УДК 004.052.32

Адаптивный алгоритм децентрализованной самодиагностики распределённых вычислительных систем различных топологий

О. В. Молдованова[1]

В работе предлагается адаптивный алгоритм децентрализованной самодиагностики распределённых вычислительных систем (ВС), характеризующийся параллельным выполнением фаз тестирования и распространения диагностической информации и не накладывающий ограничений на тестовую топологию системы. Приводятся результаты моделирования алгоритма для популярных топологий распределённых ВС. В результате исследований адаптивного алгоритма выявлены ситуации, при которых неисправные узлы системы формируют барьеры и циклы, препятствующие самодиагностике распределённой ВС. Разработан и исследован алгоритм обнаружения барьеров и циклов для различных топологий распределённых ВС.

Ключевые слова: децентрализованная самодиагностика, адаптивный алгоритм, распределённая вычислительная система, барьер, цикл.

1. Введение

Распределённые вычислительные системы (ВС) являются распространённым средством, используемым для решения трудоёмких научных, экономических и инженерных задач [1]. Использование распределённых ВС определяется их надёжностью и способами организации функционирования. Основная функционально-структурная единица вычислительных ресурсов в таких системах – элементарная машина (ЭМ). В качестве ЭМ могут быть использованы ЭВМ, вычислительные ядра, многоядерные процессоры, вычислительные узлы, оснащённые средствами межмашинной коммутации. Распределённые ВС характеризуются большемасштабностью – количество ЭМ в их составе может превышать миллион (например, в ВС Titan – Cray XK7, произведённой компанией Cray Inc. и занимающей первую строчку в рейтинге Top500 [2], количество вычислительных ядер равно 560640). Несмотря на высокую надёжность микроэлектронной базы, вероятность возникновения отказов в распределённых ВС повышается с ростом количества элементарных машин в них. Следовательно, организация отказоустойчивого функционирования таких систем требует создания программных средств, обеспечивающих контроль и диагностику их функционирования. Применительно к распределённым системам можно говорить о самоконтроле и самодиагностике.

[1] Работа выполнена при поддержке Совета по грантам Президента РФ (ведущая научная школа НШ-2175.2012.9), Российского фонда фундаментальных исследований (гранты 11-07-00109-а, 12-07-00019-а, 12-07-00106-а, 12-07-00145-а, 12-07-00188-а, 12-07-31015 мол_а), в рамках государственного контракта № 07.514.11.4015 с Минобрнауки РФ и интеграционного проекта №39 СО РАН.

Существуют два основных подхода к организации самодиагностики вычислительных систем. Первый подход (централизованная самодиагностика) был впервые описан в [3] и предполагает, что 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 высоконадёжном центральном обозревателе, который и определяет диагностический образ системы.

Второй подход (децентрализованная самодиагностика) был сформулирован в работах [4, 5]. Его 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 [6–10].

Опыт разработок в области самодиагностики 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.

Среди основных недостатков ранее предложенных алгоритмов децентрализованной самодиагностики следует отметить:

– накладываемые ограничения на тестовую топологию (например, в [8] используется тестовая топология в виде дерева);

– отсутствие возможности изменения состояния ЭМ во время фазы тестирования [6];

– использование диагностической модели сравнения, позволяющей установить лишь факт неисправности, т.е. выполнить самоконтроль ВС, без возможности диагностировать, какая конкретно ЭМ неисправна [10];

– передача диагностической информации после раунда тестирования.

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

– каждая исправная ЭМ тестируется только одной другой исправной машиной;

– передача диагностической информации происходит только при изменении состояния тестируемой ЭМ;

– изменение состояния ЭМ может происходить во время фазы тестирования;

– тестовая топология зависит от диагностического образа системы, полученного на предыдущем раунде тестирования (адаптивность);

– фазы тестирования и распространения диагностической информации выполняются параллельно.

