Chapter 6

# Deadlock là gì?

* Một tiến trình được gọi là deadlock nếu nó đang đợi một sự kiện mà sẽ không bao giờ xảy ra.
* Thông thường, có nhiều hơn một tiến trình bị liên quan trong một deadlock.

# Các điều kiện cần để xảy ra deadlock?

* Loại trừ tương hỗ: ít nhất một tài nguyên được giữ theo nonsharable mode.
* Giữ và chờ cấp thêm tài nguyên: một tiến trình đang giữ ít nhất một tài nguyên và đợi thêm tài nguyên do quá trình khác giữ.
* Không trưng dụng: tài nguyên không thể bị lấy lại mà chỉ có thể được trả lại từ tiến trình đó khi nó muốn.
* Chu trình đợi: tồn tại một tập các quá trình đang đợi sao cho: p0 đợi tài nguyên p1 đang giữ, p1 đợi tài nguyên p2 đang giữ…

# Đồ thị cấp phát tài nguyên là gì? Mối liên hệ giữa đồ thị cấp phát tài nguyên và deadlock?

* Là đồ thị có hướng, với tập đỉnh V và tập cạnh E.
* RAG không chứa chu trình -> không có deadlock.
* RAG chứa một hay nhiều chu trình:
  + Nếu mỗi loại tài nguyên chỉ có một thực thể -> deadlock.
  + Nếu mỗi loại tài nguyên có nhiều thực thể -> có thể xảy ra deadlock.

# Có mấy phương pháp để giải quyết deadlock? Phân tích và đánh giá ưu, nhược điểm của từng phương pháp?

* Có 2 cách: ngăn deadlock và tránh deadlock.
* Ngăn deadlock:
  + Ưu điểm: chỉ cần một trong bốn điều kiện không xuất hiện.
  + Nhược điểm: sử dụng tài nguyên kém hiệu quả.
* Tránh deadlock:
  + Ưu điểm: hữu dụng và sử dụng tài nguyên hiệu quả.
  + Nhược điểm: mọi process nói trước số lượng tài nguyên nó cần.

# Phân tích và đánh giá ưu, nhược điểm của các giải pháp đồng bộ busy waiting (cả phần cứng và phần mềm)?

# Trạng thái an toàn là gì? Mối liên hệ giữa trạng thái an toàn và deadlock?

* Một trạng thái của hệ thống được gọi là an toàn (safe) nếu tồn tại một chuỗi thứ tự an toàn.
* Nếu hệ thống đang ở trạng thái safe -> không deadlock.

# Mô tả cách thực hiện các giải thuật Banker: giải thuật an toàn, giải thuật yêu cầu tài nguyên và giải thuật phát hiện deadlock?

* Giải thuật an toàn:
  + Gọi Work và Finish là hai vector độ dài m và n.
  + Tìm *i* thỏa:

1. Finish[i] = false.
2. Needi <= Work (hàng thứ i của Need).

Nếu không tồn tại i như vậy, đến bước 4.

* + Work = Work + Allocationi

1. Finish[i] = true.

Quay về bước 2.

* + Nếu finish[i] = true, i = 1, …, n, thì hệ thống đang ở trạng thái safe.
* Giải thuật yêu cầu tài nguyên:

Request i là request vector của process Pi .

Request i [j] = k <=> Pi cần k instance của tài nguyên Rj .

* + Nếu Requesti <= Needi thì đến bước 2. Nêu không thì báo lỗi.
  + Nếu Requesti <= Available thì qua bước 3. Nếu không, Pi­ phải chờ vì tài nguyên không còn đủ để cấp phát.
  + Giả định cấp phát tài nguyên đáp ứng yêu cầu của Pi bằng cách cập nhật trạng thái hệ thống như sau:
    - Available = Available – Requesti
    - Allocationi = Allocationi  + Requesti
    - Needi  = Needi – Requesti
* Giải thuật phát hiện deadlock:
  + Gọi Work và Finish là vector kích thước m và n.
  + Tìm i thỏa mãn:
    - Finish[i] = false
    - Requesti <= Work

Nếu không tồn tại i như vậy thì đến bước 4.

* + Work = Work + Allocationi
    - Finish[i] = true

Quay về bước 2.

* + Nếu finish[i] = fasle, với một số i = 1, …, n, thì hệ thống đang ở trạng thái deadlock. Hơn thế nữa, finish[i] = false thì Pi bị deadlocked.

# Các giải pháp để phục hồi hệ thống sau khi phát hiện có deadlock?

* Báo người vận hành.
* Hệ thống tự phục hồi bằng cách bẻ gảy chu trình deadlock.

# (Bài tập mẫu)

# (Bài tập mẫu)

# (Bài làm)

Ảnh có chứa ảnh chụp màn hình, vòng tròn, thiết kế

Mô tả được tạo tự động

1. Có 8 chuỗi an toàn.
2. <P4,P2,P3,P1,P5>, <P4,P2,P3,P5,P1>, <P4,P2,P1,P3,P5>, <P4,P2,P1,P5,P3>, <P2,P3,P1,P4,P5>, <P2,P3,P1,P5,P4>,

<P2,P3,P4,P1,P5>, <P2,P3,P5,P1,P4> .

# (Bài tập mẫu)

# (Bài làm)

* 1. Chuỗi an toàn của hệ thống:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Allocation | | | | Max | | | | Need | | | | Available (Work) | | | |  |
| **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** | **A** | **B** | **C** | **D** |  |
| **P0** | 0 | 0 | 1 | 2 | 0 | 0 | 3 | 2 | 0 | 0 | 2 | 0 | 2 | 1 | 3 | 2 | P0 |
| **P1** | 2 | 0 | 0 | 0 | 2 | 7 | 5 | 0 | 0 | 7 | 5 | 0 | 4 | 4 | 8 | 6 | P3 |
| **P2** | 0 | 0 | 3 | 4 | 6 | 6 | 5 | 6 | 6 | 6 | 2 | 2 | 4 | 7 | 11 | 8 | P4 |
| **P3** | 2 | 3 | 5 | 4 | 3 | 3 | 5 | 6 | 1 | 0 | 0 | 2 | 6 | 7 | 11 | 8 | P1 |
| **P4** | 0 | 3 | 3 | 2 | 0 | 6 | 5 | 2 | 0 | 3 | 2 | 0 | 6 | 7 | 14 | 12 | P2 |

* Chuỗi an toàn: <P0,P3,P4,P1,P2> .
  1. P3 yêu cầu thêm tài nguyên (1,1,0,0):
* Request P3 (1,1,0,0) <= Need P3 (1,0,0,2) : sai
* Request P3 (1,1,0,0) <= Available (2,1,2,0)
* Vậy hệ thống không thể đáp ứng.