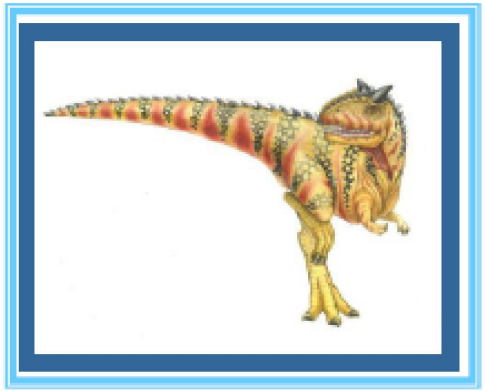
Операционные Системы

Виртуальная память



Цель

- описание перимуществ использования виртуальной памяти
- объяснение принципов загрузки по требованию, алгоритмов замещения страниц и размещения страничных кадров
- рассмотрение понятий локальности и модели рабочего набора

Определения

Виртуальная память основана на разделении логической и физической памяти

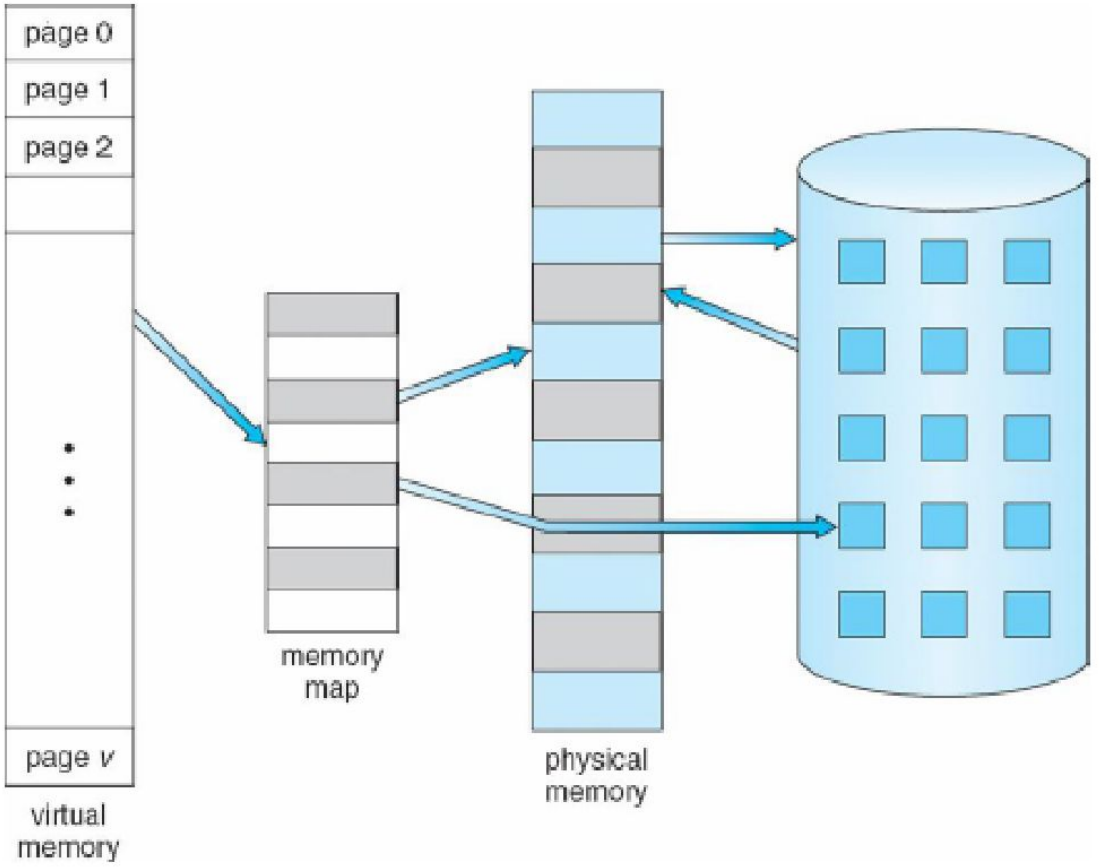
- только часть программы может находиться в оперативной памяти
- поэтому логическое адресное пространство может быть намного больше, чем физическое
- адресное пространство может быть доступно нескольким процессам

Виртуальная память может быть реализована с помощью

- подкачки страниц
- подкачки сегментов по требованию

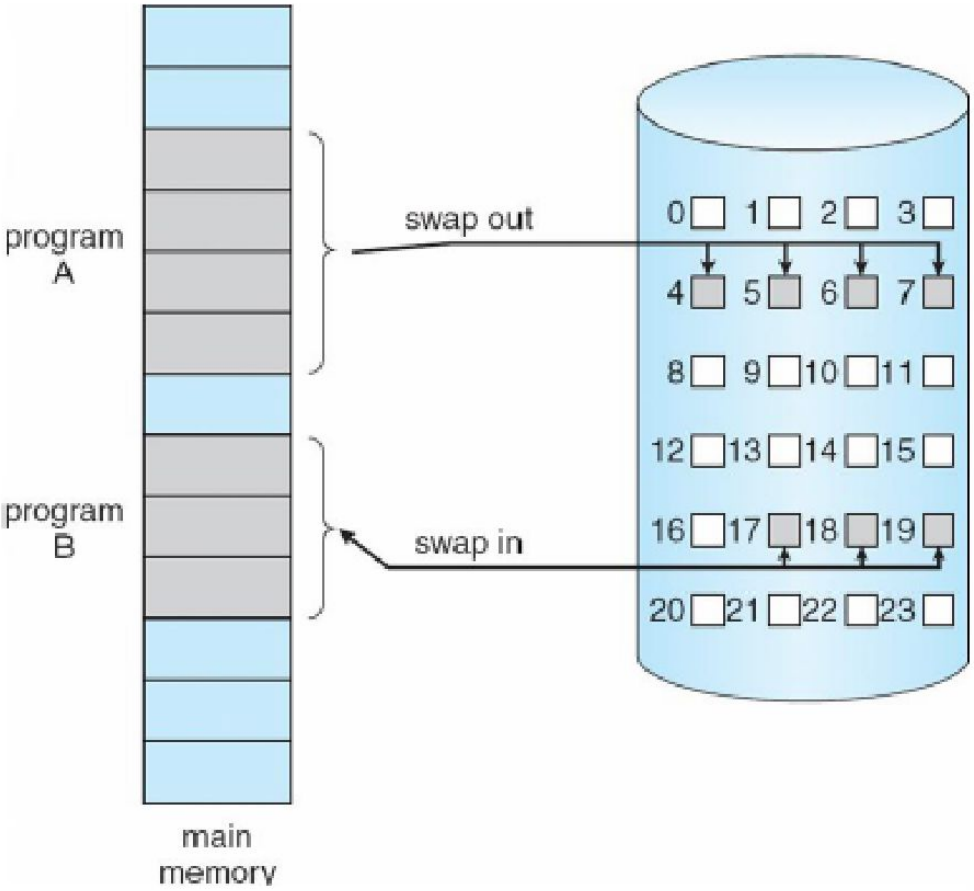
Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Виртуальная память больше физической



Gagne, Operating Systems Concepts, 8th edition, 2009

Сохранение страниц в непрерывном дисковом пространстве

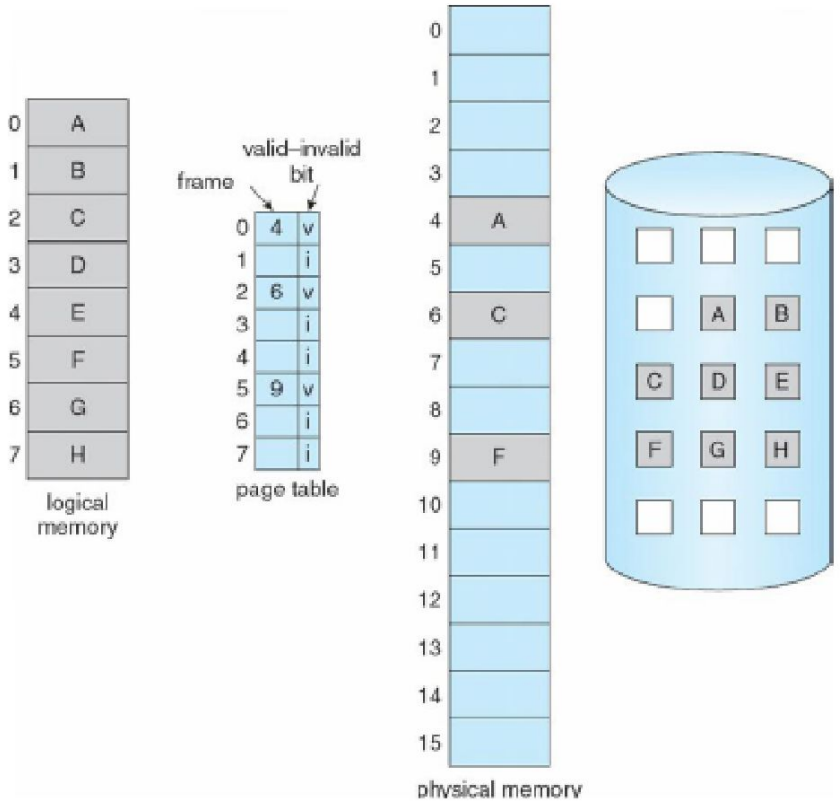


Systems Concepts, 8th edition, 2009

Бит валидности

- Каждая запись в таблице страниц содержит бит валидности, указывающий, находится ли страница в оперативной памяти (v-да, i-нет)
- В самом начале бит валидности установлен в i (invalid) для всех записей
- Во время трансляции адресов, если значение бита валидности для данной страницы i – возникает page fault

