Mata Kuliah : Sistem Operasi

Kode MK : IT-012336

10

Virtual Memori

Tim Teaching Grant
Mata Kuliah Sistem Operasi

Virtual Memory

- Latar Belakang
- Demand Paging
- Pembuatan Proses
- Page Replacement
- Alokasi Frame
- Thrashing
- Contoh Sistem Operasi

Latar Belakang

- Manajemen memori:
  - Alokasi “space” memori fisik kepada program yang diekesekusi (proses).
  - Pendekatan: Alokasi space sesuai dengan kebutuhan “logical address” => seluruh program berada di memori fisik.
    - Kapasitas memori harus sangat besar untuk mendukung “multiprogramming”.
  - Bagaimana jika kapasitas memori terbatas?
  - Pendekatan: Teknik Overlay (programming) dapat memanfaatkan kapasitas kecil untuk program yang besar.
    - Batasan (tidak transparant, cara khusus): program sangat spesifik untuk OS tertentu.

Latar Belakang (cont.)

- Q: Apakah sesungguhnya diperlukan seluruh program harus berada di memori?
  - Mayoritas kode program untuk menangani “exception”, kasus khusus dll. (sering tidak dieksekusi).
  - Deklarasi data (array, etc) lebih besar dari yang digunakan oleh program.
- IDEA:
  - Sebagian saja program (kode yang sedang dieksekusi) berada di memori, tidak harus serentak semua program berada di memori.
  - Jika kode program diperlukan maka OS akan mengatur dan mengambil page yang berisi program tersebut dari “secondary storage” ke “main memory”.

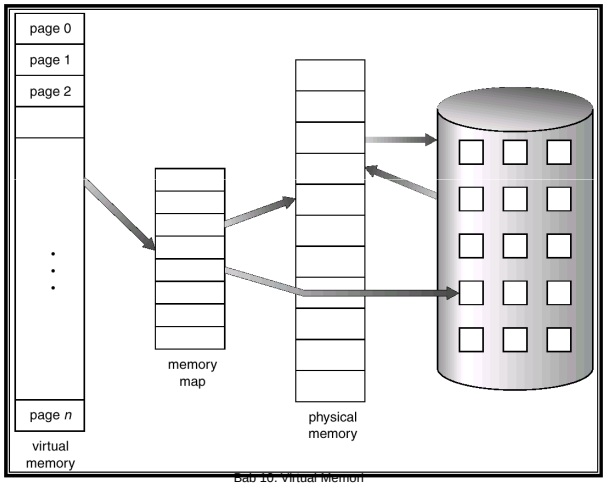
Latar Belakang (cont.)

- Pro's (jika OS yang melakukan "overlay")
  - Programmer dapat membuat program sesuai dengan kemampuan "logical address" (virtual address) tanpa harus menyusun modul mana yang harus ada di memori.
    - Fungsi OS sebagai "extended machine": memberikan ilusi seolah-olah memori sangat besar, memudahkan penulisan program dan eksekusi program.
  - Proses dapat dieksekusi tanpa memerlukan memori fisik yang besar => banyak proses.
    - Fungsi OS sebagai "resource manager": menggunakan utilitas memori yang terbatas untuk dapat menjalankan banyak proses.

Latar Belakang (cont.)

- Konsep Virtual Memory:
  - Pemisahan antara "user logical memory" (virtual) dengan "physical memory".
  - Logical address space (program) dapat lebih besar dari alokasi memori fisik yang diberikan.
  - Hanya sebagian kecil dari program yang harus berada di memori untuk eksekusi.
  - Terdapat mekanisme untuk melakukan alokasi dan dealokasi page (swapped out dan in) sesuai dengan kebutuhan (referensi program).
  - Terdapat bagian dari disk menyimpan sisa page (program) yang sedang dijalankan di memori.
- Virtual memory dapat diimplementasikan melalui :
  - Demand paging
  - Demand segmentation

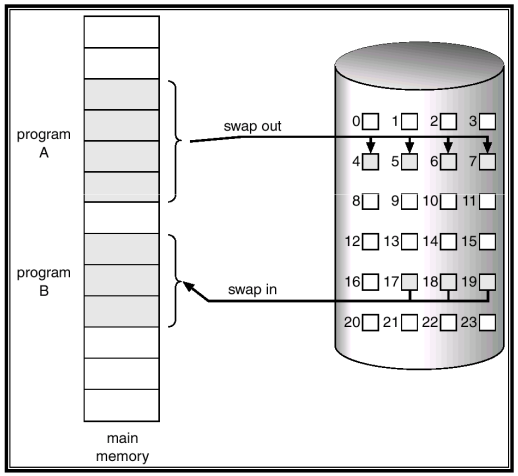
Virtual Memory Lebih Besar daripada Memori Fisik



