs using transactional memory

In this work, investigate model simulation of distributed programs using transactional memory. Shows the feature of the model during simulation, and proceeds to the next stage of verification. Demonstrated a model of a distributed system using WSTM written in language Promela. The results of the asynchronous computing components has been analyzed. The distribution system TORQUE in model comparison is made. In carrying out computation tasks on clustered computers, the distribution model of the nodes is written and analyzed and verification using Spin.
Key words: verification, clusters, parallel computing, distributed systems, Spin, Promela, TORQUE.

А.Т. Бектемесов, Д.Ж. Ахмед-Заки, Н.Т. Данаев

Транзакциялы жадыны қолданған үлестірмелі программалардың симуляциясы және верификациясы

Бұл мақалада транзакциялы жадыны қолдана отырып, үлестірілген программалардың симуляцияланған моделіне зерттеу жүргізіледі. Симуляция кезіндегі модель ерекшеліктеріне тоқталып, келесі верификациялау кезеңіне өтеді. Promela тілінде жазылған WSTM және үлестірімді программалардың моделі ұсынылған. Программа моделінің жұмыс істеу барысында алгоритмнің асинхронды есептелінетін компоненттер нәтижелеріне анализ жасалынады. MPI үлестірімді программасы ретінде қолданатын TORQUE жүйесіне моделді салыстыру жүргізілген. Кластерленген компьютерлердегі күрделі есептеулер барысында түйіндер арасындағы үлестірімді алгоритм моделіне сараптама жүргізіледі және Spin арқылы моделі құрылып, верификацияланады.
Түйін сөздер: верификация, кластер, параллельді есептеулер, үлестірімді жүйелер, Spin, Promela, TORQUE.

Введение

Параллельные программы с конечным числом состояний возникают во многих разделах теории вычислений. Серьезные логические ошибки, возникающие на последних

стадиях разработки, очень важная проблема для проектировщиков и программистов. Началом моделирования при проверке корректности системы является спецификация свойств. Надо показать, что некоторая параллельная система не попадает в тупик. Но стоит отметить что, часто встречается нехватка ресурсов или мощности узлов при высокопроизводительных вычислениях. В таких ситуациях решением проблем считаются распределение нагрузки между узлами.

Распределенные системы состоят из множества компонентов, которые выполняются совместно. Обычно компоненты взаимодействует с некоторыми средствами. Режим выполнения и тип взаимодействия в разных средах могут отличатся. Можно представить два типа исполнения: асинхронное, в котором в любой момент времени только один компонент делает шаг вычисления и синхронное, в котором все компоненты вычисляют одновременно. Компоненты могут вычислять путем изменения значения общих переменных, либо с помощью обмена сообщениями. В обоих случаях могут быть использованы как очереди, так и синхронные взаимосвязи.

В этой работе продемонстрирована модель распределенных систем с использованием WSTM, учитывая то, что производительность вычислений растет с точки зрения пересылки данных и сериализуемость выполнения параллельных программ. В транзакционной памяти оконного режима (Window-based STM) [1, 2] в определенный промежуток времени потоки транзакции обращаются в память для редактирования значения выделенной области (рис. 1). Таким образом, поток транзакций выполняется в промежуточно-согласованное время. Это облегчит рутинную работу проектировщика для синхронизации общей памяти или пересылки сообщений.

