단일사용자 다중프로그래밍 환경에서 비례배분이 가능한 페이지대치 알고리즘

서의성[0] 박도현 오승택 이준원
한국과학기술원 전산학과
(ses[0], dhpark, stoh, joon)@calab.kaist.ac.kr

A Proportional Page-replacement Algorithm for Single-user Multi-programming System

Euiseong Seo[0], Dohyun Park, Seungtaek Oh Joonwon Lee
CS Dept. Korea Advanced Institute of Science and Technology

요 약

컴퓨터 시스템의 성능이 발전하면서 동시에 실행되는 프로그램의 수도 증가하게 되었다. 이로 인해 자원을 두고 경쟁하는 프로세스들의 우선순위에 따라 분배되는 자원의 양을 세밀하게 조절하는 기능이 필요하게 된다. 우선순위에 따른 자원의 비례배분 알고리즘에 대한 연구가 많이 있었고, 이를 바탕으로 대부분의 자원을 관리하는 운영체계들도 개발 되었다. 하지만, 가상메모리 자원에 대해서는 자원의 할당량의 모호함과 프로세스가 사용할 메모리 필요량이 예측불가능하기 때문에 적은 연구가 이루어졌을 뿐이다. 본 논문에서는 프로세스가 단위 작업을 수행할 때 발생하는 페이지폴트의 양이 프로세스의 우선순위에 반비례하는 것으로 가상 메모리에 대한 비례배분 개념을 정의 하고, 프로세스 마다 메모리에 대한 요구 형태가 다른 환경에서 이를 가능하게 하는 페이지대치 알고리즘을 제안한다.

제안한 페이지대치 알고리즘을 리눅스 커널상에 구현한다. 구현된 페이지대치 알고리즘을 사용하면 우선순위가 높은 프로세스가 우선순위가 낮은 프로세스에 비해서 페이지폴트를 적게 발생하고, 그 발생량은 반비례 관계에 있음을 보인다.

1. 서 론

컴퓨터의 성능이 발전하면서, 대부분의 시스템들은 다중처리 기능이 점점 더 강화되고 있다. 다중처리 시스템에서는 자원을 두고 경쟁하는 다수의 프로세스가 존재하게 된다. 이 때, 프로세스들간에는 우선순위의 차이가 있고, 운영체계의 바람직한 자원 배분 정책은 자원 사용량을 프로세스의 우선순위에 비례하도록 하는 것이다.

이를 위해 프로세스들 간의 자원을 우선순위에 비례하게 배분하는 개념인 비례배분(proportional sharing)이 제시되었다. 이러한 비례배분 개념을 통해 UNIX의 우선권 시스템 보다 정확한 CPU 자원의 배분이 가능하게 되었다[1]. 비례배분 관련 연구로 CPU 및 네트웍대역과 같은 시분할이 가능한 자원의 비례배분을 위한 알고리즘과 이를 바탕으로 시스템의 자원을 관리하는 운영체계들이 개발되었다.

하지만, 메모리 자원에 대해서는 프로그램들마다 사용하는 메모리의 양과 사용 형태가 예측 불가능하며 그로 인해서 우선순위에 비례하게 배분한다는 개념이 모호한 이유로 극히 적은 연구만이 이루어졌다.

본 논문에서는 전체적인 성능을 위해서는 프로세스가 할당 받는 자원의 양이 정확하게 우선순위에 비례하지 않는 것을 용납하는 단일 사용자 환경을 가정하고, 프로세스가 메모리 때문에 겪게 되는 시간지체가 우선순위에 반비례하도록 하는 것을 가상메모리 시스템에서의 비례배분이라고 정의한다. 이 정의를 바탕으로 가상메모리 시스템의 비례배분이 가능한 페이지대치 알고리즘을 제안하고 Linux에 구현하여, 프로그램의 종류와 상관 없이 우선순위가 높을수록 적은 페이지폴트를 발생시키는 것을 확인하였다.

2. 관련 연구

2.1 비례배분 알고리즘

비례배분 알고리즘은 가중된 순환(weighted round-robin) 알고리즘을 시작으로 공정분배(fair-share), 공정큐잉(fair-queueing)등의 알고리즘이 제시 되었다. 본 논문에서는 제비뽑기 알고리즘(lottery scheduling)[2]을 사용하였다.

제비뽑기 알고리즘은 프로세스가 갖고 있는 우선순위를 제비의 갯수로 환산한 후 운영체계가 자원을 배분할 때, 제비를 뽑아서 뽑힌 제비를 갖고 있는 프로세스에게 배분하는 방식이다. 이 방식은 간단한 구현과 여러 가지 형태의 자원에

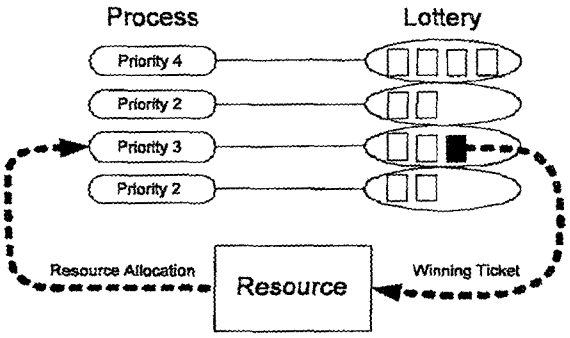

그림 1. 제비뽑기 알고리즘

적용 가능하다는 장점을 지니고 있다.

