QoS 을 보장하는 멀티미디어 서버 설계

현은실*, 김태윤*
*고려대학교 컴퓨터학과
e-mail : eunsil@netlab.korea.ac.kr

Design of QoS Multimedia server for Disk I/O

Eun-sil Hyun*, Tai-Yun Kim*
*Dept. of Computer Science and Engineering, Korea University

요 약

본 논문에서는 서버에서 다수의 사용자에게 효율적인 서비스를 제공하기 위하여 저장공간과 네트워크 인터페이스 사이에 직접적으로 데이터를 전달하는 Splice 알고리즘과 하나의 큐에 다수의 요청을 SCAN 순서로 정렬하고 새로운 요청을 받을 경우 우선순위와 마감시간을 고려하여 사용자의 요구를 만족시켜주는 알고리즘을 제한하였다. Splice 알고리즘은 서버가 디스크에서 사용자가 원하는 데이터를 읽어 직접 연결을 하기 때문에 자원을 절약할 뿐만 아니라 시간상에서도 많은 향상을 보여준다. 그리고 One-Queue 알고리즘은 기존의 멀티큐 알고리즘에 비해 낮은 우선순위의 요청을 처리하는 수도 많아지고 많은 양의 서비스를 처리하기 때문에 성능향상에도 이바지 할 수 있다.

1. 서론

같은 시간에 다수의 사용자에게 효율적인 서비스를 제공하게 위해 서버 시스템은 주기적으로 QOS 보장을 제공하면서 데이터를 저장 부시스템에서 네트워크 인터페이스로 전달해야 한다. 기존의 시스템은 QOS을 위한 보장뿐만아리라 저장공간과 네트워크 사이에 효율적인 데이터 전송이나 통제를 제공하지 않았다. 이 논문에서는 커널의 버퍼 캐시에서 커널의 소켓 버퍼로 데이터를 직접 전송하는 Splice 알고리즘과 하나의 큐에 사용자가 요청한 작업을 SCAN 알고리즘으로 정렬시키고 새로운 서비스가 요청되어지면 마감시간과 우선순위를 고려하여 서비스 순서를 계산하는 One-Queue 알고리즘을 제안한다.

파일시스템에서는 데이터를 사용자 공간에서 저장공간으로 보내는 데 버퍼 메커니즘을 사용하고 있는 동시에 네트워크 프로토콜 스택은 소켓 버퍼라는 사용자 공간에서 네트워크인테페이스로 직접 연결되는 다른 버퍼 시스템을 사용하고 있다. 두 번의 버퍼 구조는 불필요한 데이터 복사를 야기하고, 각각의 사용자들의 복사는 결국 처리율과 서비스가 가능한 사용자의 수를 줄일 것이다. 그러므로 저장공간과 네크워크 사이에 복사를 줄일 수 있는 Splice 알고리즘의 필요하다. 네트웨크로 전달하는 과정에 있어서 각각의 데이터 형식은 마감시간과 우선순위를 가지고 있다. 연속적인 데이터는 마감시간의 제한을 가지고 있기 때문에 요청이 마감시간안에 처리되지 않으면 부자연스러운 서비스 환경을 제공할 수 있다. 따라서 연속적인 데이터의 경우나 비연속적인 데이터의 경우를 우선순위를 기반으로 서비스를 제공하여야 한다. 그러나 기존의 다중 큐 알고리즘은 낮은 높은 우선순위의 요청을 처리하기 위해서 낮은 우선순위의 요청을 포기하는 경우가 많았다. 이 방식은 우선순위 클래스 별로 하나의 큐를 두고 SCAN 알고리즘이나 EDF 알고리즘이나 SCAN-EDF 의 알고리즘을 사용하여 요청을 처리하기 때문에 우선순위가 낮고 마감시간이 여유가 있는 요청을 기아 현상이 발생하는 경우가 많아 결국은 성능을 저하하는 결과를 초래하였다. 따라서 이 논문에서는 우선순위가 낮은 서비스의 요청도 받아들이면서 마감시간과 디스크상의 데이터의 위치를 고려한 새 알고리즘이 필요하다. 높은 우선순위의 요청을 처리하되 낮은 우선순위의 요청의 서비스 요소를 만족시킴으로써 처리율을 극대화 시키는 알고리즘을 제한