2. Постановка задачи

Рассматривается распределённая вычислительная система, состоящая из N элементарных машин (в дальнейшем – узлов), соединённых 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, отвечать в течение определенного тайм-аута на тестовые запросы своих соседей, передавать диагностическую информацию своим соседям и запрашивать их стать его тестерами. В качестве теста используются сообщения вида «Are you alive?».

Узел, находящийся в неисправном состоянии, не способен отвечать на любые сообщения от своих соседей, передавать им диагностическую информацию или запросы стать его тесте-

рами. 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.

В любой момент времени узел распределённой ВС 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 предусматриваются, поэтому отсутствует различие между неисправностью тестируемого узла и канала связи, соединяющего его с тестером.

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

3. Адаптивный алгоритм децентрализованной самодиагностики распределённых ВС

В работе предлагается адаптивный алгоритм DSLD (Distributed System-Level Diagnostics) децентрализованной самодиагностики распределённых ВС. Алгоритмом предусматривается, что узлы обнаруживают изменение состояния соседних с ними узлов, а затем передают эту диагностическую информацию остальным узлам системы. Распространение диагностической информации инициируется событиями 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 ВС 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. Элементы массива увеличиваются на единицу при каждом диагностировании события;
- 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.

Во время начальной фазы (фазы инициализации) каждый узел j обращается к своим соседям с просьбой стать его тестером (сообщение TestMe). Номер i соседнего узла, которому он отправляет запрос на тестирование, определяется по формуле: $i=(j+k) \bmod N$, где $0 \le j < N, 1 \le k < N$, N – количество узлов в вычислительной системе. Если в течение определённого тайм-аута ответ (сообщение типа AckTestMe) получен, то узел i становится узлом-тестером для j. Иначе узел j считает, что i не исправен, и начинает распространение информации об этом событии. Таким образом, диагностика отказов узлов системы происходит и во время фазы инициализации.

Обнаружение изменения состояния узлов в ВС 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 узел-тестер начинает передачу диагностической информации (сообщение типа NewEvent) своим соседям.

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 $k \in \{1,2,...,N\}$, то *i* и *j* имеют одинаковые данные о состоянии всех узлов в ВС. 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. Узел *j* обновляет свои данные и передаёт их далее своим соседям.

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. 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 тестироваться вновь. 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. В этом случае он должен обратиться ко всем своим исправным соседям по очереди с запросом на тестирование (сообщение типа TestMe).

4. Результаты моделирования

Моделирование алгоритма DSLD проводилось с использованием средства дискретного моделирования YACSIM [11]. Это событийно- и процессно-ориентированный инструментарий моделирования, позволяющий создавать взаимодействующие друг с другом процессы.

Программная реализация адаптивного алгоритма децентрализованной самодиагностики распределённых 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;
- 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;
- Init_Repair, выполняющий инициализацию и создание процессов MsgHandler и Repair для восстановленного узла;
- Repair, рассылающий сообщения после восстановления узла.

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 × 4), двумерный тор (4 × 4), четырёхмерный гиперкуб, трёхмерный тор (4 × 4 × 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.

Для полностью регулярных топологий, таких как тор и гиперкуб, исследования проводились для двух различных схем передачи диагностических сообщений. В первом случае каждый узел передавал диагностическую информацию по всем линиям связи, во втором были задействованы только линии связи, не участвующие в тестировании. Для топологии типа «решётка» вторая схема передачи сообщений не подходит, она приводит к неполному информированию узлов в системе о произошедшем событии.

В табл. 1 приведены средние значения полученных оценок. Проанализировав результаты, можно сделать вывод, что вторая схема передачи диагностической информации уменьшает количество сообщений, но незначительно увеличивает диагностическую латентность алгоритма.

