УДК 004.4'233

СРЕДСТВА МОДЕЛИРОВАНИЯ ОС И ЕЕ КОМПОНЕНТОВ

Степанов В. Н.[1], студент гр.153503

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Рогов М.Г. – ассистент кафедры информатики

Аннотация. В ходе данного исследования были рассмотрены различные средства моделирования операционных систем (ОС) и их компонентов с целью определения наиболее эффективного подхода к моделированию.

Ключевые слова. Средства моделирования, операционная система (ОС), компоненты ОС, агентно-ориентированный подход, системная динамика, сетевое моделирование.

Средства моделирования операционных систем представляют собой программные инструменты, разработанные для создания виртуальных сред, в которых можно эмулировать и анализировать работу операционных систем и их компонентов. Эти инструменты играют ключевую роль в исследовании, разработке и тестировании операционных систем, а также в разработке и отладке программного обеспечения. Они позволяют исследователям и разработчикам проводить различные эксперименты и анализировать поведение операционных систем в различных условиях, что помогает оптимизировать их производительность, эффективность и надежность.

Идея моделирования операционных систем возникла с появлением компьютеров и необходимости понимания и улучшения их работы. Первые средства моделирования операционных систем были созданы в середине XX века и использовались в основном для исследования и тестирования новых алгоритмов и стратегий управления ресурсами. С течением времени и с развитием вычислительной техники средства моделирования стали более мощными и универсальными, что позволило использовать их для широкого спектра задач, включая разработку операционных систем, анализ и оптимизацию их производительности, а также обучение и обучение студентов и специалистов в области информационных технологий.

Современные средства моделирования операционных систем предоставляют широкий спектр функциональности, включая возможность создания виртуальных моделей аппаратного и программного обеспечения, эмуляцию различных архитектур процессоров и операционных систем, а также инструменты для анализа производительности и безопасности. Они могут использоваться как для исследовательских целей, так и для практического применения в различных областях, таких как разработка новых технологий и алгоритмов, тестирование программного обеспечения и обучение студентов. Средства моделирования операционных систем играют важную роль в современной информационной технологии и являются неотъемлемой частью процесса разработки и поддержки операционных систем.

Carsim. В CarSim - это коммерческий программный пакет, который прогнозирует работу транспортных средств в ответ на действия водителя (рулевое управление, дроссельная заслонка, тормоза, сцепление и переключение передач) в заданном окружающая среда (геометрия дороги, коэффициенты трения, ветер). CarSim производится и распространяется американской компанией Mechanical Simulation Corporation с использованием технологии, разработанной в Мичиганском университете Транспортном научно-исследовательском институте (UMTRI) в Анн-Арборе, штат Мичиган.

Используется программное обеспечение более 30 производителей автомобилей (General Motors, Toyota, Honda, Ford и др.), более 60 поставщиков и более 150 исследовательские лаборатории и университеты. Математические модели имитируют физические тесты, чтобы позволить инженерам просматривать результаты, аналогичные результатам тестов, но которые могут быть получены повторно, безопасно и намного быстрее, чем это возможно при физическом тестировании. Имитационные модели часто используются для оценки еще не построенных конструкций транспортных средств. Результаты визуализируются с помощью анимации, отображаются на графике для анализа или экспортируются в другое программное обеспечение для анализа с использованием тех же методов, которые применяются к данным физических испытаний.

Математические модели воспроизводят поведение системного уровня с высокой точностью. Они содержат основные эффекты, которые определяют, как шина контактирует с дорогой и как силы от поверхности контакта шины с дорогой передаются через подвеску на шасси. Однако в них нет подробностей о связях связей или соответствии структуры. Производители неоднократно проверяли модели для воспроизведения общих движений автомобиля, необходимых для оценки управляемости,

курсовой устойчивости, устойчивости к качению, торможения и ускорения. С другой стороны, они не включают детали компонентов, необходимые для определения долговечности, усталости или высокочастотных вибраций.

