Sanders 알고리즘을 이용한 실시간 상호배제의 성능 향상

0
전상준, 김재훈
아주대학교 정보 및 컴퓨터 공학부

Real-time Mutual Exclusion Scheme Using Sanders Algorithm

0
Sang-Joon Jun, Jai-Hoon Kim
School of Information and Computer Engineering
Ajou University

요 약

본 논문에서는 실시간 시스템에서의 상호배제를 제한된 시간안에 해결하기 위한 방안을 제안한다. Sanders 알고리즘을 이용해서 임계영역을 사용하기 위해 필요한 메시지의 수가 노드마다 서로 다를 수 있는 점을 이용한다. 즉, 서로 다른 데드라인을 가진 노드들이 임계영역을 효과적으로 사용하기 위해서 데드라인이 상대적으로 가까운 노드가 임계영역 사용을 위해서 전송하는 메시지의 수를 적게 갖는 구조를 취함으로 전체적인 데드라인 miss 횟수를 줄이게 된다. 시뮬레이션을 통해서 이것의 성능 향상을 보인다.

1. 서론

상호배제는 하나의 프로세스가 공유된 자원을 사용할 때 다른 프로세스가 그 임계영역(critical section)을 동시에 사용할 수 없도록 보장하는 것으로 단일 프로세서 시스템에서는 세마포어, 모니터, 그리고 이와 유사한 구조를 이용해서 임계영역을 보호한다. 분산 시스템에서는 크게 중앙 집중(centralized) 알고리즘과 분산 알고리즘으로 나눌 수 있다. 중앙 집중적인 방법은 하나의 프로세스가 임계영역을 들어가고자 하면 코오디네이터에게 메시지를 보내서 그 임계영역에 대한 허가를 요구한다. 코오디네이터는 자신이 받은 메시지 순서대로 허용(grant) 메시지를 보내기 때문에 스타베이션(starvation)이 발생하지 않으며 구현하기 쉬운 반면에 코오디네이터에 오류가 발생하면 전체 시스템이 장애를 받게 되는 단점이 있다[3].

분산 알고리즘으로 Ricart 와 Agrawala 가 제안한 것으로 시스템에서 발생하는 전체 이벤트의 순서를 미리 정해 놓고 하나의 프로세스가 임계영역을 들어가기 원하면 들어가고자 하는 임계영역에 대한 정보, 프로세스 번호, 그리고 현재 시간등을 포함하는 메시지를 다른 프로세스에게 보낸다. 요구 메시지를 받은 프로세스는 지정된 임계영역의 상태와 메시지에 따라서 조치를 취하게 된다.

실시간 분산 시스템에서는 임계영역을 사용함에 있어서도 데드라인 안에 수행을 완료해야 하는 시간 제약을 받게 된다. 또한 분산 시스템에서의 각 노드들이 서로 다른 데드라인을 갖는 경우 데드라인 miss 로 인한 성능저하를 막기 위해서는 상대적으로 데드라인이 짧은 노드를 우선적으로 처리해 주어야 한다. 다음은 이 논문에서 사용하는 Sanders 알고리즘에 대해서 간략히 살펴본다.

2. 관련연구

Sanders 의 알고리즘에서는 분산 시스템에서 각 노드들은 하나의 프로세스가 다른 프로세스를 메시지를 보낼 수 있도록 논리적으로 연결되어 있다고 가정한다. 메시지는 어느 순간에도 도달할 수 있으며 도착순서에 따라서 처리한다. 이 알고리즘은 임계영역에 들어가기 전에 수행하는 엔트리 코드, 임계영역내에서의 메시지 처리 코드와 임계영역을 떠날 때 사용되는 엑시트 코드로 구성된다[1].

하나의 프로세스는 임계영역내에서 실행되는 IN-CS, 임계영역 밖인 NOT-IN-CS, 그리고 엔트리 코드에서 실행되거나 중지된 WAITING 의 상태를 취하고 이러한 정보는 상호배제를 위한 중요한 정보이고 다른 프로세스와의 통신을 통해서 결정된다. Sanders 의 논문에서는 다음과 같이 세 개의 세트를 프로세스간의 통신에 이용한다.

