

ДЕЦЕНТРАЛИЗОВАННЫЙ АЛГОРИТМ САМОДИАГНОСТИКИ ДЛЯ КРУПНОМАСШТАБНЫХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ РАЗЛИЧНЫХ ТОПОЛОГИЙ

О. В. Молдованова

Сибирский государственный университет телекоммуникаций
и информатики, 630102, Новосибирск, Россия

УДК 004.052.32

Предложен децентрализованный алгоритм самодиагностики крупномасштабных распределенных вычислительных систем, характеризующийся параллельным выполнением фаз тестирования и распространения диагностической информации. Приведены результаты моделирования алгоритма для распространенных топологий распределенных вычислительных систем.

Ключевые слова: распределенная вычислительная система, самодиагностика, децентрализованный алгоритм, моделирование.

A distributed self-diagnosis algorithm for large-scale distributed computer systems (CS) is proposed in the work. This algorithm is characterized by a parallel execution of a testing phase and phase of diagnostic information dissemination. Results of the algorithm simulation are provided for commonly used topologies of distributed CS.

Key words: distributed computer system, self-diagnosis, distributed algorithm, simulation.

Введение. Распределенные вычислительные системы (ВС) являются важнейшим инструментом решения сложных научных, инженерных и экономических задач [1]. Основным функциональным элементом распределенной ВС является элементарная машина (ЭМ). Такие системы характеризуются крупномасштабностью: количество ЭМ в их составе может достигать 10^5 – 10^6 . Несмотря на высокую надежность микроэлектронной базы, с увеличением количества элементарных машин вероятность возникновения отказов в распределенных ВС повышается. Кроме того, в последнее время постоянно увеличивается число трудоемких задач, решаемых на крупномасштабных ВС. Следовательно, организация отказоустойчивого функционирования таких систем требует использования алгоритмических и программных средств самоконтроля и самодиагностики.

В течение нескольких десятилетий был предложен ряд методик диагностики отказов в вычислительных системах [2–7]. Большинство работ основано на стратегии, описанной в [2], согласно которой ЭМ способны тестировать друг друга. При этом исправные ЭМ могут безошибочно определить состояние тестируемых ими элементарных машин, а результат тестирования неисправными ЭМ может быть произвольным. Все результаты тестов собира-

Работа выполнена при финансовой поддержке Совета по грантам Президента РФ по государственной поддержке ведущих научных школ РФ (грант № НШ 5176.2010.9) и Российского фонда фундаментальных исследований (коды проектов 11-07-00109-а, 09-07-00095-а), а также в рамках государственного контракта Министерства образования и науки РФ № 07.514.11.4015.

ются на высоконадежной управляющей элементарной машине (центральном обозревателе), которая и определяет диагностический образ системы.

Опыт разработок в области самодиагностики крупномасштабных распределенных ВС показывает, что использование централизованного подхода приводит к снижению их производительности и нарушению важного принципа их построения: отказ одной ЭМ обуславливает отказ всей системы. Эти проблемы могут быть решены путем децентрализации процесса диагностирования.

Основной методологии децентрализованной самодиагностики, сформулированной в работах [3, 4], является предположение, что каждая исправная ЭМ может определить корректный диагностический образ всей системы, базируясь на результатах взаимных тестов других исправных ЭМ этой ВС. Таким образом, передача тестовой информации осуществляется только между исправными компонентами системы, что гарантирует изоляцию неисправных компонентов и препятствует их влиянию на формирование диагностического образа распределенной ВС. Эта идея получила развитие в работах [5–7].

Основными недостатками предложенных ранее алгоритмов являются:

- накладываемые ограничения на тестовую топологию (например, в [7] используется тестовая топология в виде дерева);
- отсутствие возможности изменения состояния ЭМ во время фазы тестирования [5];
- последовательная передача диагностических сообщений.

В настоящей работе предлагается децентрализованный алгоритм самодиагностики крупномасштабных распределенных ВС, обладающий следующими характеристиками:

- каждая исправная ЭМ тестируется только одной машиной;
- передача диагностической информации происходит только при изменении состояния тестируемой ЭМ;
- изменение состояния ЭМ может происходить во время фазы тестирования;
- фазы тестирования и распространения диагностической информации реализуются параллельно.

