Микроядерные Операционные Системы: достижения последнего десятилетия и вызовы будущего

Введение

Микроядерные Операционные Системы (ОС) привлекают внимание разработчиков и исследователей последние 30 лет. Действительно, с одной стороны эти системы обеспечивают жесткое разделение исполнимых модулей, вынесение всех служб из ядра в пространство пользователя, способствует повышение отказоустойчивости и безопасности системы. С другой стороны издержки, возникающие из-за поддержки такой распределенной структуры (вызванные частыми переключениями контекста) требуют дополнительных вычислительных ресурсов. Это противоречие заставляет ученых и разработчиков искать новые архитектурные решения для минимизации негативных и усиления позитивных сторон микроядерных ОС.
За последнее десятилетие технологии вычислительных систем осуществили огромный скачек вперед и теперь да же в мобильных устройствах присутствуют процессора с частотой большей 1Ghz и модули оперативной памяти объемом 512-1024 MB. Кроме того, внутренняя архитектура программного обеспечения существенно увеличила свою сложность, появились большие программные стеки изолированных модулей осуществляющих интенсивный обмен данными. Например: При просмотре видео в интернете, в браузере исполняется код flash проигрывателя, являясь отдельным исполнимым модулем. Этот модуль взаимодействует с библиотеками кодека, декодирующего видео, с оконным менеджером, с графической подсистемой, и, в конечном счете, с ядром ОС. Такая архитектура, как можно заметить, основана на идее жесткого разделения исполнимых модулей и коммуникации между ними. Если в такой системе ядро будет построено в соответствии с

микроядерной идеологией, то добавится дополнительный уровень абстракции связанный с аппаратным вводом-выводом. Иными словами на фоне интенсивного межпроцессного взаимодействия (IPC) в современных системах, дополнительный IPC возникающий от микроядра не значителен, а для некоторых приложений с соответствующей архитуктурой может быть даже более оптимальным. Таким образом, если брать во внимание архитектуру системы целиком, а не только ядра и его компонентов, можно сделать вывод что идея использования микроядерных ОС снова может стать актуальной. Данная статья представляет собор обзор наиболее ярких событий и достижений последнего десятилетия в области микроядерных ОС.

Первое поколение микроядер

Первым поколением микроядерных ОС называют системы разработанные и созданные в 80ых года XX века. Наиболее известными проектами являются ядро Mach и операционная система QNX. Первый проект был результатом исследований в области архитектуры ОС осуществленный командой ученых в Carnegie Mellon University в конце 80-ых — начале 90-ых. Это ядро было медленным[1] и поддерживало только архитектуру x86, но дало огромный импульс разработке микро ядерных ОС. После закрытия проекта в CMU, микроядро развивала команда исследователей в Utah University и в то время ядро стало частью проекта GNU, получив название GNU/Mach. В настоящее время GNU/Mach является частью ОС GNU/Hurd и развивается командой энтузиастов при поддержке компании Debian.

