Optimisasi Penjadwalan Proses Pada *Central Processing Unit* Dengan Menggunakan Algoritma *Greedy*

Irdham Mikhail Kenjibriel (13508111)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
irdhamkenjibriel@students.itb.ac.id

***Abstract* - Makalah ini membahas tentang penjadwalan proses pada Central Processing Unit (CPU) dan bagaimana cara melakukan optimisasi yang baik agar CPU dapat berjalan dengan lebih cepat dengan mengaplikasikan algoritma greedy ketika CPU menjadwalkan proses- proses yang dilakukannya. Tujuan dari penjadwalan proses yang dilakukan oleh CPU sendiri adalah memberikan pembagian yang adil terhadap alokasi waktu dan memori bagi setiap proses, memberi prioritas kepada proses yang lebih penting, meningkatkan efisiensi dari penggunan waktu dan memori, mendukung kerja proses yang berat, dan memberikan kemampuan adaptasi dengan beragam lingkungan.**

Secara umum semua penjadwalan proses dilakukan dengan mengeksekusi proses yang pertama kali ada di Central Processing Unit. Namun akhir- akhir ini hal tersebut dirasa kurang memberikan hasil yang optimal. Kali ini penulis akan memberikan pendekatan atau metode lain untuk melakukan proses penjadwalan yaitu dengan algoritma greedy. Algoritma greedy diharapkan dapat menyelesaikan masalah penjadwalan proses dengan lebih baik dibandingkan dengan metode yang sebelumnya.

***Kata Kunci**: penjadwalan, proses, efisien, algoritma greedy, Central Processing Unit.*

I. PENDAHULUAN

Proses- proses yang terjadi pada CPU komputer memerlukan pengaturan akan proses mana saja yang akan dieksekusi terlebih dahulu CPU. Penjadwalan proses pada CPU mempunyai makna melakukanpemilihan terhadap proses selanjutnya yang akan dieksekusi. Kesalahan pada penjadwalan proses pada CPU komputer dapat menyebabkan kekacauan pada proses- proses yang akan dieksekusi atau yang lebih kita kenal dengan istilah *deadlock.* Oleh karena pentingnnya penjadwalan proses ini maka sejak pertama kali computer dibuat para teknisi-teknisi dalam bidang komputer berusaha untuk memodelkan penjadwalan proses sehingga sebagaimana mungkin membuat kerja dari CPU menjadi lebih efisien dan efektif.

Penjadwalan proses yang pertama kali dipakai oleh CPU memiliki prinsip yang dikenal dengan *first in first serve*. Algoritma ini melakukan penjadwalan proses dengan mengeksekusi proses yang pertama kali masuk kedalam CPU. Oleh karena itu proses yang lain akan dieksekusi setelah proses sebelumnya yang dating dieksekusi oleh CPU. Namun hal tersebut akan sangat mengurangi keefisienan dari CPU bila process yang dikeskusi setelahnya memiliki waktu eksekusi yang lebih kecil. Jika hal tersebut terjadi maka waktu tunggu total dari setiap proses akan sangat besar sehingga dapat menyebabkabkan beberapa proses mengalami *deadlock.*

Metode *first in first serve* kurang efektif jika ada suatu proses yang waktu eksekusinya lebih besar datang lebih awal dibandingkan dengan proses yang datang setelahnya dengan waktu eksekusinya lebih kecil. Oleha karena hal tersebut disini kita akan menerapkan algorima *greedy* yang pasti akan mengahasilkan total waktu tunggu eksekusi yang paling kecil sehingga kinerja dari CPU menjadi optimum. Hal ini dapat terjadi karena algoritma *greed*y akan mengeksekusi proses dengan waktu eksekusi paling kecil lebih dahulu, *shortest job first.*

II. DASAR TEORI

A. Algoritma *Greedy*

Algoritma *greedy* merupakan salah satu algoritma yang paling populer didalam memecahkan suatu masalah mengenai pengoptimisasian kinerja dari suatu aplikasi. Algoritma *greedy* kebanyakan (tetapi tidak selalu) gagal untuk mencari solusi global optimal, karena mereka biasanya tidak beroperasi secara mendalam pada semua data. Meskipun demikian, algoritma *greed*y sangat berguna karena algoritma ini sangat cepat untuk menyelesaikan masalah dan sering meberikan prakiraan yang baik untuk nilai yang optimal

Algoritma *greedy* membentuk solusi langkah per langkah. Pada setiap langkah, terdapat banyak pilihan yang perlu dieksplorasi. Dari pilihan tersebut, dipilih yang merupakan nilai optimum pada langkah tersebut (optimum lokal) dengan harapan bahwa langkah selanjutnya akan mengarah ke optimum global.

