Аппаратная поддержка контролируемого выполнения

В.А. Галатенко
Научно-исследовательский институт системных исследований РАН
Россия, 117218, Москва, Нахимовский пр-т, 36, к. 1
E-mail: galat@niisi.msk.ru

К.А. Костюхин
Научно-исследовательский институт системных исследований РАН
Россия, 117218, Москва, Нахимовский пр-т, 36, к. 1
E-mail: kost@niisi.msk.ru

Н.В. Шмырёв
Научно-исследовательский институт системных исследований РАН
Россия, 117218, Москва, Нахимовский пр-т, 36, к. 1
E-mail: shmyrev@niisi.msk.ru

В данной работе рассматривается расширение концепции контролируемого выполнения на аппаратные компоненты сложных систем. Анализируются современные технологии разработки вычислительной аппаратуры, предлагаются новые решения, основанные на концепции контролируемого выполнения.

HARDWARE SUPPORT OF CONTROLLED EXECUTION / V.A. Galatenko (Scientific Research Institute for System Analysis, Russian Academy of Sciences, 36-1 Nakhimovskiy pr., Moscow 117218, Russia, E-mail: galat@niisi.msk.ru), K.A. Kostyukhin (Scientific Research Institute for System Analysis, Russian Academy of Sciences, 36-1 Nakhimovskiy pr., Moscow 117218, Russia, E-mail: kost@niisi.msk.ru), N.V. Shmyrev (Scientific Research Institute for System Analysis, Russian Academy of Sciences, 36-1 Nakhimovskiy pr., Moscow 117218, Russia, E-mail: shmyrev@niisi.msk.ru). The present work considers new extension of controlled execution paradigm. This extension deals with hardware components of complex systems. Authors made a brief analysis of hardware development modern methods and offer new solutions, based on controlled execution paradigm.

Введение

Современные информационные системы являются распределенными и разнородными, их программная часть состоит из десятков миллионов строк исходных текстов, а аппаратная – из десятков миллионов транзисторов. Современная технология проектирования и реализации подобных систем не позволяет избавиться от ошибок, что особенно опасно для критически важных систем.

Как правило, поиск и исправление ошибок относят на этапы разработки и тестирования. Программные и аппаратные ошибки проявляются и на этапе эксплуатации, а исправление ошибок не означает уменьшения их числа. Тестирование, как отметил еще Дейкстра, может служить лишь для доказательства наличия ошибок, но не для доказательства их отсутствия.

Многие информационные системы считаются надежными [1], если были успешно выполнены тесты, соответствующие типичным вариантам использования системы. Однако в этом подходе кроется серьезное упущение с точки зрения информационной безопасности, поскольку многие уязвимости вскрываются как раз в нетипичных режимах функционирования информационной системы.

Даже одна, редко возникающая аппаратная или программная ошибка может стать причиной как финансовых потерь, так и угроз человеческим жизням. Достаточно вспомнить известную ошибку в процессоре Pentium, связанную с реализацией операции деления в

сопроцессоре плавающей арифметики [2]. Несмотря на "незначительное" расхождение правильного и вычисленного результата, в конечном счете, эта ошибка привела к отзыву процессоров и потерям Intel в размере 500 миллионов долларов США. Следует заметить, что в современных аппаратных компонентах информационных систем критические ошибки могут возникнуть (и возникают) не только в вычислительных блоках процессоров, но и в кэш-памяти [3], подсистемах ввода-вывода и прерываний, подсистеме памяти.

Второй пример также хорошо известен: ошибка в программном обеспечении прибора для проведения радиационной терапии Therac-25 [4]. Предполагая надежность программного обеспечения, унаследованного от предшественников (Therac-6 и Therac-20), создатели Therac-25 решили отказаться от механизма аппаратной защиты, применявшегося в старых приборах. Эта аппаратная защита служила для предотвращения небезопасных для здоровья пациентов операций. В результате несколько пациентов погибли от радиационной передозировки, вызванной крайне редко возникающей ошибкой в ПО прибора Therac-25. Отметим тот факт, что ошибка была и в старых приборах, однако в силу постоянно включенной аппаратной защиты ошибка не имела таких трагических последствий.

