УДК 004.031.43

Т.В. ВАСИЛЕНКО

Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Украина

СОЗДАНИЕ ФУНКЦИОНАЛЬНО БЕЗОПАСНЫХ ПРОГРАММНЫХ СИСТЕМ НА ОСНОВЕ ВИРТУАЛЬНЫХ МАШИН

Рассмотрена существующая проблема обеспечения функциональной безопасности программного обеспечения. Приведен обзор нормативных документов, которые регламентируют функциональную безопасность программных средств. Рассмотрены возможности построения функционально безопасных программных систем на основе виртуальных машин.

функциональная безопасность, виртуальные машины, системы реального времени, критические системы, параллельные вычисления

Введение

Известно множество примеров гибели дорогих систем, катастроф, больших финансовых и материальных потерь в авиации, в военных системах и на транспорте из-за относительно простых ошибок в программах. Несмотря на опережающее развитие информационных технологий, это утверждение остается справедливым практически полвека.

Так, 28 июля 1962 г. во время старта космического аппарата Mariner I к Венере из-за поломки антенны он потерял связь с земными службами управлениями и перешел на собственную систему пилотирования. Крохотная ошибка программного обеспечения системы навигации (при вводе одной из формул был пропущен один символ) привела в итоге к необходимости подрыва космического аппарата над Атлантическим океаном [1].

4 июня 1996 г. ракета-носитель Ariane 5 взорвалась через 40 секунд после своего первого старта. Система автоподрыва сработала после остановки обоих процессоров в результате цепочки ошибок, началом которой стало переполнение буфера, вызванное, в свою очередь, некорректным переносом ПО ракеты-носителя Ariane 4. Только прямые потери от этой некорректности составили $500 млн [2].

В наиболее тяжелых случаях ущерб измерялся ценой жизни и здоровья людей или большими материальными потерями. В то же время они имели вполне объяснимую природу и источники, влияние которых могло быть снижено путем глубокого анализа, выявления дефектов и корректировок программ или информации баз данных [3].

Общая постановка проблемы. Критические информационные системы (КС) – это класс систем, для которых нарушение их функциональной стабильности крайне опасно, поскольку полный или частичный отказ системы может привести к значительному экономическому, политическому, моральному и другим ущербам. Очевидно, что аэрокосмическая техника с повышенными требованиями к функциональной безопасности и надежности относится именно к этому классу.

Зависимость критических систем от программных средств порождает необходимость придания применяемым в них программным средствам заданных свойств безопасности и способности противостоять разрушению, нарушениям функционирования системы, сбоям, преднамеренным воздействиям злоумышленников и ошибкам различных видов при выполнении критической системой основной целевой функции.

Имеющиеся достижения в области теории и практики управления безопасностью сложной промышленной продукции, как правило, не известны и не используются специалистами, создающими и применяющими военные системы на базе программных средств. Это во многих случаях определяет дефекты и отказовые ситуации при применении программных систем (ПС), конфликты между заказчиками и разработчиками из-за неопределенностей значений их безопасности и неконкурентоспособность создаваемых ПС на мировом рынке. Формализация и оценивание реальной безопасности программных средств и систем часто зависит от интуиции и квалификации их разработчиков, заказчиков и пользователей. В результате проекты ПС не соответствуют исходному, декларированному назначению и первоначальным спецификациям требований к характеристикам безопасности и качества, не укладываются в согласованные графики и бюджет разработки [3].

Все вышесказанное вынуждает считать проблему обеспечения функциональной безопасности программного обеспечения (ПО) критических систем актуальной и требующей разрешения.

Целью работы является обоснование возможности применения виртуальных машин, как инструментальной среды обеспечения функциональной безопасности ПО критических систем.

Анализ основных положений нормативных документов, регламентирующих функциональную безопасность программных систем

Угрозы безопасности информации и программного обеспечения КС возникают как в процессе их эксплуатации, так и при создании этих систем, что особенно характерно для процесса разработки ПО, баз данных и других информационных компонентов КС. Базовым задачам обеспечения безопасности программных средств и систем с использованием ЭВМ посвящена значительная группа стандартов. Например, согласно стандарту Международной электротехнической комиссии (МЭК 61508) «Функциональная безопасность электрических/электронных/программируемых электронных систем, связанных с безопасностью», функциональная безопасность (ФБ) любых изделий должна определяться двумя требованиями: к тем функциям безопасности, которые должны выполняться, и к полноте безопасности этих функций безопасности [3].

Технология создания и всего жизненного цикла комплексов программ для обеспечения функциональной безопасности специализированных ЭВМ, встроенных в аппаратуру объектов и систем наиболее четко представлена в стандарте IEC 61508:1-6: 1998 Функциональная безопасность электрических / электронных/программируемых электронных систем безопасности. Третья часть стандарта IEC 61508-3 "Требования к программному обеспечению" устанавливает общий подход ко всем видам деятельности на протяжении цикла обеспечения безопасности для систем, содержащих программируемые электронные компоненты (ПЭС), которые используются для выполнения различных функций и, в частности, для обеспечения безопасности систем [3].

