Deadlock 검출을 위한 분산알고리즘

°원 종 권,* 류 근 호,* 박 규 태**
*충북대학교 전산통계학과 **연세대학교 전자공학과

A Distributed Algorithm for Deadlock Detection

Jongkwon Won*, Keunho Ryu*, Kyutae Park**
***ChungBuk National University** ****Yonsei University**

Abstract

This paper presents a distributed algorithm for deadlock detection in distributed computing system. The deadlock could be detected optimally from cycle. This algorithm maintains the Process-Resource Table and it could be found the deadlock from the table.

I. 서 론

Deadlock은 2개 이상의 프로세스가 서로 다른 자원을 사용하기 위해 무한히 기다리는 상태에 놓인 것을 말한다. 이러한 상태는 분산된 시스템에서 더욱 인식하기 어렵다.

일반적인 시스템에서 deadlock을 해결하기 위한 방법으로는 방지, 회피 및 검출등이 있다 [1,2]. 이 방법들 중에서 분산 시스템에서는 deadlock 검출 방법을 사용한다[3].

Deadlock 검출방법은 중앙집중적인 해결방법과 분산적인 해결방법이 있다. 중앙 집중적인 해결방법은 controller가 단지 한개 존재하며 모든 정보는 control site로 보낸다. 모든 Process-Resource 정보가 control site에서 관리 보존된다. 또한 deadlock detector 도 이 site에서 실행된다. 분산적인 해결방법은 각 site에서 자기가 필요한 Process-Resource 정보를 관리 보존하고, 각 site에서 deadlock detector도 각각 실행된다. Process-Resource정보는 서로 필요한 site간에 송수신된다.

본 연구는 분산된 시스템에서 발생하는 deadlock을 분산적인 해결방법으로 검출하는 방안을 제시하고 있다. 여기서 제시된 방안은 deadlock검출에 필요한 정보를 최소한으로 송수신 할뿐 아니라, deadlock 검출을 하기 위한 cycle점검도 최소화하고 있다.

II. 몇가지 정의 및 가정

(정의 1) Process-Resource그래프, PRG=(N,E)은 bipartite directed graph이다. 여기서 N_i는 노드의 집합 N={PUR}이다. P는 Process의 집합이고 R은 Resource의 집합이다. E_i는 directed edge와 reachable edge들의 집합으로서 자원을 요청하고 할당하는 것을 나타낸다.

예로서, Process가 Resource를 요청할때는 $P_i \rightarrow R_i$로 나타내고 요청된 Resource를 할당할때는 $P_i \leftarrow R_i$로 나타낸다.

(정의 2) Process Wait for Graph PWG(V,E)은 Resource를 사용하기 위해 대기하고 있는 Process의 집합 PS_i인 directed graph이다.

여기서 V_i는 $N_i \subset V_i$이고 $P_i \subset PS_i$인 Process를나타내는 Vertex들의 집합이다. 그리고 E_i는 edge $\{(P_i, R_j, P_j)\}$의 집합으로서 P_j가 사용하고 있는 R_j를 P_i가 사용하기 위해 기다리는 것이다. $P_j \in PS_i$

예로서, $PS_i = (P_i, P_j)$

$PWG = (V_i, E_i)$

$V_i = \{P_i, P_j\}$

$E_i = \{(P_i, R_j, P_j)\}$

으로 나타낸 것으로 $P_i \rightarrow R_j \rightarrow P_j$로 표현한다.

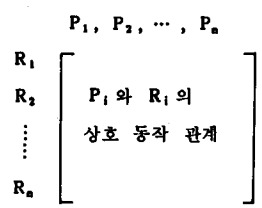
(정의 3) Reduced process graph, RPG는 PRG에서 process들의 집합만으로 표현한 것이다. RPG는 Waiting for process들로서 구성된 directed path이다. 이 RPG의 표현은 아래와 같다.

$$P_i \rightarrow P_j \rightarrow R_k \rightarrow \cdots \rightarrow P_n$$

(정의 4) Reduced reachable edge는 reachable set내의 node N_i로부터 모든 node들로 가는 directed path가 존재하는 PRG의 reduced node의 집합이다.

(정의 5) Process-Resource Table은 Process와 Resource간의 행위이다. Node $N = \{P \cup R\}$으로서 $N_i = \{P_1, P_2, \cdots, P_n\} \cup \{R_1, R_2, \cdots, R_n\}$으로 표현한 것

을 아래와 같이 나타낸다.



