УДК 681.3.06
DOI: 10.15827/2311-6749.17.4.14

МЕТОДЫ И СРЕДСТВА АВТОМАТИЗИРОВАННОГО ОБНАРУЖЕНИЯ И УСТРАНЕНИЯ ОТКАЗОВ ГЕТЕРОГЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

И.А. Сидоров, к.т.н., ivan.sidorov@icc.ru
(ФГБУН Институт динамики систем и теории управления им. В.М. Матросова СО РАН, ул. Лермонтова, 134, г. Иркутск, 664033, Россия)

Работа посвящена разработке методов и средств повышения отказоустойчивости гетерогенных вычислительных систем. Представленные методы и средства базируются на автоматической диагностике основных программно-аппаратных компонентов этих систем, устранении неисправностей и использовании механизмов автоматической реконфигурации ресурсов вычислительной системы. Оригинальность и новизна предлагаемого подхода заключаются в создании мультиагентной системы с универсальными программными агентами, способными собирать, анализировать и применять необходимые управляющие воздействия непосредственно на вычислительном узле, что позволяет добиться высокой скорости работы и необходимого уровня децентрализации системы диагностики.

Ключевые слова: *гетерогенные вычислительные системы, мониторинг, диагностика, отказоустойчивость, реконфигурация.*

При решении ресурсоемких научных задач с применением *гетерогенных вычислительных систем* (ГВС) прикладному специалисту приходится сталкиваться с рядом проблем, связанных с необходимостью оптимизации *распределенных программ* (РП) под архитектуру ГВС, со сложностью декомпозиции исходных данных задачи с учетом ограничений на доступные пользователю вычислительные ресурсы, потребностью в выборе определенного типа очереди из списка доступных в *системе управления прохождением задач* (СУПЗ), необходимостью соблюдения ограничений административных политик, определяющих приоритеты задач, максимальное число задач в очереди, максимальное время решения задач и другие условия процесса подготовки и выполнения РП. При проведении крупномасштабных вычислительных экспериментов, требующих большого числа одновременно задействованных вычислительных узлов, одной из главных проблем является обеспечение безотказного вычислительного процесса, поскольку сбой выполнения РП непосредственно влияет на экономические показатели использования ГВС: плата за вычислительные ресурсы, плата за срыв сроков проведения расчетов, плата за трудовые ресурсы, потраченные на поиск истинных проблем сбоев РП и т.д.

Причины сбоев РП при выполнении на ГВС можно классифицировать по следующим категориям: 1) ошибки разработчиков, допущенные в коде РП; 2) ошибки пользователей, допущенные в процессе подготовки исходных данных; 3) отказы в работе программно-аппаратных компонентов ГВС и, в частности, вычислительных узлов, на которых выполняются экземпляры РП. Если ошибки первых двух категорий можно выявить и устранить в ходе отладки РП на небольшом числе узлов, то последующий запуск крупномасштабного вычислительного эксперимента часто приводит к сбоям из-за возникающих технических проблем вычислительного оборудования. Так, например, при использовании библиотеки MPI выход из строя программно-аппаратных компонентов на одном узле (процессоров, модулей памяти, жестких дисков, сетевых устройств и др.) приведет к общему сбою РП на всех задействованных узлах. Используемые на сегодняшний день механизмы организации контрольных точек позволяют в некоторой степени повысить надежность вычислений, однако при выполнении РП на большом числе вычислительных ресурсов возникают проблемы существенного снижения полезной загрузки из-за того, что РП затрачивает большую часть времени на организацию контрольных точек и восстановление после очередного отказа [1].

В связи с этим все более актуальной становится задача создания и применения эффективных методов и средств повышения отказоустойчивости ГВС, что, в свою очередь, позволит снизить число сбоев РП, выполняемых на этих системах. Одним из подходов для решения этой задачи является создание и применение эффективных систем мониторинга, диагностики и автоматизированного контроля основных программно-аппаратных компонентов ГВС. Применяемые системы мониторинга (Ganglia, Nagios, Zabbix) ориентированы преимущественно на сбор данных и предоставление агрегированной информации оператору о загрузке ГВС как в целом, так и по отдельным узлам. Все больше внимания уделяется средствам мониторинга РП с целью сбора исчерпывающих данных об эффективности использования вычислительных ресурсов экземплярами РП (Lapta [2], mpiP, IPM). Существуют средства контроля объектов инфраструктуры вычислительных систем (ClustrX Watch, EMC ViRP SRM). Однако для диагностики вычислительных узлов на предмет выявления неисправностей и сбоев по-прежнему используются разраба-