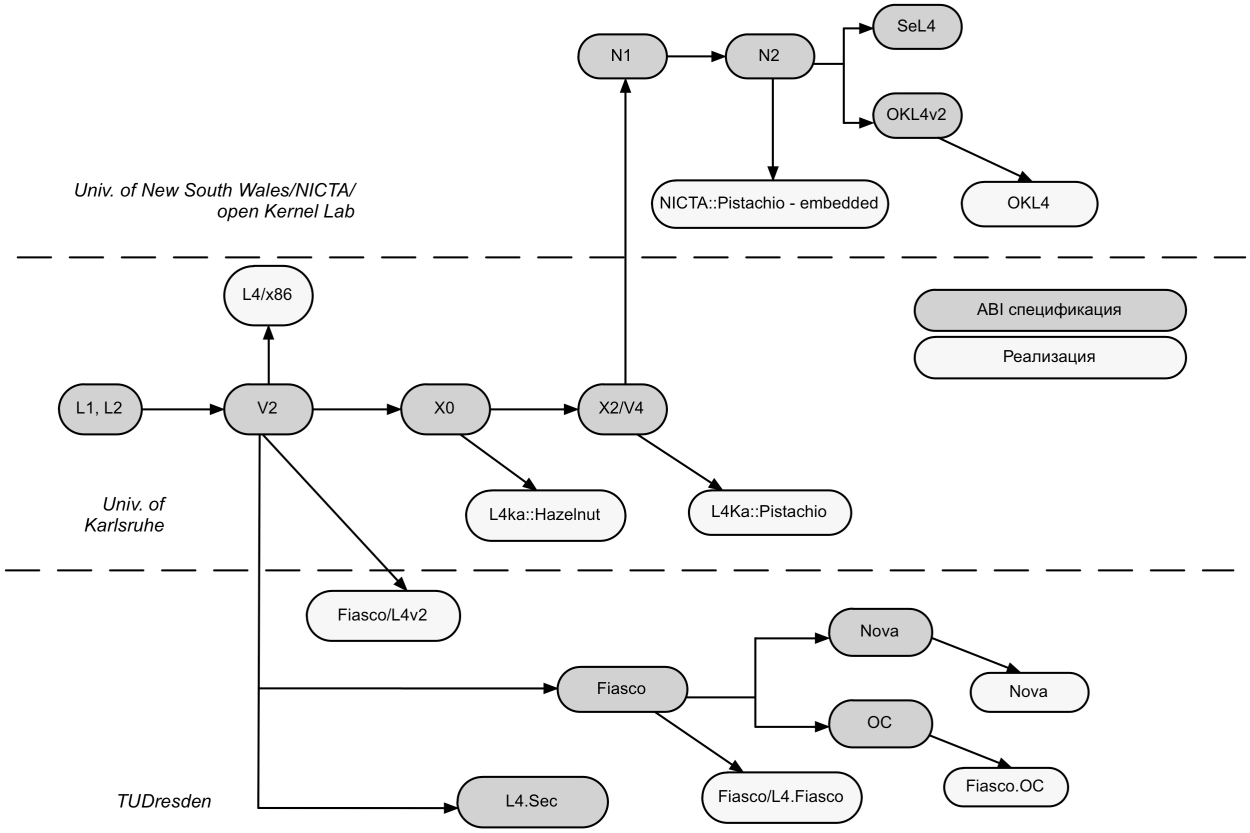

Рис. 1. Семейство микроядерных ОС архитектуры L4

Второе поколение микроядер: L4

Йохен Лидтке (Jochen Liedtke) в конце 90ых предложил новую, оптимизированную архитектуру микроядра, сначала L3, потом L4. Эта архитектура подразумевала синхронный IPC, минимальный функционал ядра, вынесение всех политик (policy) в пространство пользователя. Стремление уменьшить размер ядра помимо вопросов отказоустойчивости и безопасности ставило задачу исполнения кода ядра из кеша процессора, что должно было убрать издержки на частые переключения контекста.

L4 это идеология, и она оформлена в документах L4 eXperimental reference manual. Сейчас последняя версия[2] X.2-rev.7. На основе этой архитектуры может быть запрограммирована операционная система, благодаря чему появилось большое количество новых проектов, которые построены на L4 IPC, но преследуют разные цели и задачи.

Таких проектов за последнее десятилетие было больше двух десятков, но в этой статье представлены наиболее известные из них.

L4Ka, Karlsruhe Institute of Technology (KIT)

Операционная система L4Ka разрабатывалась командой Йохена Лидтке в соответствии с архитектурой L4. Ядро в этой ОС носило название Hazelnut и было построено в соответствии с версией X.0 L4 API. Более позднее ядро построенное в соответствии с ревизией X.2 носило название Pistachio. Это ядро существует до сих пор, но команда KIT не занимается его развитием. Поддерживает большое количество архитектур, таких как arm, power pc, x86 и другие. Ядро Pistachio получило большое распространение и на его основе было создано несколько коммерческих проектов.

