Децентрализованная самодиагностика распределённых вычислительных систем*

О.В. Молдованова

Сибирский государственный университет телекоммуникаций и информатики

Предложен децентрализованный алгоритм самодиагностики распределённых вычислительных систем (ВС), характеризующийся параллельным выполнением фаз тестирования и распространения диагностической информации. Проведено моделирование алгоритма с использованием библиотеки дискретного моделирования YACSIM. Приведены результаты моделирования для распространённых топологий распределённых ВС.

1. Введение

В последнее время постоянно увеличивается число трудоёмких задач, решаемых на распределённых вычислительных системах (ВС) [1]. Основным функциональным элементом таких систем является элементарная машина (ЭМ). Распределённые ВС характеризуются большемасштабностью – количество ЭМ в их составе может достигать $10^5 – 10^6$. Несмотря на высокую надёжность микроэлектронной базы, вероятность возникновения отказов в распределённых ВС повышается с ростом количества элементарных машин. Следовательно, организация отказоустойчивого функционирования таких систем требует создания алгоритмических и программных средств самоконтроля и самодиагностики.

В течение нескольких десятилетий был предложен ряд методик диагностики отказов в вычислительных системах [2–9]. Большинство работ базируется 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 собираются на высоконадёжной управляющей элементарной машине (центральном обозревателе), которая и определяет диагностический образ системы.

Опыт разработок в области самодиагностики большемасштабных распределённых вычислительных систем показывает, что централизованный подход ведёт к снижению производительности ВС и нарушает важный принцип их построения: отказ одной ЭМ влечёт за собой отказ всей системы. Эти проблемы могут быть решены при децентрализации процесса диагностирования.

Методология децентрализованной самодиагностики была сформулирована в работах [3, 4]. Её основой является предположение, что каждая исправная ЭМ может определить корректный диагностический образ всей системы, базируясь на результатах взаимных тестов других исправных элементарных машин этой ВС. Таким образом, передача тестовой информации осуществляется только между исправными компонентами системы, что гарантирует изоляцию неисправных и препятствует их влиянию на формирование диагностического образа распределённой ВС. В дальнейшем эта идея получила развитие в работах других исследователей [5–9].

Среди основных недостатков ранее предложенных алгоритмов следует отметить:

- накладываемые ограничения на тестовую топологию (например, в [7] используется тестовая топология в виде дерева);
- изменение состояния ЭМ не может происходить во время фазы тестирования [5];

* Работа выполнена при поддержке Совета по грантам Президента РФ (ведущая научная школа НШ 5176.2010.9), Российского фонда фундаментальных исследований (гранты 11-07-00109-а, 09-07-00095-а) и в рамках государственного контракта № 07.514.11.4015 с Минобрнауки РФ.

- использование диагностической модели сравнения, позволяющей установить лишь факт неисправности, т.е. выполнить самоконтроль ВС, без возможности диагностировать, какая конкретно ЭМ неисправна [9];
- последовательная передача диагностических сообщений.

В работе предлагается децентрализованный алгоритм самодиагностики распределённых ВС, обладающий следующими характеристиками:

- каждая исправная ЭМ тестируется только одной другой 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 выполняются параллельно.

Для исследования алгоритма используется программное средство дискретного моделирования – YACSIM [10].

2. Постановка задачи

Рассматривается распределённая вычислительная система, состоящая из N элементарных машин (в дальнейшем узлов), соединённых друг с другом каналами связи.

Каждый узел может находиться в исправном или неисправном состоянии. Исправный узел обладает информацией 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 вида «Are you alive?». Исправный узел всегда отвечает на тестовый запрос в течение определённого периода времени (тайм-аута), передаёт 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, то узел-тестер делает заключение о неисправности тестируемого им узла. Величина тайм-аута определяется как функция задержки каналов связи.

Узел распределённой ВС в любой момент времени может перейти в неисправное состояние. В свою очередь неисправный узел может быть восстановлен и снова введён в эксплуатацию. При этом он получает всю необходимую информацию, касающуюся своих соседей, но не обладает информацией о текущем диагностическом образе системы.