Prinsip algoritma *greedy* pada tiap langkah adalah mengambil pilihan yang terbaik pada saat itu tanpamemperhatikan konsekuensi ke depan dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Elemen-elemen dalam algoritma greedy :

1. Himpunan kandidat, yang berisi elemen-elemen pembentuk solusi.
2. Himpuan solusi, berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi seleksi, dinyatakan dengan predikat SELEKSI memilih kandidat yang paling memungkinkan mencapai solusi optimal pada setiap langkah.
4. Fungsi kelayakan, dinyatakan dengan predikat LAYAK, memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak dengan tidak melanggar constraints yang ada.
5. Fungsi objektif, yang memaksimumkan atau meminimumkan nilai solusi.

Dengan kata lain, algoritma *greedy* melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif.

Berikut ini adalah skema umum algoritma *greedy*,

```
function Greedy (input C : h_kandidat) → h_kandidat
{
  Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy.
  Masukan : himpunan kandidat C.
  Keluaran : himpunan solusi yang bertipe himpunan_kandidat.
}

Deklarasi
   x : kandidat
  S : h_kandidat

  function SELECTION(C : h_kandidat) □ kandidat
  {me-return sebuah kandidat yang dipilih dari C berdasarkan kriteria tertentu}

  function SOLUTION(S : h_kandidat) □ boolean
  {true  jika S adalah solusi dari persoalan;
  false jika S belum menjadi solusi}

  function FEASIBLITY(S : h_kandidat) □ boolean
  {true  jika S merupakan solusi yang tidak  melanggar kendala;
  false jika S melanggar  kendala}

Algoritma
   S ← {}   {inisialisasi S dengan kosong}
   while (not SOLUTION(S)) and (C ≠ {}) do
     x ← SELECTION(C){pilih 1 kandidat dari C}
     C ← C – {x}   {elemen himpunan kandidat berkurang 1}
      if FEASIBILITY (S or {x}) then
      S ← S or {x}
      endif
   endwhile
   {SOLUTION(S) or C ={} }

   if SOLUTION(S) then
     return S
   else
     write ("Sorry, we could not find possible answer")
  endif
```

B. Penjadwalan Proses

Penjadwalan proses didalam CPU adalah memilih pemilihan sebuah proses yang ada pada CPU untuk dieksekusi Tujuan dari penjadwalan proses sendiri adalah untuk mendapatkan kinerja CPU yang paling optimal sehingga setiap proses dapat ditangani dengan baik sehingga tidak terjadinya *deadlock*.

Deadlock adalah situasi dimana dua atau lebih proses bersaing untuk mendapatkan memori untuk dieksekusi hingga masing-masing menunggu eksekusi lain selesai, dan dengan demikian tidak pernah dilakukan. Hal ini sering terlihat pada sebuah paradoks seperti "ayam atau telur".

C. Metode *First in First serve*

Metode *first in first serve* adalah metode pertama yang digunakan didalam penjadwalan proses pada CPU. Metode ini memenuhi prinsip yang ada pada penjadwalan proses. Sehingga pada masa awal penjadwalan proses pada CPU metode ini banyak dipakai.

Berikut ini adalah skema umum dari metode *first in first serve*,

```
function FIFS (input P: process q: queue)→ process
{
 Mengembalikan solusi dari persoalan dengan metode first in first serve.
 Masukkan : process dan sebuah antrian
 Keluaran : process yang akan dieksekusi
}
Deklarasi
S: process
i: integer

i← 1;

for i=1 to i= eff

q ← P[i];

S ← q.head;

return S;
```

III. PENGUJIAN ALGORITMA GREEDY

Pada tahap ini penulis akan menguji apakah algoritma greedy dapat menghasilkan nilai yang minimum pada waktu tunggu rata- rata suatu proses sehingga dapat mengoptimalkan kerja komputer karena setiap program di proses oleh CPU dengan baik sehingga tidak terjadi deadlock.

Dibawah ini akan saya ambil salah satu soal mengenai beberapa proses yang masuk pada CPU dan bagaimana cara penjadwalan proses tersebut.

Soal yang akan dipecahkan oleh dua metode diatas akan meliputi empat proses yang kedatangan atau prioritasnya di tunjukan oleh urutan, waktu atau lama eksekusinya ditunjukkan oleh waktu, dan masing nama proses atau proses yang berjalan pada CPU ditunjukan oleh proses.

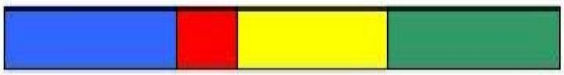
Dengan pendekatan algoritma *greedy*, soal atau masalah dan jawaban penjadwalan prosesnya dapat diilustrasikan dengan gambar dibawah ini.