И теоретические соображения, и многолетняя практика позволяют сделать вывод, что в сложных аппаратно-программных системах ошибки неизбежны. Необходима разработка и проведение в жизнь целостного подхода, обеспечивающего, несмотря на ошибки и угрозы информационной безопасности системы, выполнение системой ее миссии – то, ради чего она создавалась.

Известные частные подходы к контролю работы систем

Использование программируемых аппаратных компонентов

Этот подход [5] основывается на использовании в микросхеме процессора контролирующих программируемых схем с тем, чтобы нейтрализовывать аппаратные ошибки без замены микросхем. Существует некоторая последовательность действий, порождающая ошибку. Эта последовательность называется сигнатурой ошибки. Когда производитель аппаратных компонентов находит ошибку, он перепрограммирует соответствующие схемы так, чтобы в момент возникновения ошибки включался механизм ее нейтрализации. Если предотвратить ошибку невозможно, то, зная, характер ее проявления, можно устранить последствия, используя такие механизмы, как откатка выполнения [6], [7].

Разработка, ориентированная на отладку

Традиционно при разработке систем на кристалле применялся подход "разработка, ориентированная на тестирование". Разумеется, положительный результат тестирования не означал, что ошибок в микросхеме нет.

По мере увеличения сложности систем на кристалле стало ясно, что использование только разработки, ориентированной на тестирование, недостаточно. Ошибки в сложных системах на кристалле возникают не только внутри функциональных модулей, но и при взаимодействии между ними. Из-за подобной сложности тестирования отдельных функциональных блоков и их взаимодействия недостаточно, чтобы гарантировать надежную бесперебойную работу в течение длительного времени.

Суть технологии разработки, ориентированной на отладку [8], заключается в том, что сначала каждый функциональный модуль оснащается своим информационно-управляющим интерфейсом. Затем необходимые инструменты, осуществляющие информационно-управляющие воздействия, интегрируются уже в систему на кристалле для обеспечения контроля взаимодействия функциональных модулей внутри этой системы. Наконец, внедряются средства контроля взаимодействия нескольких систем на кристалле в рамках сети на кристалле. Это делает возможным контроль функционирования системы не только на этапе тестирования, но и на этапе эксплуатации.

Отладка систем пользователями

Разработчики осознают, что простого механизма фиксации состояния системы в момент проявления ошибки недостаточно для эффективной локализации и исправления этой ошибки. Ошибка может возникнуть в результате внешнего воздействия как на программные, так и аппаратные компоненты системы. Вести полный протокол всех событий информационной системы не представляется возможным. Необходимо предоставить механизм сбора только полезной для разработчика информации о проявившихся ошибках. В частности, можно организовывать контрольные точки, а затем проводить откатки по выполнению и повторное выполнение программы в случае возникновения ошибки [9]. При этом каждый раз при повторном выполнении кода, содержащего ошибку, автоматически порождается и проверяется очередная гипотеза о первопричине ошибки. В ходе повторного выполнения сравнивается состояние программы в тех случаях, когда ошибка проявилась, с теми, в которых выполнение прошло успешно. Такое сравнение позволяет выявить программные объекты, возможно, участвующие в порождении ошибки. Вся накопленная информация затем пересылается разработчикам.

Программирование, ориентированное на мониторинг

Программирование, ориентированное на мониторинг (ПОМ) [10], – это парадигма программирования, берущая начало в аспектно-ориентированном программировании [11].

ПОМ включает в себя два независимых механизма: наблюдения и верификации. Механизм наблюдения извлекает информацию о состоянии приложения в определенные моменты времени выполнения. Механизм верификации пользуется полученной информацией для проверки соответствия хода выполнения приложения некоторой модели. Иными словами, проверяется соблюдение определенных требований и наличие заданных свойств; при этом выполняются некоторые процедуры – обработчики положительных и отрицательных результатов проверки.

Мониторы, реализующие механизм верификации, могут быть как отдельными программами, читающими журнал событий целевого приложения, так и встраиваемыми в код приложения исполняемыми блоками, активизирующимися при наступлении определенных событий.

Внешний контроль функционирования аппаратно-программных систем

Практически все перечисленные выше подходы в той или иной мере используют внешние средства контроля функционирования информационных систем.

Перечислим основные недостатки внешнего контроля.