Проекты NICTA Group

NICTA Group[3]— Information and Communications Technology (ICT) Research Centre of Excellence В Австралии. Nicta представляет собой государственную организацию, объединяющую ведущие ВУЗы Австралии и специализируется на узком направлении исследований. В эти направления входят научные проекты в области системного программирования, управления, искусственного интеллекта и распознавания образов, оптики и наноэлектроники – наиболее перспективными направлениями начала 21ого века. Эта организация не только проводит исследования, но и целенаправленно доводит эти исследования до рынка в виде бизнес проектов. Это можно увидеть на примере проекта L4Ka. Исследователи из NICTA разработали встраиваемую микроядерную систему NICTA::Pistachio-embedded на основе проекта L4Ka из KIT. Далее была создана компания OKL (Open Kernel Lab), которая занималась разработкой коммерческих продуктов на основе микроядра Pistachio-embedded. В 2007 году компания OKL анонсировала собственно микроядро OKL4, и это микроядро получило большое распространение в коммерческих продуктах в партнерстве с Qualcomm. В 2012 году OKL4 была продана[4] компании General Dynamics.

L4.verified — формально верифицированное ядро на архитектуре L4. Это научная разработка NICTA Group, и можно предположить, что трек у этого проекта будет аналогичен проекту OKL. Основная идея этого проекта заключается в математическом доказательстве корректности реализации ядра. Микроядерные ОС обладающие минимальным размером могут быть формально верифицированы. То есть по каждой строчке кода может быть написано математическое выражение, все функции превращены в теоремы, и дальше, используя математический аппарат можно доказать что реализация операционной системы соответствует полностью модели по которой она построена. В частности для L4.verified доказаны соответствие реализации модели и отсутствие вечных циклов. Стоимость этого исследования была порядка 6 млн$, длился проект 7 лет. Объем работы был оценен[5] в 25 человеко-лет. Это колоссальная и очень сложная работа. Вероятными областями применений такой системы являются военное оборудование и системы безопасности.
seL4 это еще один проект NICTA и является операционной системой для систем безопасности. Информации об этом проекте мало, но с официального сайта есть возможность скачать исполнимые образы.

Проекты Technische Universität Dresden (TUD)

Научная группа под руководством профессора Hermann Härtig в TUD участвовала в разработке микроядерных ОС совместно с Йохеном Лидтке. Эта группа создала собственную реализацию архитектуры L4 в микроядрах L4.Sec, Fiasco/L4.Fiasco и Fiasco.OC. Кроме того ими была разработана операционная система реального времени DROPS[6] (Dresden Real-Time Operating System Project), разработана доверенная графическая система Nitpicker и многое другое.
Современные ядра:
Fiasco.OC[7] микроядро реализованное на языке программирования C++, построенное по концепции Everything Is Object. Поддерживает архитектуры

ARM, x86, ppc, sparc v8, а так же современные отладочные средства и платформы на подобие Pandaboard и Beagleboard. Самостоятельно микроядро функционировать не может как следствие у него должно быть окружение – набор прикладных программ и модулей реализующих полезный функционал. Таким окружением для микроядра Fiasco.OC является L4Re[8] (L4 Runtime Environment). В его состав входят паравиртуализированный L4Linux - позволяющий использовать скомпилированное для Linux прикладное ПО, DDEKit - набор исходного кода (wrapper), позволяющий использовать драйвера ОС Linux, а так же большой набор библиотек и прикладных программ, адаптированных для запуска поверх микроядра. Среди таких приложений стоит выделить TCP/IP Стек LwIP с набором драйверов сетевых устройств, доверенный графический интерфейс (GUI) Nitpicker и графическую подсистему Mag. Это все позволяет создавать на основе микроядра Fiasco.OC современные аппаратно-программные комплексы.

