1

УДК 004.451.87 + 004.451.88

ХАРАКТЕРИСТИКИ МИКРОЯДЕРНЫХ ОПЕРАЦИОННЫХ СИСТЕМ В СРАВНЕНИИ С ДРУГИМИ ВИДАМИ АРХИТЕКТУР ОС

Дмитрук С.С., Лебединский А.А., Теплинский С.В.
Донецкий национальный технический университет

Рассматривается общая классификация подходов к структурированию составляющих частей операционных систем. Выполняется краткое сравнение систем на основе микроядра с другими архитектурами. Приводятся основные характеристики микроядерных систем, которые определяют их уникальные по сравнению с другими архитектурами свойства.

Введение

Операционная система, сокр. ОС (англ. *operating system, OS*) – комплекс управляющих и обрабатывающих программ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой стороны – предназначены для управления устройствами, управления вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надёжных вычислений.

Компьютерное аппаратное и программное обеспечения прошли длительный путь развития. Но в то время, как прикладное программное обеспечение претерпело огромное количество изменений, системное ПО (СПО) хоть и значительно продвинулось вперёд, всё же и сегодня имеет очень много черт характерных СПО середины XX-ого века. Так что в некотором роде СПО отстаёт в своём развитии от прикладного программного обеспечения. Поэтому в целях форсирования исследований в области разработки системного программного обеспечения, следует уделять больше внимание альтернативным (не тем, которые реализованы в доминирующих на рынке системах, а реализованным преимущественно в исследовательских проектах) подходам в разработке как всего СПО в целом, так и самой главное его части – ядра операционной системы.

Целью данной статьи является описание преимуществ, которыми обладает архитектура ОС на основе микроядра, по сравнению с доминирующим в современных условиях типом систем – систем с ядром монолитного типа.

1 Виды архитектур операционных систем

Как в истории развития программного обеспечения любого типа, в разработке операционных систем первые образцы представляли собой в основном монолитные структуры. Причиной этому было то, что несмотря на накопленный опыт разработки других типов систем, ещё не было разработано такое количество ОС, которое необходимо для проведения структурного анализа с целью выделения функциональных частей, обязательно присутствующих в любой универсальной ОС (именно универсальной, так как это наиболее распространённый тип систем).

Со временем, из-за необходимости эффективно справляться с всё возрастающей сложностью был предложен ряд способов разбиения ядра на более-менее обособленные функциональные составляющие. Как это часто бывает, предложения поступали прежде всего из академической среды, а проверялись преимущественно в коммерческих разработках. В качестве примера можно привести разработку известного голландского учёного Эдсгера Вайба Дейкстры – операционную систему THE[3], или же учебную микроядерную операционную систему Эндрю Таненбаума – Minix [1], опираясь на пример которой в качестве любительского проекта Линусом Торвальдсом было разработано ядро ОС Linux [4].

В результате сегодня, спустя более чем полвека развития компьютерной науки и

промышленности, выделяют следующие основные виды структуры ядра операционной системы:

– монолитные ядра (монолиты);
– микроядра;
– экзоядра (экзо – приставка, обозначающая нечто внешнее, находящееся снаружи).

Несмотря на название «монолитные ядра», нельзя проводить аналогию между современными монолитными ядрами и ядрами первых операционных систем, так как современные монолиты имеют чётко выраженную внутреннюю структуру в виде модулей (возможно, динамически загружаемых/выгружаемых). Две другие архитектуры можно представить как современные монолиты, у которых большая часть функционала вынесена во множество приложений пользовательского уровня исполнения (микроядра) или должна быть реализована в так называемых библиотеках ОС (экзоядра). Экзоядра продвинулись дальше в вынесении своих частей в отдельные модули чем микроядра и их можно даже сравнивать с системами виртуализации, в которых экзоядро выполняет роль гипервизора, а процессы – роли гостевых систем [5]. Также были реализованы и различные комбинации названных подходов.

Каждый из типов архитектур можно кратко охарактеризовать с точки зрения их производительности и применимости. Так монолитные ядра обеспечивают наибольшую производительность, жертвуя при этом надёжностью. Высокая производительность объясняется функционированием всех основных и даже большей части вспомогательных (драйверы уровня ядра) систем в одном адресном пространстве, которое при этом является частью адресного пространства любого процесса, работающего в системе. Микроядра имеют несколько меньшее быстродействие (лучшие экземпляры на 8-10% медленнее от производительности монолитов), но обладают лучшей отказоустойчивостью, так как могут продолжать функционировать (для восстановления либо аварийного завершения) даже при отказе критических для работы всей системы частей. Экзоядра призваны обеспечить наилучшую производительность для ряда задач, под которые они разрабатываются, но в отличии от первых двух типов являются менее применимыми для персональных ЭВМ и больше предназначены для решения узкоспециализированных задач. Это в значительной степени объясняется трудоёмкостью разработки для таких систем, т. к. разработка эффективного ПО под экзоядерные системы может потребовать специальных знаний из области системного программирования, которыми далеко не всегда обладают прикладные программисты.

2 Свойства систем на основе микроядра

В связи с примерным выполнением закона Мура (эмпирическое наблюдение, утверждающее, что удваивание числа транзисторов на кристалле происходит примерно каждые 24 месяца [6]) и, следовательно, значительным повышением производительности компьютерных систем, появилась возможность использования систем на основе микроядра с менее заметным проявлением их главного недостатка – низкой производительности. А раз вопрос с производительности более не стоит так критично как раньше, на первый план выходят повышенная отказоустойчивость и модульность как неотъемлемые свойства микроядерных операционных систем.