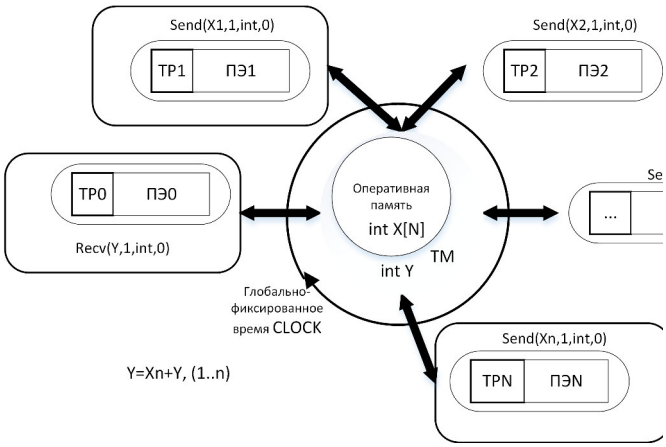

Рисунок 1 – Схема работы WSTM в параллельной программе

Модельная проверка программ

Один из наиболее перспективных и широко используемых подходов к решению проблемы автоматизации отладки и проверки правильности программ является Model Checking. Для заданной анализируемой программы строится ее абстрактная формальная модель. Проверяемое свойство или требование выражается на формальном методе в виде логической формулы: $M \models \phi$, которая означает, что некоторая булева формула ϕ удовлетворяет модели M, верификация программы сводится к проверке выполнимости фор-

мализованного требования спецификации на абстрактной модели программ [3, 4].

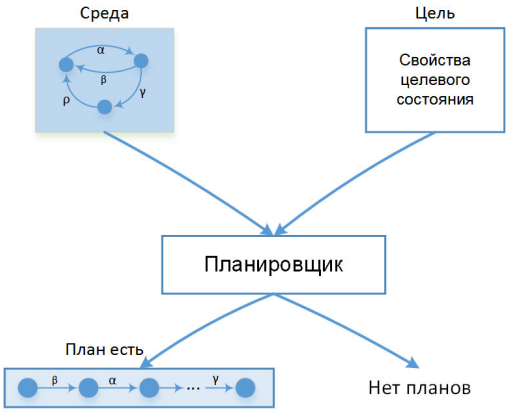

Рисунок 2 – Система планирования Model Cheking.

Система распределенных вычислений

Система пакетной обработки заданий (СПО) осуществляется при синхронном выполнении нагрузки несколькими пользователями на общем вычислительном комплексе. После применения СПО вычислительные ресурсы используются оптимально: перегрузка какого-либо одного узла. Torque часто применяется в сферах, где большая интенсивность использования вычислительных мощностей.

Таким образом, torque контролирует вычислительные ресурсы, освобождая их для выполнения других задач. Также torque контролирует выполнение заданий, используя приоритетность и планировщик заданий. Принцип работы torque заключается в том что, задания создаются и управляются сервером заданий [5]. Клиенты СПО взаимодействуют с сервером заданий, который предоставляет соответствующие службы. Осуществлена верификационная симуляция при реализации функций TORQUE и WSTM на распределенных узлах (рис. 3).

Поведение двух параллельных процессов представляется недетерминированном интерливингом последовательного процесса. Обозначим операции двух неделимых операций α и β. Эти операций прямой зависит от последовательных выполнении в любом порядке: $\alpha;\beta$ или $\beta;\alpha$.

Асинхронное выполнение параллельных процессов, осуществленных с помощью рандеву, определяется следующим образом. Пусть

$$P_i = (S_i, \Sigma_i, \rightarrow_i, S_{0i}), i = 1, 2$$

здесь, S_i - множество состояний двух параллельных процессов, Σ_i - множество операций, $\rightarrow_i$ - множество переходов и множество начальных состояний S_{0i}. Пусть операции выполнения взаимосвязи рандеву обозначим для обоих процессов идентично. Тогда, $H = \Sigma_1 \cap \Sigma_2$ - обозначим множества как взаимосвязь двух операций. Из алфавита записываем последовательность операции процесса P_0:

$$\rho = P_1||P_2 = (S_1 \times S_2, \Sigma_1 \cup \Sigma_2, \rightarrow_i, S_{01} \times S_{02}),$$