2.2 가상메모리 시스템의 비례배분

VINO의 비례배분 확장[3], Eclipse[4], Nemesis[5]등 비례배분 알고리즘을 이용한 운영체계들은 대부분 CPU, 네트웍 대역, I/O 대역에 대해서는 비례배분이 이루어지고 있지만, 메모리 시스템에 대해선 기존 UNIX의 NRU(not recently used) 방식을 사용하거나 그림 2와 같이 프로세스의 우선순위에 비례하게 물리메모리의 양을 할당하는 방식을 사용하고 있다.

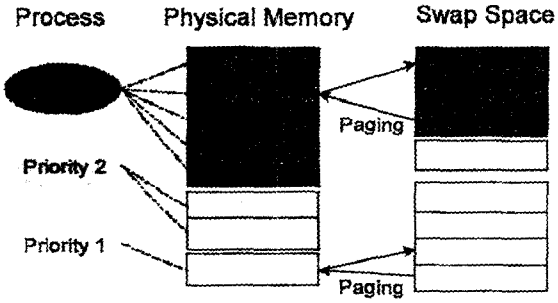

그림 2. 우선순위에 비례한 물리메모리 배분

이러한 방식은 우선순위가 낮지만 사용하는 메모리의 양이 적은 프로세스가 우선순위가 높지만 사용하는 메모리의 양이 많은 프로세스에 비해서 월등히 빠른 수행속도를 갖게 된다. 또한, 프로세스가 사용하는 메모리의 크기는 일정하지 않고 수행과정에 따라 크게 변화하기 때문에 사용할 메모리의 크기를 미리 예측해서 우선순위를 주는 것도 불가능하다. 따라서, 이러한 형태의 접근 방법은 제한된 상황에서만 효과가 있다.

제비뽑기를 이용한 페이지대치 알고리즘[2]도 제안 되었다. 이 방식은 페이지대치시에 제비뽑기를 통하여 희생될 페이지를 고르는 형태이다. 이 알고리즘은 계속해서 우선순위가 낮은 프로세스가 희생되게 되어 기아현상이 발생할 가능성이 높다. 또한, 희생페이지의 선정과 메모리 자원의 할당량과는 직접적인 연관이 없기 때문에 실제로 사용되고 있지는 않다.

3. 비례 배분이 가능한 페이지 대치 알고리즘

3.1 가상메모리 시스템의 비례배분

자원의 비례배분은 프로세스가 자원의 단위 사용으로 인해 소모되는 시간이 우선순위에 비례한다는 개념이다. 메모리의 단위 사용이라는 개념은 모호하기 때문에 본 논문에선 메모리의 비례배분은 단위 작업을 할 경우 메모리에서 소모되는 시간이 우선순위에 반비례함으로 정의하였다. 메모리에서 소모되는 시간은 물리메모리에 접근하는 시간과 페이지폴트가 났을 경우 대치공간에서 물리메모리로 적재하는 시간으로 이루어져있다. 페이지폴트를 처리하는 시간은 물리메모리를 접근하는 시간에 비해서 수천에서 수십만 배의 시간이 필요하게 된다. 따라서, 메모리가 충분하지 않은 상황에서 메모리에서 소모되는 시간은 대부분 페이지폴트로 인해 발생한다고 가정할 수 있다. 제안하는 페이지대치 알고리즘은 여러 프로세스들이 동시에 실행될 때, 발생하는 페이지폴트의 양이 우선순위에 반비례 하게 발생하여야 한다.

프로그램들은 메모리 사용량이 다르기 때문에 단순히 우선순위의 비율로만 페이지폴트의 발생량을 맞추려고 한다면 메모리를 많이 사용하는 프로세스로 인해서 다른 모든 프로세스들이 큰 피해를 볼 수 있다. 따라서, 페이지폴트의 발생량은 프로세스가 사용하는 물리 메모리의 양에도 비례하여야 한다. 이를 통해 우선순위에 비례한 메모리 성능과 전체 시스템의 성능사이에서 균형을 이룰 수 있게 된다.

