Pencegahan *Deadlock* pada Alokasi *Resource* dalam Sistem Operasi Menggunakan Algoritma Greedy

Timotius T. Safei (13509017)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509017@std.stei.itb.ac.id

***Abstract*—Sistem operasi menangani berbagai fungsi, salah satunya pengalokasian sumber daya. Pada pengalokasian sumber daya dalam sistem operasi sangat mungkin terjadi masalah. Salah satu masalah tersebut adalah *deadlock*. Karena itu diperlukan cara-cara khusus untuk menanganinya. Selain penangan, dapat juga dilakukan pencegahan. Oleh karena itu dalam makalah ini akan dibahas mengenai pencegahan *deadlock* menggunakan algoritma greedy.**

***Index Terms*—deadlock, sistem operasi, pencegahan, greedy.**

I. PENDAHULUAN

Sistem operasi adalah suatu *software* yang berguna untuk menjembatani *user* dan *hardware* atau komputer itu sendiri. Sistem operasi memiliki banyak tugas dan fungsi. Contoh tugas dari sistem operasi adalah mengatur jadwal proses, mengatur sumber daya, dan juga menjadi penerjemah antara bahasa tingkat dasar menjadi bahasa yang lebih mudah digunakan.

Sstem operasi juga memilik fungsi mengatur penggunaan *hardware* yang ada, baik dari prosesor, memory, perangkat input/output dan banyak hal lainnya. Penggunaan perangkat tersebut harus diatur karena sumber daya. Baik sumber daya waktu, energi dan sumberdaya lainnya.

Dalam melakukan pengaturan tersebut tentu diperlukan mekanisme khusus untuk mencegah terjadinya masalah. Masalah yang mungkin mucul sangat beragam di setiap bagiannya. Masalah tersebut tetntu saja perlu bermacam-macam cara penanganannya.

Bagian yang akan dibahas pada makalah ini adalah bagian pembagian sumber daya. Sumber daya perlu diatur karena sumber daya tersebut terbatas, sedangkan proses yang membutuhkannya lebih banyak. Tentu saja hal tersebut menimbulkna masalah.

Secara khusus, masalah yang akan dibahasa adalah mengenai *deadlock*. Masalah *deadlock* muncul ketika sumber daya tertentu diperlukan oleh 2 proses atau lebih dalam periode yang sama.

Penyelesaian masalah tersebut dapat dibagi menjadi dua bagian, yaitu perbaikan dan pencegahan. Perbaikan dilakukan apabila *deadlock* sudah terjadi. Metode ini memiliki beberapa contoh seperti algoritma ostrich, roll back, dan killing proses.

Selain itu, permasalahan *deadlock* bisa dihindarkan dengan cara dicegah. Kita bisa membuat mekanisme sedemikian hingga kejadian *deadlock* tidak akan terjadi.

II. TEORI DASAR

A. Alokasi Sumber Daya

Sistem operasi mengatur pembagian sumber daya untuk setiap proses yang akana diproses. Setiap proses tersebut membutuhkan sumber daya yang berbeda dengan jumlah yang berbeda. Sumber daya yang ada tentu saja terbatas dan perlu dikelola.

Satu sumber daya hanya mungkin digunakan untuk satu proses untuk satu periode. Oleh karena itu apabila satu sumber daya sudah diberikan kepada suatu proses maka proses lain harus menunggu sumber daya tersebut dilepaskan baru kemudian dapat digunakan lagi.

Pada pengalokasian terdapat dua jenis sumber daya, preemptive dan non-preemptive. Suber daya preemptive dapat diambil dari proses yang sedang berjalan. Proses tersebut akan masuk ke status tunggu dan menyerahkan sumber daya yang ia miliki.

Pada sumber daya non-preemptive, kita harus menunggu proses tersebut sampai benar-benar seslesai untuk dapat menggunakan sumber daya tersebut. Oleh karena itu, pengalokasian sumber daya harus dilakukan dengan hati-hati.

B. Deadlock

Deadlock adalah suatu keadaan yang mengakibatkan tidak ada proses yang dapat diselesaikan sampai tuntas karena adanya saliong tunggu antar proses. Saling tunggu dapat dikarenakan prasyarat dari setiap proses adalah proses lain, ataupun saling menunggu sumber daya yang sedang digunakan proses lain..

Ada 4 penyebab *deadlock*:

a. Mutual Exclution
   Keadaan di mana setiap sumber daya hanya bisa digunakan untuk satu proses sajapada satu periode

tertentu.

b. Hold and Wait
   Suatu keadaan di mana proses dapat masuk ke dalam status hold dan menunggu *resource* lain yang sedang digunakan proses lain.
c. Non-preemptable
   Suatu sumber daya tidak bisa diambil setiap saat dari suatu proses. Sumber daya hanya dapat diambil apabila proses tersebut telas selesai digunakan.
d. Circular Wait
   Keadaan dua proses saling menunggu secara circular karena proses saling menunggu sumber daya.