〈그림 1〉 Process-Resource Table

우리는 그림 1로부터 PWG, PRG를 만들 수 있다.

Deadlock은 lock을 사용한 병행 수행 제어에서 발생한다. Lock방법에서는 resource을 액세스할때 process가 미리 lock을 시켜야만 한다. 따라서 이 논문을 위하여 다음과 같이 가정한다.

1) Protocol은 2phase locking protocol이다. 만일 process가 resource를 사용한다면 다른 process는 이 resource을 사용할 수 없다. 즉 nonpreemptive이다.
2) 하나의 process는 한번에 하나의 resource만을 요청한다.
3) 어느한 process는 여러개의 Resource를 사용할 수 있다. 그러나 한 resource는 단지 한 process에게 할당된다. 즉, 모든 resource는 exclusive이다.
4) 모든 resource는 fully disjoint하다.

Ⅲ. Deadlock검출 알고리즘

각 Site는 자기의 Site에서 만들어진 Process에 대한 정보와 자기 Site에 존재하는 Resource에 관한 정보를 Process Resource Table로서 유지 관리한다.

3.1 알고리즘

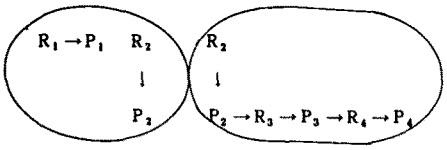
1. {Remote resource의 요청}
   a. Request edge를 PRG에 추가시키고 cycle을 점검한다.
   b. Cycle이면 request edge의 송신을 보류한다.
   c. Cycle이 아니면 request message를 request resource site로 송신한다.
2. {Local resource의 요청}
   a. Resource를 사용할 수 있는 경우
      a.a. 다른 Site로 부터 온 request message이면 resource를 할당시키고 PRG에 edge를 추가시킨다. 그리고 allocation message를 request site로 송신한다.
      a.b. 알고리즘을 종료시킨다.
   b. Resource를 사용할 수 없는 경우
      b.a. Waiting for edge를 PRG에 추가시킨다.
      b.b. Deadlock인가를 점검하여 deadlock이면 victim을 선정하여 해결한다.
      b.c. Deadlock이 아니면 단계 4로 간다.
3. {Allocation message의 수신}
   a. PWG의 할당된 process와 waiting process가 동일하다면 reachable edge로 만들어진 것이므로 할당 edge와 reachable edge를 삭제한다.
   b. Request edge가 있으면 request edge을 allocation edge로 바꾼다.
4. {Reduced reachable message 생성 및 전파}
   a. Directed outgoing path나 incomming path가 존재하는가를 점검한다. 만일 그 path가 존재한다면 reduced reachable message를 생성하여 site의 순서(daisy chain)에 따라 outgoing이나 incomming site로 송신한다.
   b. 알고리즘을 종료시킨다.
5. {Reduced reachable message의 수신}
   a. Reduced reacable message에 포함된 process와 PRG내의 process와 동일한가를 점검하여 동일하지 않으면 reachable message를 무시하고 단계 4로 간다.
   b. PRG를 구성한다.
   c. Cycle을 점검하고, cycle인 경우 deadlock을 해결한다.
   d. 만일 resource가 이미 Waiting process 에게 할당되었다면 알고리즘을 종료시킨다.
6. {Local resource release}
   a. PRG에서 Process-Resource edge를 삭제한다. Waiting edge가 있으면 Waiting process 에게 released resource를 할당한다.
   b. Process의 site와 resource site가 동일하지 않다면 allocated message를 process의 site로 송신한다.
   c. 알고리즘을 종료시킨다.

3.2 검출의 예와 Process Table의 표현

1. 검출의 예

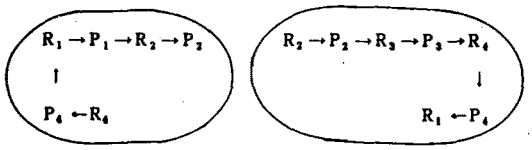
예를 간단히 하기 위하여 2개의 site가 존재하고, site간의 순서는 site 1 > site 2라고 가정한다. 〈그림 2〉와 같이 site 1에는 resource R_1과 R_2가 존재하고, site 2에는 resource R_2, R_3, R_4가 존재한다.

