Praktikum 12

Dead Lock

**POKOK BAHASAN:**

Membuat program simuliasi tentang

* Metode untuk Menangani Deadlock
* Mencegah Deadlock
* Menghindari Deadlock
* Mendeteksi Deadlock
* Perbaikan dari Deadlock

**TUJUAN BELAJAR:**

Setelah mempelajari materi dalam bab ini, mahasiswa diharapkan mampu:

* Memahami latar belakang terjadinya deadlock pada sistem.
* Membuat program yang dapat mesimulasikan mendeteksi deadlock

**DASAR TEORI:**

Permasalahan deadlock terjadi karena sekumpulan proses-proses yang di-blok dimana setiap proses membawa sebuah sumber daya dan menunggu mendapatkan sumber daya yang dibawa oleh proses lain. Misalnya sistem mempunyai 2 tape drive dan terdapat dua proses P1 dan P2 yang masing masing membawa satu tape drive dan masing-masing memerlukan tape drive yang dibawa proses lain sehingga terjadi keadaan saling menunggu resource dan sistem di-blok.

Contoh lain, misalnya terdapat semaphore A dan B yang diinisialisasi 1 dan terdapat dua proses P0 dan P1 masing-masing membawa semaphore A dan B. Kemudian P0 dan P1 meminta semaphore B dan A dengan menjalankan operasi wait. Hal ini mengakibatkan proses di-blok dan terjadi deadlock.

*P0 P1*

*wait (A); wait(B);*

*wait (B); wait(A);*

1. **MENGHINDARI DEADLOCK**

Metode alternatif untuk menghindari deadlock adalah digunakan informasi tambahan tentang bagaimana sumber daya diminta. Misalnya pada sistem dengan satu tape drive dan satu printer, proses P pertama meminta tape drive dan kemudian printer sebelum melepaskan kedua sumber daya tersebut. Sebaliknya proses Q pertama meminta printer kemudian tape drive. Dengan mengetahui urutan permintaan dan pelepasan sumber daya untuk setiap proses, dapat diputuskan bahwa untuk setiap permintaan apakah proses harus menunggu atau tidak. Setiap permintaan ke system harus dipertimbangkan apakah sumber daya tersedia, sumber daya sedang dialokasikan untuk proses dan permintaan kemudian serta pelepasan oleh proses untuk menentukan apakah permintaan dapat dipenuhi atau harus menunggu untuk menghindari deadlock.

Model yang sederhana dan sangat penting dibutuhkan adalah setiap proses menentukan jumlah maksimum sumber daya dari setiap tipe yang mungkin diperlukan. Algoritma deadlock avoidance secara dinamis memeriksa status sumber daya yang dialokasikan untuk menjamin tidak pernah terjadi kondisi menunggu sirkular. Status alokasi sumber daya ditentukan oleh jumlah sumber daya yang tersedia dan yang dialokasikan dan maksimum permintaan oleh proses-proses.

Untuk penghindaran deadlock diperlukan pengertian mengenai state selamat (safe state) dan state tak selamat (unsafe state).

* 1. **State Selamat (Safe State)**

Ketika suatu proses meminta sumber daya yang tersedia, sistem harus menentukan apakah alokasi sumber daya pada proses mengakibatkan sistem dalam state selamat. Sistem dikatakan dalam state selamat jika sistem dapat mengalokasikan sumber daya untuk setiap proses secara berurutan dan menghindari deadlock. Urutan proses <P1, P2, …, Pn> selamat jika untuk setiap Pi, sumber daya yang masih diminta Pi masih memenuhi sumber daya yang tersedia dan sumber daya yang dibawa oleh setiap Pj, dimana j < i. Jika sumber daya yang diperlukan Pi tidak dapat segera disediakan, maka Pi dapat menunggu sampai semua Pj selesai. Ketika Pj selesai, Pi dapan memperoleh sumber daya yang diperlukan, mengeksekusi, mengembalikan sumber daya yang dialokasikan dan terminasi. Ketika Pi selesai, Pi+1 dapat memperoleh sumber daya yang diperlukan dan seterusnya.

Jika sistem dalam state selamat maka tidak terjadi deadlock, sedangkan jika sistem dalam state tidak selamat (unsafe state) maka kemungkinan terjadi deadlock seperti Gambar 12-1. Metode menghindari deadlock menjamin bahwa sistem tidak pernah memasuki state tidak selamat.

