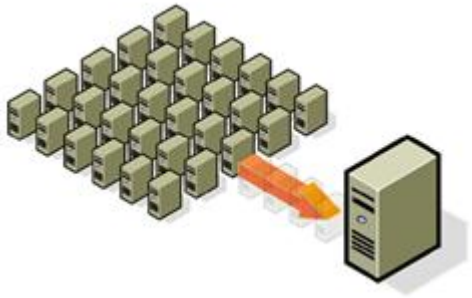
Технологии виртуализации

Иртегов Дмитрий Валентинович

Доцент факультета информационных технологий Новосибирского гос. Университета

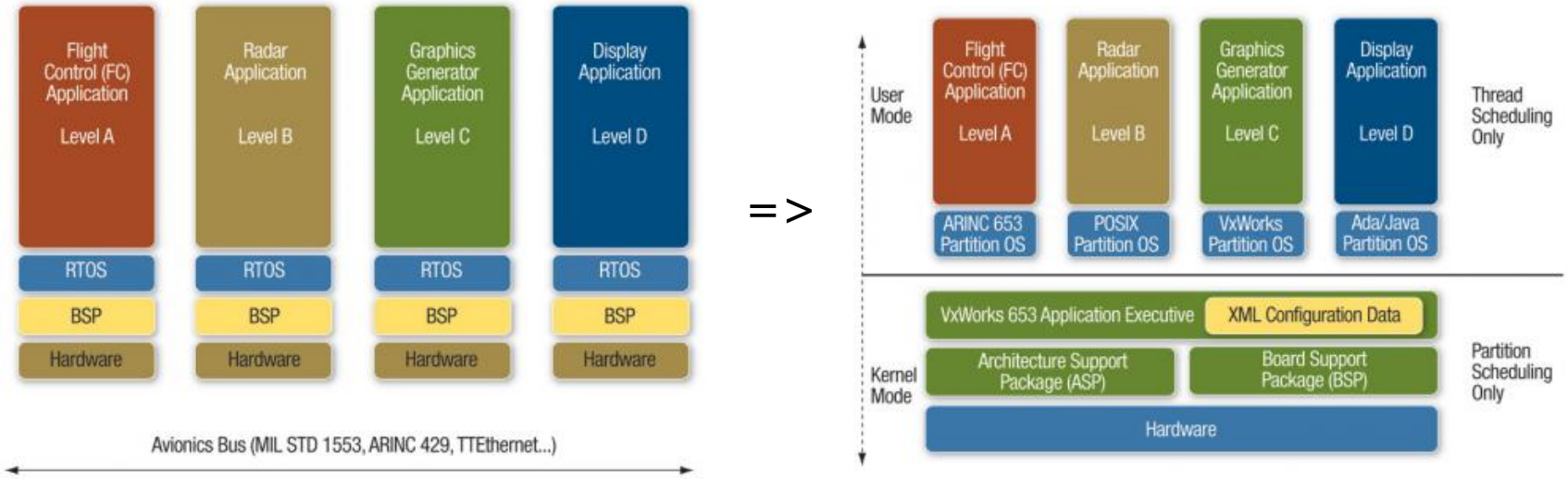
Для чего нужна виртуализация?

- Консолидация серверов
  - Облачные инфраструктуры
  - Хостинг
  - …
- Персональные компьютеры
  - Run Windows on your Mac
  - Run Mac on your Linux
  - Run Ubuntu on your Android
  - …



Для чего нужна виртуализация?

- Встраиваемые приложения
  - Заменить 10 бортовых компьютеров на 1



Для чего нужна виртуализация?

Для чего нужна виртуализация?

Для чего нужна виртуализация?

+

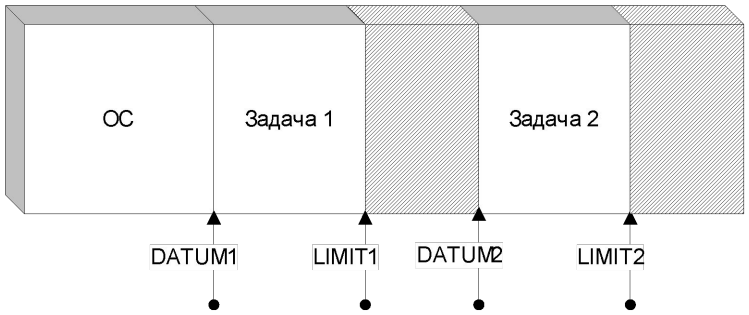
Сначала маленький экскурс в историю

- Когда компьютеры были большими…

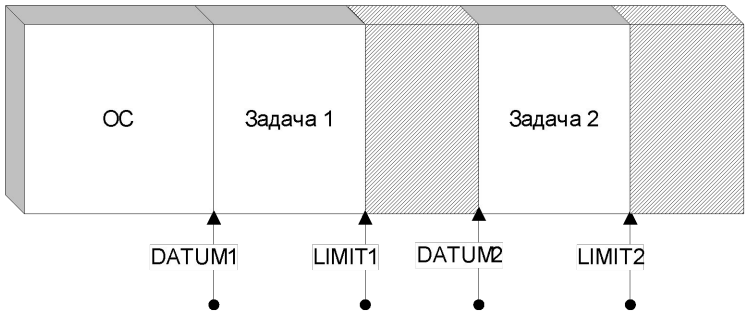
Системы разделения времени

- Один компьютер, много пользователей
- Что будет, если кто-то из пользователей запустит плохую программу?

