Использование технологии объединения ресурсов для создания безопасных отказоустойчивых военных систем

Сетевые военные технологии в значительной степени зависят от взаимосвязанных платформ, средств вооружения и связи, а также сенсорных систем. Безусловно, каждая такая "система систем" содержит в себе интеллектуальные возможности, что делает ее уязвимой для программных ошибок, злоумышленных атак и просто неправильно написанного кода. Одним из способов гарантированной защиты Глобальной информационной сети (Global Information Grid — GIG) от воздействия неисправностей её отдельных систем является использование операционных систем с технологией объединения ресурсов. Кроме того, среды с объединением ресурсов обеспечивают защиту от инверсий приоритетов, недостатка ресурсов и других непреднамеренных, но нередких действий программного обеспечения.

Информационная система GIG предназначена для того, чтобы помогать участникам военных действий распознавать и уничтожать неприятельские угрозы с беспрецедентной эффективностью благодаря обеспечению доступа к точной и своевременной информации почти из любого местоположения. Тем не менее, требование GIG о том, что все военные системы должны иметь сетевую архитектуру, неизбежно приводит к появлению в сети собственных угроз, в том числе вирусов, атак на отказ в обслуживании (DoS) и других форм компьютерных диверсий. Чтобы остановить распространение таких атак и обеспечить непрерывную готовность критически важных служб, технология объединения предоставляет гарантированную долю компьютерных ресурсов каждой программной подсистеме.

Ограниченные ресурсы

Обеспечение надёжности и безопасности сетевых военных систем является существенной проблемой, которую значительно усугубляет сложность современного программного обеспечения. Сложность системы способна подорвать её надёжность просто потому, что с ростом объёма кода системы повышается вероятность проявления ошибок кодирования и непредвиденных взаимодействий между программными компонентами системы в процессе её эксплуатации. Сложность программного обеспечения также способна причинить ущерб безопасности, поскольку хакеры, как правило, используют такие ошибки при желании повредить систему или проникнуть в неё. К сожалению, никакой объём тестирования не может полностью устранить эти проблемы, так как ни один тестовый комплект не способен предусмотреть все ситуации, которые могут произойти в сложной программной системе.

Процесс разработки программного обеспечения для военных систем создаёт дополнительные сложности. Например, интегратору, который выполняет сборку наземного транспортного средства, может потребоваться скомпоновать прикладные программы одного поставщика, стеки протоколов — другого, операционную систему реального времени — третьего и встраиваемую базу данных — четвертого. Затем интегратор должен объединить эти компоненты с многочисленными программными подсистемами внутренней разработки, которые написаны разными группами разработчиков. На рис. 1 показаны некоторые из многочисленных программных компонентов, которые может включать в себя такая система.

Рис. 1

При использовании параллельных процессов проектирования неизбежно возникают проблемы с производительностью на стадии интеграции, когда различные подсистемы впервые начинают конкурировать друг с другом за процессорное время и другие системные ресурсы. Подсистемы, которые хорошо работали в отдельности, реагируют медленно или не реагируют вообще. К сожалению, многие из таких проблем проявляются только во время интеграции и проверочных испытаний, когда затраты на повторное проектирование и кодирование программного обеспечения максимальны.

Чтобы обеспечить безопасное функционирование системы и успешную интеграцию её программных подсистем, интегратор должен реализовать архитектуру, которая не позволяет компонентам и подсистемам повреждать и монополизировать компьютерные ресурсы (например, память и процессорное время), необходимые другим подсистемам. Теоретически интеграторы могли бы достичь этой цели с помощью технологии аппаратного объединения — подхода, в котором каждая программная подсистема или группа подсистем работает на отдельной плате или узле. Такой подход способен локализовать сбои и снизить конкуренцию за совместно используемые ресурсы. Тем не менее, в военных системах существует тенденция к увеличению объема функциональности, интегрируемой в устройство. Это не только снижает затраты ("чем меньше устройств, тем дешевле"), но и сводит к минимуму вес и энергопотребление системы.

Безопасные разделы

Для того, чтобы решить проблемы безопасности и интеграции, некоторые архитектуры помещают группы программных процессов в виртуальные разделы или *блоки,* каждому из которых выделяется заранее определенное множество ресурсов, в том числе процессорное время (см. рис. 2). Таким образом, система не позволяет процессам, которые находятся в каком-либо блоке, непреднамеренно или злоумышленно монополизировать ресурсы, необходимые процессам в других блоках. Многие системы оборонно-космической промышленности, использующие технологию объединения, соответствуют спецификации ARINC 653, которая обеспечивает хорошо известный, хотя и несколько жёсткий и неэффективный подход к объединению ресурсов.

Рис. 2

Среди других возможностей блоков приложений — защита памяти в операционных системах, которые управляют доступом ко всей памяти с помощью диспетчера памяти. Например, операционная система на основе микроядра может разделить приложения, драйверы устройств, стеки протоколов и файловые системы на отдельные процессы с защищенной памятью. Если какой-либо процесс, такой как драйвер устройства, попытается осуществить доступ к памяти, находящейся за пределами своего контейнера, диспетчер памяти уведомит об этом операционную систему, которая затем сможет завершить процесс и перезапустить его.