Таблица страниц, некоторые страницы не находятся в оперативной памяти



Systems Concepts, 8th edition, 2009

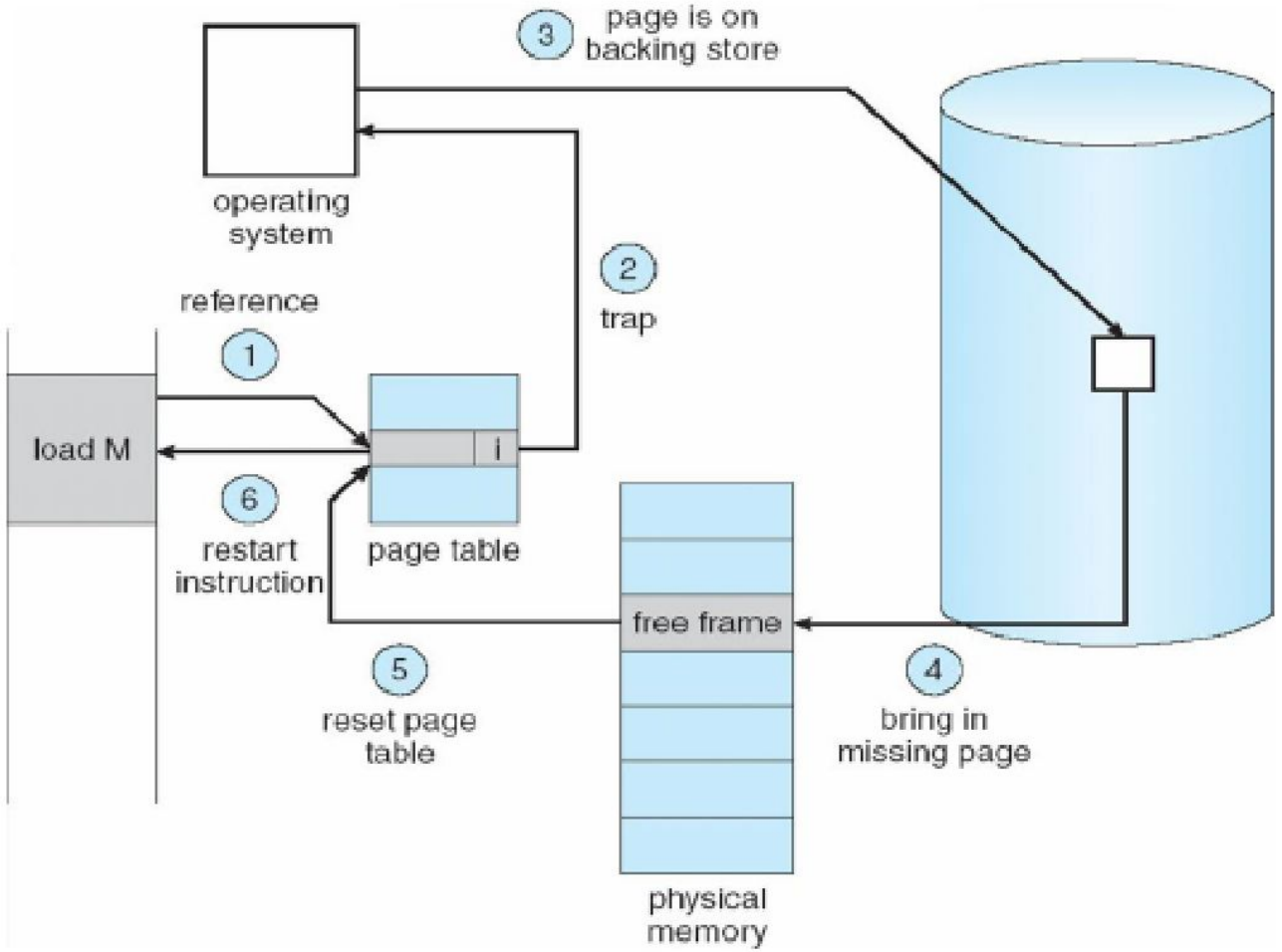
Страничное Прерывание - Page Fault

Первая ссылка на страницу всегда требует вмешательства операционной системы, то есть генерирует страничное прерывание – page fault.

1. Операционная система проверяет ссылку
   - неправильная ссылка на память – процесс завершается
   - нужной страницы просто нет в памяти – страница загружается с диска
2. Отыскивается пустой кадр (фрейм)
3. Страница загружается в найденный фрейм с диска
4. Запись в таблице страниц обновляется (бит валидности устанавливается в v)
5. Инструкция, приведшая к страничному прерыванию, исполняется заново

Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Обработка страничного прерывания



Systems Concepts, 8th edition, 2009

Производительность загрузки по требованию

- Коэффициент страничных прерываний 0<=p<=1
  - p=0 – страничных прерываний нет
  - p=1 – каждая ссылка вызывает page fault
- Эффективное время доступа к адресу (Effective Access Time - EAT)

  EAT=(1-p)*время обращения к памяти + p*(время обслуживания страничного прерывания)

Пример

- Время обращения к памяти – 200 нс
- Среднее время обслуживания страничного прерывания – 8 мс
- EAT=(1-p)*200 + p*8000000 = 200 + p*7999800
- Если страничное прерывание возникает на одну ссылку из 1000 (p=0,001)

  EAT=8.2 мкс – медленнее в 40 раз!

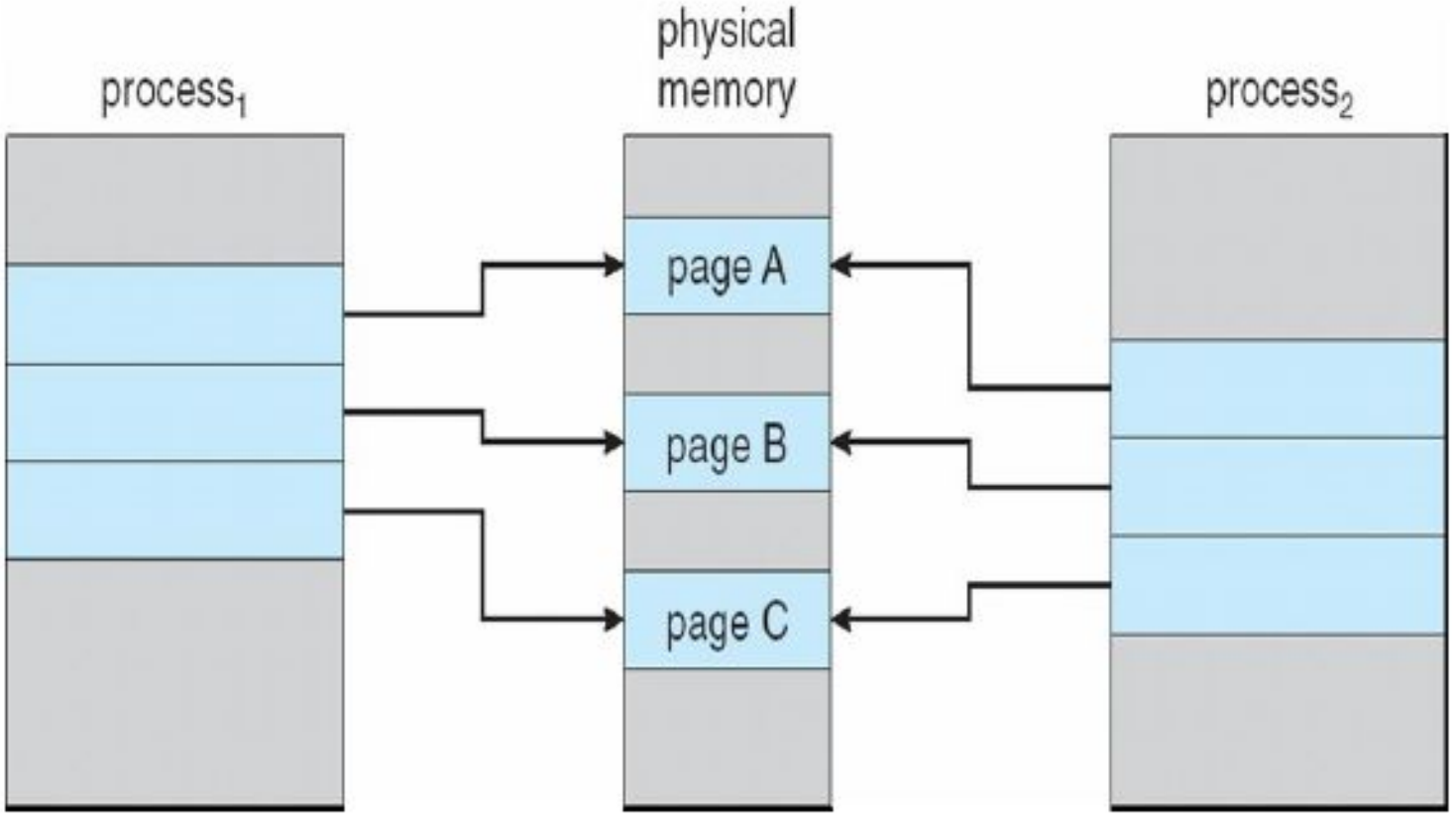
Copy-on-Write (Копирование при записи, отложенное копирование)

- Copy-on-Write(CoW) позволяет процессу-родителю и процессу-потомку сразу после выполнения fork() обращаться к одним и тем же страницам памяти

  Страницы копируются только если один из процессов модифицирует разделяемую страницу
- CoW делает создание нового процесса более эффективным, так как реальное копирование происходит только при модификации (часто модификации не требуется)
- Свободные страницы предоставляются из пула «обнуленных» страниц