тываемые администраторами ГВС скрипты, запускаемые по расписанию. Зачастую эти скрипты работают неэффективно, отсутствуют учет и анализ возникающих неисправностей и сбоев, несвоевременно выполняется вывод проблемных узлов из пула доступных ресурсов СУПЗ. Таким образом, существует потребность в комплексных средствах автоматической диагностики узлов и автоматической реконфигурации ГВС, которые в совокупности позволят повысить отказоустойчивость ГВС и, как следствие, снизить число сбоев РП, выполняемых на таких системах.

Модель системы диагностики

Информационно-вычислительную модель системы диагностики можно представить в виде структуры $S = <O, Z, T, C, F, R, P, Q, L, I>$, где

O – множество исследуемых компонентов в узлах ГВС;

Z – множество измеряемых характеристик элементов множества O;

T – множество типов значений характеристик;

C – множество предикатов;

F – множество контрольно-диагностических операций;

R – множество типов операций;

P – множество продукций;

Q – очередь операций, готовых к исполнению;

L – журнал диагностики;

I – периоды запуска диагностики вычислительного узла, соответствующие режимам эксплуатации узла в разные временные интервалы.

Элементами множества O являются различные компоненты вычислительных узлов ГВС, отказ в работе которых может привести к сбою выполнения заданий. Среди элементов множества O выделяются аппаратные компоненты узлов (системы активного охлаждения, процессоры, модули оперативной памяти, запоминающие устройства, сетевые устройства и др.), а также программные компоненты (системные службы операционной системы, подключения к сетевым дискам, вычислительные процессы, выполняемые в операционной системе узла и др.).

Характеристики множества Z делятся на следующие категории:

– характеристики объемов вычислительной нагрузки компонентов узла (нагрузки процессоров, ядер, оперативной памяти, сетевых элементов, систем хранения данных и других структурных элементов);

– характеристики физического состояния компонентов узла (температура процессоров и материнских плат, работоспособность систем бесперебойного питания, жестких дисков и других структурных элементов);

– характеристики процесса выполнения задания (приоритет и статус задания, использованное процессорное время, объем используемой оперативной памяти, число обращений к жесткому диску и сетевым элементам и другие сведения).

Множество T включает следующие типы значений характеристик: логические (булевы), целые, дробные, процентные (беззнаковые целые числа в диапазоне от 0 до 100), символьные, строковые (до 256 символов), текстовые (до 2^{20} символов).

Множество F включает операции, назначением которых является выполнение контрольно-диагностических действий по сбору данных о работе программных и аппаратных компонентов узлов ГВС, а также действий по локализации и устранению выявленных неисправностей. Операции из F реализуют отображения вида $F : O \times Z \to Z$ или $F : O \to Z$.

Множество R включает следующие типы операций:

– операции сбора данных r_1 (снятие показаний о текущем состоянии систем активного охлаждения, температуры материнской платы и процессоров, объема оперативной памяти, объема свободной памяти на запоминающих устройствах, статуса сетевых устройств, статуса системных и сетевых сервисов операционной системы, статуса подключений к сетевым дискам и др.);

– операции тестирования r_2 (выполнение быстрых и более глубоких тестов оперативной памяти, тестирование запоминающих устройств, тестирование пропускной способности сетевых компонентов и др.);

– операции локализации неисправности r_3 (выявление номера банка отказавшего модуля оперативной памяти, процесса, использующего доступный объем оперативной памяти, физического расположения неисправного жесткого диска и др.);

– операции устранения неисправности r_4 (очистка временных файлов на запоминающих устройствах, перезапуск процессов операционной системы и др.);

– операции проверки устранения неисправности r_5 (проверка высвободившегося дискового пространства, проверка высвободившегося объема оперативной памяти, проверка снятия процесса операционной системы и др.);

– операции действий в критических ситуациях r_6 (выключение узла, вывод узла из СУПЗ, уведомление администратора ГВС).