Demand Paging

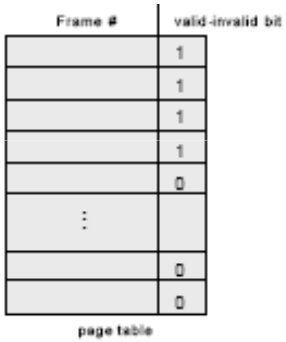
- Umumnya basis VM => paging.
- Demand (sesuai dengan kebutuhan):
  - Ambil/bawa page ke memory hanya jika diperlukan.
  - Umumnya program memerlukan page sedikit (one by one).
  - Less I/O & less memory (more users).
  - Transfer cepat (faster response).
- Kapan page dibutuhkan?
  - Saat ekesekusi proses dan terjadi referensi logical address ke page tersebut.
    - invalid reference ⇒ abort
    - not-in-memory ⇒ bring to memory
  - Page table menyimpan daftar page frame yang telah dialokasikan untuk proses tersebut.

Transfer Page Memory ke Contiguous Disk Space

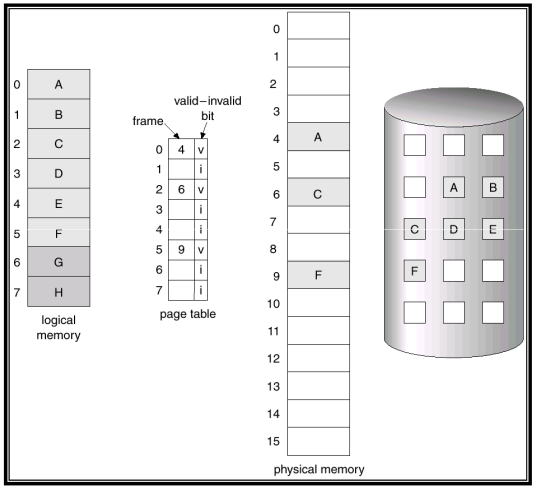


Valid-Invalid Bit

- Setiap entry pada page table terdapat bit: Valid dan Invalid mengenai keberadaan page di memori fisik (1 ⇒ in-memory, 0 ⇒ not-in-memory)
- Saat awal: page belum berada di memori maka bit adalah 0 (not in memory).
- Jika terjadi referensi dan page frame yang akan diakses bit Valid-Invalid 0 => page fault.

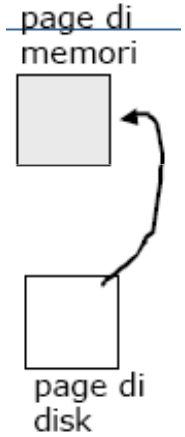


Page Table Ketika beberapa Page Tidak Berada di Main Memory

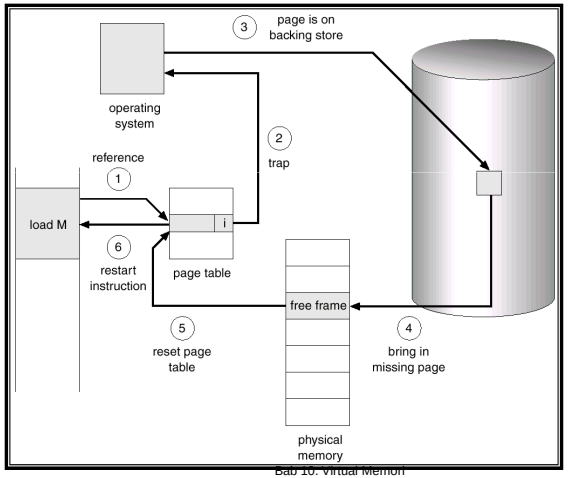


Page Fault (OS tasks)

- Saat pertama kali referensi ke page, trap ke OS => page fault.
- OS melakukan evaluasi, apakah alamat logical tersebut “legal”? OK, tapi belum berada di memori.
  1. Get empty frame (frame free list).
  2. Swap page into frame.
  3. Reset tables, validation bit = 1.
  4. Restart instruction: yang terakhir eksekusi belum selesai, mis.
     - block move



