MANAJEMEN MEMORI DENGAN METODE SWAPPING

By JKusnendar

MANAJEMEN MEMORI DENGAN SWAPPING

- **Pengertian :**

  Suatu metode pengalihan proses yang bersifat sementara dari memori utama ke suatu tempat penyimpanan sementara (*disk*) dan dipanggil kembali ke memori jika akan melakukan ekseskusi.

  Adapun proses yang dipindahkan yaitu proses yang di-*blocked* ke *disk* dan hanya memasukkan proses-proses *ready* ke memori utama.

- **Tujuan Swapping**

  Meningkatkan kinerja saat multiprogramming pada sistem time sharing

Masalah yang harus diatasi multiprogramming dengan swapping :

- Pemartisian secara dinamis
- Strategi pengelolaan memori bebas
- Algoritma penempatan proses ke memori
- Strategi penempatan ruang swap pada disk

A. Multiprogramming dengan Pemartisian Dinamis

Memori dipartisi menjadi bagian-bagian dengan jumlah dan besar tidak tentu ketika suatu proses telah masuk ke memori utama.

a. Kelebihan

Meningkatkan utilisasi memori

b. Kelemahan

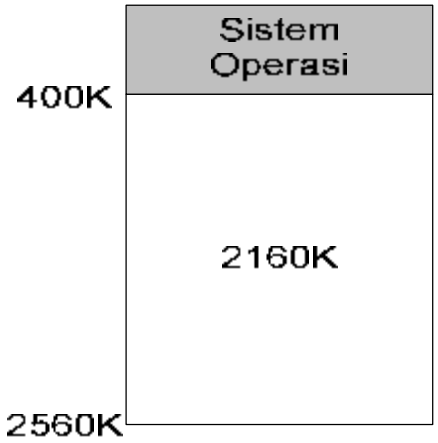
- Merumitkan alokasi dan dealokasi memori, juga dalam melacak alokasi memori.
- Dapat terjadi lubang-lubang kecil memori antara partisi-partisi yang dipakai.

? **Masalah lain yang timbul dari partisi dinamis** yaitu adanya segmen data proses yang berkembang sebagi akibat adanya *heap* atau *stack* yang memanggil prosedur variabel-variabel lokal

ü **Solusi dari masalah tersebut :**

a. Proses dipindah ke lubang memori yang cukup dapat memuatnya.
b. Satu proses atau lebih di-*swap* ke *disk* agar memberi lubang cukup besar untuk proses yang berkembang.
c. Jika proses tidak dapat tumbuh di memori dan daerah *swap* di *disk* telah penuh, proses harus menunggu atau disingkirkan.

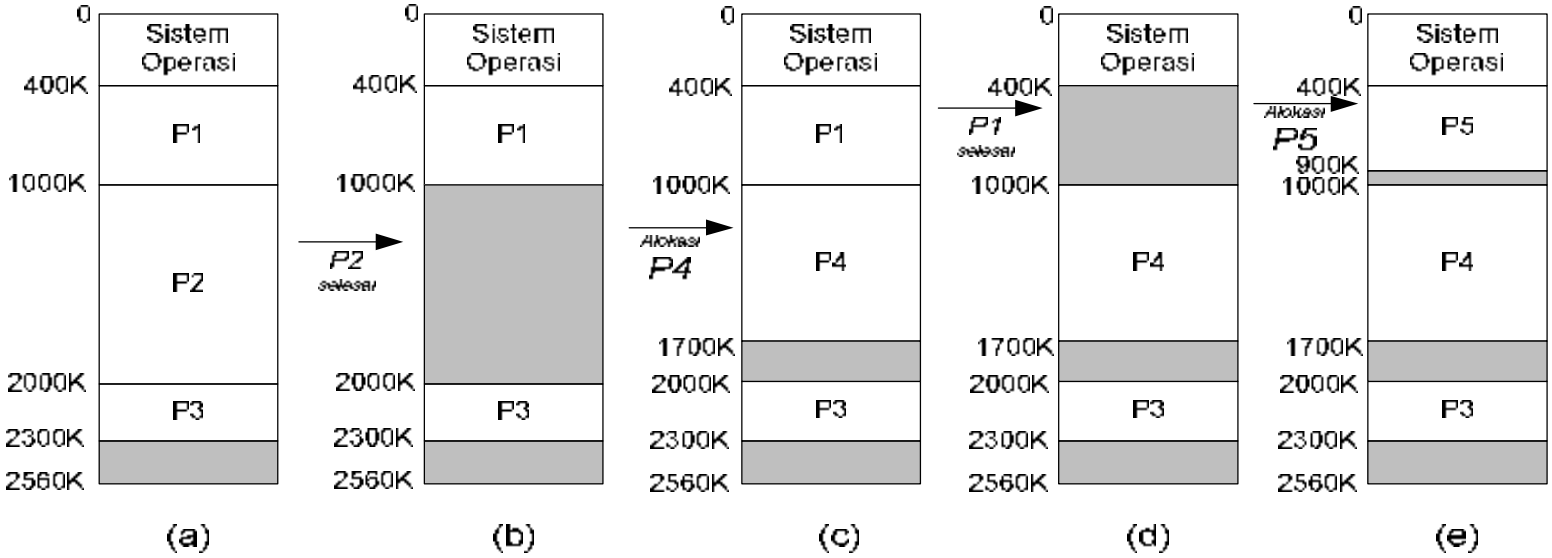
Contoh :