Тип операции определяет приоритет ее выполнения в очереди Q: r_1 – приоритет 1, r_2 – приоритет 2, r_3 – приоритет 3, r_4 – приоритет 4, r_5 – приоритет 5, r_6 – приоритет 0 (наивысший).

Следует отметить, что типы операций r_1, r_2, r_3 и r_5 относятся к классу контрольно-диагностических действий, позволяющих извлечь информацию о состоянии компонентов узла без внесения каких-либо изменений в его работу. Типы операций r_4 и r_6 предназначены для исполнения в автоматическом режиме управляющих воздействий, нацеленных на устранение выявленных неисправностей программных и аппаратных компонентов узла. Управляющие воздействия могут изменять режимы работы компонентов с целью обеспечения отказоустойчивой работы узла в составе ГВС, а в случае невозможности устранения критических неисправностей применяются управляющие воздействия по автоматическому выводу узла из пула доступных ресурсов ГВС.

Анализ значений контрольно-диагностических характеристик из Z задается предикатами из множества C и записывается выражением вида: $c_i : z_j$ <*логический оператор*> *const*, где $c_i \in C$, $z_j \in Z$, $i = \overline{1, n_C}$, $j = \overline{1, n_Z}$.

В левой части указывается характеристика из Z, значение которой должно быть проанализировано. Среди логических операторов могут использоваться операторы <, ≤, >, ≥, == и !=. Значение *const* в правой части и значение характеристики z_k должны быть одного типа. Допускается построение сложных выражений с помощью операторов *AND*, *OR*, *XOR* и *NOT*.

Множество P содержит продукционные правила вида *IF* c_k *THEN* f_i. Эта конструкция интерпретируется следующим образом: если предикат c_k истинен, необходимо выполнить операцию f_i. Продукция является допустимой, если заданы значения для всех характеристик, включенных в ее предикат. Продукция является выполненной, если значение предиката истинно и операция, стоящая в правой части, добавлена в очередь Q.

Схема диагностики вычислительного узла строится следующим образом. С некоторым периодом дискретности из I агенты мониторинга запускают процедуру диагностики узла. Исходным для процесса диагностики является вектор, содержащий значения характеристик логического типа из Z, зависящий от текущего интервала из I и определяющий основные режимы работы (использовать или нет операции устранения неисправностей и т.п.) и качество процесса диагностики (быстрое тестирование, глубокое тестирование или расширенное тестирование).

Имеется определенный порядок интерпретации продукций и добавления операций в очередь Q. Выше было указано, что в системе выделено шесть типов операций. На первом шаге выбираются допустимые продукции, у которых в правой части указаны контрольно-диагностические операции r_1, r_2 и r_3 типов (операции сбора данных). Осуществляется процедура интерпретации выбранных допустимых продукций, которая включает вычисление предикатов, и в случае их истинности выполняется добавление соответствующих операций в очередь Q.

Обработка операций, находящихся в очереди исполнения Q, осуществляется в следующем режиме. Одним из требований системы диагностики вычислительного узла является снижение общего времени выполнения операций при использовании доступных вычислительных ресурсов узла. В связи с этим выбрана стратегия выполнения максимального числа операций в единицу времени, в соответствии с которой все контрольно-диагностические операции, попадающие в очередь на исполнение, незамедлительно запускаются в отдельном потоке параллельно с другими операциями. При этом единственным условием для запуска операции, стоящей в очереди Q, является проверка того, что на текущий момент не выполняется другая операция, использующая тот же компонент из O. Данное условие проверяется с целью исключения выполнения взаимно противоречивых операций.

После завершения выполнения какой-либо операции в очереди Q расширяется множество определенных характеристик Z и соответственно расширяется список допустимых продукций. Запускается процедура интерпретации допустимых и ранее не выполненных продукций.

Следует отметить, что на втором и последующих шагах интерпретации списка продукций, помимо продукций, в правой части которых стоят операции r_1, r_2 и r_3 типов, выбираются продукции, в правой части которых стоят операции r_6 типа (действий в критических ситуациях). В случае истинности предиката такой продукции операция ставится в начало очереди. Данный подход позволяет незамедлительно реагировать на выявленные критические неисправности.

