УДК 004.415.538.052.3 **doi: 10.32620/reks.2018.4.06**

В. А. РОМАНКЕВИЧ, А. В. РОМАНКЕВИЧ, Д. Н. АХМЕДОВА

Национальный технический университет Украины "Киевский политехнический институт имени Игоря Сикорского", Украина

МЕТОД УМЕНЬШЕНИЯ КОЛИЧЕСТВА ВЗАИМОПРОВЕРОК ПРИ САМОТЕСТИРОВАНИИ МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Предметом *исследования в данной работе являются процессы тестирования в многопроцессорных системах, в первую очередь - отказоустойчивых систем управления, когда число* ***m*** *допустимых отказов из* ***n*** *процессоров заранее известно и ограничено. Топология межпроцессорных связей может быть описана направленным графом типа циркулянт, где число входных и выходных каналов связи каждого процессора с другими процессорами системы не менее* ***m***. ***Целью*** *является минимизация количества взаимопроверок в системе при выполнении ее самотестирования.* ***Задачи:*** *разработать эффективный метод организации взаимотестирования процессоров в многопроцессорных системах и алгоритм его выполнения, дать оценку количества элементарных проверок, доказать ее справедливость. Получены следующие* ***результаты****. На базе известного метода диагностирования многопроцессорных систем с регулярной структурой и* $m \le 4$ *при параллельном и независимом тестировании сформулирована задача организации процесса тестирования для любых* $m < (n / 2)$*, который назван последовательным и при котором в каждый момент времени в тестировании участвует лишь одна пара процессоров. Особенности такой организации: выбор следующей пары осуществляется на основе анализа предыстории процесса. Предложен алгоритм выполнения метода. В качестве модели неисправностей выбрана модель Препараты-Метца-Чена как наиболее близкая к реальности.* ***Выводы****. Научная новизна полученных результатов состоит в следующем: предложен метод организации самотестирования многопроцессорных систем с топологией связей, описываемых графом-циркулянтом (с не менее* ***m*** *входных и* ***m*** *выходных стрелок), который позволяет сократить количество взаимопроверок в системе. Доказано, что состояние (исправен-неисправен) всех процессоров системы может быть установлено после выполнения не более* $n+2m$ *взаимопроверок. Практическое значение – метод позволяет уменьшить потери времени, затрачиваемые системой на самотестирование, а, значит, повысить её производительность, поскольку эту задачу система выполняет постоянно в процессе эксплуатации. Преимущества выбранной топологии: она пригодна для любых целочисленных значений величины* ***n***.

Ключевые слова: *многопроцессорная система; самотестирование; диагностический граф; циркулянт.*

Введение

Как известно, процесс эксплуатации многопроцессорных систем, в том числе отказоустойчивых (ОМС) [1], включает в себя, как обязательную, процедуру взаимотестирования процессоров. Целью этой процедуры является установление состояния всех процессоров для дальнейшего исключения из работы всех неисправных.

Задача самотестирования ОМС [2] актуальна и имеет научный и практический интерес особенно в аспекте сокращения времени самотестирования системы, что связано с ее производительностью. Действительно, уменьшение времени тестирования на каком-то промежутке времени эксплуатации ОМС дает возможность увеличить время, которое система посвящает решению своих задач, в частности задач управления, что фактически повышает ее производительность. Под ОМС будем понимать системы, для которых принято обозначение k-out-of-n.

Один из вариантов уменьшения времени самотестирования ОМС рассмотрен в [3], где топология связей системы для целей диагностирования может быть представлена конструктивно регулярным диагностическим графом (частный случай графов-циркулянтов). Ниже мы будем ориентироваться именно на графы-циркулянты. Кроме того, будем ориентироваться на модель диагностирования Препараты-Метца-Чена (ПМЧ-модель, см. [2]) как на наиболее практически ценную. Ее особенностью является тот факт, что достоверным является лишь результат тестирования, который осуществляется исправным процессором, то есть при тестировании исправного процессора результат «0», при тестировании неисправного – «1». Если тестирующий неисправен, результат не зависит от исправности тестируемого, что обычно обозначается как «*x*». В работе [3] предполагается возможность для каждого процессора тестировать всего 2 других, независимые проверки осуществляются параллельно. Общее чис-

ло процессоров равно n, выполняются все проверки, число которых не превышает величины $2n+2$ при количестве допустимых отказов $m \leq 4$.