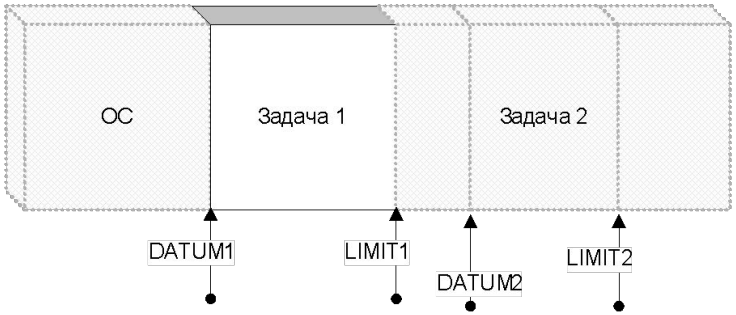
Защита памяти – базовая адресация (DEC PDP-6)



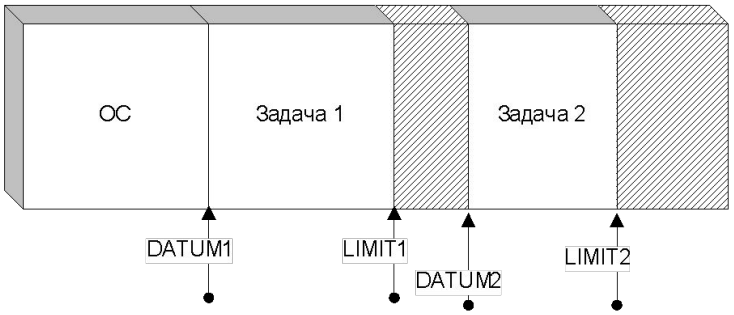
Защита памяти – базовая адресация (DEC PDP-6)