Этот подход обеспечивает прямое и значительное улучшение надёжности системы:

- не позволяет ошибкам в коде одного процесса повреждать другие процессы и ядро операционной системы;
- даёт разработчику возможность быстро обнаруживать, диагностировать и исправлять нарушения, локализация которых другими методами может занимать несколько недель;
- значительно сокращает время послеаварийного восстановления: в случае нарушения памяти система может вместо перезагрузки просто перезапустить процесс, который вызвал сбой.

Гарантия доступности ресурсов

Тем не менее, для создания надёжной системы требуется не только распределить функциональность по отдельным областям памяти. Для многих систем критически важно гарантировать доступность ресурсов. Если лишить ключевую подсистему процессорных циклов, то службы, которые она предоставляет, окажутся недоступными пользователям. Например, при атаке на отказ в обслуживании внешняя система может бомбардировать устройство запросами, которые должны обрабатываться высокоприоритетным процессом. Затем этот процесс перегружает процессор и лишает другие процессы процессорных циклов, делая систему непригодной для использования.

Кроме того, во многих случаях добавление программной функциональности в систему способно "переполнить" её и отнять процессорное время у существующих приложений. Исторически единственным решением этой проблемы была модификация оборудования или перепроектирование программного обеспечения.

Планировщики с фиксированным объединением

Для разрешения описанных выше проблем в некоторых операционных системах используется планировщик с фиксированным объединением, который обычно основан на спецификации ARINC 653 и позволяет системному разработчику объединять процессы в блоки и выделять долю процессорного времени каждому блоку. При таком подходе ни один процесс не может израсходовать больше процессорного времени, чем статически выделено его блоку (например, 20 процентов).

Тем не менее, планировщики с фиксированным объединением имеют свои недостатки. Поскольку алгоритм планирования является фиксированным, блоки, которые не выполняют работу, простаивают в течение выделенных им процессорных циклов. В это время другие блоки не могут получить доступ к непроизводительным циклам, даже если заняты работой и могли бы воспользоваться дополнительным процессорным временем. Такой подход растрачивает ценные и доступные процессорные циклы и не позволяет системе справляться с пакетными запросами. Из-за правила "пользуйся или теряй" планировщики с фиксированным объединением способны использовать процессор лишь на 70 процентов.

Этот предел использования процессора ставит системного разработчика перед несколькими нежелательными альтернативами: воспользоваться более быстрым, тепловыделяющим и дорогим процессором, ограничить объем функциональности программного обеспечения до уровня, с которым способна работать система или просто смириться со сниженной производительностью.

Ограниченное использование процессора также является "кошмаром" для архитектур, в которых необходимо резервировать значительную долю процессорного времени для будущих приложений и расширений системы.

Кроме того, согласно стандарту ARINC, приложениям необходимо использовать интерфейс APEX для того, чтобы запрашивать службы операционной системы и взаимодействовать с другими приложениями. Это ограничение не позволяет приложениям использовать преимущества безопасного объединения.

Объединение ресурсов на многоядерных процессорах

Как и системы в любой другой индустрии, военные компьютеры и подсистемы управления радарами, полётами и объединения датчиков становятся всё более сложными и потребляют значительную вычислительную мощность. Многоядерные процессоры предоставляют идеальную возможность повысить производительность таких систем, обеспечивая значительно более высокую производительность на единицу мощности, веса и площади, чем традиционные однопроцессорные микросхемы. Фактически у системных разработчиков почти нет альтернативы использованию многоядерной технологии, поскольку она лежит в основе большинства новых процессорных архитектур.

Таким образом, операционная система должна иметь возможность поддерживать объединение ресурсов на многоядерном оборудовании. К сожалению, большинство существующих операционных систем реального времени, в том числе тех, которые включают в себя планировщиков с поддержкой блоков, способно управлять лишь одним процессором или процессорным ядром в каждый момент времени. В результате разработчики вынуждены запускать отдельную копию ОСРВ на каждом ядре многоядерной микросхемы. Поскольку ни одна такая копия не управляет всей системой, решение сложной задачи управления общедоступными аппаратными ресурсами микросхемы, в том числе физической памятью и периферийными устройствами, а также обработкой прерываний, возлагается на разработчика приложений, а не на операционную систему. Чтобы избежать этой сложности, системным разработчикам следует выбрать операционную систему реального времени, которая способна одновременно управлять множественными ядрами, контролировать общие ресурсы и обеспечивать динамическое распределение загрузки между ядрами, при этом гарантируя выделение ресурсов.