- Сложность внешнего аппаратно-программного комплекса, осуществляющего информационно-управляющие воздействия на систему, сопоставима со сложностью управляемой системы. Это влечет за собой увеличение сроков разработки конечного продукта и дополнительные ошибки в управляющем комплексе.
- Сложность синхронизации управляющего комплекса и управляемой системы. Управляемая система постоянно эволюционирует, изменяется ее архитектура, интерфейсы и протоколы взаимодействия между компонентами, появляются новые компоненты, реализующие новую функциональность. В этих условиях разработчикам приходится постоянно обновлять управляющий комплекс, при этом обеспечивая также и обратную совместимость управляющего комплекса со старыми версиями управляемой системы.

Контролируемое выполнение

Общие положения

Под контролируемым выполнением понимается специально организованный процесс функционирования информационных систем, целью которого является выполнение системами своей миссии, несмотря на возможное проявление ошибок, атаки и отказы. Основными положениями концепции контролируемого выполнения являются:

- интеграция средств информационной безопасности, отладки, профилирования и управления;
- распространение контролируемого выполнения на все этапы жизненного цикла информационных систем;
- широкий спектр набора средств контролируемого выполнения, различающихся по степени воздействия на целевые системы, возможность взаимодействия между этими средствами.

Частными случаями контролируемого выполнения являются:

- применение средств управления информационными системами;
- интерактивные информационно-управляющие воздействия;
- мониторинг функционирования систем;
- самоконтроль систем;
- воспроизведение предыдущих сеансов работы систем.

Средства управления информационными системами ориентированы на такие крупные сущности, как сети, узлы сетей, хосты, устройства, приложения. Эти средства применяются на этапе эксплуатации систем.

Под интерактивным информационно-управляющим воздействием понимается опрос характеристик функционирования системы и, возможно, воздействие на ход функционирования (например, интерактивная отладка). Интерактивное информационно-управляющее воздействие применяют на этапах разработки, тестирования и, возможно, опытной эксплуатации систем. В соответствии с концепцией контролируемого выполнения интерактивные информационно-управляющие воздействия допустимы (и желательны) на всех этапах жизненного цикла систем.

Развитие программного обеспечения с открытыми исходными текстами по крайней мере в принципиальном плане дает возможность распространить эти воздействия на этап эксплуатации. Однако, чтобы подобное решение стало разумным, необходимо изменение взглядов на эксплуатационную документацию к информационным системам, а также на информацию, доступную во время выполнения. Их следует дополнить, предоставив пользователям и средствам контроля достаточно сведений для анализа нештатных ситуаций и их разрешения.

Под мониторингом системы понимается сбор и анализ (в реальном времени или постфактум) информации о ходе ее функционирования. Мониторинг может являться составной частью как управления на этапе эксплуатации, так и тестирования и отладки на этапах разработки и тестирования. Важно отметить, что на этапе эксплуатации обычно осуществляется мониторинг описанных выше крупных сущностей; на этапах разработки и тестирования он может носить сколь угодно детальный характер, определяемый потребностями тестирования и отладки.

Самоконтроль систем состоит в проверке значений параметров, отражающих уровень безопасности состояния, в котором находится система, и реагировании в случае, если состояние оказалось небезопасным (некорректным).

Воспроизведение предыдущих сеансов работы систем организуется на основе накопленной трассировочной информации и служит целям выработки и проверки гипотез о первопричинах и местах ошибок.

Роль контролируемого выполнения и его составляющих

Информационные системы создаются для решения определенных задач, достижения заданных целей. Для краткости будем говорить, что у каждой системы есть своя миссия.

У миссии есть несколько относительно независимых аспектов, к которым, помимо предоставляемой функциональности, принадлежат обеспечение заданного уровня информационной безопасности, количество расходуемых ресурсов, качество обслуживания пользователей и т.д.

Факторами, способными привести к нештатному функционированию систем, являются:

- аппаратные и программные ошибки;
- сбои и отказы аппаратуры;
- ошибки пользователей и обслуживающего персонала;
- нарушение эксплуатационных требований и регламентов;
- злоумышленные действия

и т.д.

Контролируемое выполнение направлено на то, чтобы помочь системе выполнить свою миссию вопреки внутренним ошибкам, а также случайным или преднамеренным внешним деструктивным воздействиям.