Пользовательский режим

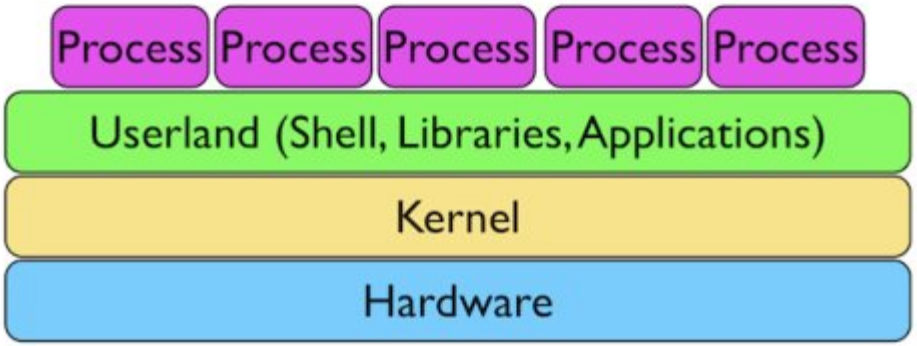


Системный режим

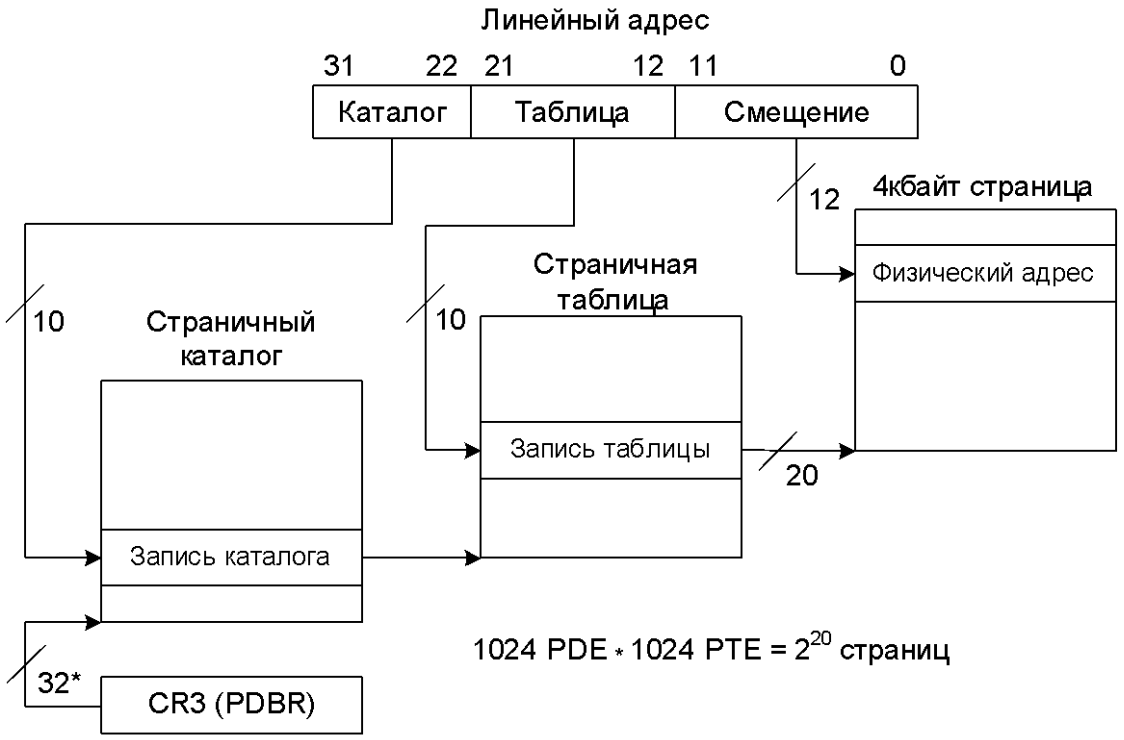


Терминология

- Ядро (kernel) – код, исполняющийся в системном режиме
- Пользовательское пространство (userland) – код, исполняющийся в пользовательском режиме
- Процесс (задача, task) – контейнер, создаваемый для исполнения пользовательского кода. Каждый процесс имеет свое адресное пространство.



Защита памяти – современные процессоры (x86)

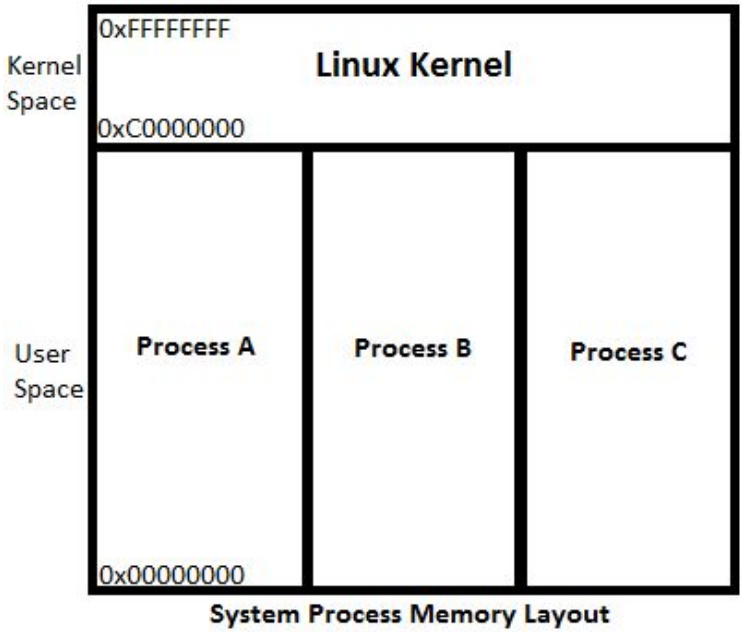


*32 бита выровненные на 4кбайта

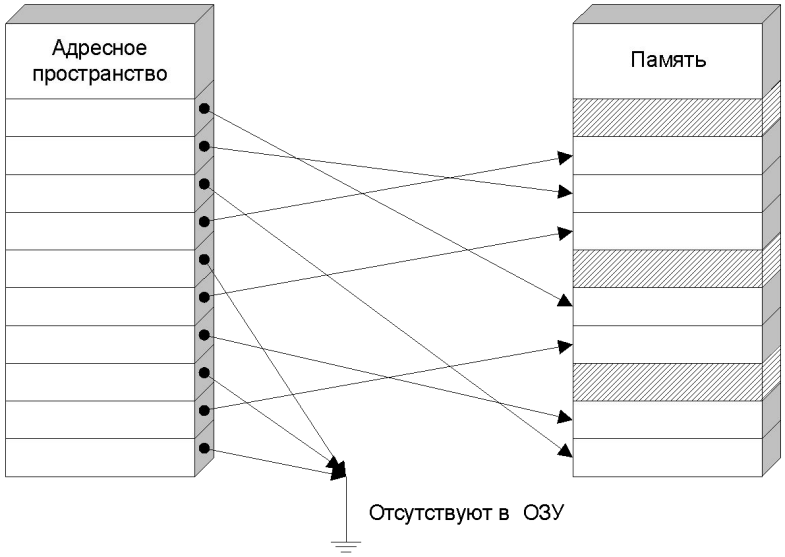
Защита памяти – современные процессоры (x86)