На рис. 1 показана зависимость количества сообщений от выполняемой фазы адаптивного алгоритма и используемой схемы передачи диагностической информации для топологии «трёхмерный тор» (4 × 4 × 4). Наибольшее количество сообщений приходится на фазы, связанные с появлением событий: фаза тестирования на первом раунде (здесь происходит совмещение тестирования и передачи диагностической информации в связи с обнаружением неисправной ЭМ) и фаза восстановления. Во время фазы тестирования на втором раунде со-

бытий не происходит, поэтому количество передаваемых сообщений заметно меньше. Применение схемы передачи диагностической информации, не использующей линии связи, задействованные в тестировании, сокращает количество сообщений на 14.3%.

Таблица 1. Результаты моделирования

Топологии / Полученные оценки	Решётка (4 × 4)	Двумерный тор (4 × 4) Схема 1	Двумерный тор (4 × 4) Схема 2	Гиперкуб (4) Схема 1	Гиперкуб (4) Схема 2
Обнаружение события	9.5	9.2	9.2	9.8	9.8
Время, когда последний узел системы узнает о событии	78.2	58.9	65.3	60.89	68.5
Количество передаваемых диагностических сообщений	52	78	60	76	62

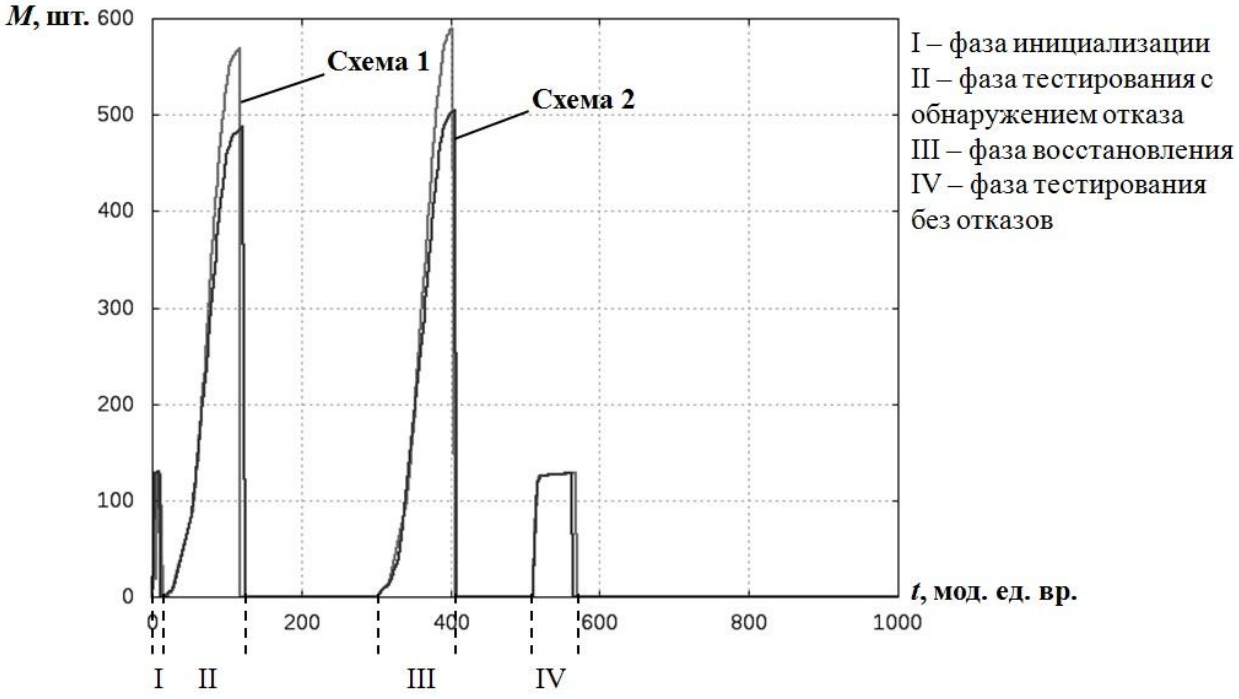


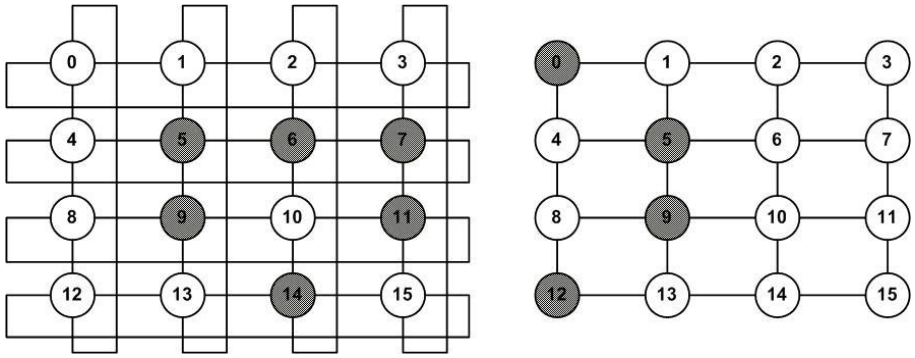
Рис. 1. Зависимость количества сообщений от выполняемой фазы алгоритма самодиагностики и используемой схемы передачи сообщений для топологии «трёхмерный тор» (4 × 4 × 4)

Для топологии «трёхмерный тор» (4 × 4 × 4) получены также оценки загрузки системы при выполнении предложенного алгоритма в единицах времени и в процентном соотношении от 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 и одного восстановления увеличивается до 131.6 (13.16%) в случае первой схемы передачи диагностической информации и до 95.6 (9.56%) – при использовании второй схемы.

5. Алгоритм обнаружения барьеров и циклов