1. Суть метода

Ниже рассматривается более общий случай, когда топология ОМС предполагает наличие достаточного количества диагностических связей. Более точно: в диагностическом графе каждая вершина имеет m входящих и столько же исходящих стрелок. Как уже говорилось, это – граф-циркулянт. Подобные графы нами выбраны не случайно по нескольким причинам. Во-первых, в отличие от известных топологий с регулярной структурой, таких как, например, матричная или гиперкуб, рассматриваемая топология пригодна для любого количества процессоров. Во-вторых, как показано в [3], по крайней мере, для небольших значений величины m, тестовый эксперимент может быть достаточно коротким. Имеются и другие преимущества (см. например, [4, 5]).

Цель исследований – показать, что общее число проверок может быть доведено до величины $n+m$ за счет изменения процесса тестирования и некоторого усложнения анализа результатов. Изменение заключается в том, что анализ выполняется после каждой отдельной проверки, что возможно лишь тогда, когда проверки осуществляются последовательно, одна за другой, то есть в каждый момент времени в эксперименте участвует лишь одна пара процессоров. Новая пара (тестирующий-тестируемый) для следующего такта тестирования выбирается на основе анализа результатов предыдущих проверок подобно тому, как это предложено в [6], где диагностический граф является полным.

Здесь следует отметить, что время, которое многопроцессорная система затрачивает на самотестирование, фактически определяется количеством взаимопроверок процессоров независимо от того, сколько проверок осуществляется в одном такте: одна или несколько. Под тактом тестирования понимается время, в течение которого выполняется проверка одного процессора.

Пусть имеется ОМС R с диагностическим графом-циркулянтом $G(m,n)$, где m выходных и m входных различных стрелок у каждой вершины, и в которой допускается не более $m<n/2$ отказов.

Утверждение. Для установления состояния всех процессоров системы R достаточно не более $n+2m$ тестовых проверок.

Для доказательства рассмотрим наиболее плохое расположение исправных и неисправных процессоров (если их ровно m), которое приводит к тому, что все неисправные процессоры тестируют один и тот же процессор, и, следовательно, появляется неопределенность типа $xx\ldots x$ [3]. Однако при этом такой процессор является единственным. Все неисправные будут иметь хотя бы одну входящую связь с исправным процессором, и, следовательно, их состояние может быть установлено. Поскольку число неисправных ровно m, то становится понятным, что неопределенность исчезает – упомянутый процессор исправен. Ситуация с иным расположением неисправных процессоров и случаи реального числа отказов менее m понятны: всегда будет иметь место связь с каким-то исправным процессором, и, следовательно, состояние любого процессора может быть установлено.

Перейдем к сути метода. Её можно выразить следующей последовательностью действий.

Первый этап. Формирование 0-цепочек.

Выбирается произвольная пара процессоров, которые имеют связь в диагностическом графе, и тестирующий (пусть это первый) процессор тестирует второй – проверка v_{12}. При нулевом результате v_{12} осуществляется проверка v_{23}, где цифра 3 символизирует любой процессор, еще не участвовавший в эксперименте. Так продолжается до тех пор, пока результат проверки не станет равным 1, и соответствующая пара исключается из цепочки (по крайней мере один из процессоров неисправен в соответствии с ПМЧ-моделью) и переводится в множество «подозреваемых». Последний из оставшихся в цепочке процессоров тестирует любой, доступный ему и еще не участвующий в эксперименте процессор. Далее по тому же принципу формируется новая 0-цепочка и т.д. Весом вершины графа является общее количество вершин графа (включая данную), которые подходят к ней через 0-проверки.

Второй этап: Образование деревьев.