Deadlock dapat diatasi dengan beberapa cara sebagai berikut:

a. Detect & Recovery
   Membiarkan *deadlock* tersebut sempat terjadi, baru kemudian melakukan perbaikan. Contohnya: roll-back, killing process.
b. Ostrich Algorithm
   Kondisi *deadlock* diabaikan karena sangat jarang terjadi.
c. Prevention
   Pencegahan *deadlock* terjadi dengan cara menghilangkan salah satu dari empat penyebab *deadlock* yang ada.
d. Dynamic Avoidance
   Menghindari terjadinya *deadlock*. Salah satunya dengan cara pengalokasian sumber daya dilakukan dengan hati-hati.

Selain keadaan *deadlock*, ada juga keadaan safe dan unsafe. Keadaan safe adalah keadaan di mana sumber daya yang ada apabila dialokasikan ke proses manapun tidak akan menyebabkan *deadlock*. Dengan kata lain, bagaimanapun pengalokasian sumber daya, semua proses tetap dapat diselesaikan.

Keadaan unsafe adalah keadaan yang tidak menjamin semua proses akan selesai. Bahkan sangat besar kemungkinannya untuk terjadi *deadlock*.

C. Algoritma Greedy

Algoritma greedy adalah algoritma yang mengambil satu langkah selanjutnya berdasarkan kondisi yang paling menguntungkan. Algoritma ini tidak memperhitungkan langkah-langkah selanjutnya. Pengambilan langkah hanya berdasarkan fungsi yang sudah dibuat untuk menentukan langkah yang paling menguntungkan pada saat itu.

Algoritma greeedy terdiri dari 5 elemen.

a. Himpunan Kandidat, C
   Kumpulan dari semua kemungkinan pilihan yang dapat diambil.
b. Himpunan Solusi, S
   Kumpulan dari pilihan yang diambil
c. Fungsi Seleksi
   Fungsi yang menentukan pilihan mana yang paling menguntungkan pada saat itu.
d. Fungsi Kelayakan
   Fungsi yang menentukan apakah pilihan yang ditawarkan feasible.
e. Fungsi Obyektif
   Fungsi yang menentukan tujuan akhir atau keadaan optimum global.

III. PERMASALAHAN

Dalam mengalokasikan sumber daya, perlu diperhatikan jumlah sumber daya yang akan diberikan. Selain itu perlu diperhatikan juga proses yang mana yang harus diberikan sumber daya. Apabila sumber daya diberikan ke sembarang proses dalam keadaan tertentu maka akan terjadi *deadlock*.

Pada makalah ini, permasalah yang akan diangkat adalah bagaimana mengalokasikan sumber daya agar tidak terjadi *deadlock*. Hal ini termasuk dalam prevention. Kita mencegah agar *deadlock* tidak akan muncul. Dalam masalah ini, asumsi yang digunakan adalah:

a. Sumberdaya bersifat non-preamptable
b. Proses dan sumber daya dianggap homogen

Pemasalah yang ada bukanlah permasalahan secara menyeluruh, tetapi merupakan model dari permasalahan yang ada.

Sebagai contoh berikut adalah tabel alokasi sumber daya

Tabel I

	Alokasi	*Kebutuhan*
Proses 1	3	5
Proses 2	2	4

Sisa sumber daya: 1.

Keadaan di atas belum merupakan *deadlock*, tetapi merupakan keadaan unsafe. Dapat dilihat bahwa dalam pengalokasian selanjutnya, akan terjdai *deadlock*. Tujuan dari penggunaan algoritma greedy adalah mencegah keadaan tersebut terjadi sehingga dapat menghindarai *deadlock*.

Berikut ini contoh kedua

Table II

	Alokasi	*Kebutuhan*
Proses 1	2	5
Proses 2	2	4

Sisa sumber daya: 2.

Keadaan pada contoh kedua sistem dalam keadaan safe karena dengan pengalokasian yang tepat, maka semua proses pasti dapat diselesaikan. Akan tetapi jika kita salah mengalokasikan sumber daya yang ada maka dapat terjadi

deadlock. Apabila kita mengalokasikan sisa sumber daya tersebut pada proses 1 maka akna terjadi keadaan seperti pada contoh satu dan jika dilanjutkan lagi akan terjadi *deadlock*.

Dapat dilihat dari contoh bahwa perlu cara yang tepat untuk menangani pembagian sumber daya agar tidak terjadi *deadlock*. Bahkan walaupun dalam keadaan safe sekalipun.