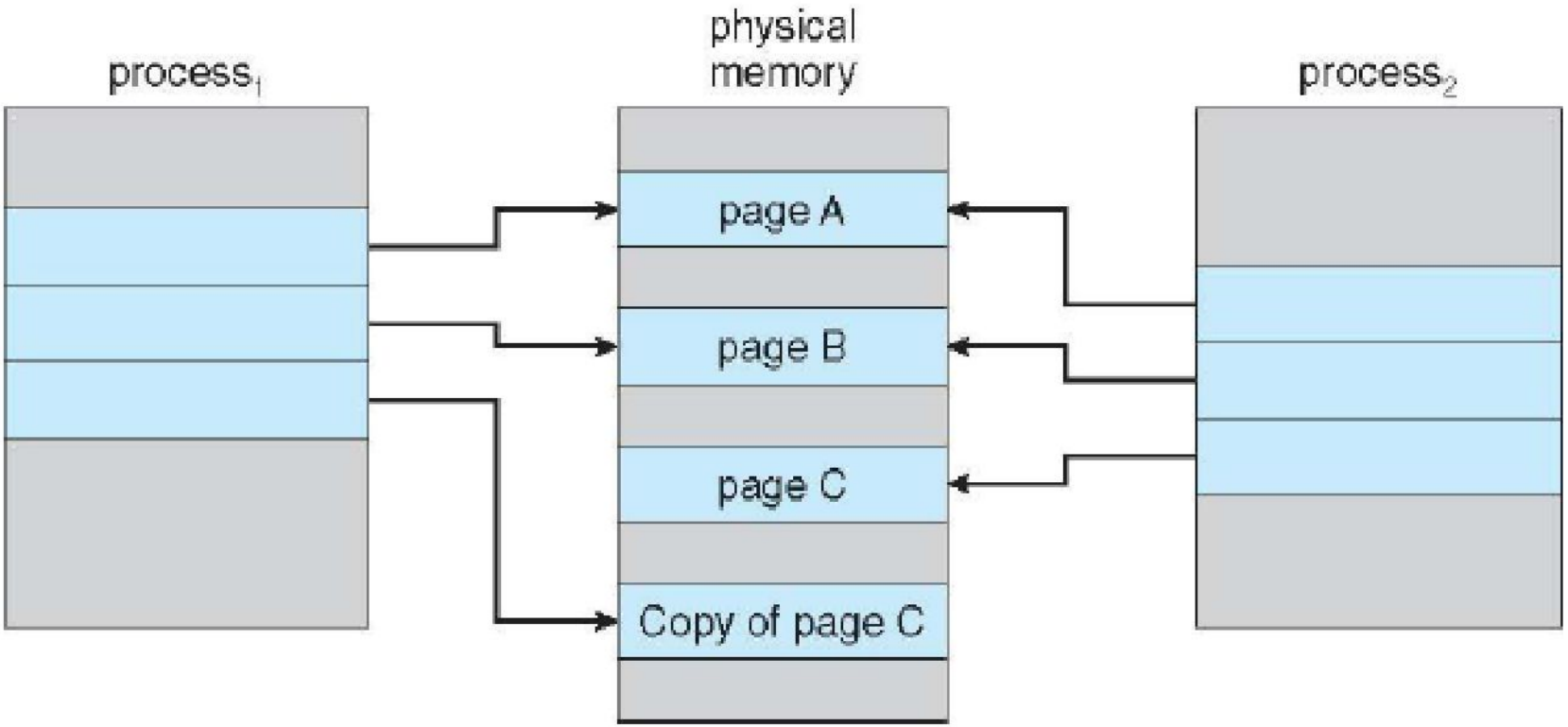
Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Перед модификацией страницы C Процессом 1



Gagne, Operating Systems Concepts, 8th edition, 2009

После модификации



Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Что происходит, если свободных кадров нет?

- Замещение страниц (Page Replacement) – нужно найти в ОП ту страницу, которая не используется, и выгрузить ее
  - алгоритм?
  - производительность – нужен алгоритм, который будет вызывать наименьшее количество страничных прерываний
- Страницы могут выгружать ся на диск и снова загружаться в ОП несколько раз

Замещение страниц

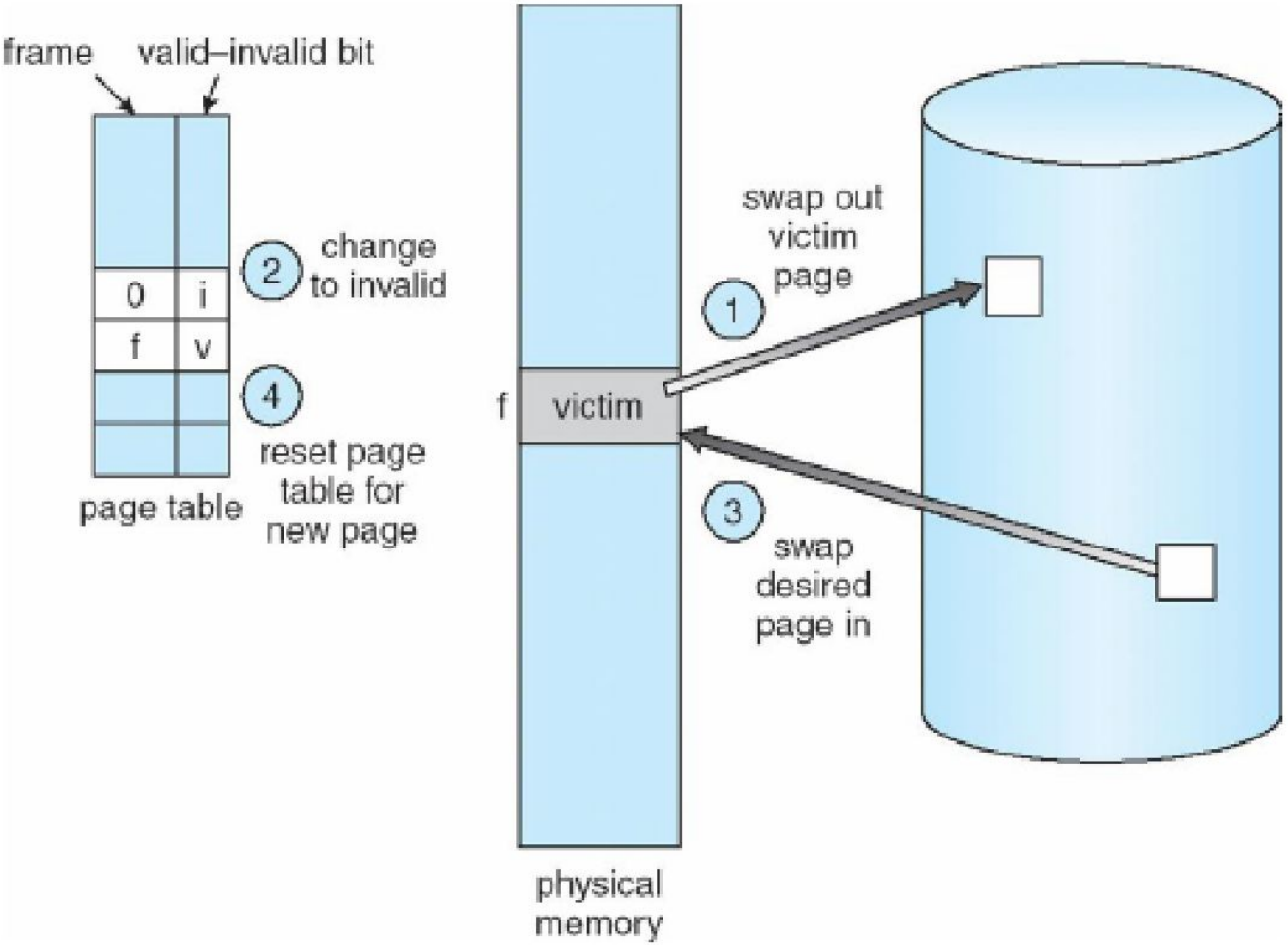
- Предотвращаем перезагрузку памяти – изменяем процедуру обслуживания page fault, так, чтобы она содержала замещение страниц
- Используем бит модификации (“dirty”, “грязный” бит для уменьшения количества перемещения страниц с диска и на диск – на диск записываются только модифицированные («грязные») страницы
- Замещение страниц завершает разделение между логической и физической памятью – большое пространство логической памяти может быть предоставлено на меньшей физической

Простое замещение страниц

1. Найти расположение нужной страницы на диске
2. Найти свободный фрейм (кадр)
   - если такой имеется, используем его
   - если нет, используем алгоритм замещения для поиска страницы-жертвы
3. Загрузить нужную страницу с диска в освобожденный фрейм, обновить таблицу страниц и фреймов (инвертированную ТС)
4. Запустить процесс

Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Замещение страницы



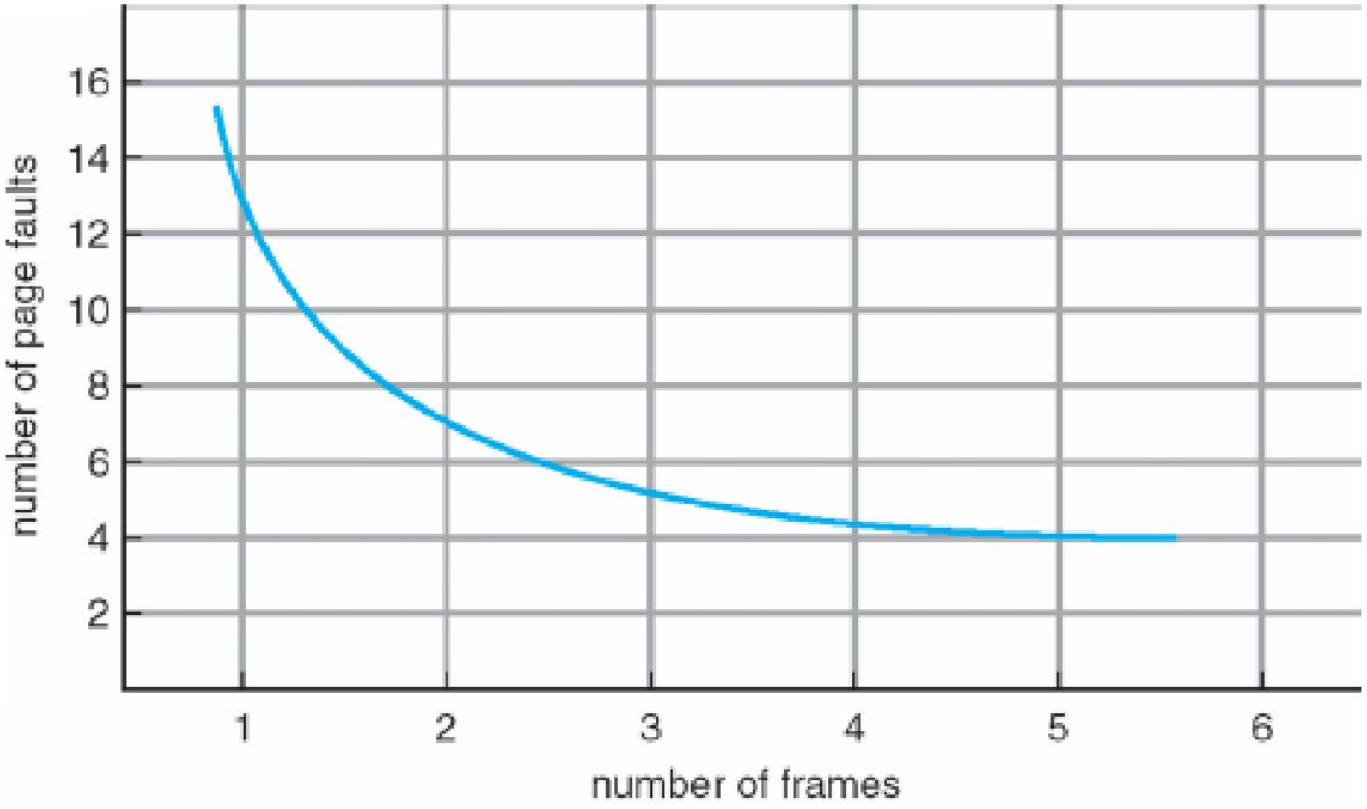
Silberschatz, Galvin, Gagne, Operating System Concepts, 8th edition, 2009