3.2 제비뽑기를 이용한 페이지대치 알고리즘

제안하는 페이지대치 알고리즘은 제비뽑기 페이지대치 알고리즘을 확장하였다. 페이지폴트가 발생하여 희생페이지를 선정해야 할 경우 운영체계는 3.1의 조건에 의해 프로세스에게 식1과 같은 양의 제비를 할당한다.

$$제비수 = \frac{프로세스가\ 사용하는\ 물리메모리}{우선순위 \times 지난\ 수행에서의\ 페이지폴트량}$$

수식 1. 프로세스가 갖는 제비표의 양

식1에서 지난 수행에서의 페이지폴트 발생량은 프로그램의 메모리 요구형태가 예측불가능하기 때문에 피드백을 통해 원하는 조건을 얻기 위한 항이다. 프로세스는 위와 같은 수의 제비를 할당 받기 때문에 우선순위가 높

은 프로세스일수록 희생프로세스가 될 가능성이 적어진다. 또한, 우선순위가 높다 하더라도 사용하는 메모리가 많거나 페이지폴트의 발생량이 우선순위에 비해 상대적으로 적다고 하면 희생프로세스가 될 가능성이 높아지게 된다. 이러한 적응적인 방식으로 프로그램의 메모리 사용형태를 알 수 없는 상황에서도 우선순위에 비례한 메모리 성능을 기대할 수 있다.

4. 구현 및 성능평가

제안한 알고리즘을 리눅스 2.2.18 커널에 구현하였다. 구현한 커널을 사용하여 SPECCPU 2000 벤치마크에서 메모리의 사용량이 10메가 바이트에서 100메가 바이트까지 증가하는 vortex와 약 50메가 바이트의 메모리를 할당 받아 끝까지 사용하는 equake 프로그램을 기준으로 다음과 같은 실험을 하였다.

a. vortex : 두 개의 vortex 프로세스를 우선순위를 1:2로 주고 동시에 실행하였다.

b. equake : 세 개의 equake 프로세스를 우선순위를 1:2:3으로 주고 동시에 실행하였다.

성능평가에 사용한 환경은 표1과 같다.

CPU	Pentium III 600Mhz
RAM	128 Mbytes
OS	RedHat 7.0
Kernel	Linux 2.2.18
Run Level	1 (single user)

표 1. 성능평가 환경

프로세스들이 한 번의 수행을 하는데 발생한 페이지폴트의 발생량은 다음과 같았다.

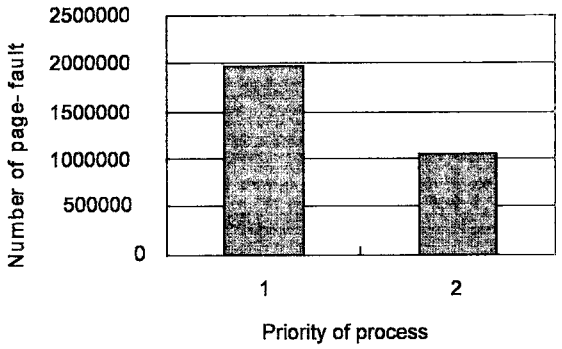

표 2. vortex의 페이지폴트 발생량

표 2에서 볼 수 있듯이 vortex의 경우 거의 우선순위에 반비례하게 페이지폴트의 비율이 나타났다.

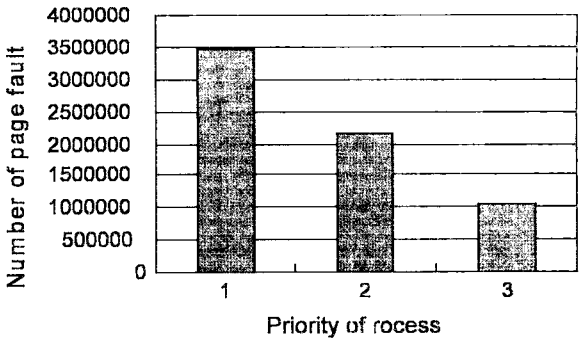

표 3. equake의 페이지폴트 발생량

equake의 경우 페이지폴트의 발생량은 정확하게 반비례하지는 않았지만, 3:2:1의 비율로서 1:2:3의 우선순위의 비에 반비례인 6:3:2의 비율에 매우 가까운 결과를 나타냈다.

이를 통하여 제안한 페이지대치 알고리즘은 프로세스가 단위 작업을 할 때의 페이지폴트 발생량이 프로세스의 우선순위에 반비례 하게 발생하도록 한다는 것을 알 수 있다.

5. 결론

본 논문에서는 프로세스가 발생시키는 페이지폴트양을 이용하여 가상메모리 시스템의 비례배분 개념을 정의하고, 그 정의를 바탕으로 비례배분이 가능한 페이지대치 알고리즘을 제안하였다. 제안한 알고리즘은 우선순위에 반비례 관계를 보이는 페이지폴트 발생량을 보여주었다. 향후 과제는 알고리즘을 확장하여 보다 정확하고 능동적인 알고리즘을 개발하는 것이다.

7. 참고 논문

[1] J. Nieh 외 2인, *Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler*, Proceedings of the 2001 USENIX Annual Technical Conference, 2001

[2] C. Waldspurger 그리고 W. Weihl, *Lottery Scheduling: Flexible Proportional-share Resource Management*, The 1st Symposium on Operating System Design and Implementation, 1994

[3] D. Sullivan 그리고, M. Seltzer, *Isolation with Flexibility: A Resource Management Framework for Central Servers*, Proceedings of the 2001 USENIX Annual Technical Conference, 2001

[4] S. Hand, *Self-Paging in the Nemesis Operating System*, The 3rd Symposium on Operating System Design and Implementation 1999