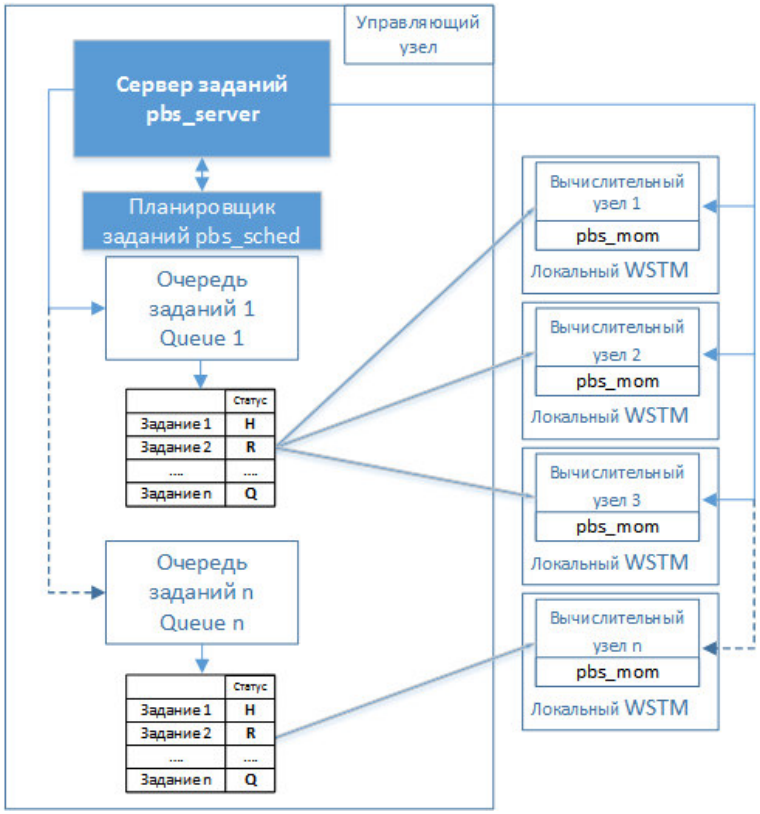

Рисунок 3 – Схема работы WSTM в системе Torque.

где множество $\rightarrow$ переходов процесса P_0 определяется общим действием процессов P_1 и P_2:

$$\frac{(s_1 - \alpha \rightarrow_1 s_1' \wedge s_2 - \alpha \rightarrow_2 s_2')}{< s_1, s_2 > -\alpha \rightarrow < s_1', s_2' >} \text{ при } \alpha \in H$$

данное правило определяет взаимодействия параллельных процессов.

Рассмотрим параллельную программу, состоящих из двух процессов P_1 и P_2, частично независимо выполняющие последовательности простых операций суммы от 1-го до 100. Более того, в алгоритме внесен режим распределения и планирования по нагрузке или свойству вычисляющих узлов.

Схема задачи состоит следующим образом:

```
P1:                          P2:
m0: Distr_WSTM               t0: Distr_WSTM
m1: sum(1-50)                t1: sum(51-100)
m2: end                      t2: end
```

```
function Proc_send(addr,i,cap)
{       s = cap - 49; k = cap;
        if ( busy1 && i == 0 ) then k = 0;
```

```
            else skip
        end if;
        for ( s <= k; s = s + 1)
                MEMORY[addr].value = MEMORY[addr].value + s;
            else send MEMORY[addr].value by chanel chan_p1;
            if ( i == 1 && busy1 ) then MEMORY[addr].value = 0;
            receive b by chanel chan_busy; k = b; s = k - 49;
            for ( s <= k; s = s + 1) then
             MEMORY[addr].value = MEMORY[addr].value + s;
            else send MEMORY[addr].value by chanel chan_p1;
               break
            end for; else skip
            end if; break
        end for
}

function Proc_recv(addr)
{       for (infinitive) then receive p1 by chanel chan_p1;
        MEMORY[addr].value = MEMORY[addr].value + p1;
        end for;
}

main function MemoryWrite(addr, value, date, i)
{       wby[addr].tran[i] = true; task = 100;
        if ( i == 0 ) then Proc_send(addr,i,task/2);
        if ( i == 1 ) then Proc_send(addr,i,task);
        end if; end if;
        MEMORY[addr].date = date;
        wby[addr].tran[i] = false;
}
```

Все возможные варианты операций между двумя параллельными процессами можно определить по методу интерливинга (рис. 4).