- Inform Set : I_i
- Request Set : R_i
- Status Set: $S_i : j \in S_i$, if $i \in I_j$

프로세스는 자신이 IN-CS 에서 NOT-IN-CS 로나 NOT-IN-CS 에서 WAITING 으로 변할때 자신의 Information Set(I_i)에 있는 모든 프로세스에게 메시지를 보낸다. 임계영역에 들어가기 전에 각 프로세스는 반드시 Request Set(R_i)에 있는 모

든 프로세스에게 허가를 요청하고 받아야 한다.

2.1 일반화된 상호배제 알고리즘

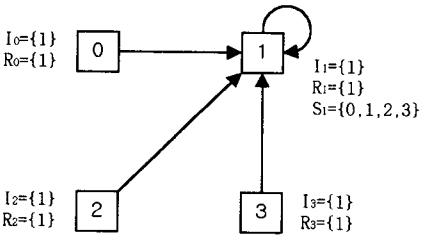
상호배제를 보장하기 위해서는 프로세스가 임계영역을 들어가기 전에 수행하는 엔트리(Entry) 코드와 임계영역을 나갈때의 엑스트(Exit) 코드가 요구된다.

▮ 엔트리(Entry) 코드
① 타임 스템프(time stamp)를 선택하고 REQUEST 메시지를 Request Set(R_i)에 있는 모든 프로세스에게 보낸다.
② R_i에 있는 모든 프로세스로부터 GRANT 메시지를 기다리고 받으면 임계영역에 들어감
▮ 엑시트(Exit) 코드
① RELEASE 메시지를 Inform Set(I_i)에 있는 모든 프로세스들에게 보낸다.

다음에서 언급되는 우선권 큐에 REQUEST 메시지를 받고 GRANT를 보내지 않은 프로세스를 저장하고 CSSTAT는 각 프로세스의 임계영역의 최신 정보를 가지고 있다. 프로세스가 메시지를 받으면 지역 변수를 수정하고 메시지에 따라 다음과 같이 한다.

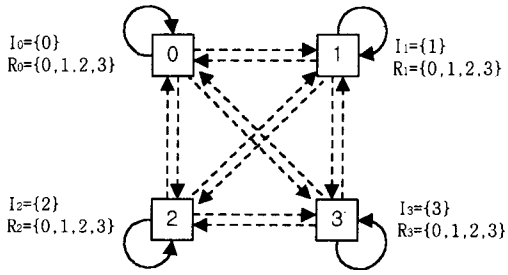
▮ REQUEST 메시지를 받을 때
① 메시지를 보낸 프로세스의 ID를 우선권 큐에 쓴다
② CSSTAT이 프리(free)이면 GRANT를 큐의 맨 처음에게 보낸다.
▮ RELEASE 메시지를 받을때
① CSSTAT를 프리(free)로 세트한다.
② 큐의 맨앞에 있는 프로세스에게 GRANT를 보낸다.
③ CSSTAT가 프로세스가 임계영역에 있음을 나타내거나 큐가 비어있을 때까지 ②를 계속 반복한다.

이와 같은 알고리즘을 사용해서 두가지 다른 형태의 알고리즘을 [그림 1], [그림 2]와 같이 나타낼 수 있다. 각각은 4개의 노드로 구성되며 각 노드는 서로 메시지를 주고 받을 수 있도록 연결되어 있다고 가정한다. 그림에서 실선은 REQUEST 메시지만을 보내는 것이고 점선 화살표는 REQUEST와 RELEASE 메시지 모두를 보내는 것이다. [그림 1]은 중앙 집중적인 알고리즘을 나타낸 것으로 0번, 2번 그리고 3번 노드는 임계영역에 들어가기 위해서 노드 1번으로 메시지를 보내는 방식이다.



[그림 1]

[그림 2]는 각 노드가 균등한 구조를 갖는 알고리즘을 보여주는 것으로 모두 동일한 Request Set(R_i)와 Inform Set(I_i)를 갖는다. 따라서 어느 한 노드가 임계영역에 들어가기 위해서는 다른 모든 노드들에 메시지를 보내야 한다.