Process P_1 은 site 1 에서 Process P_2, P_3, P_4 는 site 2 에서 동작한다.



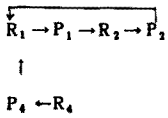
〈그림 2〉 초기상태

Site 1의 P_1 은 local resource R_2 를 요청하고 site 2의 P_4 는 site 1의 resource R_1 을 요청한다. 이 상태는 〈그림 3〉과 같이 바뀌어질 것이다.



〈그림 3〉 요청 메세지 수신

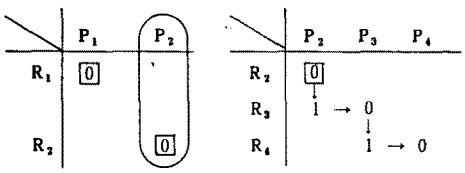
알고리즘은 〈그림 3〉으로부터 reduced reachable edge 를 생성한다. 즉, site 순서에 따라 site 2에서의 reachable message $\{P_2, R_1\}$ 을 생성하여 site 1로 전송한다. Site 1에서는 reachable message을 수신하여 PRG에 첨가시키므로서 deadlock이 검출된다(〈그림 4〉).



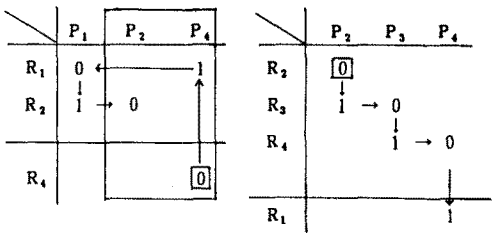
〈그림 4〉 Deadlock 검출

2. Process-Resource Table의 표현

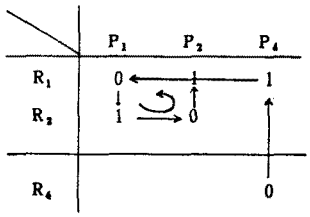
〈그림 5, 6, 7〉은 위의 예에서 나타낸 그래프의 형태를 Process-Resource Table로서 표현한 것이다.



〈그림 5〉 그림 2의 Process-Resource Table 표현



〈그림 6〉 그림 3의 Process-Resource Table 표현



〈그림 7〉 그림 4의 deadlock 상태표현

N. 결 론

시스템이 분산된 경우는 중앙 집중화된 시스템보다 deadlock 문제가 더욱 중요하다. 분산된 경우 deadlock의 해결방법으로는 검출기법을 사용하는 것이 적당하다 [3].

본 논문은 deadlock검출 기법중 분산적인 처리방안을 사용하고 있다. 각 site에 필요한 최소한의 정보를 분산시켜 PRG와 동일한 구조인 Process-Resource Table로서 관리하고 있다. 따라서 deadlock검출에 필요한 모든 정보를 송수신하는 것이 아니라 최소의 정보량(reduced reachable message)을 만들어 보낸다. 또한 deadlock의 cycle점검도 간단히 할 수 있음을 보여주고 있다.

참 고 문 헌

1. R.C. Holt, "Some Deadlock Properties of Computer Systems", Computing Survey, Vol.4, No.3, Sept., 1972.
2. S.S. Isloor and T.A. Marsland, "The Deadlock Problem: An Overview", IEEE Computer, Sept., 1980.
3. D.A. Menasce and R.R. Muntz, "Locking and Deadlock Detection in Distributed Databases", IEEE Trans. on S/W Eng., Vol. SE-6, Sept., 1980.

4. B.Goldman, "Deadlock Detection in Computer Networks", MIT Tech. Rep. TR-MIT/LCS/TR-185, Sept., 1977.
5. R. Obermark, "Distributed Deadlock Detection Algorithm", ACM TODS., Vol.7, No.2, June, 1982.
6. D.Z. Badal, "The Distributed Deadlock Detection Algorithm", ACM TODS, Vol.4, No.4, Nov., 1986.
7. W.C. Tasi, "Distributed Deadlock Detection in Distributed Database Systems", Ph.D thesis, Univ. of Illinois at Urbana-Champaign, 1982.
8. K.H. Ryu, S.B.Rhee, J.H. Kim and K.T. Park, "A Study on an Effective Distributed Deadlock Detection". KIEE, Vol.8, No.1, June, 1985.
9. T.G. Lewis, "Software Engineering Analysis and Verification", A Prentice-Hall Company, 1982.