Tahap Penanganan Page Fault



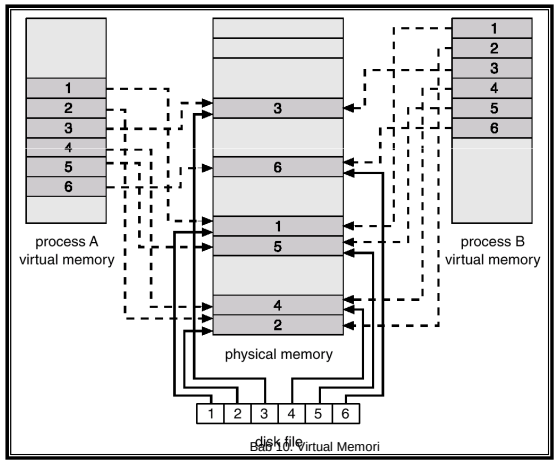
Tidak ada Frame yang bebas ?

- Jika terdapat banyak proses, maka memori akan penuh (tidak ada page frame yang free).
- Page replacement (penggantian)
  - Mencari kandidat “page” untuk diganti di memori dan “kemungkinan tidak digunakan” (allocate but not in used).
  - Swap page tersebut dengan page yang baru.
  - Algoritma: efisien dan mencapai min. jumlah page faults (karena kemungkinan page yang diganti harus di swap in lagi).
    - Page yang sama akan masuk ke memori pada waktu mendatang.

Memory-Mapped File

- Memory-mapped file I/O membolehkan file I/O diperlakukan sebagai rutin akses memori yang dipetakan sebagai blok disk ke dalam page memori
- Suatu file diinisialisasikan menggunakan demand pagin. Suatu bagian page file dibaca dari file sistem ke page fisik. Subsequent membaca/menulis ke/dari file yang diperlakukan dalam urutan memori akses.
- Secara sederhana file akses memperlakukan file I/O melalui memori melalui **read() write()** system calls.
- Beberapa proses juga dapat dipetakan pada fiel yang sama pada memori yang di-share.

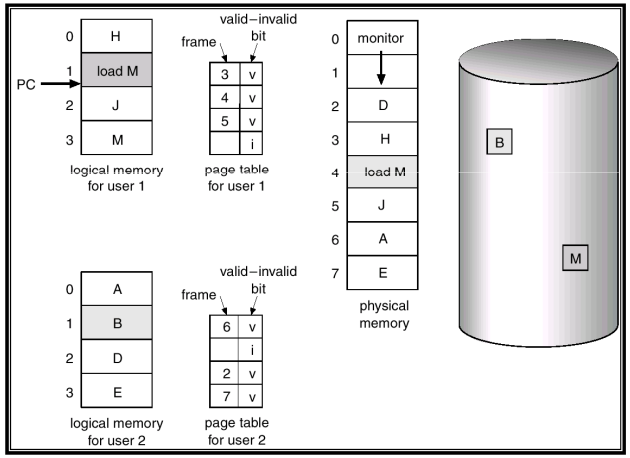
Memory Mapped Files



Page Replacement

- Mencegah alokasi yang berlebihan dari memori dengan memodifikasi layanan rutin page-fault melalui page
- Menggunakan *modify bit* untuk mengurangi overhead transfer page – hanya modifikasi page yang ditulis di disk.
- Page replacement melengkapi pemisahan antara memori logik dan memori fisik – virtual memori yang besar dapat memenuhi kebutuhan memori fisik yang kecil.

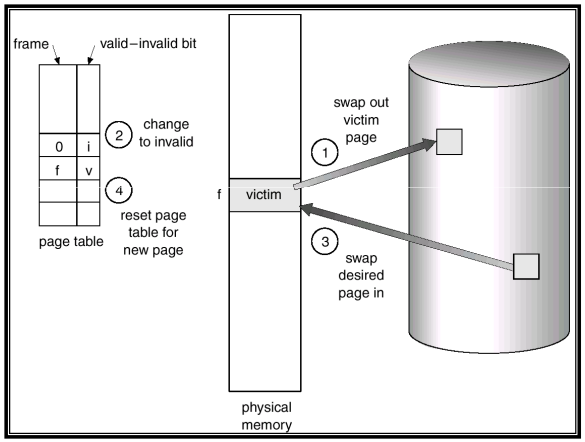
Kebutuhan Page Replacement



Basic Page Replacement

1. Tentukan lokasi yang diminta page pada disk.
2. Tentuka frame bebas :
   - Jika tersedia frame bebas, maka dapat digunakan
   - Jika tidak tersedia frame bebas, gunakan algoritma penggantian untuk memilih kandidat frame.
3. Baca page yang dituju ke dalam frame bebas (yang baru). Update page dan frame table.
4. Restart process.

