# Лекция 12:
# Виртуальная память
# Стратегии замещения страниц

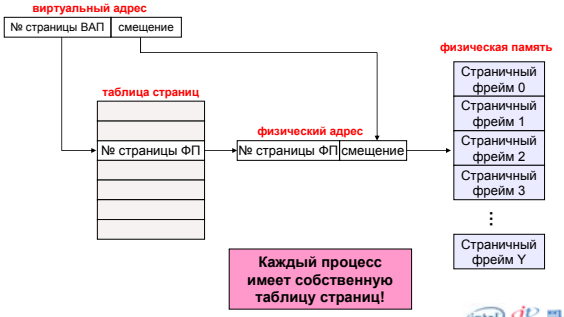
Алексей Линёв
Александр Мощук
Кирилл Погорельский

somes slides are adapted from the OS course at the University of Washington

## Выгружаемость виртуальной памяти

- Мы отмечали, что не обязательно всем страницам ВАП присутствовать в оперативной памяти
  - ВАП (его используемая часть) полностью хранится на жестком диске в виде множества страниц
  - ОС использует оперативную память качестве кэша (страниц)
  - требуемая страница перемещается в свободный страничный фрейм ФП
  - если свободных страничных фреймов в ФП не осталось – какая-то страница должна быть "замещена"
    - замещаемые страницы отправляются на жесткий диск (при этом запись на диск необходима только в том случае, если у страницы установлен признак изменения)
  - данный механизм прозрачен для прикладных программ (но сказывается на производительности)
    - в управлении участвуют аппаратное обеспечение и ОС
- Обычно описанную схему называют выгружаемой виртуальной памятью (paged virtual memory)

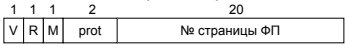
## Страничное преобразование
## Механизм преобразования адреса




## Страничные сбои (page faults)

- Что произойдет, если процесс обратится по виртуальному адресу из страницы, вытесненной на жесткий диск?
  - при замещении страницы ОС устанавливает в ее дескрипторе признак недействительности и сохраняет место расположения страницы на диске в специальной структуре (похожей на таблицу страниц, но содержащей адреса страниц на диске)
  - когда процесс пытается обратиться к отсутствующей странице, при обращении к недействительному дескриптору возникнет исключение (страничный сбой)
    - то есть, будет инициировано прерывание
  - в результате, ОС запустит обработчик страничных сбоев
    - обработчик найдет в структуре, похожей на таблицу страниц, адрес страницы на диске
    - обработчик считает страницы в страничный фрейм оперативной памяти и скорректирует дескриптор страницы, записав в него новый физический адрес страницы и сбросив признак недействительности
    - ОС запустит прерванный процесс
    - вообще-то, существует еще тысяча и одна деталь…

## Дескрипторы страниц
## Page Table Entries (PTEs)




- Дескрипторы страниц управляют отображением
  - признак достоверности (valid bit) определяет, можно ли использовать данный дескриптор
    - указывает, является ли использованный виртуальный адрес корректным
    - проверяется каждый раз при использовании виртуального адреса
  - признак использования (referenced bit) указывает, выполнялся ли доступ к данной страница ВАП
    - устанавливается процессором при выполнении любой операции над содержимым страницы
  - признак изменения (modified bit) указывает, изменялось ли содержимое страницы
    - устанавливается при выполнении записи в страницу
  - биты прав доступа (protection bits) определяют, какие операции разрешены для содержимого страницы
    - чтение, запись, исполнение (read, write, execute)
  - № страницы ФП задает номер фрейма страницы ФП, на которую отображена данная страница ВАП
    - адрес начала страницы ФП = № страницы ФП :: 0…0

## Подкачка страниц по требованию
## Demand paging

- Страницы загружаются в оперативную память только при обращении к ним
  - только код и данные, которые нужны процессу (затребованы!) должны быть загружены
    - естественно, требования со временем изменяются
  - Такой подход называется подкачкой по требованию
- Очень немногие системы пытаются предугадать, что потребуется процессу в будущем
  - низкая эффективность модулей предсказания – общеизвестна
- Но это не относится к кластерам страниц
  - ОС отмечает, какие страницы желательно загружать и вытеснять вместе
  - соответственно, если к одной из них произошло обращение – загружаются все
  - возможна организация механизма, позволяющего программисту или компилятору определять кластеры