На этом этапе цепочки объединяются, начиная с наиболее длинных, причем последний процессор (вершина графа) с весом R более короткой цепочки длиной l компонент тестирует процессор, имеющий такой же вес R другой цепочки. Нулевой результат увеличивает вес последнего элемента дерева, единичный переводит $2l$ компонент в множество «подозреваемых», и остается одна цепочка, укороченная на l компонент. При этом вес каждой из ее компонент также уменьшается на l.

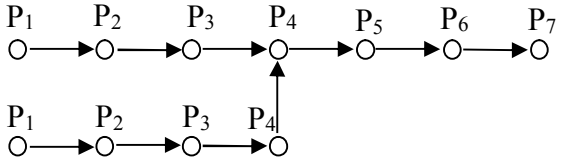

Рис.1. Пример объединения двух цепочек

На рисунке 1 дан пример объединения двух цепочек, содержащих седьмую и четвёртую вершины (последние вершины имеют вес 7 и 4 соответственно). После того, как четвёртый процессор короткой цепочки проверит четвёртый процессор более длинной цепочки, вес последнего процессора (последней вершины) длинной цепочки изменяется: при нулевом результате он увеличивается до 11, при единичном уменьшается до 3, причем в последнем случае в множество подозреваемых добавляется 8 процессоров.

Таким же образом объединяются другие цепочки. Далее объединяются деревья, хотя это иногда несколько сложнее, однако суть остается прежней: выбирается пара элементов одинакового веса из двух разных деревьев.

Теперь перейдем к установлению исправного процессора, для чего сформулируем почти очевидную Лемму.

Лемма. Последний процессор с весом R 0-цепочки или дерева исправен, если $R+\alpha > m$, где α – суммарное количество установленных и «подозреваемых» неисправных процессоров вне цепочки.

Как наиболее неблагоприятную ситуацию мы принимаем, что число неисправных процессоров составляет ровно половину множества «подозреваемых», хотя реально их в этом множестве может быть больше. Сформулированная Лемма дает возможность определить на этом этапе исправный процессор. Последнее всегда возможно, так как число исправных процессоров вообще в системе и, в частности, в множестве процессоров, которые не входят в число «подозреваемых», превышает число неисправных.

Третий этап: окончание тестирования и анализ результатов.

Установленный на предыдущем этапе исправный процессор тестирует другие следующим образом:

– тестирует доступные ему (кроме тех, которые он тестировал на предыдущих этапах), начиная либо с ещё непроверенных процессоров (если число 1-проверок к этому моменту меньше m), либо с первых процессоров каких-то цепочек, пока не обнаружит исправный процессор. Если все, доступные ему, оказываются неисправными, то все остальные в ОМС – исправны, и тестовый эксперимент закончен;

– вновь установленный и все последующие исправные процессоры выполняют по очереди те же действия, устанавливая состояние всех процессоров системы. Исключение, как уже отмечалось выше, может составить ситуация, когда в системе имеется максимальное число неисправных процессоров, и все они тестируют один и тот же процессор. В этом случае состояние последнего определяется косвенно, после того, как установлены все неисправные процессоры: он исправен. Все неисправные будут установлены, поскольку в нашем диагностическом графе, как уже упоминалось, для каждого неисправного процессора среди связанных с ним найдется такой исправный, который его тестирует. При этом многие неисправные доступны для тестирования несколькими исправными, что облегчает определение их состояния. Все сказанное подтверждает справедливость Утверждения.

Для иллюстрации выполнения этапов рассмотрим пример выполнения тестирования в ОМС для случая m=3 и n=7, предполагая, что диагностический граф это граф-циркулянт со скачками 1, 2 и 4.