Для сложных, ответственных систем целесообразно постоянно контролировать характер их функционирования, по возможности выявляя и ликвидируя проблемы на ранних стадиях возникновения. Поведение таких систем определяется не только программными компонентами, но и темпом поступления внешних событий, наличием ресурсов и т.п. Если не располагать регистрационной информацией, то на практике не удастся воспроизвести ситуацию, приведшую к ошибке и, соответственно, не удастся эффективно устранить саму ошибку и/или последствия ее возникновения.

Очевидно, применение информационно-управляющих воздействий влияет на (вносит возмущения в) поведение систем (меняются временные характеристики, распределение памяти и т.п.). Ошибка, проявившаяся в эксплуатационном режиме работы системы, в режиме поиска и устранения ошибок может не воспроизводиться. Из этого обстоятельства можно сделать два вывода:

- необходимо учитывать возмущения, вносимые информационно-управляющими воздействиями в поведение систем;
- при разработке системы целесообразно включить в эксплуатационный вариант средства информационно-управляющего воздействия как минимум для всех критически важных компонентов системы, так как их изъятие после этапа тестирования повлияет на поведение, возможно, сделав его некорректным, а поиск и устранение ошибок – невозможным. При этом накладные расходы должны быть приемлемыми, но запас мощности у системы должен оставаться.

Корректное функционирование – необходимое условие выполнения миссии. Средства поиска и устранения ошибок (интерактивные информационно-управляющие воздействия, мониторинг и воспроизведение предыдущих сеансов работы систем) помогают в обеспечении корректности функционирования. В то же время, цель, состоящая в выявлении и исправлении всех ошибок, при существующей технологии программирования представляется недостижимой. Следовательно, помимо средств внешнего контроля, для выполнения миссии необходимо привлечение дополнительных механизмов. На эту роль предлагаются средства самоконтроля программ и аппаратуры.

Применение средств самоконтроля можно рассматривать как распространение объектно-ориентированного подхода на контролируемое выполнение систем. Программные и аппаратные компоненты системы в таком случае трактуются не как пассивные объекты, к которым внешним образом применяются информационно-управляющие воздействия с внешней же интерпретацией семантики выполнения, а как активные сущности, самостоятельно контролирующие корректность своего функционирования.

При интерактивных информационно-управляющих воздействиях и мониторинге сложных систем вопрос о том, является ли некоторое состояние, достигнутое в процессе функционирования, корректным, может оказаться весьма сложным, а для его разрешения может потребоваться анализ не только исходных текстов, но и больших объемов данных. Подобный анализ трудно производить вручную; целесообразно использование некоторых аппаратных и программных средств. В подобной ситуации средства самоконтроля имеют то преимущество, что они записываются на языке достаточной для автоматической проверки выразительной силы и имеют доступ ко всем необходимым объектам.

В начале 1980-х годов Д. Кнут в работе [12] ввел понятие (много)аспектного программирования и предложил инструментальные средства для работы с многоаспектными исходными текстами. Идеи аналогичной направленности развивались в инструментальной системе ЭСКОРТ [13]. Средства самоконтроля можно рассматривать как отдельный, относительно независимый аспект программы, который должен отвечать за все грани информационной безопасности, такие, например, как доступность, контроль целостности программ и данных.

В концепции контролируемого выполнения средствам самоконтроля отведена ключевая роль. Только сама система "знает" свою семантику, поэтому все, что необходимо для ее контролируемого выполнения, должно присутствовать в ней самой, как в аппаратных компонентах, так и в исходных текстах программных компонентов (в том числе процедуры и функции, необходимые для мониторинга и интерактивных информационно-управляющих воздействий). Средства самоконтроля являются связующим звеном между аппаратно-программной системой и другими средствами контролируемого выполнения – реакция на выявление некорректной или подозрительной ситуации может заключаться в инициации или повышения уровня детализации мониторинга или в открытии сеанса интерактивных информационно-управляющих воздействий. Это дополнительно подчеркивает объектно-ориентированный характер контролируемого выполнения, активность программных и аппаратных компонентов системы.