Математические модели генерируются с помощью генератора символьного многотельного кода под названием VehicleSim Lisp (первоначально названного AutoSim), который был разработан одним из основателей компании в UMTRI. Машинно-сгенерированный код сильно оптимизирован для достижения быстрых вычислений, так что математические модели работают намного быстрее, чем в реальном времени. Начиная с 1998 года, версии CarSim в реальном времени были доступны для тестирования аппаратного обеспечения в цикле (HIL). Математические модели используются непосредственно в более чем 350 симуляторах вождения, чтобы предоставить физические модели, которые были проверены в большинстве представляющих интерес условий.

Основными приложениями программного обеспечения CarSim являются:

- Инженеры-испытатели заранее моделируют сотни тестов для выявления проблем или четких схем, которые не показывают проблем;
- Разработчики расширенных средств управления (тормоза, устойчивость, тяга и т. д.) тестируют моделируемые конструкции управления с моделированным транспортным средством. В этих приложениях CarSim имитирует базовое динамическое поведение транспортного средства в качестве подключаемого модуля к программному обеспечению для проектирования контроллеров, например Simulink (из Mathworks), LabVIEW (из National Instruments), Activate (из solidThinking) или пользовательский код (MATLAB, Visual Basic, C / C ++ и т. Д.);
- Car производители и поставщики тестируют фактическое оборудование контроллера, используя системы HIL в реальном времени.
- Исследователи и другие специалисты используют математические модели CarSim в симуляторах вождения, начиная от недорогих систем, использующих игровой контроллер, до полноценных -масштабируйте большие симуляторы движения, такие как симулятор Toyota.
- Та же технология используется в BikeSim, программе для моделирования динамики мотоциклов и скутеров, и TruckSim, используемом для моделирования динамики грузовых автомобилей с двойными шинами., прицепы и более двух осей на единицу.

Network Simulator. Программный продукт network simulator, известный также как REAL, был основой для создания ns-2. Работа над ns-2 началась в 1996 году в рамках проекта VINT по инициативе DARPA. Разработка ведется при участии научных центров, таких как USC/ISI, Xerox PARC, LBNL и UCB, а также в рамках проектов SAMAN и CONSER, финансируемых DARPA и NSF соответственно. Участвуют также исследователи из других организаций, включая некоммерческий исследовательский институт ICIR.

Особеннности. Симулятор ns-2 осуществляет имитационное моделирование сетей на уровне пакетов, то есть, моделирует генерацию пакетов и прохождение их по сети. На прикладном уровне моделируется характер трафика, порождаемого различными приложениями: Web, FTP, Telnet, RealAudio; кроме того, имеются абстрактные модели трафика, например Constant Bitrate. Возможно моделирование работы протоколов транспортного уровня UDP и различных реализаций TCP, multicast-протоколов, различных протоколов маршрутизации в проводных и беспроводных сетях, очередей с дисциплинами обслуживания DropTail и RED. Кроме того, моделируются некоторые факторы, относящиеся к физическому уровню: задержка пакетов в каналах, возникновение ошибок, видимость/невидимость узлов в беспроводных сетях (как наземных, так и спутниковых), расход энергии батарей в устройствах с автономным питанием.

Результатом работы симулятора являются выходные текстовые файлы, в которых регистрируется ход моделирования (моменты генерации/получения пакетов, состояние очередей, отброс пакетов в очередях и т. д.). Кроме того, в модель могут быть включены инструкции, вычисляющие любые величины, измерение которых требуется в конкретной задаче (задержка пакетов, пропускная способность и т. п.). Значения этих величин в ходе моделирования также могут регистрироваться в выходных файлах.