****

***Gambar 12-1 : Ruang state selamat, tak selamat dan deadlock***

Untuk menggambarkan sistem dapat berpindah dari state selamat ke state tidak selamat dapat dilihat ilustrasi berikut ini. Misalnya sistem mempunyai 12 magnetic tape drive dan 3 proses P0, P1 dan P2. Proses P0 membutuhkan 10 tape drive, proses P1 membutuhkan 4 dan proses P2 membutuhkan 9 tape drive. Misalnya pada waktu t0, proses P0 membawa 5 tape drive, P1 membawa 2 dan P2 membawa 2 tape drive sehingga terdapat 3 tape drive yang tidak digunakan.

Kebutuhan Maksimum Kebutuhan Sekarang

P0 10 5

P1 4 2

P2 9 2

Pada waktu t0, sistem dalam state selamat. Urutan < P1, P0, P2> memenuhi kondisi selamat karena P1 dapat segera dialokasikan semua tape drive dan kemudian mengembalikan semua tape drive sehingga sistem tersedia 5 tape drive. Kemudian P0 dapat memperoleh semua tape drive dan mengembalikan semua sehingga system tersedia 10 tape drive dan terakhir proses P2 dapat memperoleh semua tape drive dan mengembalikan semua tape drive sehingga system tersedia 12 tape drive.

Sistem dapat berubah dari state selamat ke state tidak selamat. Misalnya pada waktu t1, proses P2 meminta tambahan alokasi 1 tape drive. Sistem menjadi tidak selamat. Pada saat ini, hanya proses P1 yang mendapatkan semua tape drive dan kemudian mengembalikan semua tape drive sehingga hanya tersedia 4 tape drive. Karena proses P0 sudah dialokasikan 5 tape drive tetapi membutuhkan maksimum 10 tape drive sehingga meminta 5 tape drive lagi. Karena tidak tersedia, proses P0 harus menunggu demikian juga P2 sehingga system menjadi deadlock.

* 1. **Algoritma Banker**

Algoritma resource allocation graph tidak dapat diaplikasikan pada sistem yang mempunyai beberapa anggota pada setiap tipe sumber daya. Setiap proses sebelum dieksekusi harus menentukan jumlah sumber daya maksimum yang dibutuhkan. Jika suatu proses meminta sumber daya kemungkinan proses harus menunggu. Jika suatu proses mendapatkan semua sumber daya maka proses harus mengembalikan semua sumber daya dalam jangka waktu tertentu.

Struktur data yang digunakan untuk mengimplementasikan algoritma Banker akan menentukan state dari sumber daya yang dialokasikan oleh sistem. Misalnya n = jumlah proses dan m = jumlah tipe resource. Struktur data yang diperlukan :

• Available : Vektor panjang m. Jika Available[j] = k, terdapat k anggota tipe sumber daya Rj yang tersedia.

• Max : matrik n x m. Jika Max[i, j] = k, maka proses Pi meminta paling banyak k anggota tipe resource Rj.

• Allocation : matrik n x m. Jika Allocation[i, j] = k maka Pi sedang dialokasikan k anggota tipe resource Rj.

• Need : matrik n x m. Jika Need[i, j] = k, maka Pi membutuhkan k anggota tipe resource Rj untuk menyelesaikan task. Need[i, j] = Max[i, j] – Allocation[i, j].

Beberapa notasi yang perlu diketahui adalah misalnya X dan Y adalah vector dengan panjang n. X ≤ Y jika dan hanya jika X[i] ≤ Y[i] untuksemua i = 1, 2, .., n. Sebagai contoh jika X = (1, 7, 3, 2) dan Y = (0, 3, 2, 1) maka Y ≤ X.

* 1. **Algoritma Safety**

Algoritma ini untuk menentukan apakah sistem berada dalam state selamat atau tidak.

1. Work dan Finish adalah vector dengan panjang m dan n. Inisialisasi : Work = Available dan Finish[i] = false untuk i = 1,3, …, n.

2. Cari i yang memenuhi kondisi berikut :

(a) Finish [i] = false

(b) Needi ≤ Work

Jika tidak terdapat i ke langkah 4.

3. Work = Work + Allocationi

Finish[i] = true

Kembali ke langkah 2.

4. Jika Finish [i] == true untuk semua i, maka sistem dalam state selamat.

* 1. **Algoritma Resouce Request**

Requesti adalah vector permintaan untuk proses Pi. Jika Requesti[j] = k, maka proses Pi menginginkan k anggota tipe sumber daya Rj. Jika permintaan untuk sumber daya dilakukan oleh proses Pi berikut ini algoritmanya.