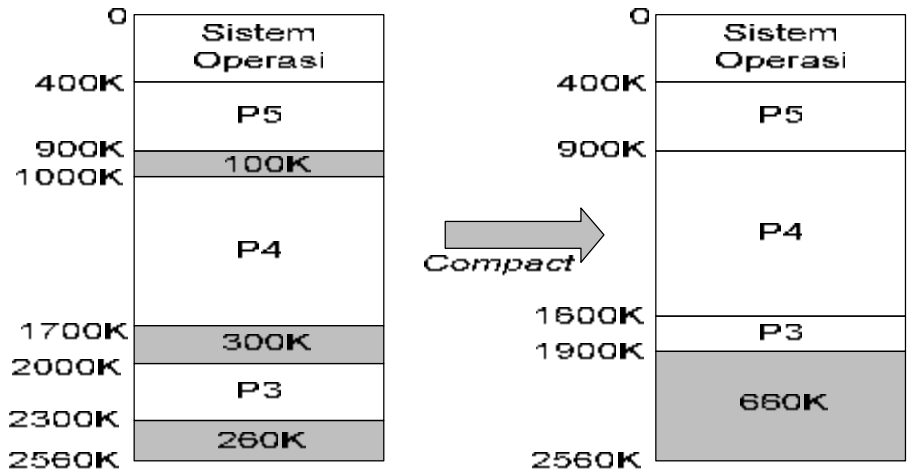
Job Queue		
Proses	Memori	Waktu
P1	600K	10
P2	1000K	5
P3	300K	20
P4	700K	8
P5	500K	15

Keterangan :
= Proses
= Lubang

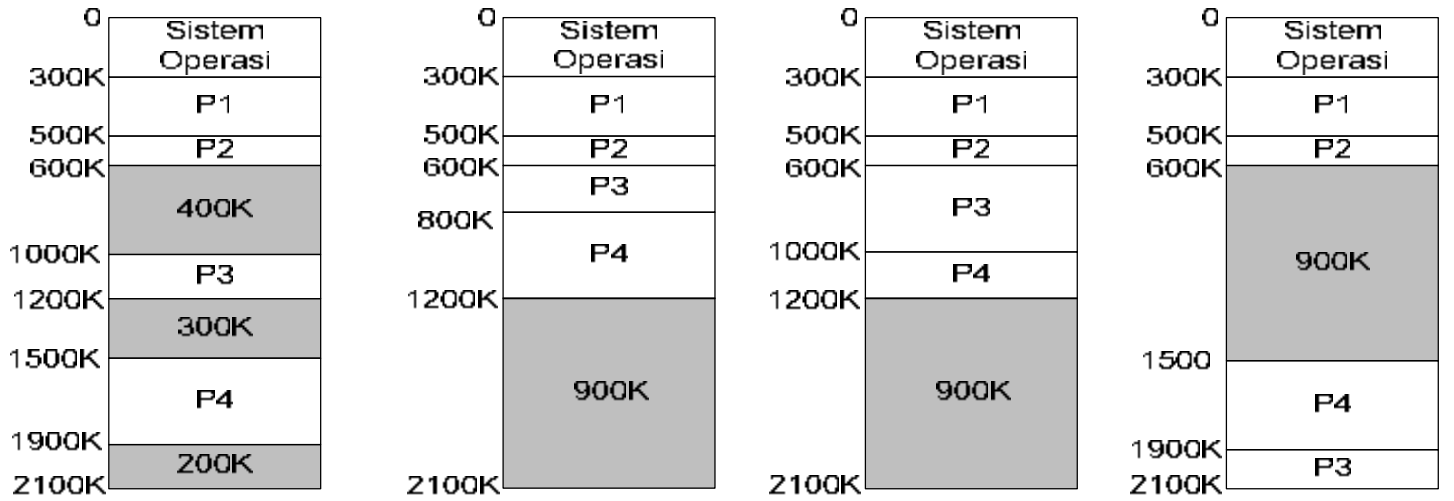
Gambar 1 Contoh Penjadwalan



Gambar 2 Alokasi memori dan *long-term scheduling*



Gambar 3 *Compaction*



Gambar 2 Perbandingan beberapa pemadatan memori

B. Pengelolaan Pemakaian Memori

Memori yang tersedia harus dikelola. Untuk itu, harus dilakukan pencatatan pemakaian memori yang bertujuan agar dapat diketahui lokasi-lokasi mana saja di memori utama yang masih kosong dan yang sudah terisi.