Архитектура. Симулятор ns-2 состоит из двух частей. Одна из них написана на языке C++ и должна быть перекомпилирована в случае внесения изменений и дополнений; другая написана на интерпретируемом языке OTcl (объектно-ориентированное расширение языка сценариев Tcl) и, соответственно, не требует компиляции. При этом иерархии классов в обеих частях имеют совпадающие части и в терминологии ns-2 называются компилируемой и интерпретируемой иерархиями, соответственно. Взаимодействие между частями, написанными на таких принципиально

разных языках программирования, осуществляется согласно спецификации, определяющей способ обращения из tcl-сценария (скрипта) к любому методу классов компилируемой иерархии и возвращение назад результатов, а также способ обращения из программы на C++ к любому методу, описанному в Tcl-сценарии. Во втором случае, фактически, интерпретатор языка Tcl вызывается из C++ как функция. По замыслу создателей ns-2, все методы, имеющие дело с отдельными пакетами и потому требующие высокого быстродействия, относятся к компилируемой части. Интерпретируемая же часть отвечает за менее частые события, чем передача пакетов, обеспечивающие управление ходом моделирования, и манипуляцию объектами, описанными в компилируемой части. Также, Tcl-скриптом является собственно описание модели сети, подлежащей исследованию.

Такой подход позволяет быстро построить требуемую модель сети с помощью скриптового языка OTcl без необходимости вникать в структуру компилируемой части ns-2. В случае, если необходима модификация или дополнение компилируемой части, это может быть сделано путем добавления (или изменения) C++ кода и перекомпиляции системы. Единственный недостаток такого подхода – трудности при изучении системы и отладке программ (моделей), возникающие вследствие использования двух языков.

Согласно объектному подходу в имитационном моделировании в ns-2, сеть представляет собой совокупность сетевых объектов, каждый из которых реагирует на события. Планировщик извещает сетевые объекты о событиях, храня их в хронологической таблице. В системе ns-2 пакеты являются основным типом событий, генерируемых объектами-отправителями и принимаемые объектами-получателями. Остальные события, такие как at-события, создаются пользователями для управления моделью.

Планировщик событий. Планировщик по очереди (в хронологическом порядке) выбирает из таблицы события и извещает соответствующие сетевые объекты о наступлении этих событий, сообщая также момент времени, в который произошло это событие. С точки зрения программиста, такое «извещение о событии» представляет собой просто вызов метода-обработчика для соответствующего объекта. Вся сопутствующая информация передается через параметры этого метода.

Помещать события в таблицу планировщика могут как сами сетевые объекты, так и пользователь при описании модели. Как говорилось выше, в первом случае событие называется пакетом, во втором – at-событием.

Таким образом, любое взаимодействие между сетевыми объектами осуществляется через посредство планировщика. Отправка пакета одним сетевым объектом другому есть ни что иное, как постановка отправителем в таблицу события, адресованного получателю и помеченного тем моментом времени, когда пакет должен до этого получателя дойти (возможно, с учетом задержки при передаче). Получение пакета, соответственно, представляет собой извещение планировщиком получателя о событии и обработка этого события получателем.

При этом (если речь не идет о планировщике реального времени) модельное время не имеет никакого отношения к реальному. Планировщик извлекает события из таблицы, обращая внимание лишь на порядок их следования. Интервал между обработкой последовательных событий определяется лишь скоростью их обработки компьютером и не зависит от того, какими моментами модельного времени помечены эти события в таблице. Модельное время находит свое отражение лишь в трассировочных файлах, куда сетевые объекты записывают моменты отправки и получения пакетов (естественно, в модельном времени).

В системе ns-2 предусмотрен также планировщик реального времени (RealTime Scheduler), который синхронизует модельное время с реальным, ожидая между выборкой последовательных событий из таблицы по возможности в точности столько времени, сколько составляет разность между моментами, которыми помечены эти события. Такие планировщики используются, когда компьютерная модель входит в состав реальной сети. Например, создав на одном или нескольких компьютерах модель большой сети с планировщиком реального времени и подключив к такой модели два других компьютера, можно реалистично имитировать ситуацию, когда эти компьютеры связываются друг с другом, находясь в любых двух заданных точках этой большой сети. Этот планировщик, однако, далее в данной лекции не рассматривается.

Структура пакета. Из соображений простоты все пакеты в системе имеют одинаковый формат (являются объектами одного класса), и поэтому каждый пакет, независимо от того к какому протоколу он относится, имеет в своем составе заголовки всех протоколов, используемых в модели. Типичная структура пакета в ns-2 изображена на рис. 1. Поле данных используется лишь в случаях, когда ns-2-модель входит в состав реальной сети; если модель чисто компьютерная, то достаточно иметь в заголовке поле, содержащее размер пакета. Для ускорения процесса моделирования необходимо указывать явным образом, какие протоколы будут использоваться в модели, чтобы исключить заголовки неиспользуемых протоколов из заголовка пакета (рис 3.1).