Proses	Waktu	Urutan
P1	6	4
P2	8	1
P3	7	3
P4	3	2

Gambar 1. Solusi penjadwalan proses yang diselesaikan dengan algoritma *greedy*

Sedanghkan dengan pendekatan metode *first in first serve*, soal atau masalah dan jawaban penjadwalan prosesnya dapat diilustrasikan dengan gambar dibawah ini.

Proses	Waktu	Urutan
P1	6	4
P2	8	1
P3	7	3
P4	3	2



Gambar 2. Solusi penjadwalan proses yang diselesaikan dengan algoritma *first in first serve*

Kembali kita ingat bahwa parameter dari pendekatan *greedy* pada ilustrasi pertama adalah waktu atau lama eksekusi yang nilainya dapat dilihat pada kolom waktu. Sedengkan pada pendekatan dengan metode *first in first serve* yang diilustrasikan pada gambar kedua mempunyai parameter urutan kedatangannya yang nilainya dapat dilihat didalam kolom urutan.

IV. PEMBAHASAN

Pada ilustrasi pertama atau pendekatan dengan algoritma *greedy* dengan parameter lama waktu eksekusi, CPU akan melakukan penjadwalan proses dengan memprioritaskan proses yang memiliki waktu atau lama eksekusinya yang paling cepat dengan begitu maka proses yang akan diekskusi pertama adalah proses P4 yang mempunyai waktu atau lama eksekusi 3 detik, yang kedua adalah proses P1 yang mempunyai waktu atau lama eksekusi 6 detik, yang ketiga adalah proses P3 yang mempunyai waktu atau lama eksekusi 7 detik, dan yang terakhir adalah proses P2 yang mempunyai waktu atau lama eksekusi 8 detik.

Pada pendekatan dengan algoritma *greedy* ini dapat kita lihat bahwa elemen- elemen yang ada dalam algoritma *greedy* terbagi menjadi:

1. P1, P2, P3, dan P4 secara berurutan adalah himpunan kandidat
2. Himpunan solusinya secara berurutan anggotanya berupa P4, P1, P3, dan P2.
3. Fungsi seleksinya berupa pengecekkan apakah nilai proses yang akan dieksekusi memiliki nilai waktu atau lama eksekusi yang paling cepat jika ya maka proses akan dieksekusi. Jika tidak maka akan dicari proses yang lainnya.
4. Fungsi kelayakan adalah mengetahui apakah suatu proses layak untuk dieksekusi yang ditandai dengan pengecekan waktu eksekusi bila mendekati tak hingga maka dia dianggap bukan proses yang layak untuk dieksekusi dalam persoalan kali ini semua anggota himpunan

kandidat memiliki kelayakan sebagai proses yang dapat atau layak dieksekusi.

5. Fungsi objektifnya adalah bertujuan untuk mengembalikan anggota proses yang waktu atau lama eksekusinya paling kecil dan masih ada didalam himpunan kandidat atau proses- proses yang belum dieksekusi oleh CPU.

Dengan pendekatan algoritma *greedy* dengan parameter waktu atau lama eksekusi didapatkan waktu tunggu rata-rata tiap proses adalah:
P1: (0 + 3) = 3 detik
P2: (0 + 3 + 6 + 7) = 16 detik
P3: (0 + 3 + 6) = 9 detik
P4: (0) = 0 detik

Waktu tunggu rata = 28/4 = 7 detik

Pada pendekatan dengan metode *first in first serve* yang kita dapat lihat pada gambar 2. Sebenarnya memiliki karakteristik yang sama dengan algoritma *greedy* yang kita terapkan diatas, namun berbeda dalam hal parameternya. Pada metode *first in first serve* parameter yang diacu adalah urutan prosesnya sehingga akan mengambil nilai dengan nilai urutan yang terkecil. Dengan elemen- elemen yang bisa dibagi menjadi seperti dibawah ini:

1. P1, P2, P3, dan P4 secara berurutan adalah himpunan kandidat
2. Himpunan solusinya secara berurutan anggotanya berupa P4, P1, P3, dan P2.
3. Fungsi seleksinya berupa pengecekkan apah nilai proses yang akan dieksekusi memiliki nilai urutan yang paling dahulu atau awal jika ya maka proses akan dieksekusi. Jika tidak maka akan dicari proses yang lainnya.
4. Fungsi kelayakan adalah mengetahui apakah suatu proses layak untuk dieksekusi yang ditandai dengan pengecekan waktu eksekusi bila mendekati tak hingga maka dia dianggap bukan proses yang layak untuk dieksekusi dalam persoalan kali ini semua anggota himpunan kandidat memiliki kelayakan sebagai proses yang dapat atau layak dieksekusi.
5. Fungsi objektifnya adalah bertujuan untuk mengembalikan anggota proses yang memiliki nilai urutan paling kecil atau awal dan masih ada didalam himpunan kandidat atau proses- proses yang belum dieksekusi oleh CPU.

