Организация виртуальной памяти при меньшем размере оперативной памяти

1. Общее 1
2. Алгоритмы подкачки 1
2.1. Алгоритм подкачки Unix 1
3. Страничное замещение основной памяти и swapping 2
4. Копирование страниц при попытке записи (copy on write). 2

1. Общее

В элементе таблицы страниц предусматривается поле под специальный флаг – признак отсутствия страницы. При его установке аппаратура компьютера вместо нормального отображения виртуального адреса в физический прерывает выполнение команды и передает управление соответствующему компоненту операционной системы (т.н. «Demand Paging» - листание по требованию). После загрузки недостающей страницы в память программа продолжает свое выполнение с прерванного места.

Если в ОЗУ отсутствует место под загрузку недостающей страницы то операционная система должна в соответствии с заложенными критериями найти неиспользуемую страницу основной памяти и выгрузить ее во внешнюю память. (Совокупность критериев – «политика замещения», основанный на них алгоритм – «алгоритм подкачки»)

2. Алгоритмы подкачки

Алгоритмы подкачки делятся на:

- глобальные;
- локальные.

В обоих вариантах популярными являются FIFO (first in first out) и LRU (least Recently Used).

При использовании алгоритма FIFO для замещения выбирается страница, которая дольше всего остается приписанной к виртуальной памяти. Алгоритм LRU предполагает, что замещать следует ту страницу, к которой дольше всего не происходили обращения.

Для полной реализации алгоритма LRU необходимо поддерживать связный список всех содержащихся в памяти страниц. Сложность заключается в том, что список должен обновляться при каждом обращении к памяти. Поэтому существуют различные модификации алгоритма LRUю

2.1. Алгоритм подкачки Unix

Хотя на концептуальном уровне все аппаратные механизмы поддержки виртуальной памяти практически эквивалентны реальные реализации часто весьма различаются. Невозможно создать полностью машинно-независимый компонент управления виртуальной памятью. В ОС Unix существует разделение средств управления виртуальной памятью на аппаратно-независимую и аппаратно-зависимую часть подсистемы управления виртуальной памятью.

Основная идея состоит в том, что UNIX опирается на некоторое собственное представление организации виртуальной памяти, которое используется в аппаратно-независимой части подсистемы управления виртуальной памятью и связывается с конкретной аппаратной реализацией с помощью аппаратно-зависимой части.

Виртуальная память каждого процесса представляется в виде набора сегментов:

- Сегмент программного кода (text segment) [только для чтения, разделяемый] - содержит только команды.
- Сегмент данных (data segment) [чтение и запись, частный] - инициализированные и неинициализированные статические переменные программы, выполняемой в данном процессе.
- Сегмент стека (stack segment) [чтение и запись, частный] - область виртуальной памяти, в которой размещаются автоматические переменные программы, явно или неявно в ней присутствующие.

- Разделяемые сегменты (shared memory) [чтение и запись, частный] - образуется при подключении к ней сегмента разделяемой памяти.
- Сегменты файлов, отображаемых в виртуальную память (mapped files) [чтение и запись, разделяемый] - представляют собой разновидность разделяемых сегментов.

Три типа обязательны для каждой виртуальной памяти и сегменты этих типов присутствуют в виртуальной памяти в одном экземпляре для каждого типа.

3. Страничное замещение основной памяти и swapping

В ОС Unix используется «облегченный» вариант алгоритма подкачки, основанный на использовании понятия «рабочего набора». Большинство страниц виртуальной памяти процесса характеризуются локальностью обращений, означающей, что во время выполнения любой фазы процесс обращается только к определенной набору своих страниц. Это и называется «рабочий набор». Также, ОС стремится отслеживать рабочий набор процессов и при повторном запуске программы сразу загружать его - такой подход носит название модели рабочего набора, загрузка страниц перед тем как разрешить процессу работать называется опережающей подкачкой страниц.

Периодически для каждого процесса производятся следующие действия: просматриваются таблицы отображения всех сегментов виртуальной памяти этого процесса. Если элемент таблицы отображения содержит ссылку на описатель физической страницы, то анализируется признак обращения. Если признак установлен, то страница считается входящей в рабочий набор данного процесса и сбрасывается на ноль счетчик старения данной страницы. По ходу просмотра таблиц отображения в каждом из них признак обращения сбрасывается.

Выгрузку во внешнее хранилище страниц, не входящих в рабочие наборы процессов, производит специальный системный процесс-stealer (pagedaemon). Он начинает работать когда количество страниц в списке свободных страниц достигает установленного нижнего порога.

Если список описателей свободных страниц пуст, то начинает работать механизм свопинга. Основной повод для применения другого механизма состоит в том, что простое отнятие страницы у любого процесса потенциально вело бы к ситуации trashing, поскольку разрушало бы рабочий набор некоторого процесса. При этом любой процесс, затребовавшей страницу не из своего текущего рабочего набора, становится кандидатом на своппинг. Причем, в очереди к процессу-swapper может находиться несколько процессов. Процесс-swapper по очереди осуществляет полный своппинг этих процессов, до тех пор, пока количество свободных страниц не достигнет установленного в системе верхнего предела. После завершения выгрузки каждого процесса одному из процессов из очереди к процессу-swapper дается возможность попытаться продолжить свое выполние (в расчете на то, что свободной памяти может быть достаточно)

4. Копирование страниц при попытке записи (copy on write).

При выполнении системного вызова fork() ОС Unix образует процесс-потомок, являющийся полной копией своего предка. В данной ситуации создавать физическую копию виртуальной памяти процесса нерационально – требуется время на создание копии, а также существует ситуация дублирования информации.

Несмотря на то, что в сегмент запись разрешена, для каждой его страницы устанавливается блокировка записи. Тем самым, во время попытки выполнения записи возникает прерывание, и ОС на основе анализа статуса соответствующего сегмента принимает решение о создании копии именно этой страницы/сегмента.