ABRAHAM SILBERSCHATZ - PETER BAER GALVIN - GREG GAGNE

Operating System:

Abraham Silberschatz, Peter Baer Galvin, Greg Gagne, *Operating System Concepts Essentials*, 2012, 2th Edition, John Wiley & Sons. Inc.

CHAPTER 5 PART 2: DEADLOCK

PRIO HANDOKO, S.KOM., M.TI.

An Overview

In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources; if the resources are not available at that time, the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting processes. This situation is called a **deadlock**.

Ch. 5: Deadlock

Perhaps the best illustration of a deadlock can be drawn from a law passed by the Kansas legislature early in the 20th century. It said, in part: "When two trains approach each other at a crossing, both shall come to a full stop and neither shall start up again until the other has gone."

Chapter Objectives.

- To introduce deadlock condition including system model and deadlock characteristics.
- To describe various methods for handling deadlocks.

Ch. 5: Deadlock

Agenda.

- Definition
- System Model
- Deadlock Characterization
- Methods for Handling Deadlocks
  - Deadlock Prevention
  - Deadlock Avoidance
  - Deadlock Detection and Recovery

Definition

Suatu kondisi dimana setiap ***thread*** atau ***process*** yang dieksekusi dalam waktu bersamaan saling tidak saling melepaskan sumber daya (*resources*).

System Model

- Meskipun proses yang harus dieksekusi CPU jumlahnya banyak, sistem hanya memiliki sebuah CPU dan sumber daya yang terbatas.
- Agar CPU dapat melakukan pengolahan terhadap proses, maka sumber daya yang tersedia harus dapat mengakomodir kebutuhan proses.
- Proses dapat meminta sumber daya sebanyak mungkin sesuai kebutuhan proses tetapi tidak boleh melebihi kapasistas sumber daya yang tersedia.

System Model

- Proses harus melakukan permintaan (*request*) penggunaan sumber daya dan melepasnya (*release*) ketika selesai menggunakan.
- Dalam kondisi normal, proses dapat menggunakan sumber daya **jika** mengikuti urutan berikut:
  - **Request.**
  - **Use.**
  - **Release.**
- ***Bagaimana sistem operasi memastikan bahwa semua proses telah memiliki sumber daya yang diinginkan?***

System Model

- ***Bagaimanakah jika sebuah proses meminta sumber daya yang telah dialokasikan untuk proses yang lain?***
- ***Kondisi bagaimanakah yang menggambarkan bahwa sekumpulan proses berada dalam kondisi deadlock?***

Deadlock Characterization

Neccesary Condition.

- Kondisi ***deadlock*** dapat dipicu oleh 4 kondisi "***hold***" berikut:
  - **Mutual Exclusion Condition.**
  - **Hold and Wait Condition.**
  - **No Preemption Condition.**
  - **Circular Wait Condition.**

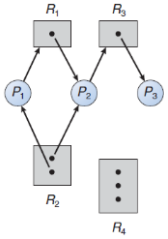
Deadlock Characterization

Resource-Allocation Graph.

- Kondisi ***deadlock*** dapat digambarkan dalam bentuk ***directed graph*** (***digraph***) yang disebut dengan ***system resource-allocation graph***.
- Graph terdiri dari himpunan simpul/***node*** (*V*) dan himpunan busur/***line*** (*E*).
- Himpunan simpul/***node*** (*V*) terbagi lagi menjadi 2 tipe simpul/***node***: (1) proses/***process*** (*P*); dan (2) sumber daya/***resource*** (*R*).
- *Directed edges* dari proses P_i ke R_j dinotasikan dengan $P_i \rightarrow R_j$.

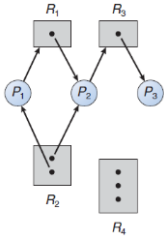
Deadlock Characterization

- Jika proses P_i meminta (*request*) sumber daya R_j, maka dinotasikan dengan $P_i \rightarrow Rj$ (***request edge***), diwakilkan oleh simbol lingkaran.
- Jika sumber daya R_j telah dialokasikan untuk sebuah proses P_i, maka dinotasikan dengan $R_j \rightarrow P_i$ (***assignment edge***), diwakilkan oleh simbol persegi.
- Titik (***dot***) mewakili banyaknya alokasi sumber daya setiap tipe sumber daya.



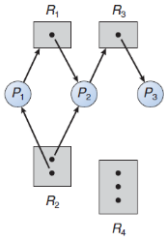
Deadlock Characterization

- ***Jelaskan kondisi system resource-allocation graph di samping ini!***
- ***Jelaskan hubungan kondisi state dengan hold and wait condition!***
- ***Berapakah jumlah siklus yang terdapat dalam system resource-allocation graph di samping ini?***



Deadlock Characterization

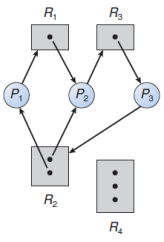
- Jika graph tidak mengandung suatu siklus, maka tidak ada proses yang mengalami ***deadlock***, dan jika sebaliknya, maka dimungkinkan ***deadlock*** terjadi.



Deadlock Characterization

Kasus.