Simics. Структура Simics является полноплатформенным симулятором, созданным для запуска неизмененных исполняемых файлов целевой платформы. Разработка Simics началась в Шведском институте информатики, но позже была передана компании Virtutech для коммерческой эксплуатации. В настоящее время симулятор Simics распространяется компанией Wind River Systems, дочерней структурой Intel, приобретшей Virtutech в 2010 году. Simics поддерживает широкий спектр архитектур, таких как Alpha, x86-64, IA-64, ARM, MIPS, MSP430, PowerPC, POWER, SPARC-V8 и x86, и позволяет запускать множество операционных систем, включая MS-DOS, Windows, VxWorks, OSE, Solaris, FreeBSD, Linux, QNX и RTEMS. Simics часто используется в качестве виртуальной платформы для разработки ПО для встроенных систем. Одной из его особенностей является возможность воспроизведения симуляции в обратном направлении.

Simics является симулятором – программным решением, симулирующим аппаратуру. Такой симулятор часто называют виртуальной платформой, так как он предоставляет виртуальную аппаратную платформу для запущенного на нём ПО. Следует отметить, что Simics виртуализирует встроенную аппаратуру по-другому, нежели гипервизоры, такие как Wind River Hypervisor.

Гипервизор ожидает, что операционная система, запущенная в нём, поддерживает определённую архитектуру виртуальной машины, тогда как виртуальная платформа Simics симулирует конкретную аппаратуру. Simics представляет собой инструмент, позволяющий разрабатывать ПО без наличия соответствующей аппаратуры, в то время как гипервизор является способом управления аппаратурой при её работе. Можно запустить гипервизор и операционные системы, исполняемые в нём, в симуляторе Simics. Wind River Hypervisor был даже разработан с использованием симулятора Simics.

Аппаратура, которая может быть просимулирована с помощью симулятора Simics, варьируется от базовых встраиваемых платформ с одним процессором или системой на чипе, вплоть до серверов и отказоустойчивых кластеров, собираемых в стойках. Виртуальная платформа достаточна полна и точна, чтобы заставить работать целевое ПО так, как оно работает на реальном железе, и достаточно быстра для того, чтобы использовать её для обычных работ по разработке ПО.

С точки зрения целевого ПО, Simics выглядит как настоящая платформа. Simics способен запускать те же исполняемые файлы, что и физическая целевая система, и исполнять их точно также, как они были бы исполнены на реальной машине. Программный стек, исполняемый в симуляторе, включает в себя всё, начиная от загрузочного кода и заканчивая гипервизорами, операционными системами и пользовательскими приложениями. Целевой код способен исполняться на симуляторе без изменений (хотя дополнительные возможности для модификации кода имеются и используются, когда они нужны).

Simics – это обычное приложение. Ему не требуется специального оборудования, плат с FPGA или специальных эмуляторов. Simics может работать на любом ПК, везде, в любое время (по крайней мере, если на ПК используется Windows или Linux). Вы можете отправить виртуальную платформу, реализованную с помощью симулятора Simics, буквально в электронном письме. Таким образом можно заменить аппаратные платы для команд разработчиков, располагающихся на большом удалении друг от друга, прямо-таки волшебным способом.

Одна из значительных частей Simics симуляции – это симуляция целевого оборудования. Она включает в себя модели процессорных ядер, шин и других соединений, модели запоминающих устройств, периферийных устройств и модели сетей.

Ключевым ядром симулятора являются системы моделирования набора команд, которые могут моделировать ARM, DSP, MIPS, Power архитектуру, SPARC, x86/IA и другие процессоры. Тем не менее, сама по себе модель процессора не позволит запустить на ней операционную систему. Виртуальные платформы Simics по этой причине также включают в себя модели блоков управления памятью (MMU), а также запоминающих и прочих устройств, которые доступны процессору.