Page Replacement

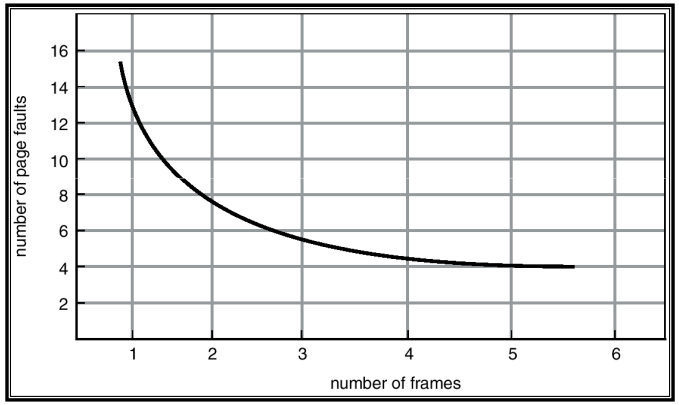


Algoritma Page Replacement

- Pilih page fault terendah.
- Evaluasi algoritma dengan menjalankan particular string dari memori acuan (reference string) dan menghitung jumlah page fault dari string.
- Contoh, reference string sebagai berikut :

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

Graph Page Faults vs. Jumlah Frame



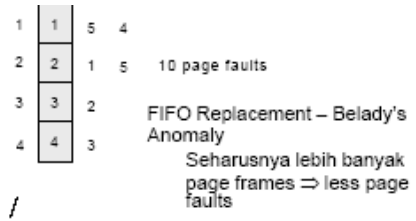
FIFO

- FIFO
  - Mengganti page yang terlama berada di memori.
  - Data struktur FIFO queue yang menyimpan kedatangan pages di memori.
  - Masalah: menambah page frame => page fault tidak berkurang.

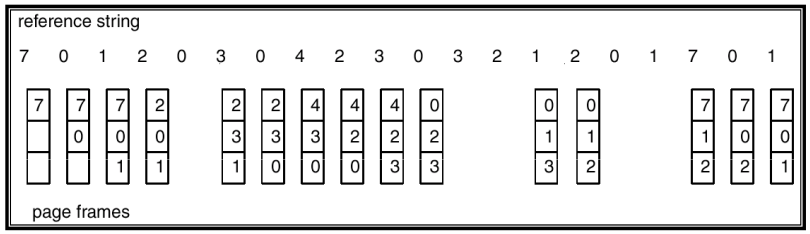
Algoritma FIFO

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 page yang dapat berada di memori pada suatu waktu per proses)

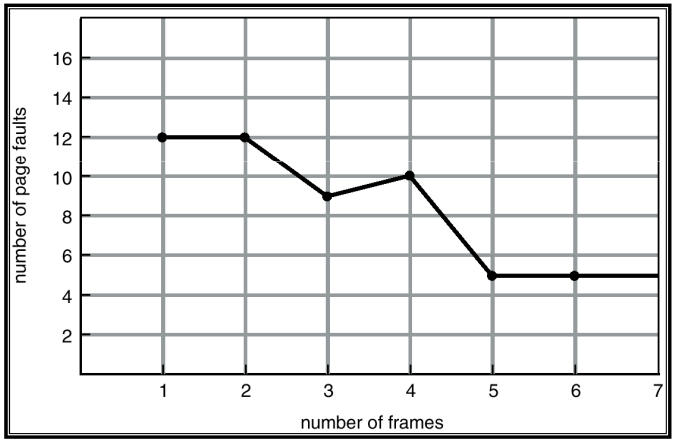
- 4 frames



FIFO Page Replacement



Ilustrasi Anomali Belady pada FIFO



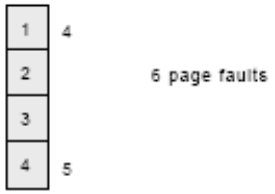
Optimal (Prediction)

- OPT (optimal)
  - Mengganti page yang tidak digunakan dalam waktu dekat (paling lama tidak diakses).
  - Menggunakan priority lists page mana yang tidak akan diakses (“in the near future”).
  - Sulit diterapkan (prediksi): terbaik dan “benchmark” untuk algoritma yang lain.