Genode Framework[9] является Open Source проектом выпускников TUD. Его разработчики ставили перед собой задачу создания унифицированного окружения для множества разных микроядерных проектов. Важными качествами этого проекта является наличие доверенного графического интерфейса пользователя (GUI), поддержка Qt, а так же специальных возможностей микроядер. Например, для микроядра Fiasco.OC разработан паравиртуализированный L4Linux, представляющий собой ядро Linux запускаемое в пространстве пользователя. Фреймворк Geode позволяет использовать L4Linux в свое окружении. Аналогично присутствует поддержка L4Linux для ядра OKL4. Кроме того Genode Фреймворк содержит в себе DDE_Kit для использования исходного кода драйверов Linux. Genode поддерживает разнообразное количество ядер, таких как Fiasco, Fiasco.OC, OKL4, Pistachio, HelenOS, Linux. Кроме того оно содержит собственное ядро с минимальным функционалом называемым base-hw. С недавних пор Genode стал способен заменить полностью Linux для сборки, то есть сборку и отладку окружения можно полностью производить на Genode. Помимо этого Genode

поддерживает механизмы аппаратной виртуализации ARM TrustZone, что позволяет использовать этот фреймворк для создания доверенных решений. Микро гипервизор NOVA[10] (NOVA OS Virtualization Architecture) является гипервизором первого порядка (bare metal), то есть запускается напрямую на аппаратной платформе. Он разработан выпускниками TUD и сейчас развивается в Intel. В основе этого микрогипервизора находится микроядерная ОС архитектуры L4. Архитектура системы такова, что позволяет исполнять прикладное ПО параллельно с виртуальными машинами. Благодаря этому на основе гипервизора NOVA можно создавать доверенные решения, в которых доверенное ПО (например криптография) будет исполняться в отдельном, жестко отделенном от виртуальных машин окружении. Таким окружением например может быть фреймворк Geode, так как в Genode есть поддержка этого гипервизора. Помимо Genode NOVA поддерживает собственное окружение NOVA User Land (NUL).

Другие микроядерные проекты

Помимо семейства L4 в последнее десятилетие создавались и развивались другие микроядерные проекты. Микроядро QNX продолжало развиваться и стало де-факто стандартом для применения в системах промышленной автоматизации.

MINIX3

MINIX3 являлся учебным проектом по разработке операционной системы. Эндрю Таненбаум, архитектор и разработчик системы, является автором множества учебников по архитектуре вычислительных и операционных систем. Основной упор в микроядре MINIX3 сделан на отказоустойчивости и безопасности. Эта система обладает многофункциональным графическим интерфейсом пользователя (GUI) и может быть использована в качестве настольной (Desktop) системы. В настоящее время MINIX3 продолжает свое развитие

HelenOS

Микроядерная операционная система HelenOS[11] так же являлась учебно-научной работой как и MINIX3. Эта система разработана студентами Faculty of Mathematics and Physics at Charles University in Prague. Изначально в этом проекте было множество архитектурных недостатков. Например, система позиционировалась как микроядерная, в то время как часть драйвера каждого устройства находилась в пространстве ядра. Кроме того HelenOS не поддерживала оперативную память больше 2Gb, а набор драйверов был минимальным и устаревшим. Эта система не позволяет использовать драйвера Linux а так же запускать исполнимый код для Linux. В последние несколько лет архитектура операционной системы была серьезно переработана, весь код драйверов был полностью убран из ядра в пространство пользователя, а после сотрудничества с Genode Labs микроядро HelenOS получило поддержку в Фреймворке Genode.

Singularity

Singularity[12] - это исследовательский проект компании Microsoft Research. Эта операционная система работает в общем адресном пространстве, а изоляция[13] процессов создана программно, а не аппаратно. В обычных монолитно-модульных, или микроядерных L4 системах, разделение адресных пространств делает по средством аппаратного обеспечения: центральный процессор обладает разными режимами работы (Ring0-Ring3), и привилегированный код ядра и код пользовательских программ находятся в разных режимах (кольцах). Передача данных между ними осуществляется при помощи прерываний, а жесткая изоляция процессов гарантируется блоком MMU (Memory Management Unit). Такая архитектура для микро ядерных систем добавляет большое количество лишних операций при передаче данных, и разработчики Singularity решили отказаться от разделения на user space и kernel space, благодаря чему производительность Singularity на операциях ввода-вывода больше да же чем монолитно-модульный Linux.