Алгоритмы замещения страниц

- Нам нужен низкий коэффициент страничных прерываний, поэтому
- Мы оцениваем алгоритмы, запуская их на определенной последовательности обращений к памяти (строка обращений, reference string), и вычисляя количество страничных прерываний для этой строки
- Во всех примерах это будет строка обращений

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Соотношение страничных прерываний и количества фреймов



Silberschatz, Operating Systems Concepts, 8th edition, 2009

Алгоритм First-In-First-Out (FIFO)

- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- для процесса в ОП выделяется три фрейма

  1 1 4 5
  2 2 1 3 – 9 страничных прерываний
  3 3 3 4

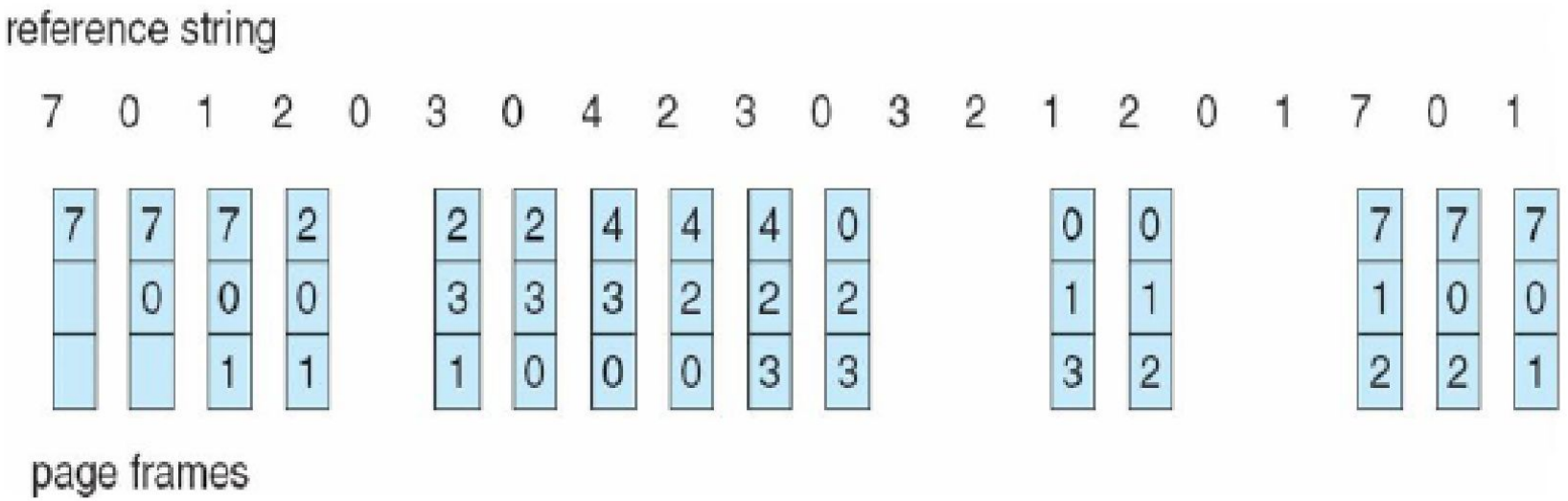
- 4 фрейма

  1 1 5 4
  2 2 1 5 – 10 страничных прерываний
  3 3 2
  4 4 3

- Это аномалия Билэди – фреймов больше, но и page faults тоже больше

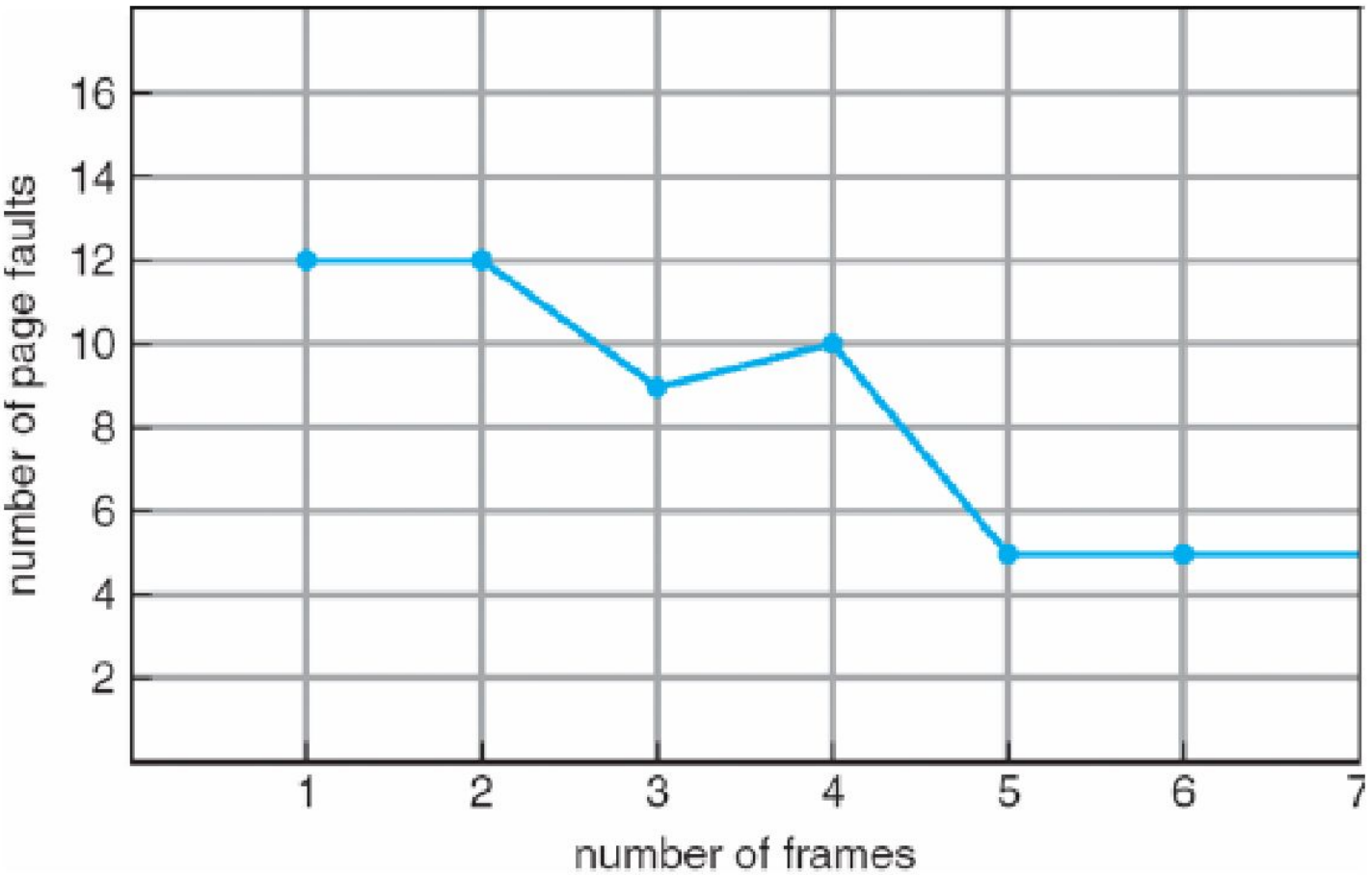
Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

FIFO, пример



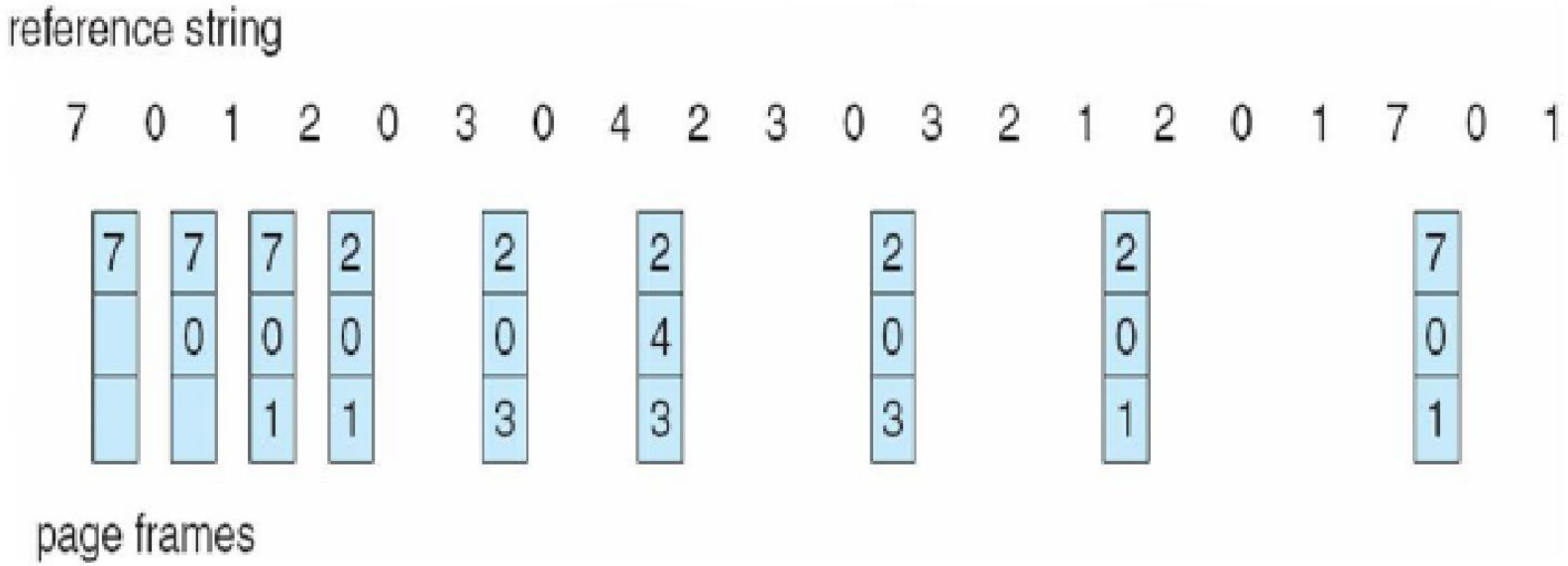
Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Аномалия Билэди