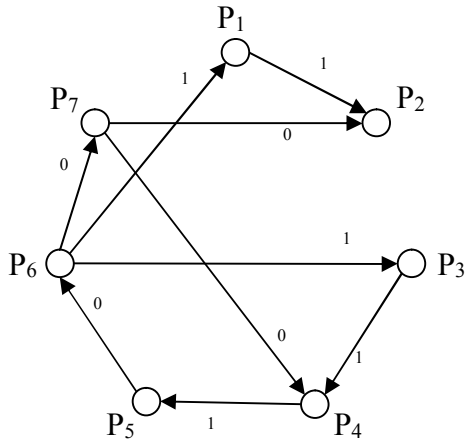

Рис. 2. Пример диагностического графа

Выполнение тестового эксперимента, как было описано выше, начинается с проверки v_{12} (см. рис. 2). Поскольку результат равен 1, цепочка заканчивается, и далее выполняется проверка v_{34}=1 и затем v_{56}=0. Последняя проверка устанавливает исправность шестого процессора (будем обозначать P_6=И) в соответствии с Леммой: вес последней вершины цепочки 5-6, то есть вершины P_6, равен 2, а в множестве оставшихся после образования двух цепочек допускается лишь 1 неисправность (напомним, что m=3). Далее выполняются проверки

v_{67}=0, что даёт P_7=И
v_{61}=1, что даёт P_1=Н
v_{63}=1, что даёт P_3=Н
v_{72}=0, что даёт P_2=И
v_{74}=0, что даёт P_4=И
v_{45}=1, что даёт P_5=Н

Тестовый эксперимент закончен, другие возможные проверки не понадобились, состояния всех процессоров установлены. Можно отметить, что в

данном процессе исправный процессор P_2 тестировался дважды, однако неисправные P_1 и P_5 потребовали по одной проверке. Безусловно, выполнение проверок может быть выполнено по-другому, однако результат будет таким же.

2. Количество проверок

Попробуем теперь определить сложность проводимого диагностического эксперимента, то есть подсчитать количество проверок S, достаточное для установления состояния всех процессоров системы.

После установления исправного процессора следуют только достоверные проверки, то есть такие, которые устанавливают состояние проверяемого процессора окончательно. Проверяются лишь те процессоры, состояние которых еще не установлено.

Рассмотрим цепочку, образованную на первом этапе. Проверка первого элемента цепочки уже установленным исправным процессором определяет его состояние. Если он исправен, то состояние всех процессоров цепочки определено, то есть дополнительные проверки не требуются. Если он неисправен, то следует проверить следующий элемент цепочки; если и он неисправен – следующий и т. д. Из этих простых действий, если не вникать в детали, можно сделать следующий вывод: для установления состояния исправного процессора в среднем требуется одна проверка, неисправного – две, что в сумме приведет к $S = n + m$ проверкам. Учитывая тот факт, что в начале третьего этапа не всегда имеется доступ к началу цепочек, что может привести к лишним проверкам, можем принять окончательно

$$S = n + 2\,m,$$

хотя реально их может быть меньше. Многочисленные эксперименты, выполненные авторами с различными графами для различных значений n и m, подтверждают это. В рассмотренном примере (см. рис. 2) понадобилось всего девять проверок, что меньше даже $n + m = 7+3$.

В качестве примера, иллюстрирующего возможность меньшего числа проверок, возьмем следующий. Пусть все проверки первого этапа дают единичный результат. В худшем случае, когда m равно почти половине n остается один процессор, который исправен, и m пар процессоров с единичными проверками. В этом случае может получиться (опять же в худшем случае, когда тестирующий неисправен) так, что на третьем этапе придется проверять каждый из процессоров каждой пары, то есть на каждую пару придется потратить в сумме 3 проверки, а значит в общей сложности $n+m$ проверок.

Заключение

Рассматривается задача минимизации потерь времени, которое система расходует на диагностирование. Фактически это время определяется числом взаимопроверок независимо от того, выполняются они последовательно или параллельно. Предполагается, что система имеет n процессоров и в ней допускается $m<n/2$ неисправностей. Топология *диагностических* связей отображается орграфами-циркулянтами с m выходными и m входными дугами. Отличительной особенностью является осуществление анализа на каждом шаге диагностического эксперимента, который в этом случае осуществляется последовательно: в любой момент времени работает лишь одна пара процессоров, и новая пара выбирается на основе этого анализа. Показано, что для установления состояния всех процессоров системы достаточно не более $S = n+2\,m$ проверок.

Литература

1. Avizienis, A. Fault-tolerance: The Survival Attribute of Digital Systems [Text] / A. Avizienis // Proc. IEEE. – 1978. – Vol. 66, No. 10. – P. 1109–1126.