Отказ от аппаратного разделения пространств ядра и пользователя потребовал разработки новых механизмов безопасности. Они реализованы програмно, и основываются на свойствах языка программирования, на котором разработано ядро и прикладное программное обеспечение. Ядро в основном написано на языке Sing# (потомок C#) производящий байт-код. Этот байт-код может быть проверен математически с целью выявить операции нарушающие границы выделенных регионов памяти. Такая верификация дает гарантию что ни драйвера ни программы не осуществляют никакой вредоносной деятельности. Защита доступа к памяти со стороны устройств реализована с использованием IOMMU. Этот механизм ограничивает DMA доступ устройства к оперативной памяти, что позволяет обезопасить от повреждения ядра и программ вредоносным устройством.

Barelfish

Операционная система Barelfish[14] является первой системой построенной по принципу «Мультиядерная система». Эта экспериментальная система разработана совместно учеными из Eth Zurich и Microsoft Research. Концепция мультиядерной системы ориентирована на применение в многоядерных и многопроцессорных системах, и базируется на идее использования множества операционных систем привязанных к вычислительным ядрам внутри одного процессора или системы на кристалле. Такой подход создает структуру на подобии вычислительного кластера, но в роле отдельных станций (вычислительных комплексов) выступают вычислительные ядра процессора. Каждое из мультиядер представляет собой микроядро и передача данных между системами основывается на механизме сообщений.

Система опубликована под лицензией BSD и ее исходный код доступен.

Заключение

Анализируя проекты последнего десятилетия можно сделать вывод о неугасании интереса научного сообщества к микроядерным ОС. Микроядерные ОС за последнее десятилетие произвели качественный скачек в производительности и в настоящее время переходят из состояния исследовательских университетских проектов в реальные коммерческие продукты. Кроме того архитектура микроядерных ОС развивается для применения в распределенных системах (Barelfish), встраиваемых систем и систем общего назначения (Fiasco.OC, Genode, HelenOS, MINIX3), а так же узкоспециализированных доверенных решений и безопасности (Singularity, seL4, L4.Verified). Коммерческий успех проекта OKL4 демонстрирует потребность в микроядрах со стороны больших корпораций.
Разрабатываемые окружения и средства виртуализации стремятся повторно использовать разработанный ранее код, что способствует быстрому переходу из окружения популярных монолитно-модульных ОС в окружение микроядерных. В то же время многие проекты требует разработчик нового ПО, оптимизированного под работу поверх микроядра. Таки проекты могут стать хорошим началом как и научной карьеры студентов, так и коммерческим продуктом.

Ссылки и литература

1. Härtig H. et al. The performance of µ-kernel-based systems //ACM SIGOPS Operating Systems Review. – ACM, 1997. – Т. 31. – №. 5. – С. 66-77.
2. Dannowski U. et al. L4 experimental kernel reference manual //Version X. – Т. 2.
3. http://www.nicta.com.au
4. http://www.theaustralian.com.au/australian-it/government/nicta-sells-ok-labs-to-general-dynamics/story-fn4htb9o-1226472637476
5. Klein G. et al. Verifying a high-performance micro-kernel. – 2007.

6. http://os.inf.tu-dresden.de/drops/
7. http://os.inf.tu-dresden.de/fiasco/
8. http://os.inf.tu-dresden.de/L4Re/
9. http://genode.org
10. http://hypervisor.org
11. http://helenos.org
12. http://research.microsoft.com/en-us/groups/os/singularity/
13. Hunt G. C., Larus J. R. Singularity: rethinking the software stack //ACM SIGOPS Operating Systems Review. – 2007. – Т. 41. – №. 2. – С. 37-49.
14. http://www.barrelfish.org