Silberschatz, Operating Systems Concepts, 8th edition, 2009

Оптимальное замещение



Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

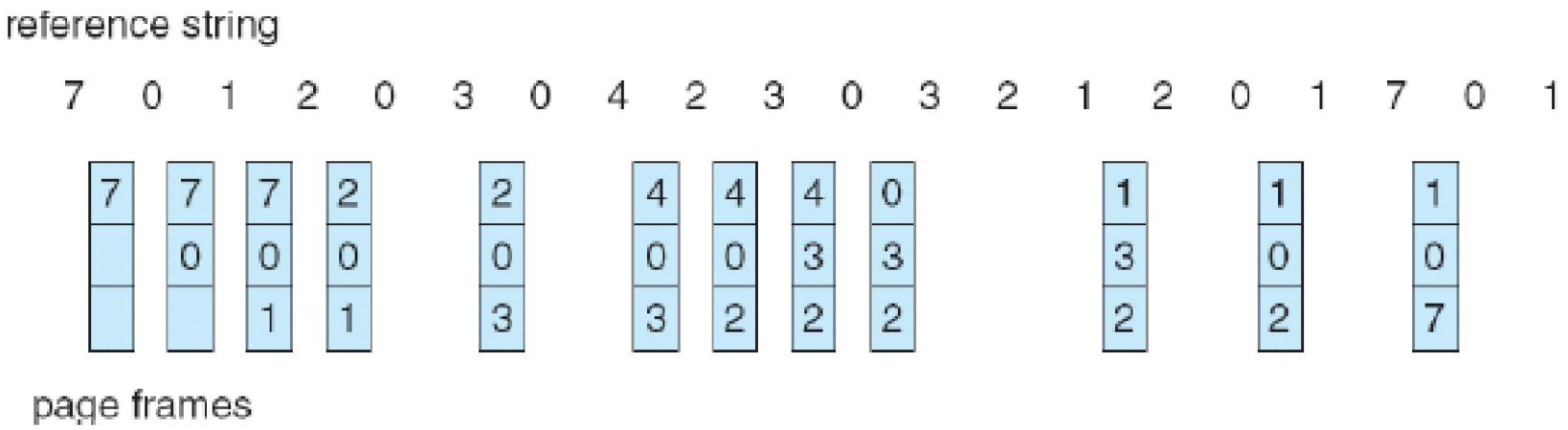
Least Recently Used – LRU, выталкиваются страницы, использовавшиеся наиболее давно

- 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

  1 1 1 1 5
  2 2 2 2 2
  3 5 5 4 4
  4 4 3 3 3

- Реализация счетчика
  - Каждая запись таблицы страниц содержит счетчик; каждый раз при обращениик этой страницы, в счетчике устанавливается текущее время
  - Когда страницу нужно заменить, просматриваются все счетчики и выбирается страница, обращение к которой было раньше всех («давнее»)

LRU

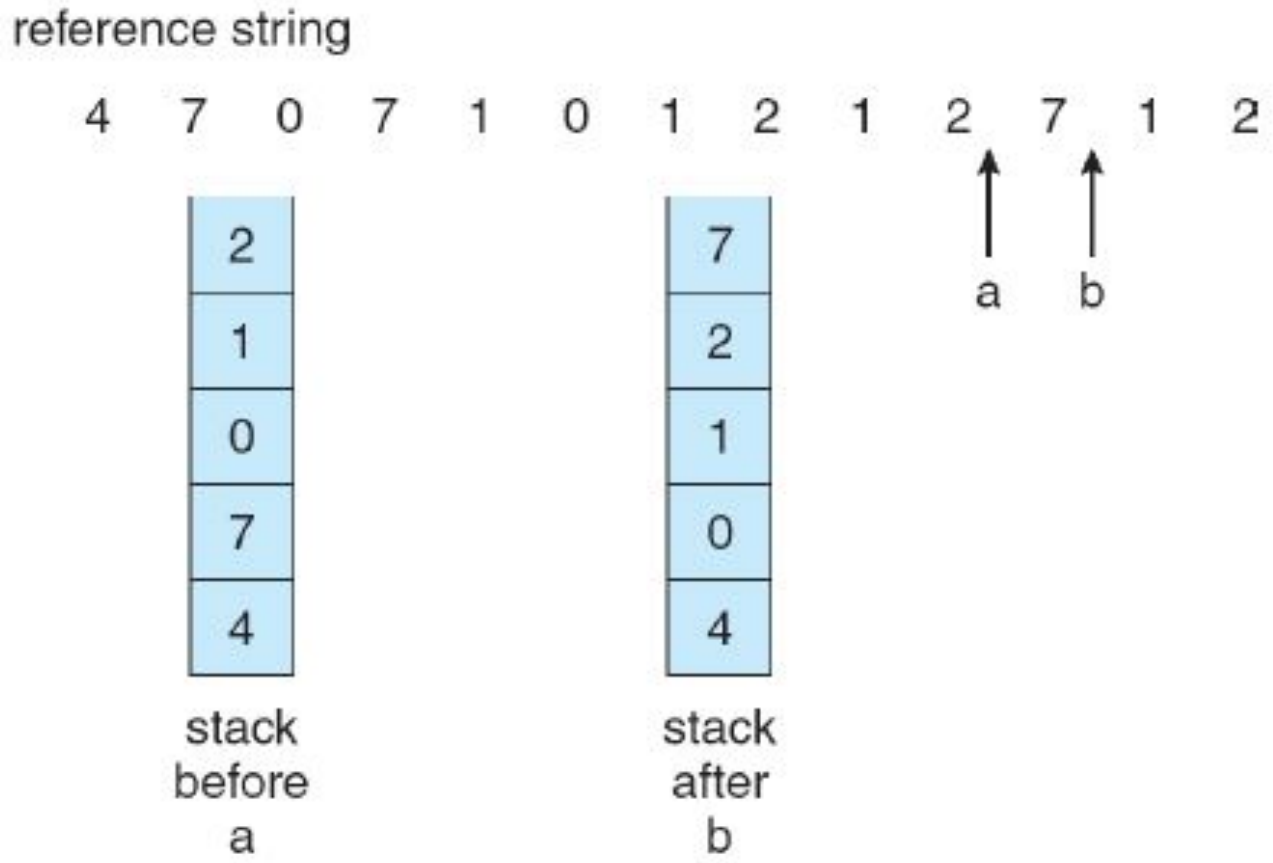


Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Реализации алгоритма LRU

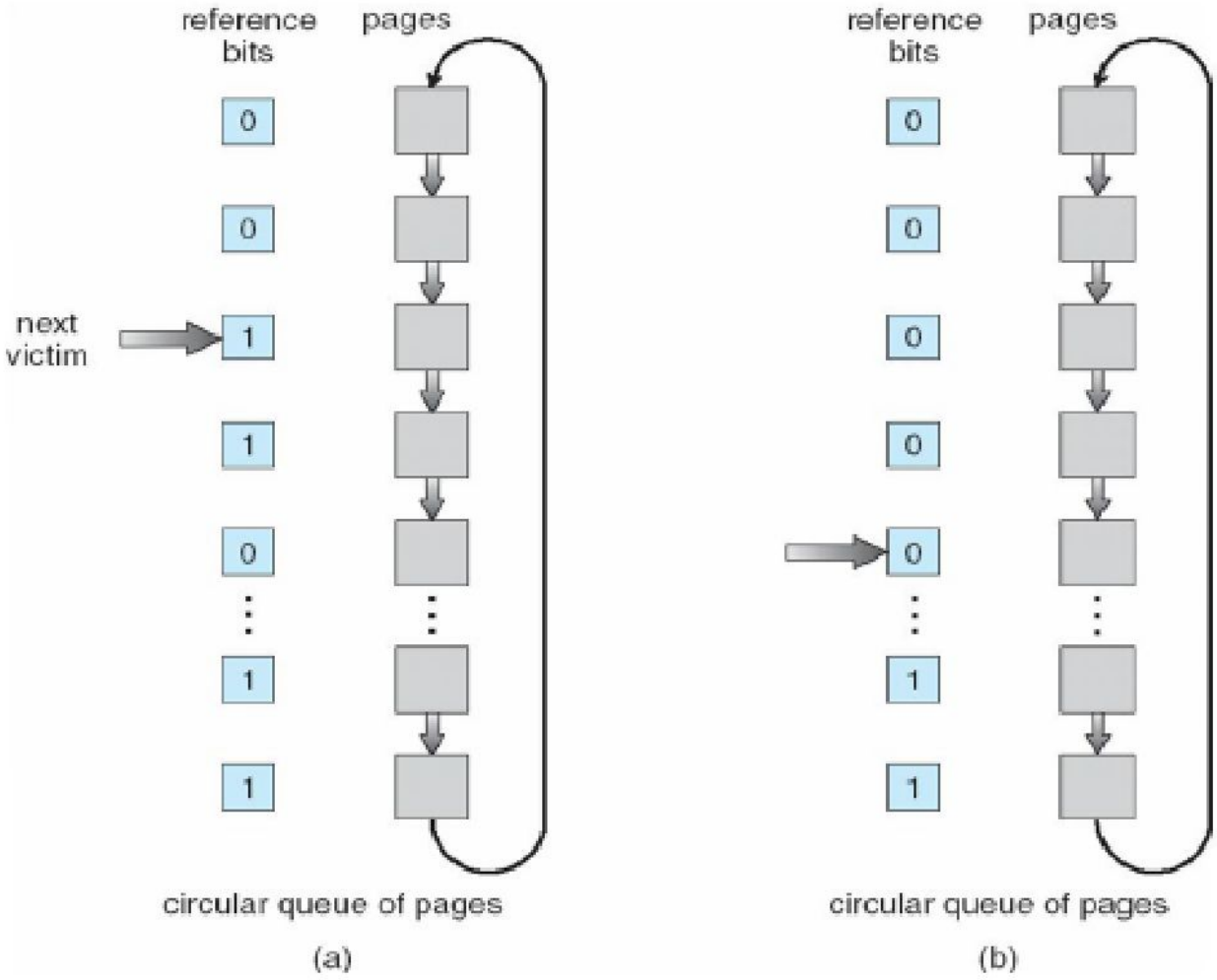
- С помощью стека – создать стек из номеров страниц с двойной связью
  - При ссылке на страницу
    - переместить ее на вершину стека
    - изменить 6 указателей
  - Зато не нужно искать страницу для замещения!