## Замещение страниц

- При считывании страницы с диска, куда она помещается?
  - если имеются свободные страничные фреймы ФП, используется один из них
    - какая структура данных может при этом использоваться?
  - если нет – какая-то другая страница вытесняется на диск
  - это называется замещение страниц
- Стратегии замещения страниц
  - попытаться найти страницу, которая не потребуется в ближайшее время
  - попытаться найти страницу, которую не нужно записывать на диск (тем самым, сэкономив на операции записи)
  - ОС, как правило, поддерживает некоторый пул свободных страниц, поэтому выделение страницы не обязательно потребует замещения
  - ОС также обычно старается поддерживать наличие "чистых" страниц (не требующих записи на диск), таким образом, даже если вам нужно заместить страницу – не обязательно выполнять операцию записи
    - реализуется посредством опережающей записи во время простоя
  - мы еще вернемся к этой теме

## Как выбрать замещаемую страницу?

- Цель алгоритма замещения:
  - уменьшить число страничных сбоев посредством выбора лучшего кандидата на замещение
    - уменьшить общее число страничных сбоев или число страничных сбоев, вызванных конкретной программой?
  - лучше всего пожертвовать страницей, которая больше никогда не будет использоваться
    - wow! как же выбрать такую страницу?
  - пожертвовать страницей, которая долго не будет использована
    - Биледи: "вытеснение страницы, которая дольше всего не будет использована, минимизирует количество страничных сбоев"
- Далее в этой лекции
  - обзор нескольких алгоритмов замещения страниц
  - предполагаем использование локальных алгоритмов замещения (каждому процессу выделяется фиксированное число страничных фреймов ФП, и когда процессу требуется считать страницу с диска, в случае использования им всех выделенных фреймов замещается именно его страница)

## Как происходит загрузка программ?

- Создается дескриптор процесса
- Создается таблица страниц
- На диске создается образ адресного пространства процесса в виде множества страниц
- Заполняется таблица страниц (одно из полей дескриптора процесса ссылается на неё)
  - во всех дескрипторах страниц сбрасывается признак достоверности
- Заполняется структура, аналогичная таблице страниц, в которой указаны места размещения страниц процесса на диске
- Процесс запускается на выполнение
  - немедленно возникает страничный сбой при обращении к странице, содержащей код программы (и, возможно, при обращении к страницам данных или стека)
  - по мере считывания страниц процесса в память, количество страничных сбоев уменьшается

## №1: Алгоритм Биледи (Belady)

- Замещается страница, которая дольше всего не будет использована
  - оптимальный: наименьшее число страничных сбоев (помните SJF?)
  - есть проблема: будущее предвидеть невозможно
- Какая польза от алгоритма Биледи?
  - он служит критерием оценки других алгоритмов
    - если алгоритм Биледи не намного лучше Вашего алгоритма – Ваш алгоритм достаточно хорош
    - а как выполнить сравнение?
- Существует ли лучший практически реализуемый алгоритм?
  - нет – эффективность алгоритма сильно зависит от сложившейся ситуации
- Существует ли худший алгоритм?
  - тоже нет, но алгоритм, случайно выбирающий замещаемую страницу, работает достаточно плохо
    - вы будете смеяться, но существуют ситуации, когда ОС используют почти случайные алгоритмы с достаточно хорошими результатами!

## В чем преимущества такого подхода?

- Принцип локальности!
  - временна́я локальность
    - недавно использованные адреса скорее всего, будут вскоре снова использованы
  - пространственная локальность
    - адреса рядом с недавно использовавшимися, наверное тоже вскоре будут использованы (объясните, почему)
- Принцип локальности означает, что перемещение страниц может возникать не часто
  - после того, как вы загрузили страницу, она будет использована много раз
  - в среднем, вы обращаетесь к страницам, которые уже находятся в оперативной памяти
  - однако, все перечисленное зависит от множества факторов
    - степень локальности прикладной программы
    - стратегия замещения страниц и последовательность обращений приложения к страницам ВАП
    - соотношение между объемом физической памяти и "рабочим множеством" процесса