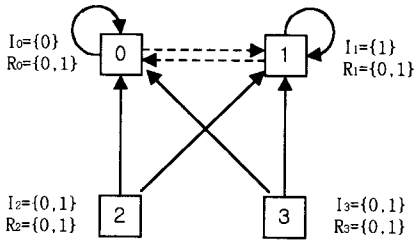


[그림 2]

3. 실시간 상호배제 알고리즘 제안 및 성능평가

3.1 알고리즘 제안

Sanders 알고리즘을 기반으로 하여 다음과 같이 4개의 노드를 위한 리퀘스트 세트와 인폼 세트를 정의하였다. 임계영역 진입에 필요한 총 메시지수 는 16이며 노드 0번과 1번 노드는 2번과 3번 노드에 비해 임계영역에 진입하기 위해 전송하는 메시지의 수가 적음을 알 수 있다. 따라서 분산 시스템에서는 자주 사용되는 노드가 0번과 1번일 경우에 유리한 구조이다.



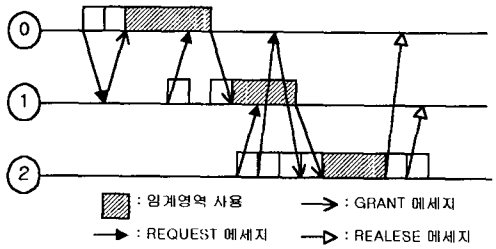
[그림 3]

본 논문에서는 이러한 구조를 실시간 시스템에 적용하여 그 성능 향상을 분석하려는 것이다. 실시간 시스템에서 주어진 타스크가 주어진 데드라인안에 끝내지 않으면 시스템의 성능저하를 초래하기 때문에 데드라인에 따라서 우선순위에 차이를 두고 타스크 스케줄링 관리를 해야한다. 또한 노드들 간에 서로 다른 데드라인을 갖게 되면 상대적으로 긴급한 데드라인을 갖는 노드는 다른 노드보다 많은 데드라인 miss를 초래하게 된다. 이로 인한 성능 저하를 막기 위해서 데드라인이 긴급한 노드에 높은 우선순위를 주어서 다른 노드보다 먼저 수행해야 한다. EDF(Earliest Deadline First) 알고리즘이 이의 한 예이다. 본 논문에서는 데드라인이 긴급한 노드에게 적은 메시지수를 이용하여 임계영역을 사용할 수 있도록 한다. 실시간 시스템에서 임계영역을 사용하기 위해서는 임계영역 진입을 위한 메시지를 줄여야 한다. 전체 메시지의 수도 줄이는 것이 필요하지만 데드라인이 급한 노드를 우선적으로 고려해야 한다. [그림 3]에서 노드 0번과 1번의 메시지수가 적기 때문에 데드라인이 급한 노드가 0번과 1번 노드이게 되면 그렇지 않은 [그림 2]

구조보다는 좋은 결과를 보여준다.

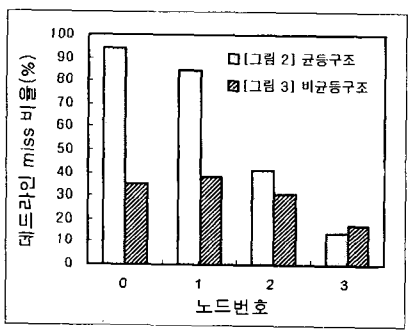
3.2 성능평가 및 분석

실시간 시스템에서 임계영역 사용의 성능을 측정하기 위하여 시뮬레이션을 하였다. 각 노드에서 임계영역을 수행하고자 하는 REQUEST는 Exponential 분포(평균도착시간: 25)를 취했고 각 노드들은 임계영역의 사용을 요구하며 임계영역내에서의 각 요구의 수행시간은 10에서 20사이로 균등분포한다고 가정했으며 각 노드당 데드라인은 각 노드의 수행시간의 1.2, 1.3, 1.5, 1.7배로 노드마다 다르다고 가정하였다. 노드 0번과 1번의 데드라인은 각각 수행시간의 1.2와 1.3배로 다른 노드에 비해 상대적으로 긴급하게 된다.



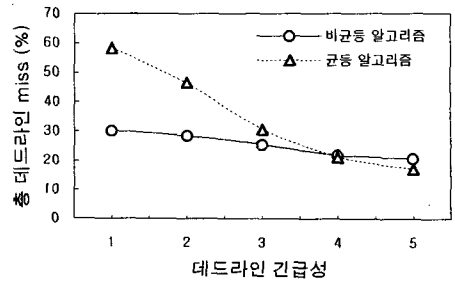
[그림 4]

[그림 4]와 같이 노드 0이 임계영역에 진입하기 위해서 노드 1에게 REQUEST 메세지를 보내고 GRANT를 받아 임계영역에 들어가고 수행을 끝내지 않았을 때 노드 1로부터 REQUEST를 받게 되면 노드 1은 노드 0번이 끝날 때 까지 기다린다. 노드 2는 REQUEST 메세지를 노드 0번과 1번 모두에게 보내고 GRANT를 기다리게 된다.