После того как все операции r_1, r_2 и r_3 типов в очереди Q выполнены, осуществляется переход к следующему этапу и выбираются продукции, в правой части которых стоят операции r_4 типа (операции устранения неисправностей).

После завершения всех операций r_4 типа в очереди Q осуществляются переход к следующему этапу и выбор допустимых продукций, в правой части которых стоят операции r_5 типа (проверка устранения

неисправности). На последнем шаге выполняется интерпретация продукций с операциями r_6 типа, после выполнения которых процесс диагностики считается завершенным.

Следует отметить, что все элементы множеств *O, Z, C, F, P, I* для структуры *S* снабжены содержательными описаниями. Для предикатов из множества *C* описывается содержательный смысл истинности предиката. Для операций из *F* описывается содержательный смысл тех действий, которые предпринимаются. В ходе интерпретации допустимых продукций в журнал диагностики *L* последовательно вносятся описания истинных предикатов и описания операций, стоящих в правой части продукции. По такому принципу осуществляется построение журнала диагностики *L*.

Программная реализация

Программная реализация рассмотренной выше модели системы диагностики осуществляется в составе инструментальных средств метамониторинга разнородных ГВС [3, 4]. Данные инструментальные средства предназначены для применения в ГВС эксафлопного уровня и базируются на применении сервис-ориентированных технологий, мультиагентных технологий, методов создания экспертных систем, методов децентрализованной обработки и децентрализованного принятия решений.

Система диагностики программно-аппаратных компонентов ГВС реализуется в составе агентов системы метамониторинга, функционирующих на вычислительных узлах в фоновом режиме. В составе системы диагностики выделены следующие основные составляющие:

– управляющая подсистема, выполняющая функции координации процесса диагностики вычислительного узла и включающая механизмы запуска процесса диагностики в определенные временные интервалы либо в моменты простоя узла от вычислительных задач;

– подсистема интерпретации продукций и предикатов;

– библиотека контрольно-диагностических операций, реализуемых на различных языках программирования;

– очередь выполнения контрольно-диагностических операций;

– подсистема исполнения операций;

– подсистема журналирования событий.

Реализация агента и всех базовых подсистем выполняется на языке программирования C++. Контрольно-диагностические операции могут быть реализованы на различных языках программирования и должны быть представлены в виде исполняемой программы, поддерживающей пакетный режим работы (работу с параметрами через файлы либо через аргументы командной строки). Для интерпретации продукций и предикатов используется интерпретатор языка ECMA Script [5]. Все спецификации системы описываются в формате JSON [6].

Пример диагностики узла ГВС

В качестве примера рассматривается простая схема диагностики оперативной памяти узла ГВС. Пусть элементы множеств для структуры *S* заданы следующим образом.

O:

o_1 – оперативная память узла ГВС;

o_2 – задание (экземпляр РП), выполняемое на данном узле.

Z:

z_0 – начальные условия диагностики (тип строковый);

z_1 – общий объем оперативной памяти (тип целочисленный);

z_2 – объем занятой оперативной памяти (тип процентный);

z_3 – номер неисправного банка оперативной памяти (тип целочисленный);

z_4 – статус отправки уведомления администратору ГВС (тип булев);

z_5 – статус отправки уведомления пользователю задачи o_2 (тип булев);

z_6 – статус выполнения команды снятия задачи o_2 с выполнения (тип булев);

z_7 – статус снятия задачи o_2 с выполнения (тип булев).

C:

c_0: z_0 == *"quick"* – выполнить быструю диагностику узла;

c_1: z_1 != 2^{34} – общий объем оперативной памяти должен быть равен 8 Гб;

c_2: $z_3 > 0$ – выявлен номер неисправного банка памяти;

c_3: $z_2 > 99$ – загрузка оперативной памяти более 99 %.