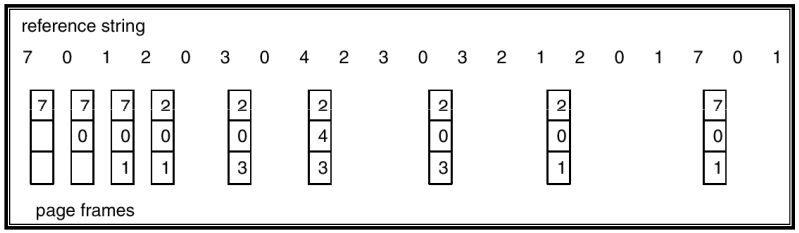
Algoritma Optimal

- Algoritma Optimal
  - Mengganti page yang tidak digunakan untuk periode waktu yang lama.
  - Contoh 4 frame

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



Optimal Page Replacement

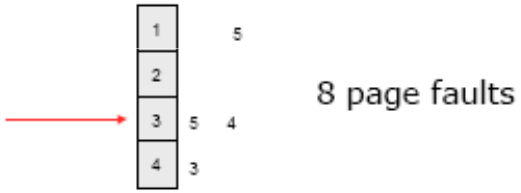


Least Recently Used

- LRU (least recently used)
  - Mengganti page yang paling lama tidak digunakan/diakses.
  - Asumsi page yang diakses sekarang => kemungkinan besar akan diakses lagi (predict?).
  - Masalah: mendeteksi (memelihara) LRU semua page => bantuan hardware yang cukup rumit.

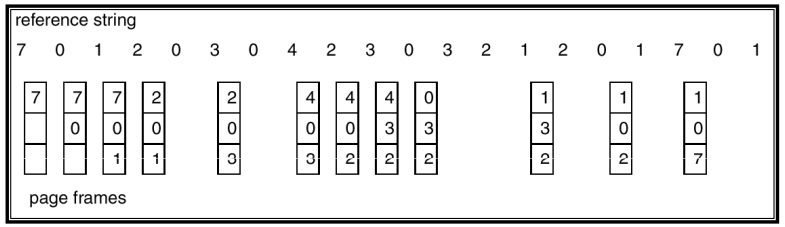
Algoritma LRU

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5



8 page faults

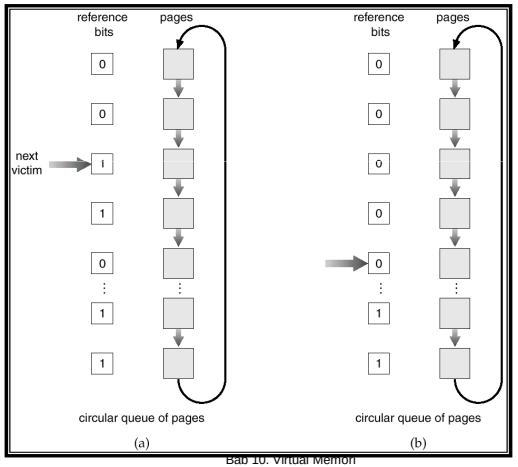
LRU Page Replacement



Algoritma Aproksimasi LRU

- Reference bit
  - Setiap page berasosiasi dengan satu bit, inisialisasinya = 0
  - Ketika page dengan reference bit di set 1
  - Ganti satu dengan 0 (jika ada satu)
- Second chance
  - Membutuhkan reference bit.
  - Jika page diganti (pada urutan clock) dengan reference bit = 1, maka
    - set reference bit 0.
    - Tinggalkan page di memori (berikan kesempatan kedua).
    - Ganti next page (dalam urutan clock) , subjek disamakan aturannya.