Terdapat 3 (tiga) cara utama pencatatan memori, yaitu :

1. Pengelolaan memaki peta bit
2. Pengelolaan memakai senarai berkait
3. Pengelolaan memakai sistem *buddy*

1. Pengelolaan Memakai Peta Bit

Dengan metode ini, memori dibagi menjadi beberapa alokasi unit, dimana tiap-tiap unit bisa terdiri dari bebrapa word atu bahkan beberapa kilobyte. Tiap-tiap unit berhubungan dengan 1 bit, yaitu akan berisi bit 0 jika unit tersebut kosong dan berisi 1 jika unit tersebut telah terisi.

Ukuran dari unit ini amatlah penting. Alokasi unit yang lebih kecil akan menyebabkan besarnya bit map. Oleh karena itu, jika suatu alokasi unit yang berukuran 4-byte untuk 32n-bit, memori hanya akan membutuhkan *n* pemetaan bit. Jika ukuran alokasi unit dibuat lebih besar, maka bit map yang dibutuhkan akan lebih kecil, namun demikian bentuk ini akan menjadi lebih buruk jika ukuran proses bukan merupakan kelipatan dari ukuran alokasi unit.

2. Pengelolaan Memakai Senarai Berkait (*Linked List*)

Setiap node list terdiri atas : informasi yang menyatakan adanya proses (P) dan hole (H), lokasi awal dan panjang lokasi.

- **Keunggulan :**

a. Memori yang dipergunakan pad metode ini lebih kecil jika dibandingkan dengan peta bit
b. Tidak perlu melakukan perhitungan blok lubang memori yang sudah tercatat di node

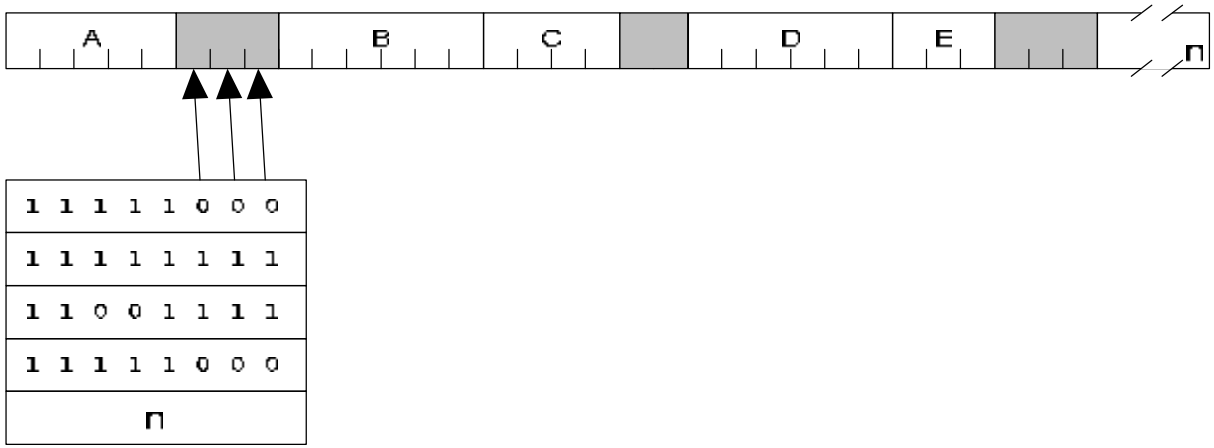
- **Kelemahan**

  Dealokasi sulit dilaksanakan, mengingat akan terjadinya penggabungan antara beberapa mode.