Предлагаемые решения

Среди предлагаемых (и реализуемых аппаратно на технологической базе НИИСИ РАН) решений можно выделить следующие:

Интерактивная отладка

Традиционный интерфейс jtag/ejtag и все возможности, им предоставляемые. Глобальный останов всех вычислителей и пошаговый режим. Под глобальным остановом понимается останов и выход в отладочный режим центрального процессора, а также останов (возможно отложенный) всех сопроцессоров. В этом режиме может быть реализована возможность исследовать и изменять состояние (регистры, память) как центрального процессора, так и сопроцессоров.

Мониторинг

Trace-буфер. Трасса нелинейных переходов. Буфер в незатираемой после перезагрузки (со)процессора памяти. В этом буфере сохраняется трасса хода выполнения программы, а именно: диапазоны адресов линейных участков выполненного кода. Предполагается наличие независимого (от ejtag), достаточно широкого канала для выборки этой информации.

Профилирование

Аппаратные счетчики команд и данных. Такие счетчики полезны при оптимизации нагрузки на процессор, а также для равномерного распределения данных между вычислителями.

Самоконтроль

Регистр watchdog, порождающий прерывание при необращении к этому регистру в течение некоторого времени. Этот регистр полезен для реализации функции самоконтроля вычислительной программы. Например, если программа в течение некоторого времени не

обращается к данному регистру, то порождается прерывание, обрабатываемое программным агентом отладки.

Заключение

Современная технология разработки сложных аппаратно-программных систем не позволяет сделать их свободными от ошибок. Кроме того, при работе систем возможны сбои и отказы, а также вредоносные воздействия.

Контролируемое выполнение направлено на то, чтобы обеспечить выполнение системой своей миссии, несмотря на внутренние ошибки, а также внешние случайные или умышленные деструктивные воздействия.

Контролируемое выполнение включает в себя широкий спектр механизмов информационно-управляющего воздействия, среди которых ключевая роль отводится средствам самоконтроля программ.

Контролируемое выполнение подразумевает приемлемый уровень вносимых возмущений, что дает возможность его использования на этапе производственной эксплуатации систем.

Базой для выполнения технологических требований к средствам контролируемого выполнения служит использование стандартов и архитектурных подходов, разработанных для средств управления, интеграция средств информационной безопасности, отладки и управления.

Список литературы

1. Jackson D. A Direct Path to Dependable Software. Communications of the ACM, vol. 52, April, 2009.
2. Halfhill T.R. The Truth Behind the Pentium Bug. http://www.byte.com, March, 1995.
3. Magee M. Intel's Hdden Xeon, Pentium 4 Bugs. http://www.theinquirer.net, August, 2002.
4. Тэллес М., Хсих Ю. Наука отладки. Москва: Кудиц-образ, 2003.
5. Sarangi S.R., Tiwari A., Torrellas J. Phoenix: Detecting and Recovering from Permanent Processor Design Bugs with Programmable Hardware. The 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06), IEEE, 2006.
6. Nakano J., Montesinos P., Gharachorloo K., Torrellas J. ReViveI/O: Efficient Handling of I/O in Highly-Available Rollback-Recovery Servers. International Symposium on High-Perfomance Computer Architecture, February, 2006.
7. Prvulovic M., Zhang Z., Torrellas J. ReVive: Cost-Effective Architectural Support for Rollback Recovery in Shared-Memory Multiprocessors. International Symposium on Computer Architecture, May, 2002.
8. Yi H., Park S., Kundu S. A Design-for-Debug (DfD) for NoC-based SoC Debugging via NoC. 17th Asian Test Symposium, IEEE, 2008.
9. Tucek J., Lu S., Huang C., Xanthos S., Zhou Y. Triage: Diagnosing Production Run Failures at the User's Site. SOSP'07, October, 2007.
10. Chen F., Rosu G. MOP: Reliable Software Development using Abstract Aspects. Dept. of Computer Science, University of Illinois, 2006.
11. Kiczales G., Lamping J. Aspect-Oriented Programming. Proceedings of the European Conference on Object-Oriented Programming (ECOOP), Finland, Springer-Verlag, June, 1997.
12. Knuth D. Literate Programming, CSLI, 1992.
13. Бетелин В.Б., Галатенко В.А. ЭСКОРТ – инструментальная среда программирования. Юбилейный сборник трудов институтов Отделения информатики РАН, том II, Москва, 1993.