## №2: First In – First Out (FIFO)

- FIFO – прост в понимании, прост в реализации
  - загружая страницу, поместите ее в конец списка
  - вытесняйте страницу, расположенную в начале списка
- Может ли этот алгоритм работать хорошо?
  - возможно, давно загруженные страницы уже не используются
- Может ли этот алгоритм работать плохо?
  - возможно, давно загруженные страницы все еще используются
  - в любом случае, FIFO не использует никакой информации за исключением порядка загрузки
- На практике, эффективность FIFO обычно ужасна
- К тому же, FIFO подвержен аномалии Биледи
  - существуют последовательности обращения к страницам (последовательности ссылок, reference strings), для которых при увеличении физической памяти, выделенной процессу, число страничных сбоев увеличивается

## №3: Least Recently Used (LRU)

- LRU использует информацию об использовании страниц для принятия более взвешенных решений
  - основная идея: имеющийся опыт – хороший помощник при предсказании будущего
  - для замещения выбирается страница, которая не дольше всех не использовалась
    - LRU смотрит в прошлое, Биледи – хочет смотреть в будущее
    - чем LRU отличается от FIFO?
  - можете придумать ситуацию, в которой производительность LRU будет ужасна?
    - обычно LRU очень хорош
- Реализация
  - в идеальном случае, при каждом обращении к странице текущее время сохраняется в ее дескрипторе, поддерживается упорядочивание страниц по времени обращения к ним или быстрый поиск,…
  - слишком дорого с точки зрения пропускной способности памяти, времени работы алгоритма и т.д.

## Распределение фреймов ФП между процессами…

- FIFO и LRU Clock могут быть реализованы в виде локальных или глобальных алгоритмов замещения
  - локальный алгоритм
    - каждому процессу выделяется пул страниц ФП, которые он может использовать
    - при замещении вытесняется страница того же процесса, для которого загружается новая
  - глобальный алгоритм
    - для замещения выбирается некоторая страница вне зависимости от того, какому процессу она принадлежит
    - состав страничных фреймов, выделенных процессу может достаточно быстро изменяться
- Преимущества/недостатки локальных алгоритмов?
- Преимущества/недостатки глобальных алгоритмов?
  - в Linux реализован глобальный алгоритм

## Аппроксимация LRU

- Существует множество аппроксимаций LRU, основанных на использовании признака обращения (reference bit) к странице
  - для каждой страницы заводится счетчик обращений
  - с некоторым постоянным интервалом для каждой страницы
    - если признак обращения = 0 (страница не использовалась), увеличить счетчик на 1
    - если признак обращения = 1 (страница использовалась), обнулить счетчик
    - сбросить признак обращения
  - счетчик будет содержать число интервалов, в течение которых страница не использовалась
    - страница с максимальным значение счетчика – дольше всего не использовавшаяся
- В некоторых архитектурах признак обращения отсутствует
  - для эмуляции признака обращения можно использовать признак достоверности (valid bit), сбрасывая его для возникновения "псевдосбоев страниц"
    - огромное количество дополнительных исключений

## Распределение фреймов ФП между процессами

- Гибридные алгоритмы
  - используется локальный алгоритм замещения
  - четко определены правила изменения количества выделенных процессу страничных фреймов ФП
- Какой из 3 представленных подходов лучше?

## №4: LRU Clock

- Также известный под названием Second Chance ("вторая попытка")
  - замещается "достаточно старая" страница
  - все фреймы ФП выстраиваются в один большой круг (часы)
    - реализуемый обычным кольцевым списком
  - "стрелка часов" указывает следующего кандидата на вытеснение
    - движется по списку страниц как стрелка часов
    - если признак обращения сброшен, значит, страница давно не использовалась, и она – подходящая жертва
      - кстати, каков ее минимальный "возраст" в этом случае?
    - если признак обращения установлен, он сбрасывается, и стрелка переводится на следующую страницу
  - чем чаще требуются страницы, тем быстрее движется стрелка
  - при достаточно большом объеме памяти дополнительные расходы невелики
  - если памяти очень много, точность используемой информации деградирует
    - и приходится использовать еще одну стрелку для сброса признаков обращения

## "Рабочее множество" процесса (working set)

- "Рабочее множество" (working set) процесса используется при моделировании потребления памяти программами
  - рабочее множество – набор "нужных" в настоящий момент процессу страниц виртуальной памяти
  - формальное определение введено Питером Деннингом (Peter Denning) в 60-х
- Определение
  - WS(t,w) = {множество страниц P, таких что к P происходило обращение в течение временного интервала (t,t-w) }
    - t – время
    - w – "окно" рабочего множества (измеряемое в количестве обращений к страницам)
    - страница относится к рабочему множеству (WS), если к ней было обращение в числе последних w обращений
  - очевидно, рабочее множество изменяется в ходе работы программы
  - как и размер рабочего множества (количество страниц в WS)