В результате исследований разработанного алгоритма самодиагностики выявлены ситуации, при которых неисправные ЭМ формируют барьеры и циклы, препятствующие самодиагностике распределённой ВС. Барьеры и циклы из неисправных машин разбивают систему на две отдельные подсистемы, которые не могут обмениваться своими диагностическими образами. Это приводит к неполному диагностированию распределённой системы.

Барьер (рис. 2б) представляет собой непрерывную цепочку соседних неисправных ЭМ, простирающуюся от одной границы топологии до другой. Таким образом, барьеры могут возникать только в топологиях типа «решётка». Цикл – это кольцо из соседних неисправных ЭМ, изолирующее свою внутреннюю область от остальной системы (рис. 2а). Циклы возможны в распределённых ВС различных топологий.



а) б)

Рис. 2. Примеры барьеров и циклов из неисправных элементарных машин:
а) цикл; б) барьер

Алгоритм обнаружения барьеров и циклов приведён на рис. 3.

Входными данными для алгоритма является множество *FAULTS* номеров узлов, диагностированных как неисправные. Множество *BARRIER* формируется во время выполнения алгоритма, в него помещаются номера узлов, которые образуют барьер или цикл. *BORDER* – это множество, состоящее из двух элементов: номера начального узла барьера и номера конечного узла. Во множество *SUSPECTS* помещаются номера неисправных узлов, «подозреваемых» в формировании барьера или цикла.

При выполнении алгоритма номера u_i узлов последовательно удаляются из *SUSPECTS* и помещаются в *BARRIER*. Затем анализируются все соседние с u_i узлы. Если оказывается, что соседний неисправный узел u_j уже находится во множестве *SUSPECTS*, но при этом ещё не помещен в *BARRIER*, значит, цикл найден, можно прекращать работу алгоритма. Иначе узел u_j добавляется к множеству *SUSPECTS*.

Барьер найден в том случае, если очередной узел из *SUSPECTS* является граничным узлом, и множество *BORDER* – непустое.