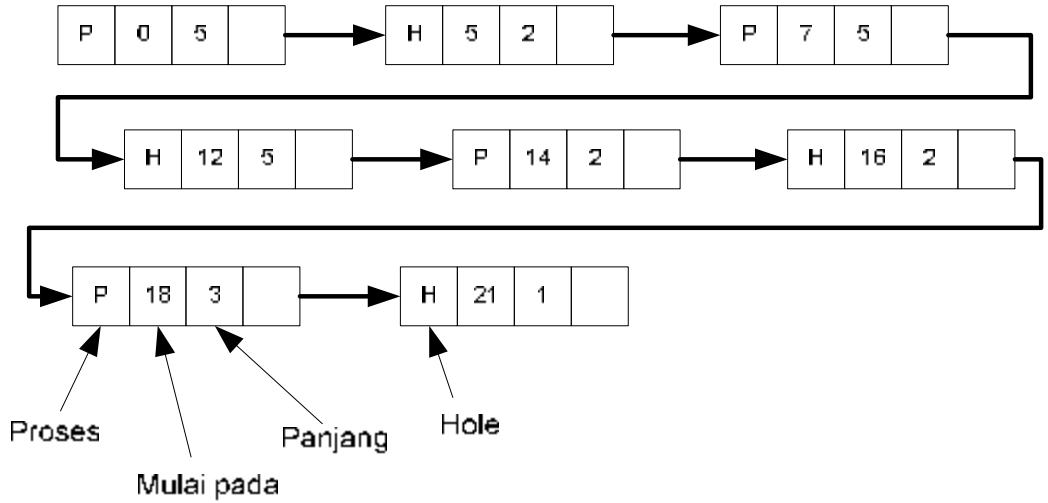
C. Algoritma Penempatan Proses ke Memori

Terdapat beragam strategi pencarian tempat kosong di memori dengan ukuran yang mencukupi untuk memuat proses yaitu lubang kkosong yang sama atau lebih besar dari yang diperlukan proses. Beragam algoritma antara lain :

a. *Fisrt-fit algorithm*
b. *Next-fit algorithm*
c. *Best-fit algorithm*
d. *Worst-fit algorithm*
e. *Quick-fit algorithm*



Gambar 5 Peta bit untuk pengelolaan pemakaian memori



Gambar 6 Pengelolaan pemakaian memori dengan senarai berkait

a. ***First-Fit Algorithm***

Manejer memori men-scan senarai segmen dari awal sampai menemukan lubang besar yang mencukupi untuk penempatan proses. Lubang kemudian dibagi dua, yaitu untuk proses dan lubang tak digunakan, kecuali ketika lubang tersebut tepat sama dengan yang diperlukan proses.

b. ***Next-Fit Algorithm***

Mekanisme ini hampir sama dengan mekanisme *first-fit*, hanya mekanisme ini dimulai dari senarai yang terakhir kali menemukan segmen yang cocok

c. ***Best-Fit Algorithm***

Mekanisme ini mencari seluruh senarai dan mengambil lubang terkecil yang mungkin ditempati proses. Bukan memecah lubang besar yang mungkin diperlukan, *best-fit* mencoba menemukan lubang yang mendekati ukuran yang diperlukan.

Mekanisme ini lebih lambat dibandingkan *first-fit* karena selalu mencari di seluruh senarai setiap kali dipanggil. *Best-fit* juga ternyata bisa menghasilkan memori tersisa banyak dibanding *first-fit* atau *next-fit*, karena keduanya selalu mengisi lubang kecil yang tak digunakan

d. Worst-Fit Algorithm

Pencarian dimulai dari awal dan akanberhenti jika ditemukan lokasi yang paling besar yang cukup untuk menempatkan ruang tersebut

e. Quick-Fit Algorithm

Algoritma mengelola sejumlah senarai lubang memori dengan beragam ukuran lubang. Misalnya terdapat senarai 8 Kb, 12 Kb, 20 Kb, 40 Kb, 60 Kb, dan seterusnya. Senarai-senarai ini mencatat lubang-lubang memori sesuai dengan jumlah terdekatnya, misalnya lubang memori 42 dimuat pada senarai 40 Kb.

Dengan beragam senarai maka alokasi dapat dilakukan dengan cepat, yaitu tinggal mencari senarai terkecil yang dapat menampung proses tersebut.