Request = request vector for process Pi. If Requesti [j] = k then process Pi wants k instances of resource type Rj.

1. Jika Requesti ≤ Needi ke langkah 2. Selain itu, terjadi kondisi error karena proses melebihi maksimum klaim.
2. Jika Requesti ≤ Available, ke langkah 3. Selain itu Pi harus menunggu karena sumber daya tidak tersedia.
3. Alokasikan sumber daya untuk Pi dengan modifikasi state berikut :

*Available = Available - Requesti;*

*Allocationi = Allocationi + Requesti;*

*Needi = Needi – Requesti;*

Jika hasil state alokasi sumber daya adalah selamat, maka sumber daya dialokasikan ke Pi , sebaliknya jika tidak selamat, Pi harus menunggu dan state alokasi sumber daya yang lama disimpan kembali.

* 1. **Contoh Penggunaan Algoritma Banker**

Diketahui sistem terdapat 5 proses yaitu P0 sampai P4, 3 tipe sumber daya yaitu A (10 anggota), B (5 anggota) dan C (7 anggota). Perhatikan gambaran sistem pada waktu T0.

Allocation Max Available

A B C A B C A B C

P0 0 1 0 7 5 3 3 3 2

P1 2 0 0 3 2 2

P2 3 0 2 9 0 2

P3 2 1 1 2 2 2

P4 0 0 2 4 3 3

Isi matrik Need didefinisikan dengan Max – Allocation.

Need

A B C

P0 7 4 3

P1 1 2 2

P2 6 0 0

P3 0 1 1

P4 4 3 1

Sistem dalam keadaan state selamat dengan urutan < P1, P3, P4, P2, P0> yang memenuhi kriteria algoritma safety.

Misalnya proses P1 meminta tambahan anggota tipe sumber daya A dan dua anggota tipe sumber daya C sehingga Request1 = (1, 0, 2). Untuk menentukan apakah permintaan dapat segera dipenuhi, pertama harus diperiksa apakah Request1 ≤ vailable ((1, 0, 2) ≤ (3, 3, 2)) ternyata benar. Maka akan diperoleh state baru berikut :

Allocation Need Available

A B C A B C A B C

P0 0 1 0 7 4 3 2 3 0

P1 3 0 2 0 2 0

P2 3 0 1 6 0 0

P3 2 1 1 0 1 1

P4 0 0 2 4 3 1

Kemudian harus ditentukan apakah sistem berada dalam state selamat. Setelah mengeksekusi algoritma safety ternyata urutan <P1, P3, P4, P0, P2> memenuhi criteria safety.

Setelah sistem berada pada state doatas, permintaan (3, 3, 0) oleh P4 tidak dapat dipenuhi karena sumber daya tidak tersedia. Permintaan (0, 2, 0) oleh P1 juga tidak dapat dipenuhi karena meskipun sumber daya tersedia, state hasil tak selamat.

TUGAS PENDAHULUAN:

1. Apa yang dimaksud dengan sumber daya ? Berikan contohnya.

2. Apa yang dimaksud deadlock ?

3. Sebutkan 4 kondisi yang menyebabkan deadlock.

4. Sebutkan cara mencegah deadlock dari 4 kondisi tersebut pada soal 3.

5. Diketahui snapshot dari suatu sistem :

Allocation Max Available

A B C D A B C D A B C D

P0 0 0 1 2 0 0 1 2 1 5 2 0

P1 1 0 0 0 1 7 5 0

P2 1 3 5 4 2 3 5 6

P3 1 6 3 2 1 6 5 2

P4 0 0 1 4 0 6 5 6

Jawablah pertanyaan berikut :

a. Bagaimana isi matrik Need ?

b. Apakah sistem dalam state selamat ?

c. Jika proses P1 meminta (0,4,2,0) dapatkah permintaan dipenuhi segera ?

PERCOBAAN:

1. Bukalah program (boleh menggunakan bahasa apa saja pemrograman yang sudah di pelajari)
2. Buatlah program untuk mengimplementasikan permasalahan Deadlock yang ada pada Tugas Pendahuluan no 7

LAPORAN RESMI:

1. Analisa hasil percobaan yang Anda lakukan.
2. Buatlah Program di atas.
3. Berikan kesimpulan dari praktikum ini.