Dengan pendekatan algoritma *greedy* dengan parameter waktu atau lama eksekusi didapatkan waktu tunggu rata-rata tiap proses adalah:
P1: (0 + 8 + 3 + 7) = 18 detik
P2: (0) = 0 detik
P3: (0 + 8 + 3) = 11 detik
P4: (0 + 8) = 8 detik

Waktu tunggu rata = 37/4 = 9.25 detik

Sehingga didapatkan selisih waktu tungggu proses- proses tersebut adalah:

$\triangle T = 9.25 - 7$
$= 2.25$ detik

Disini terlihat bahwa dengan melakukan penjadwalan proses dengan pendekatan algoritma *greedy* dengan parameter atau lama eksekusi didapatkan hasil waktu tunggu rata- rata proses yang lebih cepat jika dibandingkan dengan pendekatan dengan metode *first in first serve* atau yang sekarang bisa kita sebut algortima *greedy* dengan parameter urutan kedatangan tiap prosesnya.

Soal diatas hanya menjalankan empat proses dalam kennyataanya ratusan proses yang berjalan tiap detiknya pada CPU. Bila kita memakai pendekatan dengan metode *first in first serve* maka pemanfaatan dari CPU sendiri menjadi tidak optimal. Karena dengan selisih waktu yang tinggi dengan pendekatan algoritma *greedy* dengan parameter waktu atau lama eksekusi maka akan dapat proses yang dikerjakan lebih banyak lagi sehingga pemakaian CPU jauh lebih optimal.

V. MASALAH *STARVATION* DALAM PENJADWALAN PROSES

Penjadwalan proses dengan pendekatan algoritma *greedy* memang memberikan hasil yang maksimum pada waktu tunggu rata- rata setiap prosesnya. Namun hal tersebut tidak serta merta menjadi solusi yang tepat bagi proses penjadwalan karena dengan algoritma ini kemungkinan ada suatu kondisi dimana suatu job mungkin tidak pernah menyelesaikan eksekusinya hal ini yang kemudian disebut *starvation.*

Kondisi tersebut dapat diilustrasikan sebagai berikut:

Proses A dgn elapse time 1 jam tiba pd waktu 0. Namun, pd waktu yg sama dan setiap 1 menit berikutnya tiba proses singkat dgn elapse time 2 menit. Hasilnya: A tidak pernah mendapat jatah eksekusi.

Namun jika digunakan pendekatan dengan metode *first in first serve*hal tersebu tidak mungkin terjadi karena tidak ada proses yang akan menunggu jika dia datang belakangan sehingga setiap proses pasti dikerjakan namun waktu tunggu tiap proses kemungkinan lebih besar jika dibandingkan dengan pendekatan algoritma *greedy* dengan parameter waktu atau lama eksekusi tiap prosesnya.

VI. KESIMPULAN

Dari penjelasan, pengujian, dan pembahasan mengenai penjadwalan proses dengan dua pendekatan diatas maka dapat disimpulkan beberapa hal sebagai berikut:

1. Algoritma *greedy* dengan parameter waktu atau lama eksekusi selalu menghasilkan nilai yang minimum untuk waktu tunggu rata- rata dari

setiap prosesnya.

2. Metode *first in first serve* merupakan salah satu algoritma *greedy* juga. Namun berbeda parameternya. Untuk metode *first in first serve*, algoritma *greedy*-nya memakai parameter urutan waktunyadengan mengeksekusi proses yang pertama kali datang.
3. Ada kemungkinan satu kondisi yang menyebabkan pendekatan penyelesaian masalah penjadwalan proses pada CPU dengan menggunakan algoritma *greedy* dengan parameter waktu atau lama eksekusi menyebabkan terjadinya *starvation.*
4. Metode *first in first serve* tidak selalu menghasilkan menghasilkan nilai yang minimum untuk waktu tunggu rata- rata dari setiap prosesnya, tetapi bebas dari masalah *starvation.*

VII. DAFTAR PUSTAKA

[1] S. T.Andrew, "Modern Operating System" , 2nd ed, New Jersey: Prentice- Hall, 2001, pp. 132–150.

[2] Munir, Rinaldi, "Diktat Kuliah IF2153 Matematika Diskrit", 2006,pp.26-69.

VIII. 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, 29 April 2010

Irdham Mikhail kenjibriel (13508111)