## №6: Частота страничных сбоев (ЧСС) Page Fault Frequency (PFF)

- PFF – гибкий алгоритм, использующий очень специфичный метод
- Пытается уравнять число страничных сбоев, генерируемых разными процессами, и достичь "приемлемой" суммарной частоты страничных сбоев
  - отслеживает частоту страничных сбоев для всех процессов
  - если у процесса частота страничных сбоев выше некоторого порога, ему выделяется дополнительная ФП
    - частота страничных сбоев снижается
  - если у процесса частота страничных сбоев ниже некоторого другого порога, у него забираются излишки ФП
    - теперь он будет генерировать страничные сбои чаще, но какой-то другой процесс будет генерировать их реже

## Размер рабочего множества

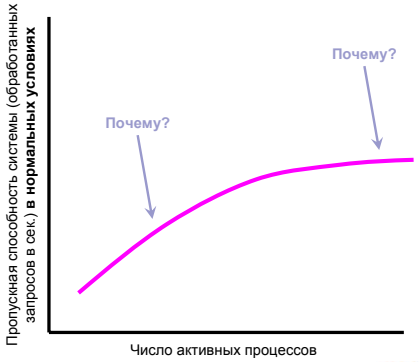
- Размер рабочего множества |WS(t,w)| изменяется в зависимости от степени локальности программы
  - в периоды с плохой локальностью программа обращается к бóльшему количеству страниц
  - в течение таких периодов размер рабочего множества больше
- Интуитивно понятно, что рабочее множество должно располагаться в оперативной памяти; в противном случае вы столкнетесь с огромным числом страничных сбоев (thrashing, режим интенсивной подкачки)
  - когда вас спрашивают "Как много памяти нужно программе XYZ?" – имеется в виду "каков средний (наибольший) размер рабочего множества программы XYZ?"

## Режим интенсивной подкачки (Thrashing)

- Режим интенсивной подкачки – ситуация, когда система бóльшую часть времени занята обработкой страничных сбоев, в оставшееся время выполняя полезную работу
  - может произойти даже при наличии достаточного количества памяти, но использования плохого алгоритма замещения (или плохо совместимого с поведением конкретных программ)
  - может произойти при слишком больших потребностях в оперативной памяти
    - например, слишком много работающих процессов

## №5: Гипотетический алгоритм, основанный на рабочих множествах

- Оцениваем |WS(0,w)| для процесса
- Запускаем процесс только в том случае, если можем выделить ему данное количество страничных фреймов ФП
- Используем локальный алгоритм замещения (LRU Clock?), обеспечивая размещение в страничных фреймах ФП "правильных" страниц (из рабочего множества)
- Следим за изменениями размеров рабочих множеств процессов, динамически перераспределяя страничные фреймы ФП между ними
- Есть сложности? Решения?
- Что такое w?




- Алгоритмы замещения страниц
  - № 1: Биледи – оптимальный, но нереализуемый
  - № 2: FIFO – замещается наиболее давно загруженная страница
  - № 3: LRU – замещается наиболее давно не используемая страница
    - приближающие алгоритмы используют признак обращения в дескрипторах страниц
  - № 4: LRU Clock – вытесняется "достаточно старая" страница
  - № 5: Working Set – поддерживает присутствие в памяти рабочего множества
  - № 6: Page Fault Frequency – увеличивает/уменьшает число страничных фреймов ФП, выделенных процессу, в зависимости от количества генерируемых им страничных сбоев

## В каких случаях выбор алгоритма замещения имеет большое значение?

- НЕ в ситуации, когда памяти очень много
  - выбор алгоритма замещение не имеет большого значения (переобеспечение памятью)
- НЕ в ситуации, когда памяти слишком мало
  - выбор алгоритма замещение опять не имеет большого значения (недообеспечение памятью)
- Выбор алгоритма имеет значение только в интервале между переобеспечением и недообеспечением

## Выводы

- Виртуальная память
- Страничные сбои
- Подкачка страниц по требованию
  - не пытайтесь предвидеть будущее
- Замещение страниц
  - локальные, глобальные и гибридные алгоритмы
- Принцип локальности
  - временная локальность, пространственная локальность
- Рабочее множество процесса
- Режим интенсивной подкачки