- **Keunggulan** *:*

  Sangat cepat dalam alokasi proses

- **Kelemahan**

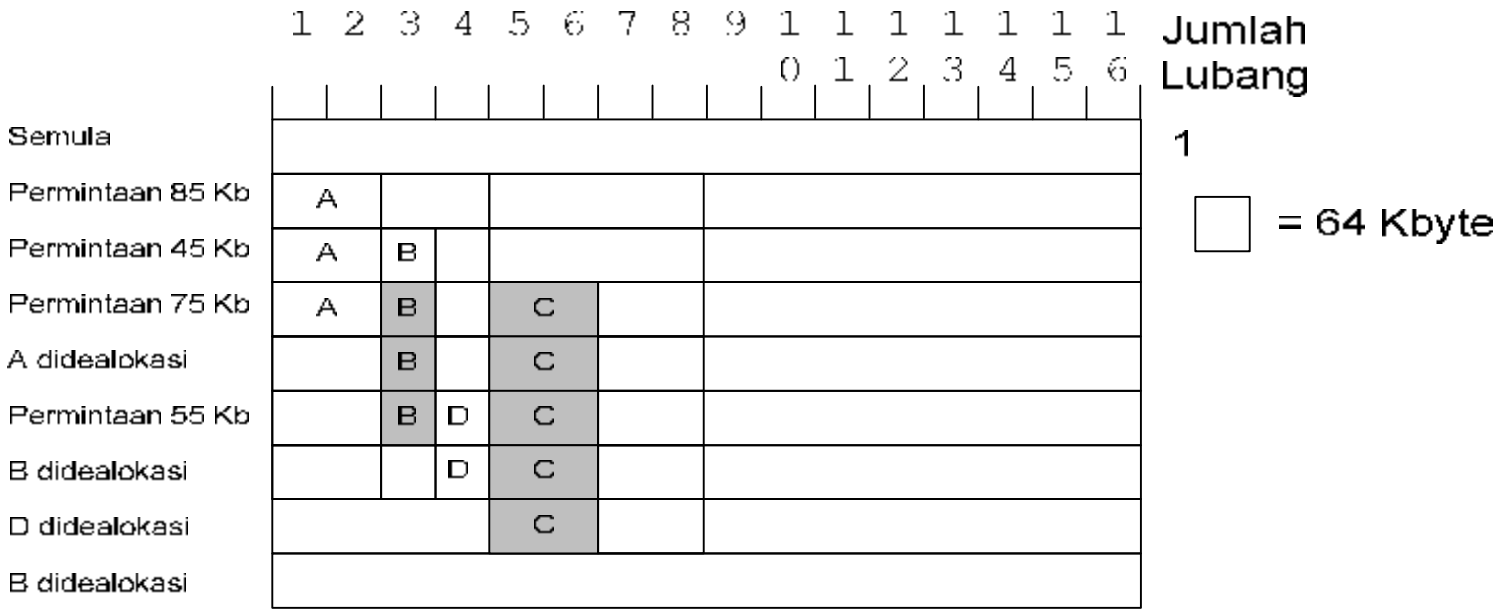
  Ketika proses berakhir atau dipindah keluar (*swap_out*) maka menemukan tetangga-tetangga memori yang dipakai proses agar dapat dilakukan penggabungan adalah sangat mahal. Jika penggabungan tidak dilakukan, memori akan segera menjadi banyak lubang kecil yang tak berguna

3. Sistem *Buddy*

Sistem *buddy* adalah algoritma pengelolaan yang memanfaatkan keunggulan penggunaan bilangan bineruntuk pengalamatan.

Contoh :

Manajer memori mengelola senarai blok-blok bebas 1, 2, 4, 8, 16 byte dan seterusnya, sampai kapasitas memori. Pada komputer dengan 1 Mb memori maka terdapat 21 senarai yaitu dari 1 *byte* sampai 1 Mb. Perhatikan Gambar berikut :



Gambar 7 Pengelolaan memori dengan Sistem Buddy

- ***Keunggulan :***

  Ketika blok berukuran 2^k dibebaskan, maka manajer hanya mencari pada senarai lubang 2^k untuk memeriksa apakah terdapat penggabungan yang dimungkinkan.

- **Kelemahan**

  Utilisasi memori tidak efisien, karena semua permintaan dibulatkan ke 2^k terdekat yang dapat memuat. Misalkan, proses berukuran 35 Kb harus dialokasikan 64 Kb, terdapat 29 Kb yang disiakan, sehingga terjadi *overhead*. Bentuk overhead ini disebut fragmentasi internal karena memori yang disiakan adalah internal terhadap segmen-segmen yang dialokasikan

D. Alokasi Ruang Swap pada *Disk*

Terdapat dua strategi utama untuk penempatan proses yang dikeluarkan dari memori utama (*swap-out*) ke *disk*, yaitu :

1. Ruang disk tempat dialokasikan begitu diperlukan
   Ketika proses harus diperlukan dari memori utama, ruang *disk* segera dialokasikan sesuai ukuran proses.
2. Ruang disk tempat swap dialokasikan lebih dahulu
   Saat proses diciptakan, ruang *swap* pada *disk* dialokasikan. Sehingga ketika proses harus dikeluarkan dari memori utama, proses selalu ditempatkan ke ruang yang telah dialokasikan.