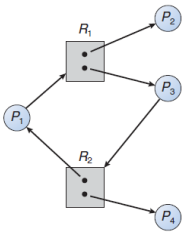
- Jelaskan kondisi *system resource-allocation graph* di samping ini dan kaitannya dengan kondisi ***hold and wait***!
- Apakah *system resource-allocation graph* di samping ini mengalami *deadlock*?
- Jika ya, jelaskan alasannya dan buktikan!



Deadlock Characterization

Contoh Kasus.

- Perhatikan *system resource-allocation graph* di samping ini!
- Apakah *system resource-allocation graph* di samping ini mengalami *deadlock*?
- Jika ya, jelaskan alasannya dan buktikan!



Methods for Handling Deadlocks

- Skema yang digunakan untuk memastikan tidak terjadinya ***deadlock*** terbagi atas:
  - **Deadlock Prevention,** metode yang memastikan setidaknya terdapat satu kondisi dimana terdapat sumber daya yang tidak dapat dipakai bersama (*nosharable resource*)
  - **Deadlock Avoidance,** memberikan informasi tambahan ke sistem operasi mengenai proses yang memiliki kemungkinan melakukan permintaan (request) sumber daya dalam waktu lama.

Methods for Handling Deadlocks

Deadlock Prevention.

- Untuk dapat mencegah terjadinya kondisi ***deadlock***, maka dapat dilakukan hal-hal berikut:
  - **Full Resources**
  - **Release Resources**
  - **Linear Queues**
  - Menghilangkan ***mutual exclusion***
  - Menghilangkan ***hold and wait***
  - Menghilangkan ***nonpreemption***
  - Menghilangkan ***circular wait***

Methods for Handling Deadlocks

Deadlock Avoidance.

- Untuk dapat melakukan penghindaran terhadap kondisi ***deadlock***, maka sistem operasi harus selalu dapat mengusahakan " ***state***" (kondisi) pemakaian sumber daya oleh proses-proses selalu dalam keadaan/kondisi aman (***safe***), ***safe state***.

Methods for Handling Deadlocks

Safe State.

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	4	8
JOB2	3	10
JOB3	5	9
Jumlah resoures tersedia		7

Methods for Handling Deadlocks

- Misalkan *job1* mendapatkan alokasi *resources* sebanyak 4 MB

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	8	8
JOB2	3	10
JOB3	5	9
Jumlah resources tersedia		3

Methods for Handling Deadlocks

- Maka setelah *job1* selesai, kondisi *resources* menjadi:

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	3	10
JOB3	5	9
Jumlah resources tersedia		11

Methods for Handling Deadlocks

- Kemudian *job3* mendapatkan *resources* sebanyak 4 MB

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	3	10
JOB3	9	9
Jumlah resources tersedia		7

Methods for Handling Deadlocks

- Maka setelah *job3* selesai, kondisi *resources* menjadi:

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	3	10
JOB3	0	0
Jumlah resources tersedia		16

Methods for Handling Deadlocks

- Berikutnya giliran *job2 yang* mendapatkan *resources* sebanyak 7 MB

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	10	10
JOB3	0	0
Jumlah resources tersedia		9

Methods for Handling Deadlocks

- Maka setelah *job2* selesai, kondisi *resources* menjadi:

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	0	0
JOB3	0	0
Jumlah resoures tersedia		19

- Dengan demikian ketiga proses dapat menyelesaikan prosesnya dengan sempurna.

Methods for Handling Deadlocks

Unsafe State.

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	4	8
JOB2	3	10
JOB3	5	9
Jumlah resoures tersedia		7

Methods for Handling Deadlocks

- Misalkan *job1* dan *job2* masing-masing mendapatkan alokasi *resources* sebanyak 4 MB dan 3 MB

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	8	8
JOB2	6	10
JOB3	4	9
Jumlah resources tersedia		0

Methods for Handling Deadlocks

- Maka setelah diproses, kondisi *resource* menjadi:

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	6	10
JOB3	4	9
Jumlah resources tersedia		8

Methods for Handling Deadlocks

- Berikutnya giliran *job2* dan *job3 yang* mendapatkan *resources* masing-masing sebanyak 4 MB dan 5 MB

Proses	Resources yang Digunakan (MB)	Maksimum resources yang dibutuhkan (MB)
JOB1	0	0
JOB2	10	10
JOB3	8 (-1)	9
Jumlah resources tersedia		0 (-1)

Methods for Handling Deadlocks

Kasus.

Perhatikan tabel di bawah ini.

Job	Sumber daya awal yang dimiliki (KB)	Maksimum sumber daya yang dibutuhkan (KB)
1	2	10
2	4	16
3	3	15
4	8	24
5	1	6
Jumlah sumber daya tersedia (KB)		**30**

Berdasarkan tabel di samping ini, **gambarkan proses pendistribusian sumber daya** untuk semua *job* agar terhindar dari ***deadlock*** jika pemberian sumber daya dapat dilakukan untuk lebih dari 1 job dalam waktu yang bersamaan dan besarnya sumber daya yang diberikan ke setiap *job* merupakan kelipatan dari nilai sumber daya yang telah dimiliki sebelumnya oleh setiap *job*!