Контроль и диагностика неисправности каналов связи алгоритмом не предусматривается. Поэтому не делается различия между неисправностью тестируемого узла и канала связи, соединяющего его с тестером.

Предполагается, что в начальный момент времени все узлы ВС исправны.

3. Децентрализованный алгоритм самодиагностики распределённых вычислительных систем

В работе предлагается децентрализованный алгоритм самодиагностики DSLD (Distributed System-Level Diagnostics) распределённых ВС. Алгоритмом предусматривается, что узлы обнаруживают изменение состояния своих соседей, а затем передают эту диагностическую информацию всем узлам системы. Диагностическая информация состоит из событий двух видов: переход узла из исправного состояния в неисправное и наоборот.

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

- массив $events_j[N]$ счётчиков событий, где N – количество узлов в диагностируемой системе. Если $events_j[i]$ – чётное число, то узел i исправен, иначе – не исправен. Начальное значение для элементов массива – 0;
- 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.

Обнаружение изменения состояния узлов в вычислительной системе осуществляется путём их периодического тестирования (сообщение типа «TestReq»). Каждый узел тестируется только одним исправным узлом. Если в течение определённого тайм-аута ответ (сообщение типа «TestResp») получен, узел диагностируется как исправный, иначе – как неисправный. Сразу после диагностирования изменения состояния тестирующий узел начинает передачу диагностической информации (сообщение типа «NewEvent») своим соседям, а те в свою очередь своим и т.д.

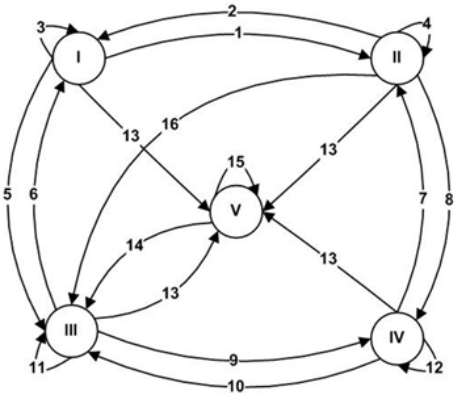
После того как узел i отправил диагностическое сообщение соседнему с ним узлу j, он ожидает от j подтверждения (сообщение типа «AckNewEvent») получения сообщения в течение определённого тайм-аута. Если такое подтверждение не поступает, i начинает распространение информации о неисправности узла j. Таким образом, удаётся получить сведения об изменении состояния узлов, не дожидаясь следующего раунда тестирования.

Если j исправен, то после получения диагностического сообщения от i он проверяет, является ли информация в этом сообщении новой, старой или такой же, какой обладает и он. Для этого используется сравнение значений из локальной версии массива $events_j$ и значений из массива $events_i$, полученного от узла i.

Если $events_j[k] = events_i[k]$ для всех $k \in \{1, 2, ..., N\}$, i и j имеют одинаковые данные о состоянии всех узлов в ВС. И узел j не передаёт полученное им сообщение от i далее своим соседям.

Если $events_j[k] > events_i[k]$ хотя бы для одного $k \in \{1, 2, ..., N\}$, то j обладает более новой информацией о состоянии некоторых узлов распределённой ВС. В этом случае j передаёт i свои данные о диагностическом образе вычислительной системы.

Если $events_j[k] < events_i[k]$ хотя бы для одного $k \in \{1, 2, ..., N\}$, то j содержит более старую информацию о состоянии некоторых узлов системы. В таком случае j обновляет свои данные и передаёт их далее своим соседям.



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

Рис. 1. Диаграмма состояний узла при выполнении алгоритма.

Таким образом, алгоритм DSLD не использует время формирования диагностического образа на конкретном узле ВС для определения актуальности этого образа, а значит дополнительная синхронизация часов в узлах распределённой вычислительной системы не требуется.

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

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

Исправный узел также может стать «брошенным», в случае если его узел-тестер поменял своё состояние на неисправное. Так же как и неисправный «брошенный» узел, он должен обратиться ко всем своим соседям по очереди с запросом на тестирование (сообщение типа «TestMeRepair»).