2. Preparata, F. P. On the Connection Assignment Problem of Diagnosable Systems [Text] / F. P. Preparata, G. Metze, R. T. Chien // IEEE Trans. Electron. Comput. – 1967. – Vol. ES-16, No. 6. – P. 848-854.

3. Romankevich, V. A. Self-testing of multiprocessor systems with regular diagnostic connections [Text] / V. A. Romankevich // Autom. Remote Control. – 2017. – Vol. 78, No. 2. – P. 289–299.

4. Dimitriev, Y. K. Necessary and sufficient conditions for t-diagnosability of multiprocessor computer systems for various models of nonreliable testing established using the system graph-theoretical model [Text] / Y. K. Dimitriev // Autom. Remote Control. – 2015. – Vol. 76, No. 7. – P. 1260–1270.

5. Dimitriev, Y. K. On t-diagnosability of multicore systems with symmetric circulant structure [Text] / Y. K. Dimitriev // Autom. Remote Control. – 2013. – Vol. 74, No. 1. – P. 105–112.

6. Self-Diagnosable Multimodular Systems: Some Estimates of Testing [Text] / V. E. Belyavskii, V. N. Valuiskii, A. M. Romankevich, V. A. Romankevich // Autom. Remote Control. – 1999. – Vol. 60, No. 8. – P. 1179–1183.

References

1. Avizienis, A. Fault-tolerance: The Survival Attribute of Digital Systems. *Proc. IEEE*, 1978, vol. 66, no. 10, pp. 1109–1126.

2. Preparata, F. P., Metze, G., Chien, R. T. On the Connection Assignment Problem of Diagnosable Systems. *IEEE Trans. Electron. Comput.*, 1967, vol. ES-16, no. 6, pp. 848-854.

3. Romankevich, V. A. Self-testing of multiprocessor systems with regular diagnostic connections. *Autom. Remote Control*, 2017, vol. 78, no. 2, pp. 289–299.

4. Dimitriev, Y. K. Necessary and sufficient conditions for t-diagnosability of multiprocessor computer systems for various models of nonreliable testing established using the system graph-theoretical model. *Autom. Remote Control*, 2015, vol. 76, no. 7, pp. 1260–1270.

5. Dimitriev, Y. K. On t-diagnosability of multicore systems with symmetric circulant structure. *Autom. Remote Control*, 2013, vol. 74, no. 1, pp. 105–112.

6. Belyavskii, V. E., Valuiskii, V. N., Romankevich, A. M. and Romankevich, V. A. Self-Diagnosable Multimodular Systems: Some Estimates of Testing. *Autom. Remote Control*, 1999, vol. 60, no. 8, pp. 1179–1183.

Поступила в редакцию 04.10.2018, рассмотрена на редколлегии 12.12.2018

METHOD OF DECREASING NUMBER OF MUTUAL TESTING DURING SELF-TESTING OF MULTIPROCESSOR SYSTEMS

V. A. Romankevich, O. V. Romankevych, D. N. Akhmedova

The subject matter of this article is the testing processes in multiprocessor systems, first of all - fault-tolerant control systems, when the number m of allowed failures from n processors is known in advance and is limited. The topology of interprocessor communications can be represented as a directed graph of the circulant type, where the number of input and output channels between each one processor and the other processors in the system is not less than m. The goal is to minimize the number of mutual checks in the system when performing its self-testing. Tasks: develop an effective method for organizing the mutual testing of processors in multiprocessor systems and algorithm for its implementation, estimate the number of elementary checks and prove its rightness. The following results were obtained. Based on the known method of diagnosis of multiprocessor systems with a regular structure, and $m \leq 4$, with parallel and independent testing, the task of organizing of testing process for any $m < (n/2)$ was formulated. In this process, only one pair of processors participate in testing in one moment of time, thus it was called sequential. what special about this organization is that the choice of the next pair is based on the analysis of the history of the process. An algorithm for performing the method has been proposed. The Preparata-Metz-Chen model was chosen as the model of faults, as the closest one to reality. Conclusions. The scientific novelty of the results is as follows: The method for organizing of self-testing of multiprocessor systems with a connection topology described by a circulant graph (with at least m input and m output edges) has been proposed, and this method allows reducing the number of mutual tests in the system. It has been proved that the state (serviceable/faulty) of all processors of the system can be determined after performing no more than $n + 2m$ checks. Practical value - the method allows to reduce the time spent by system on self-testing, and, therefore, to increase its performance, since the system performs this task constantly in the process of exploitation. The advantages of the selected topology: it is suitable for any integer values of n.