Algoritma Second-Chance (clock) Page-Replacement Algorithm



Alokasi Frame

- Setiap proses membutuhkan minimum sejumlah pages.
- Contoh : IBM 370 – 6 page untuk menangani instruksi SS MOVE :
  - instruksi 6 bytes, membutuhkan 2 pages.
  - 2 pages untuk menangani **from**.
  - 2 untuk menangani **to**.
- Dua skema besar alokasi :
  - fixed allocation
  - priority allocation

Fixed Allocation

- Equal allocation – contoh jika 100 frame dan 5 proses, masing-masing 20 page.
- Proportional allocation – mengalokasikan sesuai ukuran yang cocok dari proses

Priority Allocation

- Menggunakan skema alokasi yang proporsional dengan mengedepankan menggunakan prioritas dibandingkan ukuran.
- Jika proses P_i di-generate sebagai page fault,
  - Pilih satu replacement frame
  - Pilih replacement frame dari proses dengan prioritas terendah.

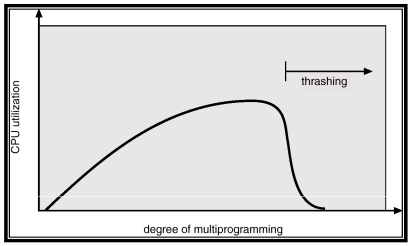
Alokasi Global vs. Local

- **Global** replacement – mengijinkan suatu proses untuk menyeleksi suatu frame yang akan fireplace dari sejumlah frame.
- **Local** replacement –proses hanya diijinkan menyeleksi frame-frame yang dialokasikan untuknya.

Thrashing

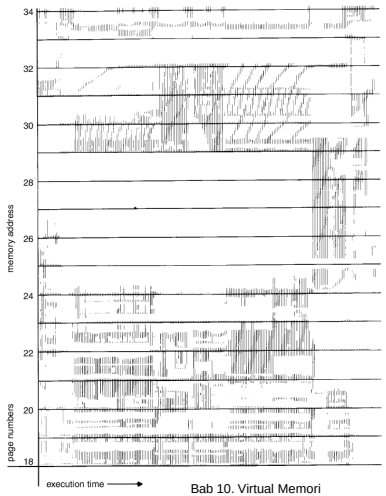
- Jika suatu proses tidak mempunyai page yang cukup, tingkat page fault menjadi tinggi If a process does not have “enough” pages, the page-fault rate is very high. Hal tersebut dapat dilihat dari :
  - Sistem operasi meningkatkan multiprogramming.
  - Utilisasi CPU meningkat sejalan dengan bertambahnya multiprogramming
  - Proses lain ditambahkan ke dalam sistem.
- **Thrashing** ≡ suatu proses yang sibuk melakukan swap page in dan out.

Thrashing



- Mengapa paging dapat bekerja ?
  Model Lokalitas
  - Proses pemindahan dari satu lokasi ke lokasi lain.
  - Terjadi overlap lokalitas.
- Mengapa thrashing terjadi ?
  Σ ukuran lokalitas > total ukuran memory

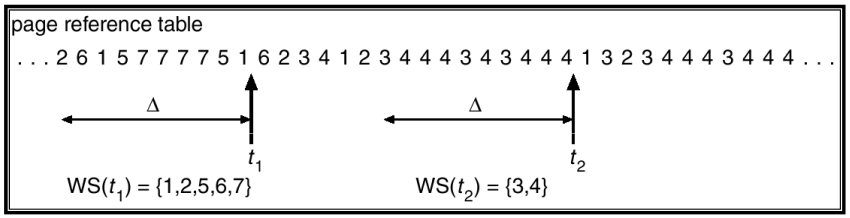
Lokalitas pada Pola Memory-Reference Pattern



Working-Set Model

- $\Delta \equiv$ jendela working-set $\equiv$ fixed number pada page references
  Contoh : 10,000 instruksi
- WSS_i (working set pada proses P_i) =
  jumlah page reference pada saat akhir Δ (beragam waktu)
  - jika Δ terlalu kecil akan mencakup seluruh lokalitas
  - jika Δ terlalu besar akan mencakup sebagian lokalitas.
  - jika $\Delta = \infty \Rightarrow$ akan mencakup seluruh program
- $D = \Sigma\ WSS_i \equiv$ total permintaan frames
- if $D > m \Rightarrow$ Thrashing
- Kebijakan, jika $D >$ m, maka menahan satu proses .

Working-set model



Pertimbangan Lain

- Prepaging
- Page size selection
  - fragmentation
  - table size
  - I/O overhead
  - locality