Использование стека для записи последних ссылок



Gagne, Operating Systems Concepts, 8th edition, 2009

Алгоритм второго шанса («Часы»)



Systems Concepts, 8th edition, 2009

Счетные алгоритмы

- Ведется счетчик количества ссылок на каждую страницу
- LFU – замещает страницу с наименьшим значением счетчика

- MFU – основан на предположении, что страница с наименьшим значением счетчика, возможно, была недавно подгружена и должна вскоре использоваться

Выделение фреймов

- Каждому процессу необходим какой-то минимум страниц в памяти
- Например: IBM 370 – инструкция SS MOVE может потребовать для выполнения 6 страниц:
  - сама инструкция 6 байт, может быть «размазана» на две страницы
  - 2 страницы для выполнения from
  - 2 страницы для выполнения to
- Две схемы размещения
  - фиксированное размещение
  - приоритетное размещение

Фиксированное распределение

- Равное размещение – например, если у вас 100 фреймов на 5 процессов, выделите каждому по 20
- Пропорциональное размещение – выделяете фреймы в соответствии с размером процесса

s(i) – размер процесса i, S – суммарный размер всех процессов, m – количество фреймов

$$a(i)=s(i)*m/S$$

Приоритетное распределение

- Используется схема пропрционального распределения, но не с размерами, а с приоритетами
- Если процесс P(i) вызывает страничное прерывание
  - для замещения выбирается один из его фреймов
  - для замещения выбирается фрейм процесса с меньшим приоритетом

Глобальная и локальная политики замещения

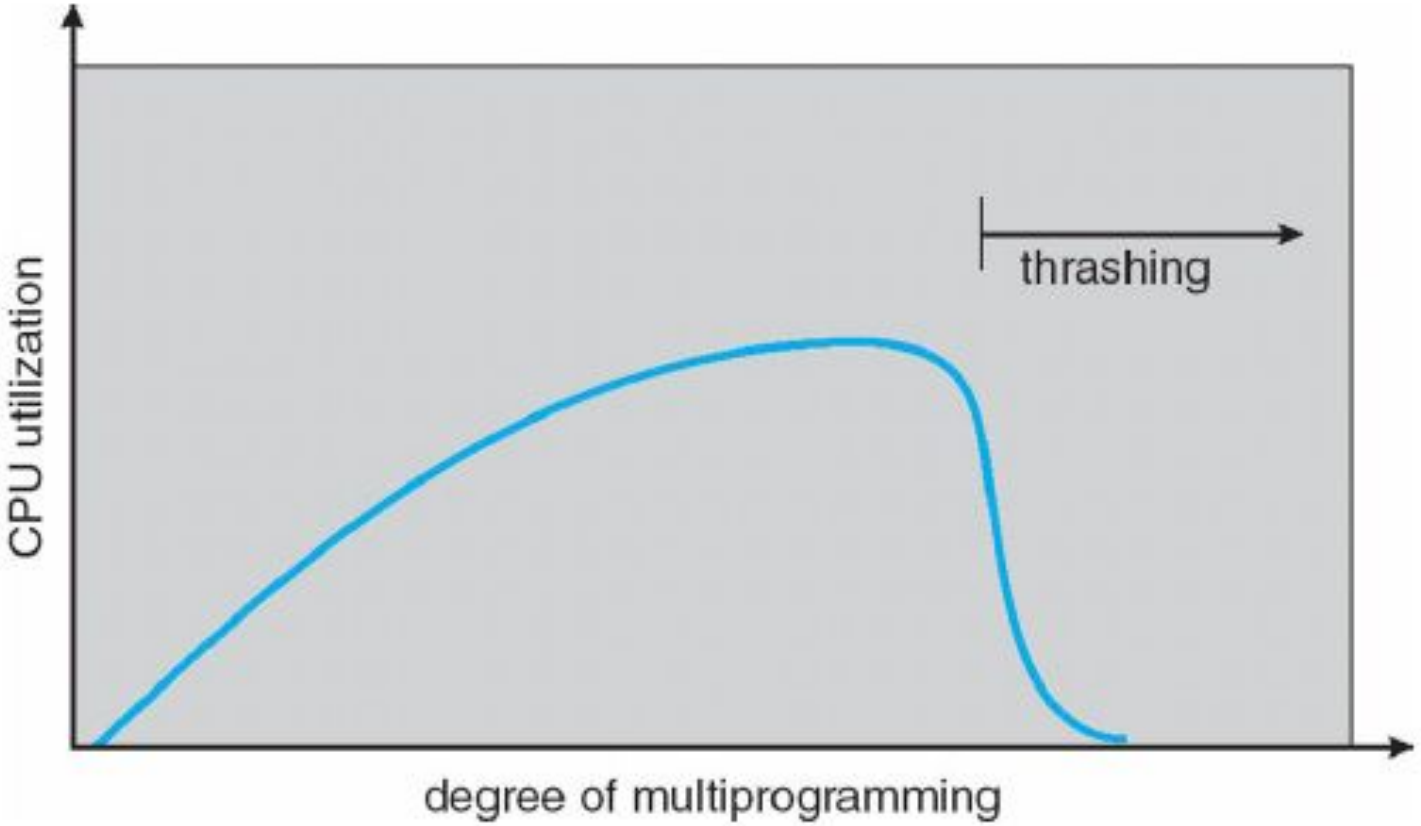
- Глобальное замещение – процесс выбирает фрейм для замещения из всего набора фреймов, процесс может забирать фреймы у других

- Локальное замещение – процесс может замещать фреймы только из своего набора

Пробуксовывание (Trashing)

- Если процессу «недостаточно» страниц, коэффициент страничных прерываний будет очень высоким. Это приводит к
  - низкой загрузке процессора
  - ОС предполагает, что нужно уменьшить «степень паралеллизма» - то есть количество процессов
  - процессы добавляются в систему
- Пробуксовывание – процессы заняты выгрузкой страниц на диск и загрузкой их в память, а не полезной работой

Trashing



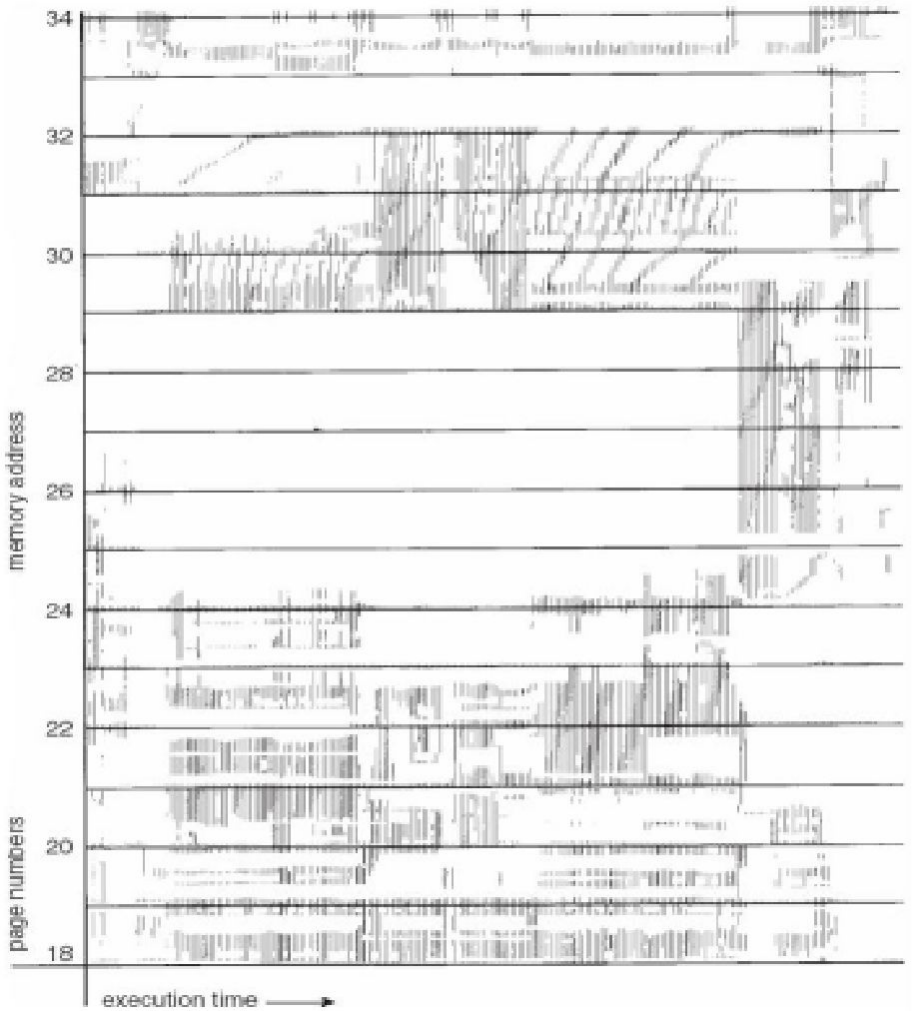
Gagne, Operating Systems Concepts, 8th edition, 2009

Загрузка по требованию и пробуксовывание

- Почему работает загрузка по требованию?
  - Принцип локальности
    - процесс мигрирует с одной локальности в другую
    - локальности могут накладываться друг на друга
- Почему вощникает пробуксовывание?
  - сумма всех локальностей всех процессов превышает размер памяти

Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Локальность при обрашении к памяти