Первая транзакция вычисляет от одного до 50, вторая транзакция вычисляет от 51 до 100. В начальном этапе верификации выполняется симуляция модели. Симуляционное выполнение должно выдать результат sum=5050. При распределенной работе каждый узел имеет метку атомарности с использованием WSTM.

Утверждение по логике LTL:

$([] <> (LT2 \wedge \neg readTran2(X)1 \wedge ([] <> readTran2(X)1))) \rightarrow \neg([] <> (readTran2(X)1 \wedge (<> Compt2)))$

$readTran2(X)v$ - событие чтения значения v в переменную X из разделяемой памяти транзакцией T. Событие LT представляет собой точку линеаризации, момент решения - принимать транзакцию T или отклонять [1]. $COMPT$ – вход транзакции T в состояние изменения данных.

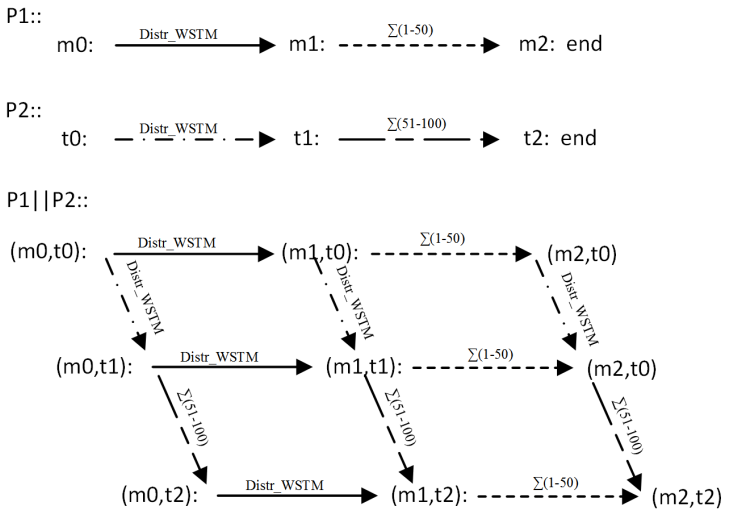

Рисунок 4 – Композиция двух параллельных процессов.

В результате, главный процесс *init* при вычислении алгоритма получает от процесса P_1 нулевой ответ о неисправности узла в состоянии 209, до того как, обнаружилась проблема на первом узле, в состоянии 197, система распределения переправила нагрузку вычисления к процессу P_2. Таким образом осуществляются параллельные вычисления нагрузки последовательно по траектории:

$(m_0, t_0) \to (m_1, t_0) \to (m_1, t_1) \to (m_1, t_2) \to (m_2, t_2) \to end.$

Метод моделирования распределения задач, а так же, верификация в работе [6] основывается на трех критериях параллелизма: параметры, локальные, публичные, совместно используемые переменные. Данный подход является решением на уровне "общих переменных"между параллельно выполняющими процессами. Однако главная цель этой работы является автоматизация построения модели для верификации в среде Spin.

В работе [7] предложен метод распределения задач между мультипроцессорами, в ней рассматривается обмен сообщений через отправку и получение данных между процессами.

В этой работе предложен гибридный вариант, то есть, обмен сообщениями между процессами с управлением транзакционной памяти WSTM. Недостатком является время выполнения алгоритма за счет обращения общих переменных каждой локальной транзакции и сложность покрытия всех возможных утверждений для модельного тестирования. В дальнейшем планируется работа по улучшению времени выполнения и оптимизация алгоритма.