Наиболее мощным современным стандартом, отражающим требования и рекомендации по обеспечению безопасности систем, содержащих программные средства, является ISO 15408:1999 — 1-3 Методы и средства обеспечения безопасности. В нем рассматривается регламентирование технологии и процессов обеспечения жизненного цикла программных средств, создаваемых для обеспечения безопасности функционирования и применения систем.

На Украине также действуют:

– государственный стандарт Союза ССР – 27.003-90. Надежность в технике «Состав и общие правила задания требований надежности». Согласс-

но ему номенклатуру задаваемых показателей надежности изделия выбирают в соответствии с положениями настоящего стандарта и согласовывают в установленном порядке между заказчиком (потребителем) и разработчиком (изготовителем);

– государственный стандарт Украины ДСТУ 2850-94 «Програмні засоби ЕОМ». Согласно данному стандарту установлена базовая номенклатура качества программных средств. Стандарт определяет основные положения выбора показателей оценки качества ПС [4].

Обеспечение функциональной безопасности и надежности ПО на основе диверсификации

Одним из основных способов обеспечения надежности программных средств является разработка многоверсионных программных систем, которые по своей идеологии реализуют такие понятия как резервирование или избыточность. Резервирование – способ обеспечения надежности объекта за счет использования дополнительных средств и возможностей, избыточных по отношению к минимально необходимым при выполнении требуемых функций. Резервирование само по себе может быть использовано не только для повышения надежности, но и для повышения точности, устойчивости, достоверности и пр. Иногда вместо термина „резервирование" используют словосочетание „введение избыточности". Между этими понятиями есть много общего, но есть и различия. Под избыточностью понимают превышение веса, габаритов, производительности, стоимости и других технико-экономических показателей изделия над минимально необходимыми [5]. Однако, введение избыточности не означает автоматического улучшения показателей надежности, достоверности и др.

Известны архитектурные решения, которые, используя принцип избыточности, позволяют создавать надежное ПО систем реального времени. Эти решения основаны на применении сторожевых таймеров, отслеживающих время выполнения каждой операции, тем самым определяя наличие ошибок в функционировании программной системы, либо параллельном решении одной и той же задачи абсолютно разными программами.

При организации параллельных вычислений выделяют два принципиально разных подхода, различающихся и целями, и способами их достижения:

– логическая параллельность применяется для снижения трудоемкости создания сложных программных проектов, в ее основе – методы и средства логической декомпозиции единого вычислительного процесса на комплекс взаимодействующих процессов меньшей сложности;

– физическая параллельность применяется для повышения производительности устройств обработки информации. В недалеком прошлом физическая параллельность могла быть достигнута только введением избыточности функциональных устройств (многопроцессорности). В настоящее время повышение производительности микропроцессоров достигается за счет развития многоядерных процессоров, преимущества которых не могут в полной мере реализоваться без эффективно распараллеливаемых программных средств [6].

Для работы на многоядерном процессоре программное обеспечение должно быть многопоточным и допускать параллельную обработку нескольких потоков данных. В этом случае каждое ядро будет обрабатывать свой поток, или при единственности потока данных процессорные ядра должны будут обеспечить последовательную обработку данных, передавая при необходимости друг другу требуемую информацию [7].

Виртуальная машина VMware Workstation

Под «виртуальной машиной» в данной работе понимается программный продукт, обеспечиваю-

щий параллельную работу нескольких операционных систем на одной аппаратной платформе. Многоверсионные программы, исполняемые под изолированными операционными системами виртуальной машины, характеризуются естественной параллельностью, поэтому многоядерные процессоры являются в данном случае той аппаратной средой, которая практически идеально соответствует общим требованиям.

Сама по себе виртуализация стала важным инструментом в разработке компьютерных систем, а виртуальные машины используются в самых разнообразных областях – от операционных систем до языков программирования и архитектуры процессоров. Освобождая разработчиков и пользователей от ресурсных ограничений и недостатков интерфейса, виртуальные машины снижают уязвимость системы, повышают интероперабельность программного обеспечения и эксплуатационную гибкость аппаратной платформы [8].

На практике применение виртуальных машин характеризуется рядом положительных сторон. Это и возможность применения различных операционных систем, что способствует лучшей дифференциации версий, и изолированность выполнения задач, что обеспечивает стабильность функционирования при исключительных ситуациях, возникающих в других разделах.

Система виртуальных машин может быть построена на базе различных аппаратных платформ при помощи разных технологий. Схема виртуализации может отличаться в зависимости, как от используемой платформы, так и от выбора определенной операционной системы. Некоторые архитектуры обеспечивают возможность виртуализации аппаратно, другие требуют использования дополнительных программных ухищрений. В нашем случае мы рассматриваем виртуализацию с использованием дополнительных программных средств, а именно продукт VMWare Workstation.