Структура дескриптора страницы (без PAE)

Страничная трансляция



System Process Memory Layout



Преимущества систем с защитой памяти

- Плохо ведущая себя программа не может уронить систему
- Возможно разделение привилегий (учетные записи пользователей, права доступа, полномочия)
- Защита от вирусов и троянских программ
- Все современные ОС используют защиту памяти
  - Windows
    - 95
    - NT (XP/Vista/7/8)
  - Unix
    - System V
    - Linux, Android
    - *BSD
    - Mac OS X
    - iOS

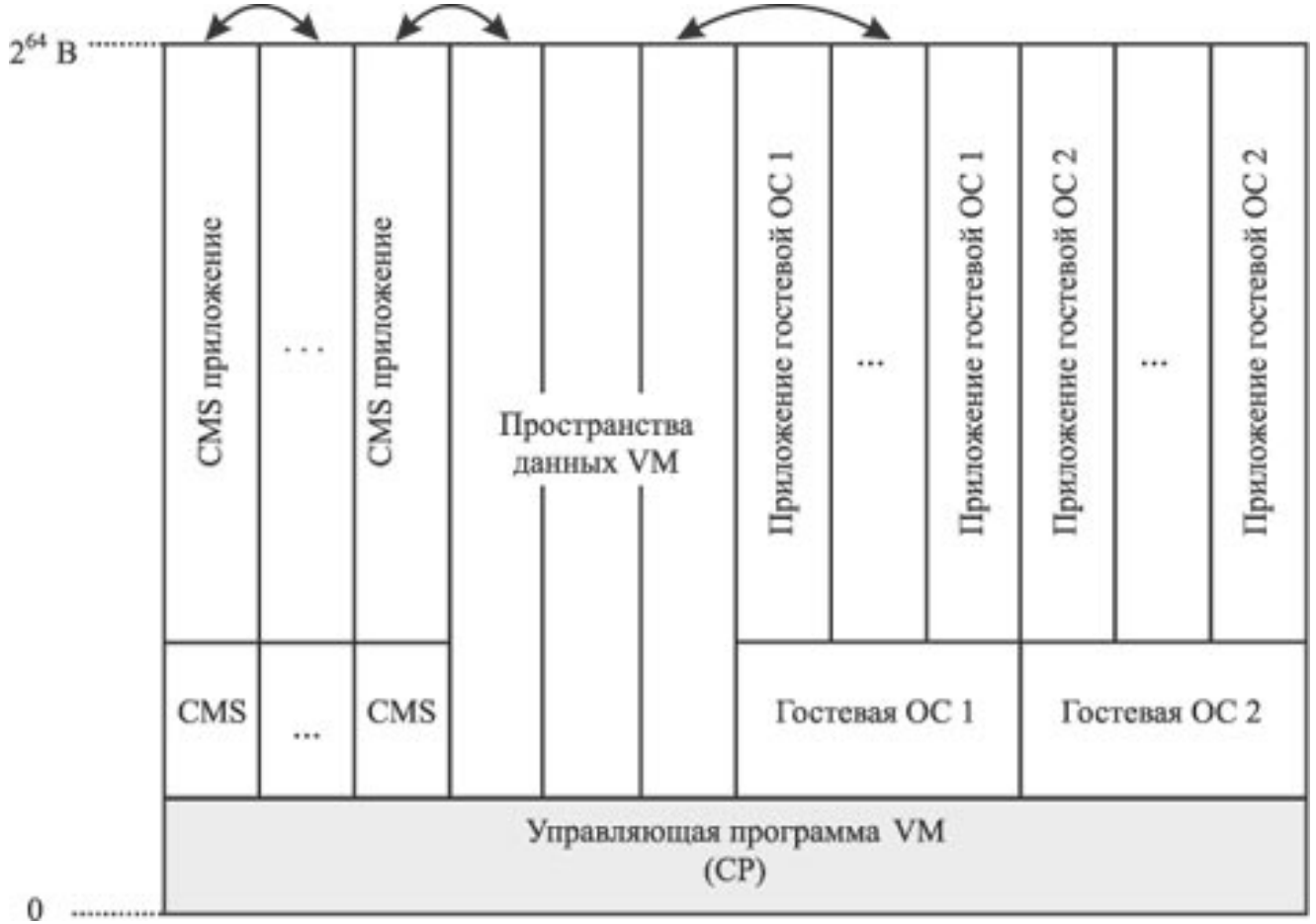
Что делать со старыми программами?

- IBM System/360 – 1965-1978
  - Защиты памяти не было
  - Различные ОС: BOS/360, DOS/360, OS/360, TSO, TSS..
- IBM System/370 Advanced Function
  - Виртуальная память
  - ОС с защитой памяти (IBM MVS)
- Как обеспечить совместимость?