Проведены экспериментальные исследования предложенного алгоритма обнаружения барьеров и циклов, которые показали его работоспособность для различных топологий распределённых ВС. Применение этого алгоритма позволяет обеспечить целостность диагностического образа системы.

```
Вход: FAULTS – множество номеров узлов, диагностированных как неисправные
BARRIER = Ø
BORDER = Ø
SUSPECTS = FAULTS[0]
while SUSPECTS ≠ Ø do
      BARRIER = BARRIER U u_i
      удалить элемент u_i из SUSPECTS
      if u_i является граничным узлом then
            if BORDER ≠ Ø then
                  BORDER[1] = u_i
                  exit // Барьер найден!
            else
                  BORDER[0] = u_i
            end if
      end if
      for all u_j ∈ NEIGHBOURS(u_i) do
            if u_j ∈ FAULTS and not(u_j ∈ BARRIER) then
                  if u_j ∈ SUSPECTS then
                        BARRIER = BARRIER U u_j
                        exit // Цикл найден!
                  end if
                  SUSPECTS = SUSPECTS U u_j
            end if
      end for
end while
Выход: BARRIER – множество номеров узлов, формирующих барьер или цикл
```

Рис. 3. Псевдокод алгоритма обнаружения барьеров и циклов

6. Заключение

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

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

Исследования адаптивного алгоритма самодиагностики показали возможность возникновения барьеров и циклов, препятствующих формированию целостного диагностического образа распределённой ВС. Предложен алгоритм обнаружения барьеров и циклов, исследования которого показали его работоспособность для различных топологий вычислительных систем.

Литература

1. *Хорошевский В.Г.* Распределённые вычислительные системы с программируемой структурой // Вестник СибГУТИ. 2010. №2. С. 3–41.
2. Top500 List – November 2012. [Электронный ресурс]. URL: `http://www.top500.org/list/2012/11/` (дата обращения: 01.03.2013).
3. *Preparata F.P., Metze G., Chien R.T.* On the connection assignment problem of diagnosable systems // IEEE Trans. Electron. Comput., 1967. Vol. EC-16. No. 6. P. 848–854.
4. *Евреинов Э.В., Хорошевский В.Г.* Однородные вычислительные системы. Новосибирск: Наука, 1978.

5. *Kuhl J., Reddy S.* Fault-diagnosis in fully distributed systems // Proc. of the 11th Int. Symp. on Fault-Tolerant Comp., 1981. P. 100–105.
6. *Bagchi A., Hakimi S.L.* An optimal algorithm for distributed system level diagnosis // Proc. of the 21st Int. Symp. on Fault-Tolerant Comp., 1991. Los Alamitos, 1991. P. 214–221.
7. *Stahl M., Buskens R., Bianchini R.* On-line diagnosis in general topology networks // IEEE Workshop on fault-tolerant parallel and distributed systems, 1992. Los Alamitos, 1992. P. 114–121.
8. *Bianchini R., Stahl M., Buskens R.* The Adapt2 on-line diagnosis algorithm for general topology networks // Proc. of IEEE Global Telecommunications Conf., 1992. Vol. 1. P. 610–614.
9. *Bartha T.* Efficient system-level fault diagnosis of large multiprocessor systems: thesis for the degree of Doctor of Philosophy. Budapest, 2000.
10. *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. P. 44–56.
11. *Jump R.J.* YACSIM: Reference Manual, V. 2.1. [Электронный ресурс]. URL: `http://oucsace.cs.ohiou.edu/~avinashk/classes/ee663/yac.ps` (дата обращения: 01.03.2013).

Статья поступила в редакцию 05.03.2013

Молдованова Ольга Владимировна
к.т.н., доцент кафедры вычислительных систем ФГОБУ ВПО «СибГУТИ» (630102, Новосибирск, ул. Кирова, 86) тел. (383) 2-698-275, e-mail: `ovm@csc.sibsutis.ru`.

Adaptive decentralized self-diagnosis algorithm for large-scale distributed computer systems

O.V. Moldovanova

In this article, an adaptive decentralized self-diagnosis algorithm for large-scale distributed computer systems (CS) is proposed. The algorithm is characterized by parallel execution of testing phases and a phase of diagnostic data dissemination without imposing constraints on a testing topology of the system. Results of the algorithm simulation using a process-oriented discrete-event simulation library YACSIM are provided for different topologies of distributed CS.

Keywords: adaptive algorithm, self-diagnosis, distributed computer system.