IV. IMPLEMENTASI

Pada pencegahan masalah ini algoritma greedy digunakan untuk mencari solusi bagaimana cara alokasi yang tidak akan mengakibatkan permasalahan di atas. Apabila kita lihat lebih jauh, kita dapat menemukan elemen greedy dari permasalahan tersebut.

Himpunan kandidat dari permasalahn di atas adalah semua kemungkinan alokasi sumber daya kepada setiap proses. Himpunan kandidat dari contoh pertama pada permasalahn adalah sumber daya dialokasikan ke proses 1 atau ke proses 2.

Himpunan solusinya adalah bentuk alokasi yang diambil setelah mengeksekusi fungsi seleksi terhadap himpunan kandidat yang ada.

Fungsi seleksi yang akan digunakan adalah fungsi yang mencari proses yang membutuhkan alokasi sumber daya paling sedikit. Dengan begitu, kita dapat menyelesaikan proses sebanyak-banyaknya dengan sumber daya yang ada. Dengan demikian diharapkan proses yang harus menunggu sumber daya berjumlah minimal.

Fungsi kelayakan yang akan digunakan adalah melihat apakah dengan alokasi yang dilakukan akan ada proses baru yang dapat terselesaikan. Fungsi akan melihat apabila kandidat yang dipilih dijalankan, apakah akan ada proses yang selesai. Apabila tidak ada proses yang selesai maka alokasi akan ditunda. Hal ini digunakan untuk mencegah *deadlock* karena kekurangan sumber daya.

Fungsi obyektif dari algoritma ini untuk menyelesaikan proses sebanyak-banyaknya. Dengan begitu, diharapkan semua proses dapat selesai tanpa terjadi *deadlock*.

V. ANALISIS

Algoritma greedy yang dibuat digunakan dari awal alokasi sumber daya, yaitu saat semua alokasi pada proses bernilai 0. Agar semua proses pasti dapat terselesaikan, maka jumlah sumber daya minimal yang dimiliki harus lebih besar sama dengan kebutuhan terbesar dari salah satu proses.

Dalam penerapannya, algoritma ini mungkin saja mendapat proses baru yang harus dijalankan. Hal ini tidak menjadi masalah karena proses baru tersebut dapat langsung dimasukkan ke dalam himpunan kandidat.

Pengeksekusian algoritma greedu ini dilakukan setiap kali ada perubahan sumber daya yang tersedia. Jadi setiap ada penambahan atau pengurangan sumber daya, alokasi akan coba dilakukan menggunakan algoritma greedy. Selain itu, algoritma akan dijalankan apabila ada proses baru yang masuk. Jadi setiap ada penambahan proses maka akan dicoba pengalokasian sumber daya. Tentu saja pengalokasian dilakukan dengan algoritma greedy yang sudah dibuat.

Sebagai bahan analisis berikut ini diberikan contoh tabel alokasi

Tabel III

	Alokasi	*Kebutuhan*
P1	0	5
P2	0	10
P3	0	6

Jumlah sumber daya yang tersedia 10.
Total sumber daya : 10.

Dari tabel III algoritma greedy yang akan mencoba mengalokasikan sumber daya yang ada ke salah satu proses. Sesuai dengan fungsi seleksi yang ada, maka kandidat yang akan diajukan adalah P1.

Setelah itu, fungsi kelayakan akan mengecek apakah kandidat yang ada sudah layak. Apabila alokasi dilakukan maka P1 dapat diselesaikan, berarti kandidat tersebut akan dijalankan dan dimasukkan ke dalam fungsi solusi.

Tabel IV

	Alokasi	*Kebutuhan*
P1	5	5
P2	0	10
P3	0	6

Jumlah sumber daya yang tersedia 5.
Total sumber daya : 10.

Dilihat dari tabel IV, setelah alokasi, maka algoritma dijalankan kembali. Fungsi seleksi akan menghasilkan kandidat P3. Akan tetapi P3 tidak memenuhi fungsi kelayakan, karena meskipun diberikan semua sumber daya yang ada, P3 tetap tidak dapat diselesaikan. Oleh karena itu P3 tidak akan dimasukkan ke dalam himpunan solusi.

Setelah P1 selesai maka pengalokasian akan dijalankan kembali. P3 menjadi kandidat untuk dialokasikan. Karena dengan pengalokasian P3 dapat diselesaikan maka P3 akan dimasukkan ke dalam himpunan solusi dan alokasi akan dijalankan. Setelah itu, algoritma akan dijalankan lagi karena terjadi perubahan jumlah sumber daya yang tersedia. Akan tetapi karena P2 tidak memenuhi fungsi kelayakan maka tidak akan dimasukkan ke dalam himpunan solusi dan alokasi tidak akan dilakukan.