Для исследования алгоритма используется программное средство дискретного моделирования YACSIM [8].

1. Постановка задачи. Рассматривается распределенная ВС, состоящая из N элементарных машин (далее — узлов), соединенных каналами связи.

Каждый узел может находиться в исправном или неисправном состоянии. Исправный узел содержит информацию о том, какие узлы являются его соседями. Кроме того, он способен инициировать тестирование соседнего узла и отвечать на тестовые запросы своих соседей. В качестве теста используются сообщения типа: “Are you alive?” Исправный узел всегда отвечает на тестовый запрос в течение определенного периода времени (тайм-аута), передает диагностическую информацию своим соседям и может запрашивать их стать его тестерами.

Узел, находящийся в неисправном состоянии, не способен отвечать на любые сообщения от своих соседей, передавать им диагностическую информацию или запросы стать его тестерами. Таким образом, узлы в системе не могут информировать друг друга о своей неисправности.

Если в течение тайм-аута ответ на тестовый запрос от узла не получен, узел-тестер делает заключение о неисправности тестируемого им узла. Величина тайм-аута определяется как функция задержки каналов связи.

В любой момент времени узел распределенной ВС может перейти в неисправное состояние. В свою очередь неисправный узел может быть восстановлен и вновь введен в эксплуатацию. При этом узел получает всю необходимую информацию о своих соседях, но не обладает информацией о текущем диагностическом образе системы.

Контроль и диагностика неисправности каналов связи алгоритмом не предусматриваются, поэтому отсутствует различие между неисправностями тестируемого узла и канала связи, соединяющего его с тестером.

2. Описание алгоритма. В алгоритме предусмотрено, что узлы обнаруживают изменение состояния своих соседей, а затем передают эту диагностическую информацию всем узлам системы. Диагностическая информация включает события двух типов: переход узла из исправного состояния в неисправное и наоборот.

Обнаружение изменения состояния узлов в ВС осуществляется путем их периодического тестирования (сообщение типа `TestReq`). Каждый узел тестируется только одним исправным узлом. Если в течение определенного тайм-аута ответ (сообщение типа `TestResp`) получен, узел диагностируется как исправный, иначе — как неисправный. Непосредственно после диагностирования изменения состояния тестирующий узел начинает передачу диагностической информации (сообщение типа `NewEvent`) своим соседям, которые в свою очередь передают ее своим соседям и т. д.

После того как узел i отправил диагностическое сообщение соседнему узлу j , он ожидает от j подтверждения (сообщение типа `AckNewEvent`) получения сообщения в течение определенного тайм-аута. Если такое подтверждение не поступает, i начинает распространение информации о неисправности узла j . Таким образом, удается получить сведения об изменении состояния узлов до начала следующего этапа тестирования.

Если узел j исправен, то после получения диагностического сообщения от i он проверяет, является ли информация в этом сообщении новой, старой или такой же, какой обладает он.

Пусть t_j , t_i — время определения диагностического образа системы узлами j , i соответственно. Если $t_j = t_i$, то i и j имеют одинаковые данные о состоянии всех узлов в ВС и узел j не передает полученное им сообщение от i далее своим соседям.

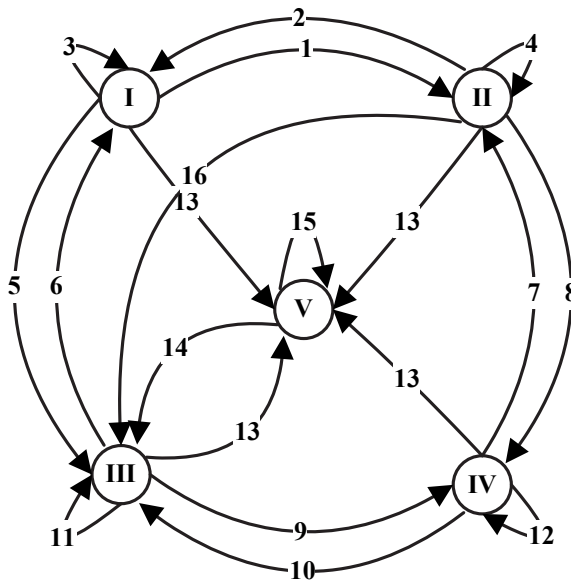
Если $t_j > t_i$, то узел j обладает более новой информацией о состоянии хотя бы одного из узлов распределенной ВС. В этом случае j передает i данные о диагностическом образе вычислительной системы.