На рис. 1 приведена диаграмма состояний, описывающая поведение узла распределённой ВС при выполнении децентрализованного алгоритма самодиагностики системы.

4. Результаты моделирования

Моделирование алгоритма DSLD проводилось с использованием средства дискретного моделирования YACSIM [10]. Это событийно- и процессно-ориентированный инструментарий моделирования, позволяющий создавать взаимодействующие друг с другом процессы.

Программная реализация децентрализованного алгоритма самодиагностики распределённой ВС включает в себя следующие процессы:

- Init, выполняющий начальную инициализацию и создающий процессы MsgHandler и Lost;
- MsgHandler, отвечающий за получение и обработку всех видов сообщений (см. рис. 1);
- 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 распределённой случайной величиной со средним значением, равным 1 единице времени. По истечении этого времени сразу же выполнялся 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 самую очередь FIFO. Общее время моделирования составляло 1000 единиц времени.

Следующие виды задержек использовались при моделировании предложенного алгоритма:

- время формирования тестового запроса (2 единицы времени);
- время формирования ответа на тестовый запрос или подтверждающего сообщения (1 единица времени);
- время обработки ответа на тестовый запрос (1 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 (1 единица времени);
- интервал между тестами (500 единиц времени).

Исследование разработанного алгоритма самодиагностики распределённых вычислительных систем проводилось для топологий: двумерная решётка (4 x 4), двумерный тор (4 x 4), четырёхмерный гиперкуб и трёхмерная решётка (4 x 4 x 4). Моделировалась ситуация, когда узел 1 переходил в неисправное состояние в момент времени 9 при первом раунде тестирования. При этом использовались 100 различных начальных значений для генерации случайных величин.

На рис. 2 показаны схема тестирования (а) и передача диагностических сообщений при обнаружении неисправности в узле 1 (б) для топологии двумерная решётка (4 x 4). Неисправность узла 1 обнаруживается его тестером, узлом 2, после чего неисправный узел полностью исключается из процесса диагностирования системы. Сплошными линиями на рис. 2, б показаны новые диагностические сообщения, после получения которых узел-приёмник начинает передачу диагностического образа системы в своей локальной окрестности. Сообщения, представленные пунктирными линиями (рис. 2, б), содержат данные, не отличающиеся от локальных данных узлов-приёмников.

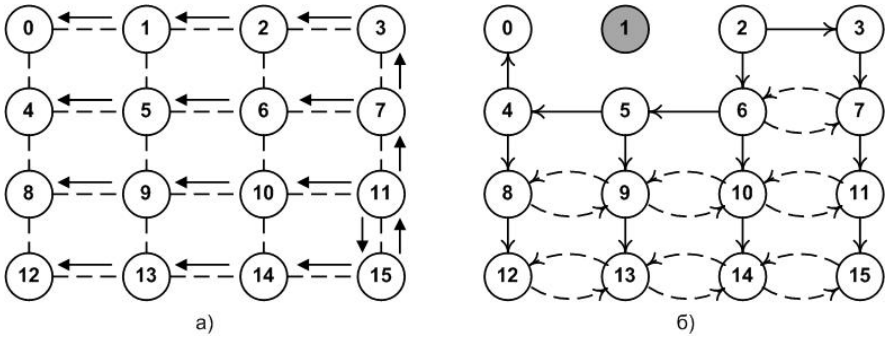


Рис. 2. Выполнение алгоритма самодиагностики на топологии двумерная решётка (4 x 4):
а) схема тестирования; б) обнаружение неисправности в первом узле;
○ – исправный узел; ● – неисправный узел;
← – направление тестирования (от тестера к тестируемому узлу);
⟵ – передача новой диагностической информации;
⇠ – передача диагностической информации, уже имеющейся на узле-приёмнике.

В табл. 1 приведены средние значения полученных оценок. Проанализировав результаты, можно сделать вывод, что увеличение размерности топологии системы не приводит к большому увеличению времени информирования всех узлов распределённой ВС о текущем диагностическом образе.