IBM CP-40

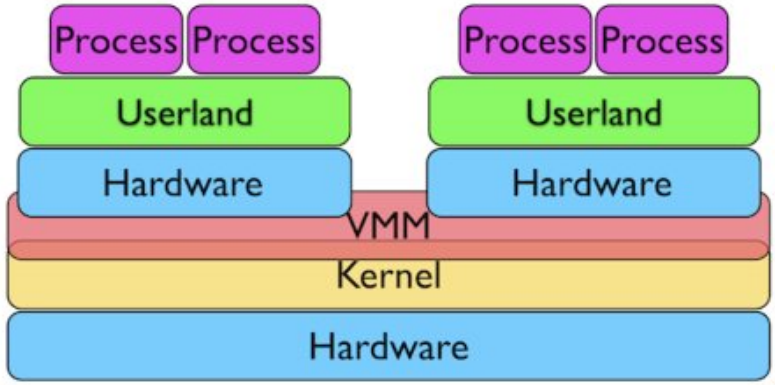
- IBM Cambridge Research Center
- Оборудование:
  IBM S360-40+”CAT Box”
- Цель: полная виртуализация
  - Среда, в которой можно установить и запустить любую существующую ОС для IBM S360
  - Виртуализация периферийных устройств
    - Исключительный (устройство закреплено за ВМ)
    - Разделяемый (устройство разделяется несколькими ВМ)
    - Накопительный (запросы к устройству, например, задания на печать, буферизуются на жестком диске)
- Результаты проекта были использованы в IBM S360-67 и System/370
- CP-40->CP/CMS->VM/370->…>VM/ESA->z/VM

Организация IBM zVM



Терминология

- Гипервизор (управляющая программа VM)
- ОС-хозяин (host)
- Гипервизоры Type I
  - Работает на голом железе
- Type II
  - Гипервизор представляет собой комплекс модулей ядра и пользовательских программ, работающих под управлением ОС-хозяина
- Гостевая ОС



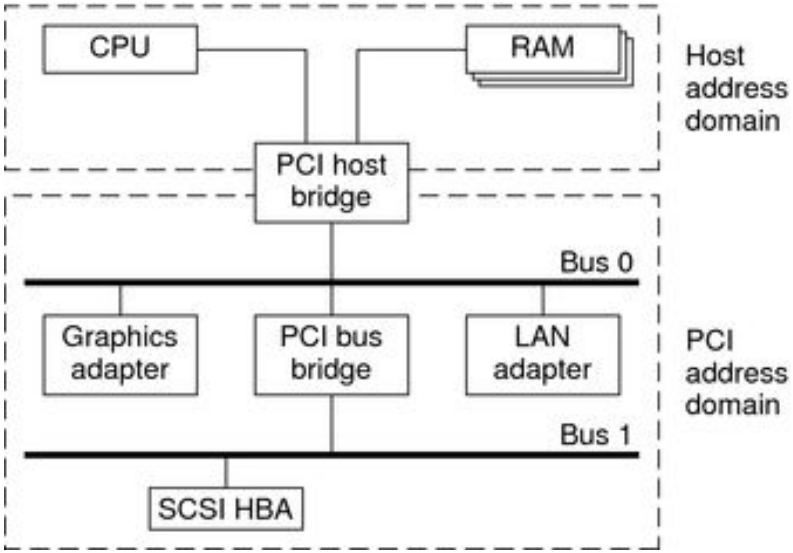
Требования Попека-Голдберга

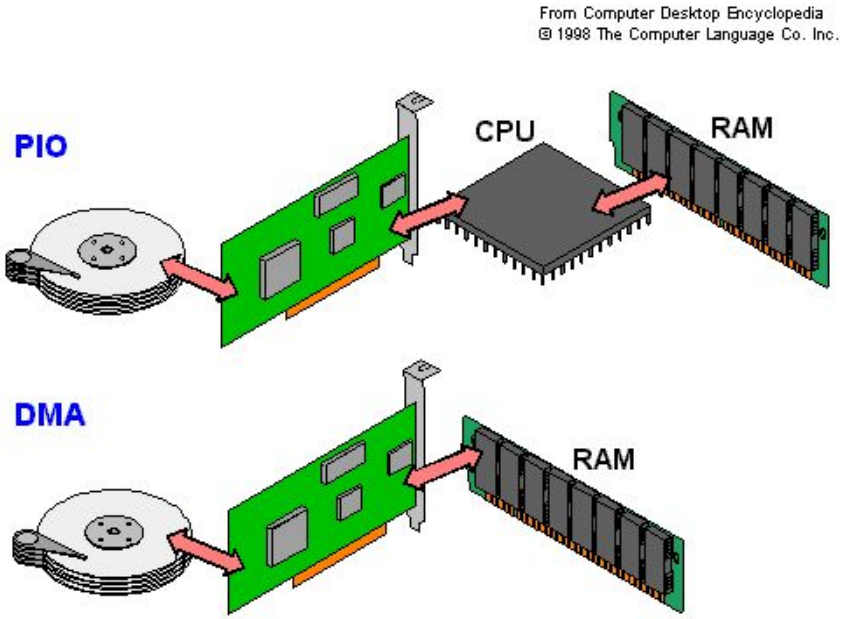
- G. Popek, R. Goldberg. 1974. Formal requirements for virtualizable third generation architectures. *Commun. ACM* 17, 7 (July 1974),
- Классы команд
  - Привилегированные (генерируют исключение при вызове из пользовательского режима)
  - Чувствительные (изменяют состояние виртуальной памяти или зависят от него)
- Теорема Попека-Голдберга
  - Если все чувствительные команды являются привилегированными, то для такой машины возможно построение гипервизора
  - Trap-and-emulate virtualization