Если $t_j < t_i$, то узел j содержит более старую информацию о состоянии хотя бы одного из узлов системы. В этом случае j обновляет свои данные и передает их своим соседям.

В результате выполнения описанного выше алгоритма тестирования один или несколько узлов в системе могут быть “брошены”, т. е. перестают тестироваться другими узлами.

Все неисправные узлы являются “брошенными”, поскольку после диагностирования их неисправности узел-тестер перестает их тестировать. Только после восстановления и нового ввода в эксплуатацию, т. е. после изменения состояния на исправное, такие узлы будут тестироваться вновь. Восстановленный узел обращается ко всем своим соседям по очереди с запросом на тестирование (сообщение типа `TestMeRepair`), до тех пор пока не получит сообщение, подтверждающее согласие стать его тестером (сообщение типа `AckTestMeRepair`). После этого узел-тестер начинает раунд тестирования. Кроме того, вновь исправный узел передает всем своим соседям информацию о своей исправности (сообщение типа `Repair`).

Исправный узел также может стать “брошенным”, в случае если его узел-тестер поменял свое состояние на неисправное. Так же как и в случае с неисправным “брошенным” узлом, он



Состояния узлов:

- I – исправный бездействующий
- II – исправный ожидающий ответа на тестовый запрос
- III – исправный "брошенный", бездействующий
- IV – исправный "брошенный", ожидающий ответа на тестовый запрос
- V – неисправный
- – отправка сообщения
- ← – получение сообщения
- ↓ – сообщение не получено в течение тайм-аута

События:

| | | | |
|---|---|----|--|
| 1 | E1: TestReq → | 9 | E9: TestReq → |
| 2 | E2 ₁ : ← TestResp E2 ₂ : ↓ TestResp E2 ₃ : ← TestMeRepair от тестируемого узла E2 ₄ : ← Repair от тестируемого узла | 10 | E10 ₁ : ← TestResp E10 ₂ : ↓ TestResp E10 ₃ : ← TestMeRepair от тестируемого узла E10 ₄ : ← Repair от тестируемого узла |
| 3 | E3 ₁ : ← TestReq E3 ₂ : ← NewEvent E3 ₃ : ← TestMe E3 ₄ : ← AckNewEvent E3 ₅ : ← TestMeRepair E3 ₆ : ← Repair E3 ₇ : ↓ AckNewEvent | 11 | E11 ₁ : ↓ AckTestMe E11 ₂ : ← NewEvent E11 ₃ : ← TestMe E11 ₄ : ← AckNewEvent E11 ₅ : ← TestMeRepair E11 ₆ : ← Repair E11 ₇ : ↓ AckTestMeRepair E11 ₈ : ← TestReq E11 ₉ : ↓ AckNewEvent |
| 4 | E4 ₁ : ← TestReq E4 ₂ : ← NewEvent E4 ₃ : ← TestMe E4 ₄ : ← AckNewEvent E4 ₅ : ← TestMeRepair E4 ₆ : ← Repair E4 ₇ : ↓ AckNewEvent | 12 | E12 ₁ : ← NewEvent E12 ₂ : ← TestMe E12 ₃ : ← AckNewEvent E12 ₄ : ← TestMeRepair E12 ₅ : ← Repair E12 ₆ : ↓ AckTestMe E12 ₇ : ← TestReq E12 ₈ : ↓ AckTestMeRepair E12 ₉ : ↓ AckNewEvent |
| 5 | E5 ₁ : ← NewEvent E5 ₂ : ← TestMe от тестера E5 ₃ : ← TestMeRepair от тестера E5 ₄ : ↓ AckNewEvent от тестера | 13 | E13: Узел не исправен |
| 6 | E6 ₁ : ← AckTestMe E6 ₂ : ← NewEvent E6 ₃ : ← AckTestMeRepair | 14 | E14: Узел восстановлен |
| 7 | E7 ₁ : ← AckTestMe E7 ₂ : ← AckTestMeRepair | 15 | E15: Узел не исправен |
| 8 | E8 ₁ : ← NewEvent E8 ₂ : ← TestMe от тестера E8 ₃ : ← TestMeRepair от тестера E8 ₄ : ↓ AckNewEvent от тестера | 16 | E16: ↓ TestResp |

Диаграмма состояний узла при выполнении алгоритма

должен обратиться ко всем своим соседям по очереди с запросом на тестирование (сообщение типа TestMe).