Рассмотрим то, как процессором выполняется базовая операция: доступ в память. Когда процессор делает запрос на чтение или запись памяти, то в первую очередь адрес транслируется блоком управления памятью, выдающим физический адрес. Физический адрес используется для создания транзакции в симуляторе Simics.

Далее транзакция проходит через карту памяти, которая определяет куда в итоге транзакция будет отправлена. Если транзакция попадает в память, то содержимое памяти считывается или модифицируется.

Если транзакция попадает в периферийное устройство, то для её обработки вызывается модель этого устройства. Далее модель устройства выполняет необходимые действия. Она может запустить таймер, отправить прерывания к другому процессору в системе, изменить конфигурацию устройств, выполнить перезагрузку, отправить пакет по сети, выставить высокое значение сигнала на выводе или сделать что-нибудь ещё. Если транзакция не находит адресата, то симулируемый процессор может сгенерировать исключительную ситуацию, связанную с ошибкой на шине.

В отличие от других виртуальных платформ, Simics может добавлять в симуляцию объекты в любой момент времени. Также можно на лету переконфигурировать объекты и изменять их внутреннее состояние.

Для того, чтобы поддерживать динамические изменения, все объекты, используемые в симуляторе, создаются из классов, которые загружаются динамически. Каждая модель, используемая в симуляторе, определяется в своём .dll или .so файле и может быть загружена динамически. Пользователю не нужно ничего перекомпилировать, чтобы создать новую конфигурацию, а только загрузить необходимые модули в симулятор динамически. Simics очень похож на среду Java и .NET в плане того, как объекты компилируются, загружаются, управляются и соединяются. Это не статическая линковка, как в простых C и C++ программах.

С практической точки зрения подход, используемый в симуляторе Simics, имеет то преимущество, что каждый симуляционный модуль может быть скомпилирован отдельно, что делает перекомпиляцию моделей устройств недорогой операцией.

Чтобы справиться со сложностью моделируемой аппаратуры, Simics позволяет создателю виртуальной платформы использовать специальный вид объекта, называемый компонентом, для объединения объектов в логические группы. Компоненты группируют устройства, запоминающие устройства, соединения и процессорные ядра вместе, в логические модули, соответствующие чипам, системам на чипе, специализированным интегральным схемам, системным платам, мезонинам, серверным стойкам и другим аппаратным узлам. Компоненты могут быть повторно использованы и вложены произвольным образом, моделируя любой тип иерархии аппаратных узлов. Обходя иерархию компонентов, легко понять структуру (виртуальной) аппаратной системы.

Список использованных источников:

1. Главная загрузочная область – [Электронный ресурс] Электронные данные. – Режим доступа: https://studfile.net/preview/9565397/page:3.

2. Главная загрузочная запись – [Электронный ресурс] Электронные данные. – Режим доступа: https://shorturl.at/brJK7.

3. Rootkit [Электронный ресурс] – Электронные данные. – Режим доступа: https://en.wikipedia.org/wiki/Rootkit.

4. Буткиты: эволюция и способы обнаружения – [Электронный ресурс] Электронные данные. – Режим доступа: https://www.ptsecurity.com/ru-ru/research/analytics/Bootkits-evolution-and-methods-of-detection.

5. Secure Boot – [Электронный ресурс] Электронные данные. – Режим доступа: https://wiki.debian.org/SecureBoot#What_is_UEFI_Secure_Boot.3F.

6. Что такое TPM – [Электронный ресурс] Электронные данные. – Режим доступа: http://al-tm.ru/stati/stati-po-setyam/trusted-platform-module.

UDC 004.4'233

TOOLS FOR MODELING THE OS AND ITS COMPONENTS

Stepanov V.N. [1]

Belarusian State University of Informatics and Radioelectronics[1]*, Minsk, Republic of Belarus*

Rogov M.G. – Assistant of the Department of Informatics

Annotation. This study examined various tools for modeling operating systems (OS) and their components in order to determine the most effective modeling approach.

Keywords. Simulation tools, operating system (OS), OS components, agent-based approach, system dynamics, network modeling.