Соответствие критерию Попека/Голдберга

- IBM S/370 – S/390 zSeries
  - Сразу разрабатывались в расчете на виртуализацию
  - Соответствуют критерию Попека/Голдберга
- X86 (Intel 80386, 486, Intel Core, AMD …)
  - Чувствительные непривилегированные команды:
    - LAR, LSL, VERR, VERW
    - POP
    - PUSH
    - CALL, JMP, INT n, RET
    - STR
    - MOV

Внешние устройства и прямой доступ к памяти





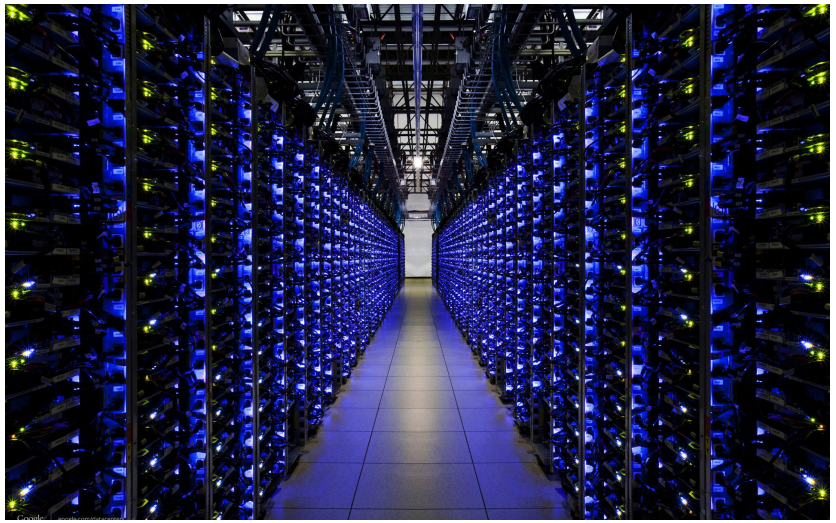
Прямой доступ к памяти

- IBM System/370
  - Ввод-вывод проходит через канальные процессоры
  - Канальный процессор может использовать те же таблицы трансляции, что и УУП центрального процессора
- PC-совместимый компьютер
  - Прямой доступ к памяти использует физические адреса
  - IOMMU появилось только в чипсетах, поддерживающих Intel VT-d и AMD Vi

Как так получилось?

=>

?



Как так получилось?

=>

Подходы к виртуализации

- Классическая (trap and emulate)
- Покомандная интерпретация
- Бинарная компиляция
- Паравиртуализация
- *Контейнерная виртуализация (VPS, VE)*

Покомандная интерпретация

- DOSEmu
- Эмуляторы Dandy/Atari/etc
- Режим интерпретатора Java Virtual Machine
- Другие интерпретаторы, использующие байт-код (UCSD Pascal, emacs lisp, basic, да тысячи их)
- Очень медленно, зато
  - Позволяет воссоздать виртуальную среду во всех деталях
  - Позволяет имитировать любое окружение на любом, например, запускать приложения для Z80 на Android

Паравиртуализация

- Использование специальной сборки ядра ОС, которое знает, что работает под гипервизором
  - Usermode Minix/Linux/Unix System V
  - Xen
- Возможна поддержка только ОС с открытыми исходными текстами
- На самом деле, многие современные VM используют элементы паравиртуализации

Бинарная компиляция

- Берется машинный код и генерируется эквивалентный машинный код для целевой архитектуры
- Вариант: jit-компиляция
  - Код некоторое время исполняется в режиме покомандной эмуляции, набирается статистика, потом генерируется оптимизированный код

История бинарной компиляции

- Первое успешное применение: IBM AS/400
  - TIMI (Technology Independent Machine Interface)
- JIT-компиляция Java
- Попытки:
  - Бинарные трансляторы DEC VAX->DEC Alpha, обещали транслятор x86->Alpha
  - Transmeta