На рисунке приведена диаграмма состояний, описывающая поведение узла распределенной ВС при выполнении децентрализованного алгоритма самодиагностики системы.

Для хранения и сбора диагностической информации в ходе выполнения алгоритма самодиагностики на каждом узле j вычислительной системы используются следующие структуры данных:

1. Массив $events_j[N]$ счетчиков событий, где N – количество узлов в диагностируемой системе. Если $events_j[i]$ – четное число, то узел i исправен, иначе – не исправен. Начальное значение для элементов массива – 0.

2. Массив $testnbs_j[N]$, где N – количество узлов в диагностируемой системе; $testnbs_j[i] = 0$, если узлы i и j не являются соседями; $testnbs_j[i] = 1$, если узел i тестируется узлом j ; $testnbs_j[i] = 2$, если узел i тестирует узел j ; $testnbs_j[i] = 3$, если узлы i и j являются соседями, но не тестируют друг друга; $testnbs_j[i] = 4$, если узлы i и j тестируют друг друга.

3. Результаты моделирования. Моделирование предложенного алгоритма проводилось с использованием средства дискретного моделирования YACSIM [8] – событийно- и процессно-ориентированного инструментария моделирования, позволяющего создавать взаимодействующие между собой процессы.

Программная реализация децентрализованного алгоритма самодиагностики распределенной ВС включает следующие процессы:

- Init, выполняющий начальную инициализацию и создающий процессы MsgHandler и Lost;
- MsgHandler, отвечающий за получение и обработку всех видов сообщений (см. рисунок);
- Lost, выполняющий рассылку сообщений TestMe для поиска соседнего узла-тестера;
- Test, реализующий рассылку тестовых запросов;
- Event, выполняющий сравнение присылаемой диагностической информации с имеющейся на узле;
- SendEvent, рассылающий диагностическую информацию соседним узлам.

При моделировании каждый узел ВС переключался между диагностическим алгоритмом и обычной рабочей нагрузкой. Время выполнения рабочей нагрузки на узле являлось экспоненциально распределенной случайной величиной со средним значением, равным одной единице времени. По истечении этого времени направлялся повторный запрос на использование процессора для выполнения рабочей нагрузки. Эти запросы обрабатывались в порядке очереди FIFO. Фоновые процессы, реализующие алгоритм самодиагностики, осуществляли запросы процессора через ту же очередь FIFO. Общее время моделирования составляло 1000 единиц времени.

При моделировании предложенного алгоритма использовались следующие виды задержек:

- время формирования тестового запроса (две единицы времени);
- время формирования ответа на тестовый запрос или подтверждающего сообщения (одна единица времени);
- время обработки ответа на тестовый запрос (одна единица времени);
- время обработки диагностического сообщения и обновления информации на узле (2,5 единицы времени);
- время определения номера соседнего узла для отправки ему диагностического сообщения (0,1 единицы времени);
- время формирования диагностического сообщения (2,5 единицы времени);
- время, затрачиваемое на пересылку любого сообщения от одного узла другому (одна единица времени);
- интервал между тестами (500 единиц времени).

Исследование разработанного алгоритма самодиагностики распределенных ВС проводилось для следующих топологий: двумерная решетка (4×4), двумерный тор (4×4), четырехмерный гиперкуб и трехмерная решетка ($4 \times 4 \times 4$). Моделировалась ситуация, когда узел 1 переходил в неисправное состояние в момент времени 10 при первом раунде тестирования. При этом использовались 100 различных начальных значений для генерации случайных величин. В таблице приведены средние значения полученных оценок. Проанализировав результаты, можно сделать вывод, что увеличение размерности топологии системы не приводит к существенному увеличению времени информирования всех узлов ВС о текущем диагностическом образе.

