김일성종합대학학보 주체107(2018)년
정보과학 제64권 제3호

C++언어를 리용한 대기해방대기렬의 한가지 실현방법

리강, 김영남

대기렬자료구조는 많은 쏘프트웨어체계들에서 리용되는 아주 기초적이며 일반적인 공유자료구조이다.

지금까지 공유자료구조에서의 교착현상을 없애기 위한 열쇠해방알고리듬들이 널리 연구되였다.

그러나 대기해방대기렬알고리듬은 많지 않으며 대부분이 Java언어를 리용하여 실현되였다.[1]

론문에서는 프로그람작성에서 널리 쓰이는 C++언어를 리용하는 대기해방대기렬의 한가지 실현방법을 제안하였다. 이 방법은 Alex Kogan이 Michael과 Scott의 열쇠해방대기렬[3]에 기초하여 Java언어를 리용하여 실현하였던 대기해방대기렬[1]을 C++언어로 실현하였다. 또한 론문에서는 기억기의 안전한 해방문제의 해결방안과 위상번호의 고속선택방법을 제안하였다.

1. 개선된 대기해방대기렬알고리듬

론문에서는 우선 매 스레드들이 대기렬에 대한 조작을 시작할 때 위상번호를 선택하는 새로운 방법을 포함하여 개선된 대기해방대기렬알고리듬의 기본원리를 주었다. 다음 대기렬에서 제거되는 마디들에 할당되였던 기억구역을 안전하게 해방하기 위한 방법을 제안하였다.

1) 개선된 대기해방대기렬알고리듬의 기본원리

선행연구[1]에서의 대기렬과 류사하게 론문에서 제안한 대기해방대기렬은 단순련결목록에 기초하고있으며 목록의 머리와 꼬리로 향하는 head와 tail이라고 불리우는 2개의 참조를 가지고있다.

대기렬에서 조작을 시작하는 매 스레드들은 조작을 위하여 이미전에 위상번호를 선택한 스레드들의 위상보다 더 높은 위상번호를 선택한다. 그후 이 수를 대기렬에서 실행시키려고 하는 조작의 추가적인 정보와 함께 특수한 state배렬에 기록한다.

대기렬에 대하여 추가 또는 꺼내기조작을 수행하려고 하는 스레드 t_i는 state배렬을 항행하면서 t_i에 의하여 선택된것보다 작거나 같은 위상번호를 포함하는 입구를 가지는 스레드를 찾는다. 이런 스레드 t_j를 찾으면(그것은 t_i자신일수도 있다.) t_i는 대기렬에서의 조작 즉 추가나 꺼내기조작을 도와주려고 시도한다. t_i는 state배렬에서 t_j의 조작의 구체적인 내용에 대하여 배우게 된다. 마지막에 t_i는 자기의 위상번호보다 크지 않은 위상번호를 가진 다른 모든 스레드들을 도와주려고 시도하였을 때 조작을 호출한 곳으로 안전하게 귀환하여 자기의 조작을 완료한다. 즉 t_i는 대기렬우에서 추가 또는 꺼내기를 진행하는 모든 스레드들의 조작이 자기자신이나 혹은 동시에 실행되고있는 다른 방조스레드에 의하여 완

료된다는것을 확정한다.

선행연구[1]에서는 매 스레드들이 대기렬에서의 조작을 수행하기 전에 위상번호를 우에서 언급한 state배렬을 순환하는 방법으로 선택하였다. 다시말하여 state배렬에 기록되여있는 매 스레드들에 할당되여있는 위상번호를 순환하면서 그중에서 제일 큰 위상번호를 선택하여 그보다 1만큼 더 큰 번호를 새로운 조작의 위상번호로 선택하였다.

그러나 론문에서는 대기렬에 현재의 제일 큰 위상번호를 보관하는 maxPh마당을 추가하여 대기렬에서 조작을 진행하는 매 스레드들이 이 값을 원자적으로 읽어서 새 조작의 위상번호로 선택하고 FAA(Fetch And Add)와 같은 원자적인 조작을 리용하여 이 마당값을 원자적으로 증가시키도록 하였다.

이렇게 함으로써 state배렬을 순환하지 않고도 최대위상번호를 신속정확히 선택할수 있도록 하였다.

2) 기억기의 안전한 해방알고리듬

선행연구[1, 2]들에서는 Java언어를 리용하여 대기해방대기렬을 구현하였으므로 프로그람작성자는 기억기관리에 관심을 돌릴 필요가 없었으며 원자적인 조작과정에 ABA문제가 발생할수도 없었다.

그러나 C언어나 C++언어를 리용하여 작성하여야 하는 경우에 프로그람작성자는 대기렬에서 마디들을 꺼낼 때 그 마디들에 동적으로 할당되였던 기억공간을 해방시켜주어야 한다. 그렇지만 이때 대기렬에서 제거되는 마디들을 아무런 고려도 없이 그저 해방시켜준다면 대기렬에서의 원자적인 CAS조작을 진행할 때 해방되였던 기억공간에 새 마디가 재할당되면서 ABA문제가 발생할수 있으며 이것은 대기렬의 파괴와 프로그람의 오유를 초래하게 된다.