[그림 5]

[그림 5]는 균등 알고리즘[그림 2]와 비균등 알고리즘[그림 3]을 시뮬레이션을 통해서 비교분석한 것으로 노드들이 균등한 구조를 갖는 것은 4개 모든 노드가 임계영역 진입에 동일한 메시지수를 필요로 하기 때문에 상대적으로 데드라인이 긴급한 노드 0번과 1번의 데드라인 miss 비율이 다른 노드 2와 3에 비해 현격히 높게 된다. 비균등한 구조를 갖는 알고리즘은 데드라인이 급한 노드 0와 1이 임계영역에 들어 갈 때 보내는 메시지의 수가 적기 때문에 균등 알고리즘에 비해서 노드 0와 1의 데드라인 miss 비율이 절반 이하로 줄어들고 어느 특정한 노드에 데드라인 miss가 집중되는 현상이 크게 향상된다.



[그림 7]

[그림 7]은 데드라인이 긴급한 경우에서부터 점차 그 긴급성을 줄이면서 4개 노드에서 발생하는 총 데드라인 miss 비율을 측정한 것이다. 1에서 가장 긴급한 데드라인을 갖고 5로 갈수록 데드라인을 길게 하였다. 비균등 알고리즘은 데드라인의 긴급성에 크게 영향을 받지 않고 고른 데드라인 miss 비율을 갖는다(30~20%). 반면에 균등 알고리즘은 데드라인이 긴급하면 많은(60%) 데드라인 miss를 하고 데드라인이 길어지면 급격히 그 비율이 줄어든다. 비균등 알고리즘은 데드라인의 긴급성이 높을 때 균등 알고리즘보다 적은 비율의 데드라인 miss를 한다.

4. 결론 및 향후 계획

분산 시스템에서 임계영역을 사용하기 위해서는 상호 배제가 보장되어야 한다. 임계영역에 진입할 때와 수행을 끝내고 나올 때에 Sanders 알고리즘에서 정의한 것과 같이 다른 노드와의 통신을 통해 해결할 수 있고 각 노드의 환경에 따라 다른 메시지수를 요구하도록 설계할 수 있다. 실시간응용에 이를 적용해서 데드라인의 긴급성에 따라 임계영역 진입을 위한 메시지의 수를 달리 함으로 데드라인 miss 가능성을 줄일 수 있음을 시뮬레이션을 통해 확인하였다.

본 논문에서 사용한 알고리즘외에도 상호 배제를 보장할 수 있는 다른 여러 구조들이 Sanders 알고리즘을 이용해서 만들어 질 수 있고 이에 대한 시뮬레이션도 추가되어야 한다.

참고문헌

[1] Beverly A. Sanders, "The Information Structure of Distributed Mutual Exclusion Algorithms," ACM Transactions on Computer Systems, Vol. 5, No. 3, pp. 284-299, August 1987.

[2] Shiwa S. Fu, Nian-Feng Tzeng, "A Circular List-Based Mutual Exclusion Scheme for Large Shared-Memory Multiprocessors," IEEE Transactions on Parallel and Distributed Systems, Vol. 8, No. 6, pp. 628-639, June 1997.

[3] Andrew S. Tanenbaum, "Distributed Operating System," Prentice-Hall International, Inc. 1995.

[4] Ye-In Chang, "A hybrid distributed mutual exclusion algorithm," Journal of Systems Architecture, V.41 N.10, pp. 715-731, 1996.

[5] Assi Khiat, Mohamed Naimi, "A Solution to the Distributed Exclusion Problem in K-Groups," Proceedings of the Twenty-Second Annual International Computer Software & Applications Conference, pp. 302-307, 1998.

[6] Ting-Lu Huang, "Fast mutual exclusion algorithms using read-modify-write and atomic read/write registers," Proceedings of the 1998 International Conference on Parallel and Distributed Systems, pp. 292-299, 1998.