История бинарной компиляции

- В 90е годы сложилось впечатление, что бинарная компиляция годится для прикладных программ, но не для ядра ОС
- Ядро ОС должно обрабатывать прерывания и сохранять контекст (регистры процессора)
- При бинарной компиляции, аппаратное прерывание может прийти во время работы команды, когда контекста не существует

VMWare Workstation (1999)

- Гипервизор Type II (ставится как пакет ОС-хозяина)
- Пользовательские программы гостевой ОС исполняются в нативном режиме
- Ядро ОС исполняется бинарной компиляцией
- Теневые таблицы трансляции
- Поддержка ограниченного набора ОС
- Эмулируется фиксированный набор оборудования

VMWare Workstation

- PCI Bus Полностью эмулируемый контроллер шины PCI
- 4x 4IDE Виртуальные диски, размещаемые в файлах ОС-хозяина, или эмулируемый доступ к заданному устройству
- 7x Buslogic SCSI Disks
- 1x IDE CD-ROM Образ ISO или эмулируемый доступ к реальному CD-ROM
- 2x 1.44MB floppy drives Физическая дискета или образ в виде файла
- 1x VMware graphics card Совместимая с VGA и SVGA. Поддержка SVGA требовала драйвер для гостевой ОС
- 2x serial ports COM1/COM2 Присоединялись к портам ОС-хозяина или файлам
- 1x printer (LPT) Мог присоединяться к порту ОС-хозяина
- 1x keyboard (104-key) Полностью эмулируемая
- 1x PS-2 mouse Полностью эмулируемая
- 3x AMD PCnet Ethernet cards (Lance Am79C970A) Режим моста или host-only
- 1x Soundblaster 16b Полностью эмулируемый

Другие попытки

- Connectix Virtual PC
  - Virtual PC for Mac (1997), динамическая байтовая компиляция пользовательского и системного кода
  - Virtual PC for Windows – компиляция только системного кода, ввод-вывод эмулируемый или паравиртуализационный через специальные драйверы гостевой ОС
  - В 2003 году продукт был куплен Microsoft
  - Microsoft Virtual PC, Virtual Server, Windows XP Mode, Hyper-V
- Innotek VirtualBox
  - В 2008 году приобретен Sun Microsystems, теперь известен как Oracle VirtualBox
  - Поддержка DirectX/OpenGL (паравиртуализация)
- Parallels Desktop/Server for Mac
  - Поддерживает x86 Mac
  - 2007 год – поддержка DirectX/OpenGL (паравиртуализация)
- QEMU
  - Бинарный компилятор с открытыми исходными текстами
  - Поддерживает очень много гостевых аппаратных архитектур: x86 PC, PowerPC, PowerMac, ARM (в т.ч. Android), SPARC, Z80 Spectrum, HP PA-RISC

Типичный набор возможностей современной VM

- Поддержка Intel VM-x/AMD Vi
- Паравиртуализационные драйверы
  - Жесткий диск
  - Сетевой контроллер
  - *Видеоадаптер*
  - *Интеграция рабочего стола (изменение разрешения видеоадаптера синхронно с изменением размеров окна, захват-освобождение мыши)*
- Расширения гостевой ОС
  - *Доступ к буферу обмена и файловой системе ОС-хозяина*
- Доступ к устройствам USB
- Режимы работы сетевого контроллера:
  - NAT
  - Bridge
  - TCP offloading

Продвинутые возможности

- Живая миграция (перенос виртуальных машин с одного хоста на другой)
- Балансировка загрузки
- Клонирование дисков и разделяемая память
- Запуск гостевых ОС по запросу
  - Облачные инфраструктуры
  - Виртуализация рабочего стола (VDI)