F:

f_1: $\{o_1\} \rightarrow \{z_1,z_2\}$ – операция получения данных оперативной памяти (тип r_1);

f_2: $\{o_1\} \rightarrow \{z_3\}$ – операция локализации неисправного банка памяти (тип r_3);

f_3: $\{o_1, z_3\} \rightarrow \{z_4\}$ – операция уведомления оператора о выявленном неисправном банке оперативной памяти (тип r_6);

f_4: $\{o_2, z_2\} \rightarrow \{z_5\}$ – операция уведомления пользователя задачи o_2 о превышении лимита оперативной памяти (тип r_6);

f_5: $\{o_2\} \rightarrow \{z_6\}$ – снятие задачи o_2 с выполнения (тип r_4);

f_6: $\{o_1, o_2\} \rightarrow \{z_7\}$ – проверка снятия задачи o_2 с выполнения (тип r_5).

P:

p_0*: IF* c_0 *THEN* f_1 – выполнить быструю диагностику узла;

p_1*: IF* c_1 *THEN* f_2 – в случае изменения объема оперативной памяти выполнить локализацию неисправного банка оперативной памяти;

p_2: *IF* c_2 *THEN* f_3 – уведомить оператора о неисправном банке оперативной памяти;

p_3: *IF* c_3 *THEN* f_4 – уведомить пользователя задачи o_2 о превышении лимита использования оперативной памяти;

p_4*: IF* c_3 *THEN* f_5 – при превышении лимита использования памяти выполнить снятие задачи o_2.

Схема диагностики для такой структуры может быть следующей. Пусть z_0 = *"quick"*. Очередь операций после интерпретации продукций на первом шаге будет включать следующие вычислительные операции: $Q = \{f_1\}$. После выполнения операций первого шага будут определены параметры z_1 и z_2. Предположим, что узел имеет два банка оперативной памяти по 4 Гб и один из банков вышел из строя: $z_1 = 2^{33}$ (в таких случаях чаще всего операционная система продолжает свою работу, выводя из адресного пространства сбойный банк). Также предположим, что $z_2 > 99$ (критическая загрузка оперативной памяти с возможным использованием swap). На следующем шаге истинными являются предикаты c_1 и c_3. В соответствии с порядком обработки продукций в очередь добавляется операция локализации $Q = \{f_4, f_2\}$. На следующем шаге истинными являются предикаты c_2 и c_3. В очередь добавятся следующие операции $Q = \{f_3, f_5\}$. И на последнем шаге добавится операция проверки снятия задачи $Q = \{f_6\}$.

Следует отметить, что приведенный пример носит демонстративный характер приведенной выше модели и методов и не содержит полного перечня операций и продукций для всесторонней диагностики оперативной памяти вычислительного узла ГВС.

Заключение

Описанный подход позволяет производить контроль, диагностику и устранение неисправностей программных и аппаратных компонентов узлов ГВС за конечное число шагов, минимизировать время выполнения диагностики и устранения неисправностей за счет применения механизмов параллельного выполнения операций, повышать отказоустойчивость узлов и, как следствие, выполняемых в этих узлах РП за счет превентивных мер по диагностике и устранению неисправностей, что в совокупности позволяет повышать надежность и эффективность функционирования ГВС.

Литература

1. Бетелин В.Б., Кушниренко А.Г., Райко Г.О. Проблемы обеспечения роста производительности отечественных суперЭВМ в период до 2020 года // Информационные технологии и вычислительные системы. 2010. № 3. С. 15–18.

2. Адинец А.В., Брызгалов П.А., Воеводин Вад.В. Мониторинг, анализ и визуализация потока заданий на кластерной системе // Высокопроизводительные параллельные вычисления на кластерных системах: матер. XI Всерос. конф. Н. Новгород, 2011. С. 10–14.

3. Bichkov I.V., Oparin G.A., Novopashin A.P., Sidorov I.A. Agent-based approach to monitoring and control of distributed computing environment. Parallel Computing Technologies. LNCS, 2015, vol. 9251, pp. 253–257.

4. Сидоров И.А., Новопашин А.П., Опарин Г.А., Скоров В.В. Методы и средства метамониторинга распределительных вычислительных сред // Вестн. ЮУрГУ. Сер. ВМИ. 2014. № 3 (2). С. 30–42.

5. Standard ECMA-262: ECMAScript language specification. URL: http://www.ecma-international.org/publications/standards/Ecma-262.htm (дата обращения: 14.11.2017).

6. The JavaScript object notation (JSON) data interchange format. URL: https://tools.ietf.org/html/rfc7159 (дата обращения: 14.11.2017).