Для топологии “трехмерная решетка” ($4 \times 4 \times 4$) получены также оценки загрузки системы при выполнении предложенного алгоритма в единицах времени и процентах общего времени моделирования. Результаты показывают, что при выполнении алгоритма в отсутствие неисправностей загрузка системы очень мала (15,1 (1,51%)) и при наличии одной неисправности незначительно увеличивается (66,47 (6,647%)).

Заключение. В работе предложен децентрализованный алгоритм самодиагностики распределенных ВС, характеризующийся параллельной реализацией фаз тестирования и распространения диагностической информации.

Результаты моделирования

| Полученные оценки | Двумерная решетка (4 × 4) | Двумерный тор (4 × 4) | Четырехмерный гиперкуб | Трехмерная решетка (4 × 4 × 4) |
|---|------------------------------|--------------------------|---------------------------|--------------------------------------|
| Обнаружение неисправности | 12,012 | 12,012 | 12,012 | 11,200 |
| Момент времени, в который последний узел системы узнает о неисправности | 79,71 | 58,22 | 63,99 | 120,32 |
| Количество передаваемых диагностических сообщений | 26 | 37 | 40 | 207 |

Проведено моделирование разработанного алгоритма с использованием средства дискретного моделирования YACSIM. Исследование проводилось для распространенных топологий распределенных вычислительных систем. Результаты моделирования показывают, что при выполнении алгоритма в отсутствие неисправностей загрузка системы очень мала и увеличивается незначительно при наличии одной неисправности.

В дальнейшем планируется провести исследование алгоритма при наличии нескольких событий (в том числе событий восстановления узла после отказа) и реализовать предложенный алгоритм как часть системного программного обеспечения самодиагностики пространственно-распределенной мультикластерной вычислительной системы Центра параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики и Института физики полупроводников им. А. В. Ржанова СО РАН.

Список литературы

1. ХОРОШЕВСКИЙ В. Г. Архитектура вычислительных систем. М.: МГТУ им. Н. Э. Баумана, 2008.
2. PREPARATA F. P., METZE G., SHEN R. T. On the connection assignment problem of diagnosable systems // IEEE Trans. Electron. Comput. Dec. 1967. V. EC-16, N 6. P. 848–854.
3. ЕВРЕИНОВ Э. В., ХОРОШЕВСКИЙ В. Г. Однородные вычислительные системы. Новосибирск: Наука. Сиб. отд-ние, 1978.
4. KUHJ J. G., REDDY S. M. Fault-diagnosis in fully distributed systems // Proc. of the 11th Intern. symp. on fault-tolerant comput., Portland (USA), June 24–26, 1981. Piscataway: IEEE Service center, 1981. P. 100–105.
5. BAGCHI A., HAKIMI S. L. An optimal algorithm for distributed system level diagnosis // Proc. of the 21st Intern. symp. on fault-tolerant comput., Montreal (Canada), June 25–27, 1991. Los Alamitos: IEEE Comput. Soc. Press, 1991. P. 214–221.
6. STAHL M., BUSKENS R., BIANCHINI R. On-line diagnosis in general topology networks // IEEE Workshop on fault-tolerant parallel and distributed systems, Amherst (USA), July 6–7, 1992. Los Alamitos: IEEE Computer Society Press, 1992. P. 114–121.
7. BIANCHINI R., STAHL M., BUSKENS R. The Adapt2 on-line diagnosis algorithm for general topology networks // Proc. of the IEEE Global telecommunications conf., Orlando (USA), Dec. 6–9, 1992. N. Y.: IEEE, 1992. V. 1. P. 610–614.
8. JUMP R. J. YACSIM: Reference Manual. [Электрон. ресурс]. 1993. V. 2.1. Режим доступа: <http://oucsace.cs.ohiou.edu/avinashk/classes/ee663/yac.ps>.

Молдованова Ольга Владимировна — канд. техн. наук, доц. Сибирского государственного университета телекоммуникаций и информатики; тел.: (383) 269-82-75; e-mail: ovm@csc.sibsutis.ru

Дата поступления — 20.01.12 г.