Таблица 1. Результаты моделирования

Топологии / Полученные оценки	Двумерная решётка (4 x 4)	Двумерный тор (4 x 4)	Четырёхмерный гиперкуб	Трёхмерная решётка (4 x 4 x 4)
Обнаружение неисправности	9,5	9,2	9,76	9,36
Время, когда последний узел системы узнаёт о неисправности	78,2	58,9	60,89	126,47
Количество передаваемых диагностических сообщений	26	39	38	204

Для топологии трёхмерная решётка (4 x 4 x 4) получены также оценки загрузки системы при выполнении предложенного алгоритма в единицах времени и в процентном соотношении от общего времени моделирования (табл. 2). Результаты показывают, что загрузка системы при выполнении алгоритма в отсутствии неисправностей очень мала и незначительно

увеличивается при наличии одной неисправности.

Таблица 2. Результаты моделирования для топологии трёхмерная решётка

Загрузка в отсутствии неисправностей	15,1 (1,51%)
Загрузка при одной неисправности	66,47 (6,647%)

5. Заключение

В работе предложен децентрализованный алгоритм самодиагностики DSLD распределённых ВС, характеризующийся параллельным выполнением фаз тестирования и распространения диагностической информации.

Проведено моделирование разработанного алгоритма с использованием средства дискретного моделирования YACSIM. Исследование проводилось для распространённых топологий распределённых вычислительных систем. Результаты моделирования показывают, что загрузка системы при выполнении алгоритма в отсутствии неисправностей очень мала, и она увеличивается незначительно при наличии неисправностей.

В дальнейшем планируется провести исследования алгоритма при наличии нескольких событий, в том числе событий восстановления узла после отказа, и реализовать предложенный алгоритм как часть системного программного обеспечения самодиагностики пространственно-распределённой мультикластерной вычислительной системы Центра параллельных вычислительных технологий федерального государственного образовательного бюджетного учреждения высшего профессионального образования «Сибирский государственный университет телекоммуникаций и информатики» и Института физики полупроводников им. А.В. Ржанова СО РАН.

Литература

1. Хорошевский В.Г. Архитектура вычислительных систем. – М.: МГТУ им. Н.Э. Баумана, 2008. – 520 с.
2. Preparata F.P., Metze G., Chien R.T. On the connection assignment problem of diagnosable systems // IEEE Trans. Electron. Comput. Dec. 1967. – vol. EC-16. – no. 6. – pp. 848–854.
3. Евреинов Э.В., Хорошевский В.Г. Однородные вычислительные системы. – Новосибирск: Наука, 1978. – 319 с.
4. Kuhl J.G., Reddy S.M. Fault-diagnosis in fully distributed systems // Proc. 11th Int. Symp. Fault-Tolerant Computing, June 1981. – pp. 100–105.
5. Bagchi A., Hakimi S.L. An optimal algorithm for distributed system level diagnosis // Proc. 21st Int. Symp. Fault-Tolerant Computing, June 1991.
6. Stahl M., Buskens R., Bianchini R. On-line diagnosis in general topology networks // IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, July 1992. – pp. 114–121.
7. Bianchini R., Stahl M., Buskens R. The Adapt2 on-line diagnosis algorithm for general topology networks // Proc. Globecorn, 1992. – pp. 610–614.
8. Bartha T. Efficient system-level fault diagnosis of large multiprocessor systems: thesis for the degree of Doctor of Philosophy. – Budapest, 2000. – 157 p.
9. Albini L.C.P., Duarte, Jr. E.P., Ziwich R.P. A generalized model for distributed comparison-based system-level diagnosis // J. Braz. Comp. Soc., 2005. – vol.10. – no. 3. – pp. 44–56. – ISSN 0104-6500.
10. Jump R.J. YACSIM: Reference Manual, V. 2.1. – March, 1993. – Режим доступа: http://oucsace.cs.ohiou.edu/~avinashk/classes/ee663/yac.ps (14.02.2012).