Заключение

Для анализа и проверки алгоритма распределенной системы на транзакционной памяти WSTM и на основе torque разработана модель и верифицирована с помощью xSpin. При тестировании на основе Model Checking, показан корректность алгоритма. Выявлен как ряд преимуществ так и недостатки по сравнению с аналогичными работами. Результат работы можно разделить на две части: первое, получена модифицированная гибридная модель распределения нагрузки между узлов. Второе, верификация распре-

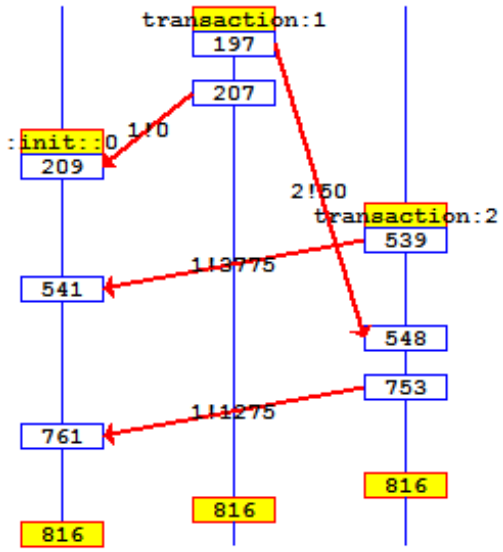

Рисунок 5 – Результат верификаций распределенной программы с транзакционной памятью WSTM на Spin.

деленной программы с транзакционной памятью WSTM.

Работа выполнена при поддержки Комитета Науки МОН РК, грант № 174/ГФ1.

Литература

[1] *Беляев А.Б* Верификация алгоритма поддержки транзакционной памяти. // Научно-технические ведомости, «Телематика-2010: телекоммуникации, веб-технологии, суперкомпьютинг». – СПбГУ, Санкт-Петербург № 101. 2010. – С. 186–192.

[2] *Imbs D., Raynal M.* Software Transactional Memories: An Approach for Multicore Programming // PaCT 2009. LNCS. – 2009. – Vol. 5698. – P. 26–40.

[3] *Карпов Ю.Г.* Model Checking: Верификация параллельных и распределенных программных систем. // «БХВ-Петербург» 2010. – 189 с.

[4] *Clarke E. M., Grumberg O., Peled D.A.* Model Checking // The MIT Press. – London, England, 1999. – 330 p.

[5] Torque administrator's guide. www.clusterresources.com (дата обращения 25.07.2014)

[6] *М.А. Лукин, А.А. Шалыто* Международная научно-практическая конференция: Инструменты и методы анализа программ. // ТМРА-2013. – 2013. – С. 78–93.

[7] *F.J. Morten* Formal Verification of Distributed Programs using Session Types and Coq. // IT University of Copenhagen. – 2013. – 68 p.

References

[1] *Belyaev A. B.* Verification algorithm of the supporting transactional memory. // Technical-science issue, «Telematic-2010: telecommunication , web-technology, supercomputing». – SPbGU, Saint-Petersburg № 101. 2010. – P. 186-192.

[2] *Imbs D., Raynal M.* Software Transactional Memories: An Approach for Multicore Programming // PaCT 2009. LNCS. – 2009. – Vol. 5698. – P. 26–40.

[3] *Karpov Y.G.* Model Checking: Verification of parallel and distributing software systems. // «BHV-Peterburg» – 2010. – 189 p.

[4] *Clarke E. M., Grumberg O., Peled D.A.* Model Checking // The MIT Press. – London, England. – 1999. – 330 p.

[5] Torque administrator's guide. www.clusterresources.com (date of treatment 25.07.2014)

[6] *M.A. Lukin, A.A. Shalyto* International scientific-practical conference: Tools and Methods of Program Analysis. // TMPA-2013. – 2013. – P. 78–93.

[7] *F.J. Morten* Formal Verification of Distributed Programs using Session Types and Coq. // IT University of Copenhagen. – 2013. – 68 p.