Повышенная отказоустойчивость обеспечивается прежде всего особенностями самой архитектуры на основе микроядра. Как и во многих структурах ОС предполагается наличие только двух уровней исполнения: *суперпользовательского* (или уровня ядра) *и пользовательского*. На уровне суперпользователя, которому, по определению, разрешено всё, исполняется только минимально-необходимая часть, которая либо не может функционировать на пользовательском уровне, либо её наличие в микроядре обусловлено соображениями улучшения каких-либо свойств системы. Само ядро в основном представляет собой средство для взаимодействия остальных *составляющих* операционной системы, которые представлены процессами, функционирующими в пользовательском пространстве. Таким образом, непосредственно в микроядро попадает лишь небольшая часть всей системы ограниченного объёма: всего лишь несколько тысяч строк кода. Подобные небольшие объёмы и функциональная ограниченность позволяют по прошествии некоторого времени прекратить дальнейшее развитие ядра и практически быть уверенными в отсутствии в нём ошибок.

Любую операционную систему на основе микроядра можно представить себе как трёх-уровневую программную структуру, в которой базовым компонентом является микроядро, промежуточным – набор модулей (серверов, в терминологии микроядерных систем, позаимствованной из клиент-серверных систем), а низший уровень представлен множеством прикладных процессов (клиентов). Таким образом реальное развитие системы происходит путём написания и модификации некоторого набора основных модулей, выполняющих функции контроля доступа к ресурсам системы, опираясь на интерфейс, предоставляемый микроядром. То, что эти модули работаю на уровне пользовательских процессов, позволяет утверждать, что неисправности в работе отдельно взятых модулей минимальным образом отразятся на функционировании остальных частей системы. Хотя дальнейшие обращения к модулям, вышедшим из строя, будут завершаться ошибками, система должна остаться в стабильном состоянии, позволяющем, по крайней мере, выполнить аварийное завершение работы, а в лучшей случае – перезапустить сбойный модуль.

Так как ядро в рассматриваемой архитектуре связано с основными модулями лишь соблюдением интерфейса взаимодействия, который является крайне ограниченным, то с использованием одного микроядра возможно создать несколько операционных систем, отличающихся наборами модулей. Возможен и обратный процесс разработки нескольких взаимозаменяемых микроядер на основе спецификации. Так например на основе спецификации микроядра L4 [5] было разработано несколько реализаций, на основе которых была построена не одна ОС [6].

В истинно микроядерной системе, каждый драйвер должен работать в отдельном адресном пространстве на пользовательском уровне исполнения, будучи полностью изолированным от критически важных модулей системы и её ядра. Это решает проблему использования ненадёжных драйверов, характерную для операционных систем с монолитным ядром. Ведь загрузка и исполнение кода на уровне ядра несёт потенциальную опасность (ошибка в любом драйвере может привести к краху всей системы). Тем не менее описанная проблема частично решается сертификацией драйверов, что само по себе доставляет дополнительные неудобства и их производителям и конечным пользователям.

Из всего вышесказанного складывается впечатление, что микроядро является очень привлекательной архитектурой в современных условиях и не до конца ясны причины их малой популярности. На самом деле развитие операционных систем на основе микроядер, помимо низкой производительности, дополнительно сдерживается сложностью разработки и процесса отладки. Ведь такие системы вполне можно рассматривать как множество асинхронно взаимодействующих процессов, между которыми могут возникать хорошо известные проблемы программирования параллельных систем. Но по сравнению с обычными параллельными системами системное ПО, разработанное с использованием подобного подхода, отлаживать на порядок сложнее, так как изначально отсутствуют необходимые для этого программные средства. Следует заметить, что последняя проблема частично решается системами эмуляции/виртуализации, поддерживающими стандартные отладочные средства гипервизора.

3 Краткое сравнение основных типов архитектур операционных систем

Сравнение архитектур ядер операционных систем, приведённое в таблице 1, выполняется по основным параметрам, имеющим наибольшее значение при разработке или выборе ОС.

Выводы

За десятилетия развития компьютерных технологий как в аппаратной так и в программной областях было предложено несколько основных видов архитектур ОС. Общей тенденцией развития таких архитектур является разбиение ядра на всё более автономные части, делая ядра более гибкими, их поддержку и развитие более простыми (если говорить об отдельно взятых частях), а систему более отказоустойчивой.

Таблица 1. Сравнение архитектур

Параметр	Монолиты	Микроядра	Экзоядра
Примерный объём (в строках кода)	Больше миллиона	Около 5 тысяч	Около тысячи
Примерное количество системных вызовов ядра	4-5 сотен	1-3 десятка	Около 10
Производительность	Высокая	Средняя	Максимальная
Распространённость	Наибольшая	Низкая	Общая – низкая, спец. – быстрее монолитов
Примеры	GNU/Linux®, Windows®	GNU Hurd, L4, Minix	Exopc, Nemesis, XOmB

Литература

[1] Таненбаум Э., Вудхалл А., Операционные системы: разработка и реализация – СПб.: Питер, 2006. – 576 с.

[2] Таненбаум Э., Современные операционные системы. 3-у изд. – СПб.: Питер, 2011. – 1120 с.

[3] The Structure of the T. H. E. Multiprogramming System – Communications of the ACM, Vol. 11, No. 5, May 1968, pp. 341-346

[4] Торвальдс Л., Даймонд Д., Ради удовольствия: Рассказ нечаянного революционера – М.: Эксмо Пресс, 2002. – 288 с.

[5] Официальный сайт микроядра L4. Электронный ресурс. Режим доступа: http://l4hq.org

[6] Версия микроядра L4 Дрезденского университета (лицензия GNU). http://os.inf.tu-dresden.de/L4