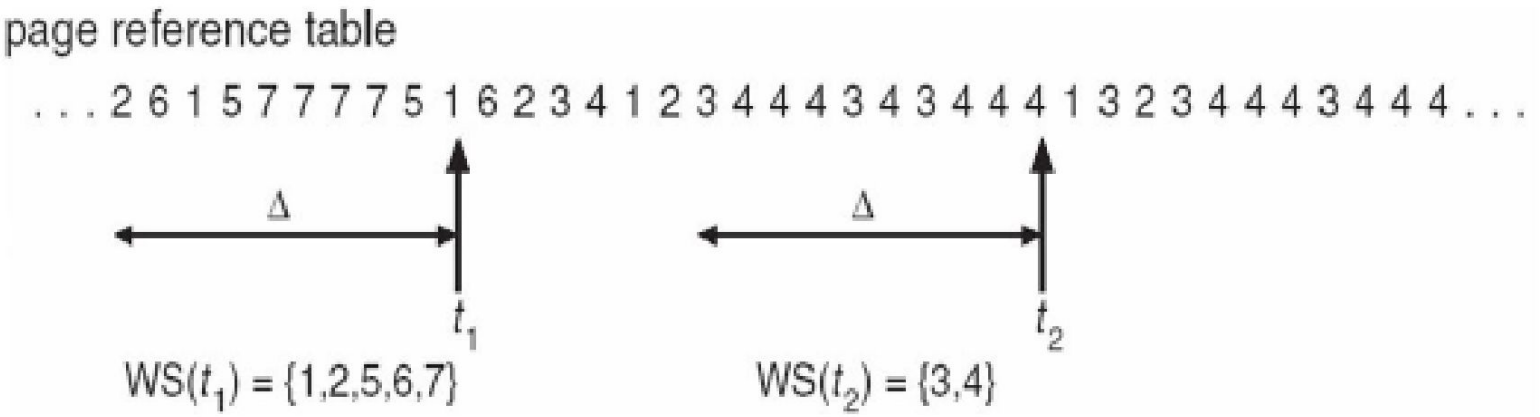
Systems Concepts, 8th edition, 2009

Модель рабочего набора

- Δ – «окно» рабочего набора – фиксированное число обращений к памяти (например 10000 инструкций)
- WSS(i) – рабочий набор процесса i - общее количество ссылок на страницы в самом недавнем «окне» (изменяется во времени)
  - если Δ очень мала, локальности не будет наблюдаться
  - если Δ очень велика, будет наблюдаться несколько локальностей
  - при Δ->∞ - локальность представляет всю программу
- D=∑WSS(i) – количество фреймов по требованию
- если D>m – пробуксовка
- Политика – если D>m – процесс приостанавливается

Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Модель рабочего набора



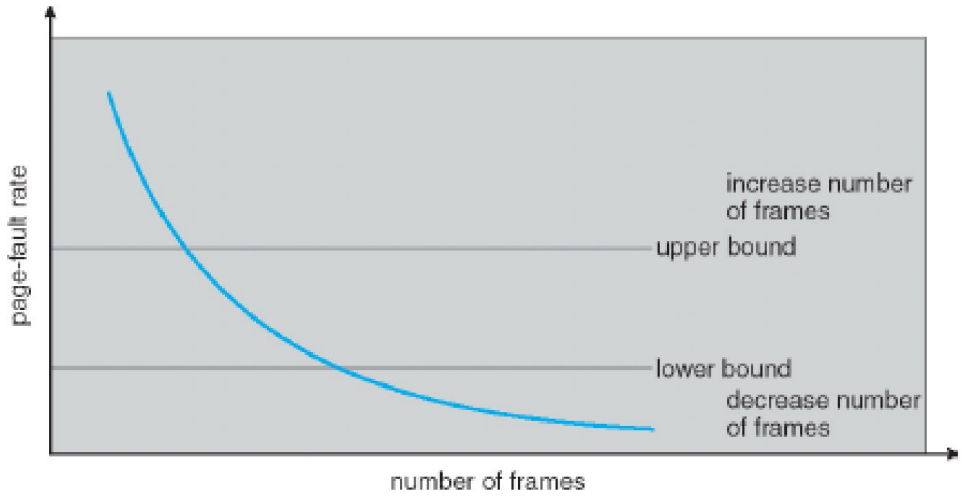
Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Отслеживание рабочего набора

- Приближение по интервалам таймера и ссылочным битам
- Пример – Δ=10000
  - Прерывание по таймеру через каждые 5000 единиц времени
  - Храним в памяти 2 байта на каждую страницу
  - При кажджом прерывании по таймеру сбрасываем (copy&set) значения всех ссылочных битов в 0
  - Если один из ссылочных битов=1 – страница в рабочем наборе
- Почему это приближение не является точным?
- Улучшение – храним 10 бит на страницу и делаем прерывание через 1000 единиц времени

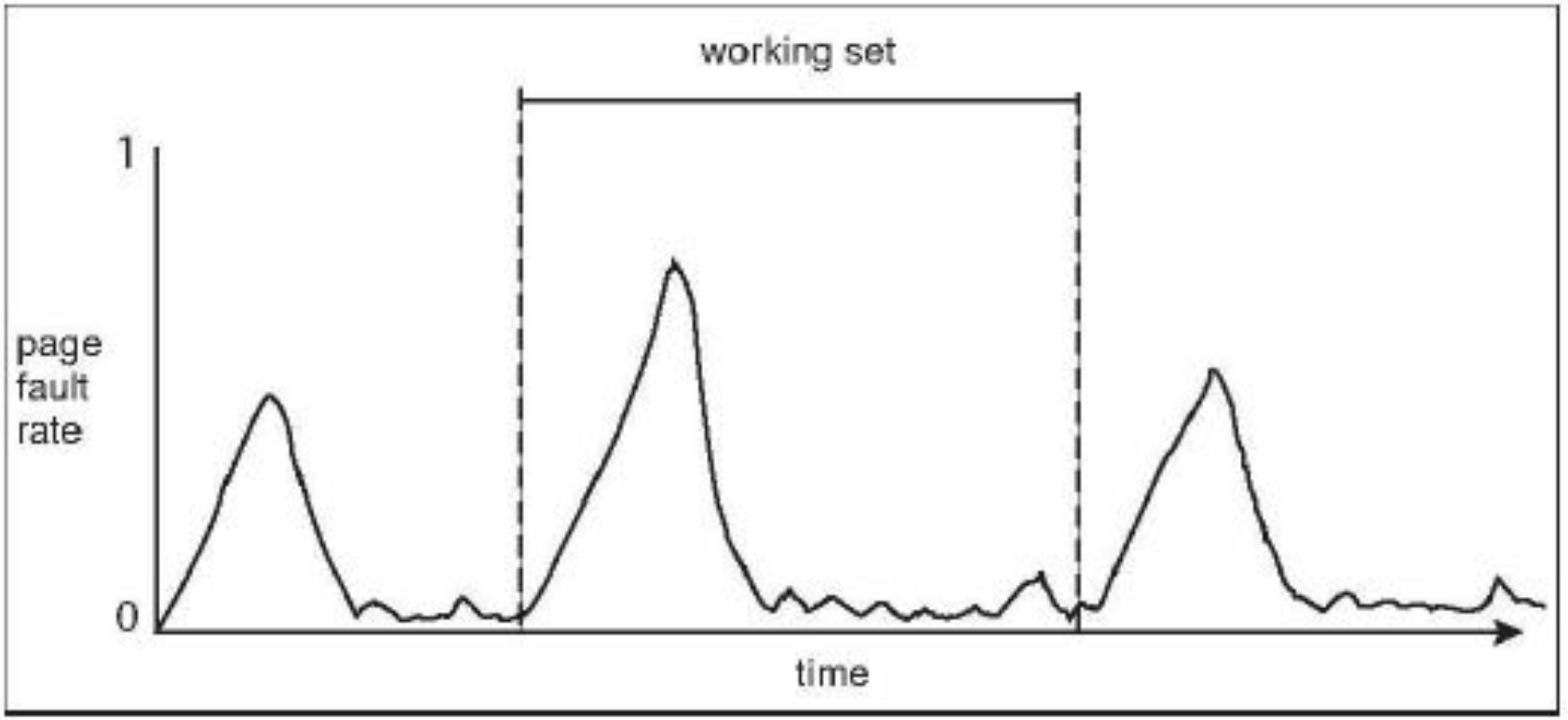
Схема частоты страничных прерываний

- Устанавливаем приемлемый коэффициент страничных прерываний
  - если реальный коэффициент слишком низкий, процесс теряет фрейм
  - если наоборот – процессу выделяются дополнительный фрейм



Gagne, Operating Systems Concepts, 8th edition, 2009

Рабочие наборы и Страничные Прерывания



Silberschatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

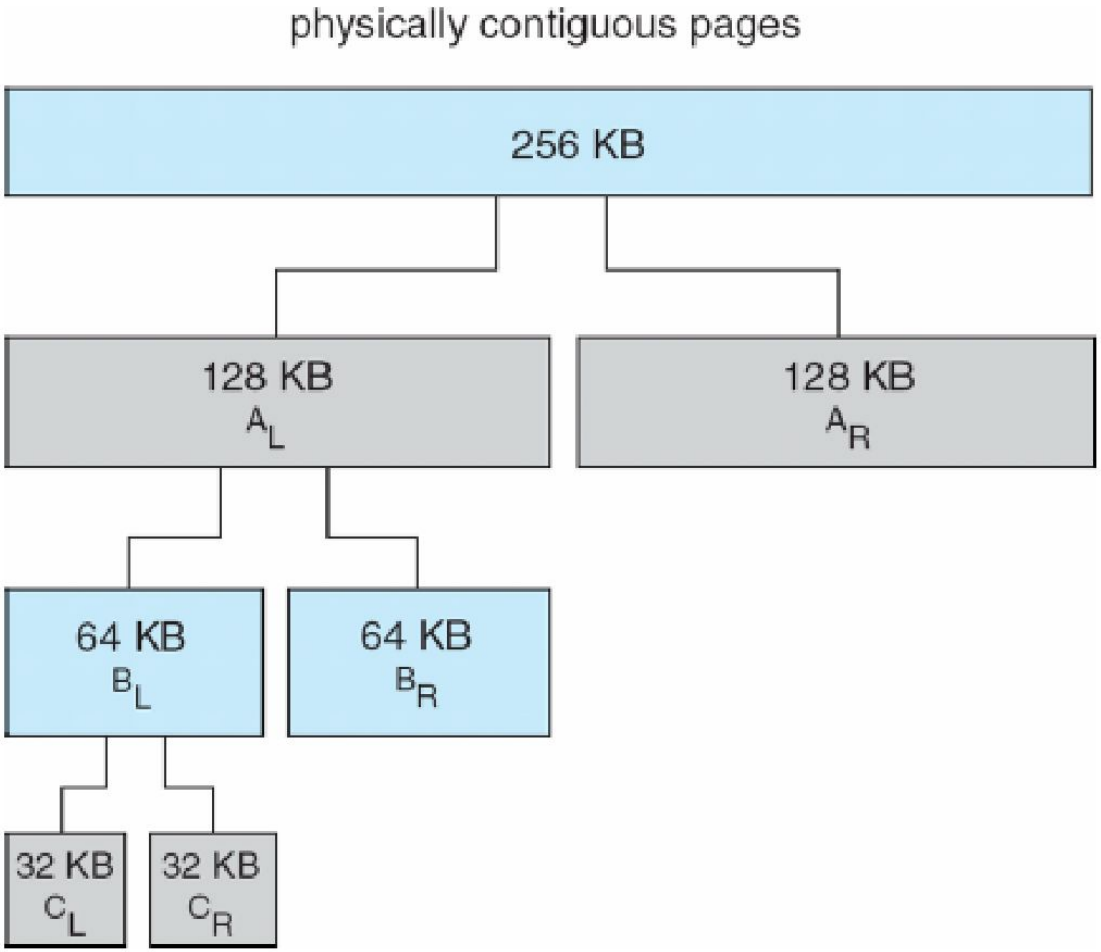
Распределение памяти ядра

- Память ядра выделяется особым образом, не так, как память для пользовательским приложений
- Часто выделяется из пула свободной памяти
  - Ядро запрашивает память для структур разных размеров
  - Некоторые участки памяти ядра должны быть непрерывными