Компания VMware производит разнообразное программное обеспечение для создания виртуальных машин, в том числе и средства для выполнения серверных операционных систем, широко применяющиеся, в частности, в некоторых решениях, поставляемых корпорацией IBM.

Продукт VMware Workstation поддерживает более разнообразный спектр как операционных систем хоста, так и гостевых операционных систем. Он предназначен для работы не только с различными версиями Windows, но и с разнообразными версиями Linux, Novell NetWare, DOS, Sun Solaris, FreeBSD. Помимо Windows-версии существует Linux-версия этого продукта.

Вложенная архитектура VMware использует оборудование совместно с базовой операционной системой как изображено на рис. 1.

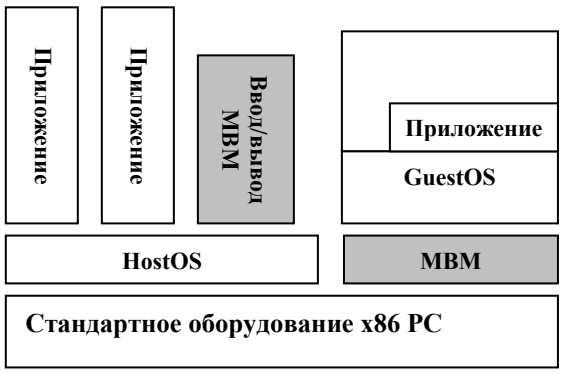

Рис. 1. Архитектура VMware

Общая системная архитектура виртуальной машины построена на взаимодействии трех основных компонентов: приложение виртуальной машины; драйвер виртуальных машин; монитор виртуальной машины [9].

Выводы

Исходя из вышеизложенного, можно сделать вывод о возможности построения надежного и функционально безопасного многоверсионного ПО на основе виртуальных машин. Такому решению способствуют и естественный параллелизм, присущий

виртуальным машинам, и современные тенденции многоядерности в развитии аппаратного обеспечения.

Основной идеей исследования является попытка оценить эффективность совместного применения:

– многоверсионности, как технологии создания сверхнадежных программных систем из не очень надежных компонентов;

– известных шаблонов проектирования, обеспечивающих надежность и функциональную безопасность. Среди шаблонов, представляющих интерес, выделим шаблоны гомогенной и гетерогенной избыточности;

– виртуальных машин, как инструментальных средств, обеспечивающих работу нескольких операционных систем одновременно на единой аппаратной платформе.

Одной из ключевых задач работы является изучение механизмов информационного взаимодействия между виртуальными машинами, в том числе оценка пропускной способности информационных каналов обмена между виртуальными машинами и измерение накладных расходов на организацию их работы. Решение данной задачи позволит дать оценку целесообразности использования такой модели для обеспечения ФБ и надежности программных систем критического назначения.

Литература

1. Wikipedia, the free encyclopedia [Электронный ресурс]. – Mariner 1, 18 January 2007. – Электронная энциклопедия. – Режим доступа: http://en.wikipedia.org/wiki/Mariner_1, свободный. – Загл. с экрана. – Яз. англ.

2. Wikipedia, the free encyclopedia [Электронный ресурс]. – Ariane 5 Flight 501, 17 January 2007. – Электронная энциклопедия. – Режим доступа: http://en.wikipedia.org/wiki/Ariane_5_Flight_501, свободный. – Загл. с экрана. – Яз. англ.

3. Липаев В.В. Технологические процессы и стандарты обеспечения функциональной безопасности в жизненном цикле программных средств // Информационный бюллетень Jet Info. – 2004. – № 8. – С. 3-28.

4. ГСУ 2850-94. Программные средства ЭВМ. Показатели и методы оценки качества. = Software Quality/ Characcteristik and methods of assessment. – Введ. 01.01.96. – Киев-6: СМП «Аверс», 1996. – 21 с. УДК 681.3. Группа П85.

5. Черкесов Г.Н. Надежность аппаратно-программных комплексов: Учеб.пособ. – СПб.: Питер, 2005. – 478 с.

6. Болинджер К. Врожденный параллелизм // Открытые системы. – 2006. –, № 2. – С.24-31.

7. Пантюхин В. Микропроцессорное многопоточие // Открытые системы. – 2006. – № 6. – С.10-18.

8. Вахрамеев К. Интенсификация виртуализации // Открытые системы. – 2006. – № 7. – С. 12-17.

9. Елманова Н. Виртуальные машины и средства их создания. Ч. 2. Vmware Worksation // Компьютер-пресс. – 2004. – 10. – С.162-163.

Поступила в редакцию 24.01.2007

Рецензент: д-р техн. наук, проф. И.Б. Туркин, Национальный аэрокосмический университет им. Н.Е. Жуковского "ХАИ", Харьков.