Планировщики с адаптивным объединением

Другой подход под названием *адаптивное объединение* устраняет перечисленные выше недостатки с помощью более динамического алгоритма планирования. Как и фиксированное объединение, адаптивное объединение позволяет системному разработчику резервировать процессорные циклы и память для процессов и групп процессов. Таким образом, разработчики могут гарантировать, что загрузка одной программной подсистемы не нанесёт ущерб работоспособности других подсистем.

Тем не менее, в отличие от подходов с использованием фиксированных блоков, адаптивное объединение учитывает, что загрузка процессора является случайной и один или более блоков могут часто простаивать. По этой причине планировщик с адаптивным объединением динамически перераспределяет неиспользуемые процессорные циклы между блоками, которые могут воспользоваться дополнительным процессорным временем. Этот подход, изобретённый компанией QNX Software Systems, сочетает два преимущества: он гарантирует выделение процессорного времени, когда в системе заканчиваются резервные циклы (для обеспечения работоспособности приложений и служб) и распределяет свободные процессорные циклы при их появлении (для максимального использования процессора и производительности). Например, блок 2 на рис. 3 потребляет не более 40 процентов процессорного времени, когда система работает на полную мощность. Тем не менее, этот блок может использовать более 40 процентов процессорных циклов, если другие блоки не полностью используют выделенный им бюджет процессора.

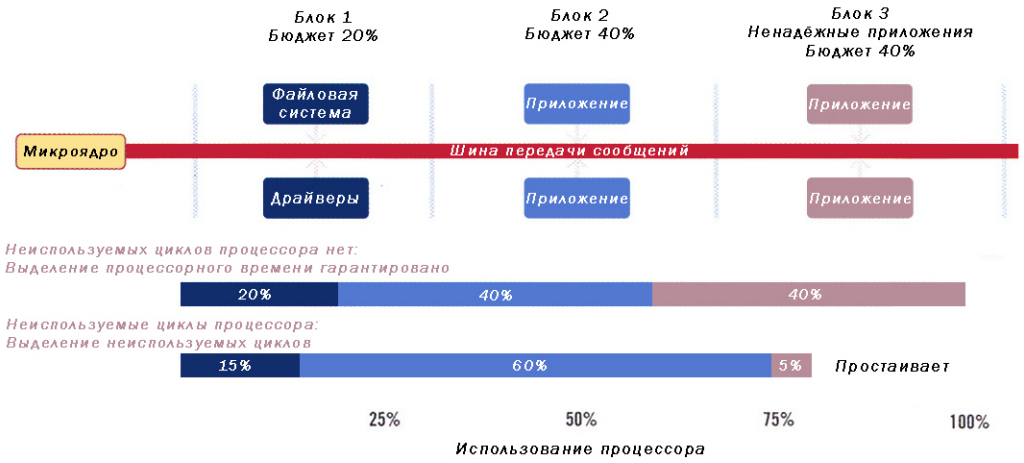

Рис. 3

Адаптивное объединение обеспечивает ряд других преимуществ, в том числе:

- возможность использования приоритетного планирования в реальном времени при небольшой загрузке, позволяющего сохранять в системах существующие алгоритмы планирования;
- возможность вводить планировщик с объединением в существующие системы, не изменяя код, что позволяет пользователям запускать существующие POSIX-приложения в блоке;
- управляемое достижение 100% использования процессора, что позволяет интеграторам реализовывать преимущества временного объединения без применения более быстрых и дорогостоящих процессоров;
- гарантированное выделение процессорных циклов процессам обнаружения сбоев и восстановления в количестве, достаточном для устранения программных сбоев, что сокращает среднее время восстановления;
- предоставление операторам возможности удалённого наблюдения системы и обнаружения неисправностей в ней, без нарушения работоспособности критически важных приложений;
- предотвращение захвата всего процессорного времени с помощью атак на отказ в обслуживании и атак с использованием вредоносных программ.

Несмотря на то, что адаптивное объединение обеспечивает большую гибкость, в некоторых ситуациях желательно использовать фиксированное планирование. Чтобы выполнить это требование, реализация адаптивного объединения должна обеспечивать системному разработчику возможность задавать в системе фиксированные бюджеты блоков без "одалживания" процессорного времени. Этот подход позволяет системному разработчику выбирать алгоритм планирования, который наилучшим образом удовлетворяет потребности его приложений.

Противоречивые требования

Рынок встраиваемых систем становится столь сложным, что без какой-либо формы объединения системным разработчикам и разработчикам программного обеспечения трудно удовлетворять противоречивые требования надёжности, производительности, безопасности, вывода системы на рынок и новых возможностей.

Объединение ресурсов позволяет поставщикам с лёгкостью интегрировать подсистемы, которые созданы множественными программными командами, субподрядчиками и сторонними разработчиками, обеспечивает работу новых и усовершенствованных компонентов без ущерба для текущего функционирования системы и останавливает распространение атак на отказ в обслуживании и других сетевых эксплойтов. Если решение с технологией объединения также предоставляет гибкий, эффективный планировщик, который позволяет использовать процессор на 100%, поставщики могут реализовать эти преимущества без затрат на более быстрое и дорогое оборудование.