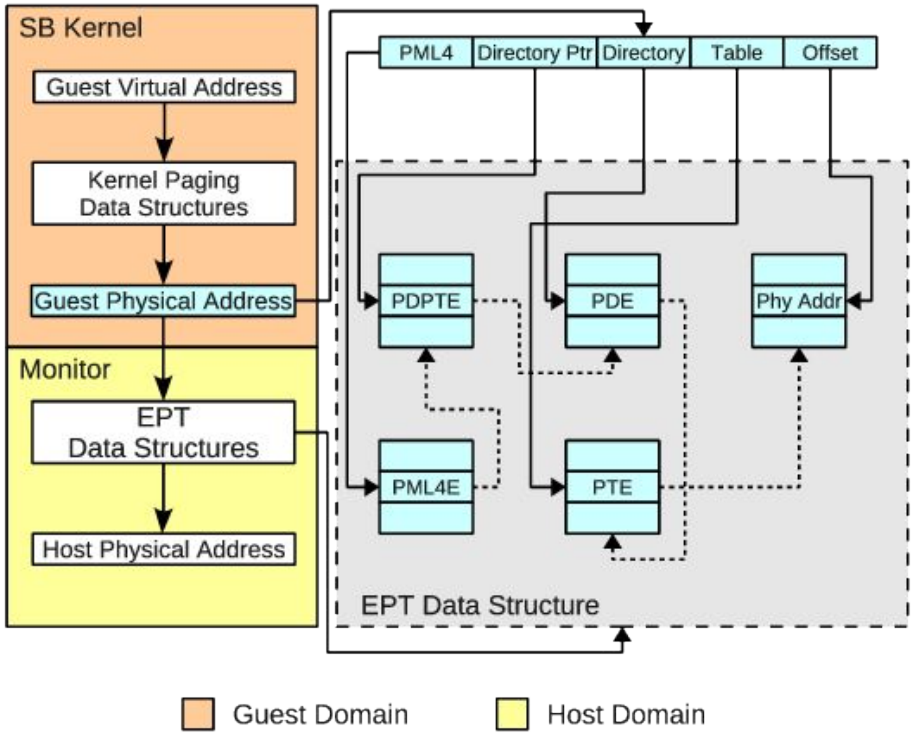
Общие проблемы

- Производительность ввода-вывода
  - В большой степени компенсируется паравиртуализационными драйверами
- Производительность операций с виртуальной памятью
  - Создание/завершение процессов гостевой ОС, увеличение и уменьшение объема памяти отдельного процесса
  - Перехват этих операций гипервизором
  - Поддержка теневых структур данных
  - Частично компенсируется аппаратной поддержкой
- Промывание кэша, конкуренция за системную шину
- Таймеры TCP/IP
  - Может быть компенсировано выгрузкой TCP

Аппаратная виртуализация x86

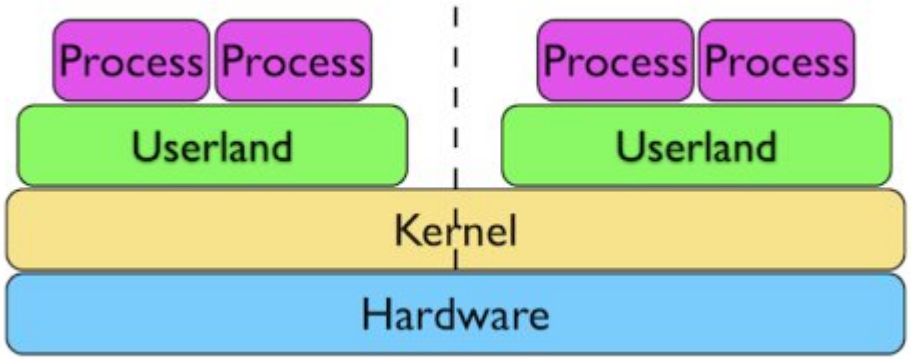
- Intel VM-x
  - VM-d (чипсет) – IOMMU
  - VMX – введение уровня привилегий гипервизора (Ring -1) и сохранения состояния гостевой ОС
    начиная с некоторых моделей P IV
  - Extended Paging Tables (EPT) – страничная трансляция второго уровня
    начиная с Nehalem
- AMD-V

Extended Paging Tables



Контейнерная виртуализация

- Небольшая переделка ядра
- Все процессы получают новый атрибут – Id зоны
- Процессы разных зон не видят друг друга



Контейнерная виртуализация

- В каждом контейнере свои:
  - Идентификаторы процессов
  - Идентификаторы пользователей
  - Полномочия супервизора
  - Ветвь файловой системы
  - Виртуальные сетевые адаптеры
  - Реестр (в Windows)

Примеры

- FreeBSD Jail (1999)
- Solaris zones (2004)
- OpenVZ (2005)
- Parallels Virtuozzo Containers for Linux
- Parallels Virtuozzo Containers for Windows
- Lxc (2008)
- Docker (2013)
- Containerd, CRI-O

Преимущества

- Нет виртуализации внешних устройств
- Нет виртуализации MMU
- Нет проблем с таймерами TCP
- Разделяемая память
- Возможность разделять дисковое пространство (Solaris+zfs, Parallels Virtuozzo, docker)

Недостатки

- Все контейнеры должны использовать одну версию ОС
  - В Linux это не проблема (Linux kernel ABI очень стабилен, можно запустить Ubuntu на ядре от CentOS)
  - Частичное исключение – Solaris BrandZ
- Сложно подсчитывать квоты оперативной памяти
  - Провоцирует oversell
- Необходим справедливый планировщик
- Неполная защита от эскалации привилегий

Чем отличается Docker?

- Контейнерные среды первого поколения (jails, openvz, lxc) предполагали запуск в контейнере нормальной Unix-среды
- Docker предполагает запуск одного процесса