론문에서는 선행연구[2]에서 제기한 Hazard지적자와 류사한 방법으로 이 문제를 해결하였다. 즉 매 스레드는 대기렬에서의 조작을 위하여 CAS원자조작을 수행하기 전에 CAS의 첫 파라메터인 주소에 보관되여있는 원래 값을 해당 스레드의 Hazard지적자로 보관하며 CAS원자조작이 끝난 후에는 이 스레드의 Hazard지적자를 NULL로 회복하여준다.

대기렬에서 마디를 꺼낸 후에 스레드는 특수한 retireNode함수를 호출하여 이 마디를 지적하고있는 Hazard지적자가 있는가를 검사하고 없으면 마디를 해방하여주고 있으면 없을 때까지 기다렸다가 해방하여준다.

이렇게 하면 CAS조작을 진행하는 동안에 값이 원래의 값과 같아지게 변하면서 생기는 ABA문제의 발생을 막을수 있게 된다.

다음의 그림에 론문에서 제안한 마디의 안전한 해방을 위한 retireNode함수의 알고리듬을 보여주었다.

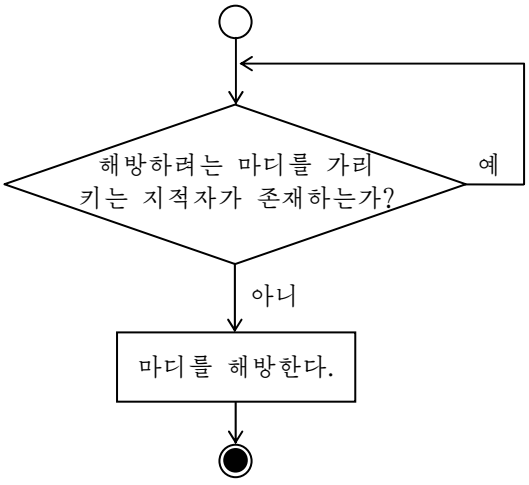

그림. retireNode함수의 알고리듬

2. 제안된 알고리듬의 성능평가

론문에서 제안한 대기해방대기렬과 선행연구[1]에서 Java언어로 실현한 대기해방대기렬에 대하여 동시적인 추가와 꺼내기조작을 진행하여 실행시간을 측정하였다.

표기를 간단하게 하기 위하여 선행연구[1]에서 제안하고 Java언어로 실현한 대기해방대기렬을 선행방법 1로, C++언어로 실현하였지만 최대위상번호를 선택하는 방식은 선행연구[1]와 같은 대기해방대기렬을 선행방법 2로, 론문에서 제안하고 C++언어로 실현한 개선된 대기해방대기렬을 제안방법으로 표시하였다.

다음의 표에 CPU가 Intel® Core™ i7-3770 3.4GHz인 Ubuntu14환경에서 우의 세가지 대기렬에 대한 삽입과 꺼내기조작을 실행한 모의시험결과를 보여주었다.

표. 각이한 대기렬실현방식에서 대기렬조작시간/ms

시험경우	대기렬실현방식		
	선행방법 1	선행방법 2	제안방법
스레드수 8개, 스레드당 조작수 10 000개일 때	2 584	1 795	1 774
스레드수 16개, 스레드당 조작수 10 000개일 때	5 198	3 619	3 573

이 시험결과들은 같은 환경설정에서 10번씩 실행한 후 그 평균값을 준것이다.

성능시험결과로부터 알수 있는바와 같이 최대위상번호를 선택하는 방법을 갱신한 제안방법은 선행방법 2보다 약간 우수한 성능을 보여주었다. 또한 C++언어를 리용한 제안방법이 Java언어를 리용한 선행방법 1보다 대기렬에서의 조작속도가 약 30%정도 빠르다는것을 보여주었다.

맺 는 말

C++언어를 리용하여 여러개의 동시적인 추가와 삭제스레드를 가능하게 하는 개선된 대기해방대기렬의 실현방법을 제안하였다.

선행연구[1]에서 제안한 알고리듬에서 위상번호를 선택하는 방법을 개선하여 성능을 향상시켰으며 C++언어를 리용하여 실현한 대기렬에서 제거되는 마디들에 할당되였던 기억구역을 안전하게 해방하기 위한 알고리듬을 제안하였다.

참 고 문 헌

[1] A. Kogan, E. Petrank; ACM Symposium on Principles and Practice of Parallel Programming (PPOPP), 223, 2011.

[2] M. M. Michael; IEEE Trans. Parallel Distrib. Syst., 15, 6, 491, 2004.

[3] M. M. Michael, M. L. Scott; ACM Symposium on Principles of Distributed Computing (PODC), 267, 1996.

주체107(2018)년 5월 5일 원고접수

A Implementation Method of Wait-free Queue Using C++ Language

Ri Kang, *Kim Yong Nam*

In this paper, we proposed a implementation method of wait-free queue capable of multiple enqueuers and dequeuers using C++ language.

Key words: wait-free, lock-free, queue, C++