Apabila ada proses baru yang masuk, maka proses tersebut aka dimasukkan ke dalam himpunan kandidat seperti yang dapat terlihat dalam tabel V

Tabel V

	Alokasi	*Kebutuhan*
P2	0	10
P3	6	6
P4	0	2

Jumlah sumber daya yang tersedia 4.
Total sumber daya : 10.

Karena ada proses baru yang masuk maka pengalokasian akan dicoba sekali lagi. Fungsi seleksi akan memilih P4 karena membutuhkan alokasi paling sedikit, yaitu 2 dan karena P4 memenuhi fungsi kelayakan maka akan dialokasikan 2 sumber daya kepada P4 dan yang tersisa adalah 2 sumber daya. Algoritma greedy akan dijalankan kembali, tetapi karena tidak ada kandidat yang layak maka tidak akan dilakukan pengalokasian.

Pemilihan fungsi kelayakan dikarenakan jika alokasi tetap dilakukan meski tidak menyelesaikan proses, maka sangat mungkin terjadi masalah jika ada proses baru yang masuk. Misalkan pada lanjutan contoh sebelumnya, sisaalokasi akan dimasukkan ke P2 meskipun tidak menyelesaikan proses, tetapi merupakan proses dengan kebutuhan sumber daya paling sedikit.

Masalah yang timbul adalah terbuangnya sumber daya dan waktu. Misalkan setelah pengalokasian, masuk 1 proses baru. Agar lebih jelas dapat dilihat pada tabel VI.

Tabel VI

	Alokasi	*Kebutuhan*
P2	2	10
P3	6	6
P4	2	2
P5	0	4

Jumlah sumber daya yang tersedia 0.
Total sumber daya : 10.

Apabila setelah ini P4 selesai, maka P5 tetap harus menunggu P3 selesai baru dapat diselesaikan. Hal ini mengurangi efisiensi waktu. Oleh karena itu isi dari fungsi kelayakan adalah setiap pengalokasian harus ada proses baru yang dapat diselesaikan.

Akan tetapi penggunaan algoritma greedy ini bukan berarti tanpa masalah. Sebagai contoh lihat tabel berikut

Tabel VII

	Alokasi	*Kebutuhan*
P2	0	10
P3	6	6
P5	4	4
P6	0	3
P7	0	5

Jumlah sumber daya yang tersedia 0.
Total sumber daya : 10.

Pada tabel VII masuk 2 proses baru yang membutuhkan alokasi lebih rendah dari P2. Apabila P3 dan P5 selesai maka algoritma greedy akan mengalokasikan sumber daya kepada P6 dan P7. Kemudian ada proses baru yang masuk. Agar lebih jelas akan digambarkan pada tabel VIII.

Tabel VIII

	Alokasi	*Kebutuhan*
P2	0	10
P6	3	3
P7	5	5
P8	0	9

Jumlah sumber daya yang tersedia 2.
Total sumber daya : 10.

Dapat dilihat dari tabel VIII, P2 memiliki kebutuhan alokasi maksimal. Oleh karena itu, apabila ada proses yang belum diselesaikan dan memiliki kebutuhan lebih kecil P2 tidak akan terproses sampai semua proses selesai. Jadi apabila masih ada proses yang sedang dikerjakan atau ada proses baru terus menerus maka P2 akan membutuhkan waktu yang sangat lama untuk diselesaikan.

Peristiwa ini disebut starving. Starving adalah peristiwa di mana proses yang memiliki prioritas kecil, dalam hal ini memiliki kebutuhan alokasi paling besar, tidak akan terselesaikan karena tidak mendapatkan sumber daya yang dibutuhkan. Peristiwa ini harus ditangani lagi dengan cara khusus. Oleh karena itu, algoritma greedy memberikan hasil optimum dalam persoalan ini. Akan tetapi sangat mungkin menimbulkan masalah lain yang harus dihindari juga.

VI. SIMPULAN

Dalam menjalankan fungsinya, sistemoperasi memiliki beberapa masalah yang dapat muncul. Salah satu masalah dalam pengalokasian sumber daya adalah *deadlock*. *Deadlock* dapat dihindari dengan mengaplikasikan algoritma greedy dalam mengalokasikan sumber daya. Algoritma greey dapat menghindarkan *deadlock* sepenuhnya selama sumber daya yang ada memang mencukupi. Akan tetapi algoritma ini juga masih memiliki kekurangan. Algoritma ini menghilangkan *deadlock*,

tetapi memungkinkan terjadinya starving. Jadi algoritma ini dapat digunakan tetapi masih membutuhkan beberapa perbaikan dan penyesuaian.

REFERENCES

[1] Munir,Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Bandung: ITB, 2009.

[2] Tanenbaum, Andrew S. "Modern Operating System", 2nd ed. Prentice Hall.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2011

ttd

Timotius T. Safei (13509017)