Buddy System

- Память выделяется сегментами фиксированного размера, состоящими из непрерывных последовательностей страниц
- Память распределяется сегментами, размеры которых являются степенью 2
  - Запросы округляются до ближайшей степени 2
    - Когда нужен сегмент, меньший чем доступные, текущий сегмент делится на два (что составляет ближайшую меньшую степень 2)
    - Деление продолжается, пока сегменты не достигнут нужного размера

Распределение памяти Buddy System

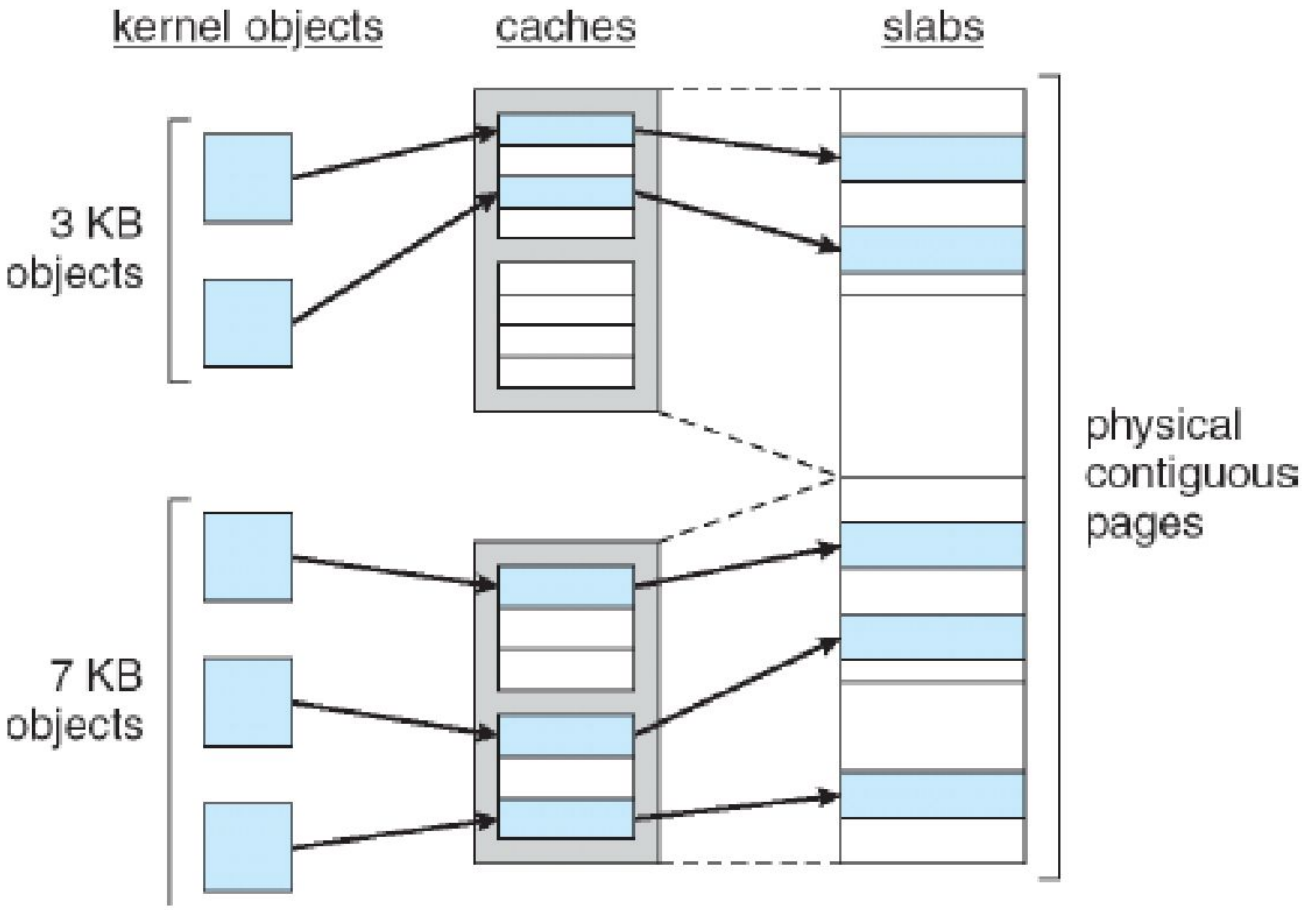


Systems Concepts, 8th edition, 2009

Slab-распределение

- Альтернативная стратегия
- Слаб (slab) - это одна и более непрерывно расположенных физических страниц
- Кэш состоит из одного или более слабов
- Отдельный кэш для каждой уникальной структуры ядра
  - Каждый кэш заполняется объектами – структурами
- При создании кэша он помечается как свободный
- При сохранении структуры объект помечается как используемый (used)
- Если слаб заполнен используемыми объектами, следующий объект создается в пустом слабе
  - Если пустых слабов нет, создается новый
- Преимущества – нет фрагментации, запросы на память удовлетворяются быстрее
- Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Слаб-распределение



Silberschatz, Galvin, Gagne, Operating System Concepts, 8th edition, 2009

Упреждающая загрузка (Pre-Paging)

- Применяется для уменьшения большого количества страничных прерываний при запуске процесса
- Загрузка всех или части страниц, которые понадобятся процессу до обращения к ним
- Но если загруженные страницы не понадобятся, операции ввода/вывода и память расходуются напрасно
- Предположим, s страниц было загружено и a – часть страниц, которая была использована (a<1)
  - Что больше – экономия за счет отсутствия s*a страничных прерываний или расходы на загрузку s(1-a) «лишних» страниц?
  - если a~0 – значит, упреждающая загрузка только уменьшает производительность

Silbershatz, Gavin and Gagne, Operating Systems Concepts, 8th edition, 2009

Другие проблемы – Размер страницы

- При выбор размера страницы нужно принимать во внимание следующее:
  - возможную внутреннюю фрагментацию
  - размер таблицы страниц
  - нагрузку на систему ввода-вывода
  - локальность

Достижимость памяти из TLB (кэша таблицы страниц – TLB Reach)

- TLB Reach – количество памяти, доступной из TLB
- TLB Reach=(TLB Size)*(Paga Size)
- Было бы идеально, если бы рабочий набор каждого процесса хранился в TLB (в ином случае коэффициент страничных прерываний будет высоким)
- Увеличение размера страницы: может привести к внутренней фрагментации, к тому же не все приложения требуют большого размера страницы
- Обеспечение нескольких размеров страниц: позволяет приложениям, требующих большого размера страниц, использовать такие страницы без увеличения фрагментации

Структура программы

- Program structure
  - Int[128,128] data;
  - Each row is stored in one page
  - Program 1

```
for (j = 0; j <128; j++)
        for (i = 0; i < 128; i++)
            data[i,j] = 0;
```

128 x 128 = 16,384 page faults

  - Program 2

```
for (i = 0; i < 128; i++)
        for (j = 0; j < 128; j++)
            data[i,j] = 0;
```

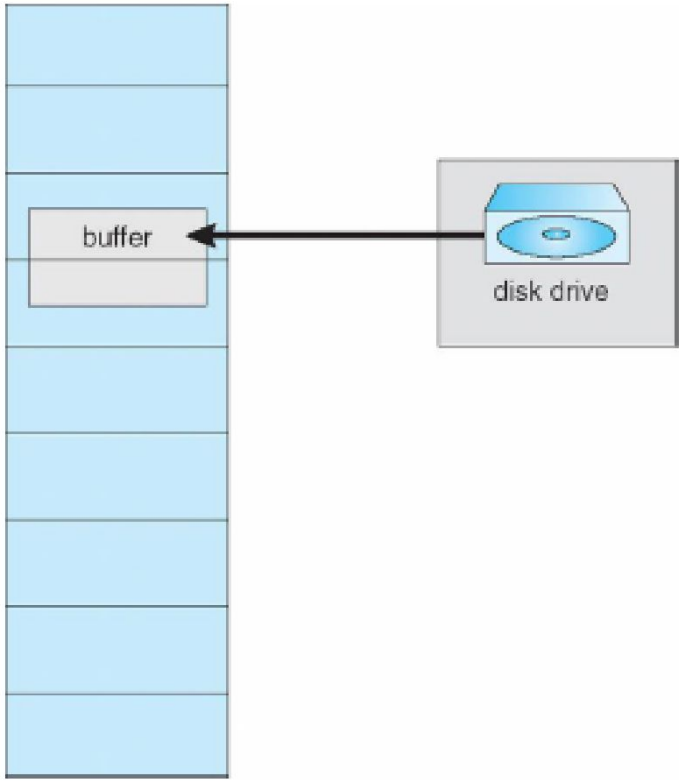
128 page faults

Silberschatz, Galvin and Gagne, Operating Systems Concepts, 8th edition, 2009

Другие вопросы – взаимная блокировка ввода-вывода (I/O interlock)

- I/O interlock – иногда нужно заблокировать страницы в оперативной памяти
- При операциях ввода/вывода – страницы, использующиеся при копировании файла с устройства должны быть заблокированы в оперативной памяти (чтобы их не выбрали в качестве жертвы при замещении страниц)

В операции ввода-вывода страницы буферизуются и не должны выгружаться



Si
Ga
Systems Concepts, 8th edition, 2009