Keywords: multiprocessor system; self-testing; diagnostic graph; circulant.

МЕТОД ЗМЕНШЕННЯ КІЛЬКОСТІ ВЗАЄМОПЕРЕВІРОК ПРИ САМОТЕСТУВАННІ БАГАТОПРОЦЕСОРНИХ СИСТЕМ

В. О. Романкевич, О. В. Романкевич, Д. Н. Ахмедова

Предметом дослідження в даній статті являються процеси тестування в багатопроцесорних системах, в першу чергу – відмовостійких системах управління, коли число **m** доступних відмов з **n** процесорів заздалегідь відомо та обмежено. Топологія міжпроцесорних зв'язків може бути описана направленим графом типу циркулянт, де число вхідних та вихідних каналів кожного процесора з іншими процесорами системи не менше **m**. **Метою** є мінімізація кількості взаємних перевірок в системі при виконанні її самотестування. **Задачі**: розробити ефективний метод взаємотестування процесорів в багатопроцесорних системах та алгоритм його виконання, дати оцінку кількості елементарних перевірок, довести її справедливість. Отримані наступні **результати**. На базі відомого метода діагностування багатопроцесорних систем з регулярною структурою

і $m \leq 4$ при паралельному та незалежному тестуванні сформульована задача організації процесу тестування для будь-яких $m < (n / 2)$, який названо послідовним і в якому в кожний момент часу в тестуванні приймає участь лише одна пара процесорів. Особливості такої організації: вибір наступної пари виконується на основі аналізу передісторії процесу. Запропоновано алгоритм виконання методу. В якості моделі несправностей обрана модель Препарати-Метца-Чена як найбільш близька до реальності. **Висновки**. Наукова новизна отриманих результатів полягає в наступному: запропоновано метод організації самотестування багатопроцесорних систем з топологією зв'язку, що описується графом циркулянтом (з не менш, ніж m вхідних та m вихідних стрілок), який дозволяє скоротити кількість взаємоперевірок в системі. Доведено, що стан (справний-несправний) усіх процесорів системи може бути встановлено після виконання не більше, ніж **n+2m** перевірок. Практичне значення – метод дозволяє зменшити витрати часу, що витрачається системою на самотестування, що означає – підвищити її продуктивність, адже цю задачу система виконує постійно в процесі експлуатації. Переваги обраної топології: вона придатна для будь-яких цілочисельних значень величини **n**.

Ключові слова: багатопроцесорна система; самотестування; діагностичний граф; циркулянт.

Романкевич Виталий Алексеевич – д-р техн. наук, доцент, профессор кафедры системного программирования и специализированных компьютерных систем, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Романкевич Алексей Витальевич – студент кафедры системного программирования и специализированных компьютерных систем, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Ахмедова Дарина Натиговна – магистрант кафедры системного программирования и специализированных компьютерных систем, Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Киев, Украина.

Romankevich Vitaliy A. – DrS on Engineering, Professor of Department of System Programming and Specialized Computer Systems, National Technical University of Ukraine "Igor Sikorskiy Kiev Polytechnic Institute" Kiev, Ukraine, e-mail: romankev@scs.kpi.ua,
ORCID Author ID: 0000-0003-4696-5935, Scopus Author ID: 57193263058,
https://scholar.google.com.ua/citations?user=nHqiOEQAAAAJ&hl=ru

Romankevych Oleksii V. – student of Department of System Programming and Specialized Computer Systems, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine, e-mail: romankev@scs.kpi.ua.

Akhmedova Daryna N. – master student of Department of System Programming and Specialized Computer Systems, National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute" Kyiv, Ukraine, e-mail: darina.akhmedova@gmail.com.