한다.
제 2 장에서는 기존의 시스템과 새로이 제한한 Splice 알고리즘을 비교하여 설명하고 3 장에서는 다중큐 방식의 스케줄링과 비교하여 One-Queue 방식의 스케줄링을 설명하겠다. 마지막으로 4 장은 결론을 내리고 향후 연구 과제를 제시한다.

2. Splice 알고리즘

2.1 기존의 디스크 I/O 메커니즘과 제한사항

디스크에서 네트워크 인터페이스로의 데이터 전달 경로는 두 번의 메모리 복사를 필요로 한다. 첫 번째 복사는 커널 버퍼 캐시에서 사용자 공간의 버퍼로, 두 번째 복사는 소켓에 의하여 사용자 공간의 버퍼에서 커널안에 소켓 버퍼에 복사되는 것을 말한다. 이러한 접근은 텍스트나 이진파일 같은 작은 양의 데이터를 접근하는 방식으로 적당하다. 그러나 오디오, 비디오, 그리고 애니메이션 같은 멀티미디어 데이터는 우선 메모리 공간을 많이 사용하고, 그들이 연속적으로 변화를 하기 때문에 아주 짧은 시간동안 만이 관계가 있기 때문에 버퍼 캐시에서 조회해온 멀티미디어 데이터는 성능 측면에서 큰 이익을 주지 않는다. 또한, 디스크에서 네트워크 인터페이스로의 데이터 전달을 하는 응용프로그램은 두 개의 다른 버퍼 시스템이 필요하기 때문에 이것은 과도한 데이터 복사이며 시스템 호출 오버헤드이다. 많은 양의 데이터를 메모리 로부터 메모리로 이동하는 것은 프로세서 시간의 낭비일 뿐만 아니라 메모리와 시스템 버스 대역폭을 소비하는 것이다.

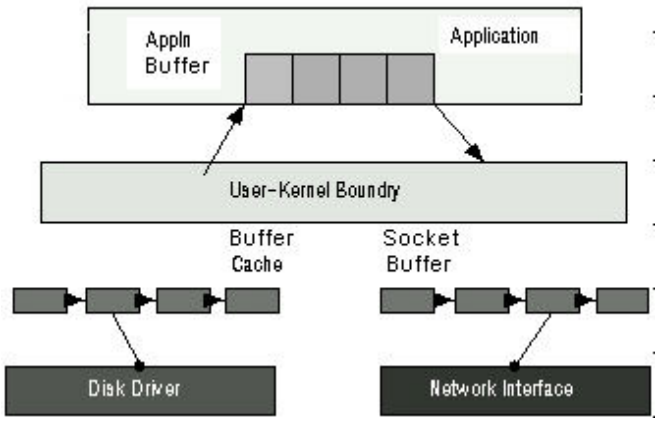


그림 1. 기존의 메모리 기반 파일의 I/O 시스템

그림 1 에서는 기존의 디바이스 사이에 데이터의 전달 방식으로 항상 사용자 공간을 통과하여 read(), write() 방식을 사용하여 데이터가 어떻게 전달되는 경로를 보여주고 있다. 데이터를 사용자 공간에서 저장공간으로 보내는 데 버퍼 메커니즘을 사용하고 있는 동시에 네트워크 프로토콜 스택은 소켓 버퍼라는 사용자 공간에서 네트워크 인테페이스로 직접 연결되는 다른 버퍼 시스템을 사용하고 있다. 두 번의 버퍼 구조는 불필요한 데이터 복사를 야기한다. 그리고 커널 버퍼에서 사용자 공간의 버퍼로 이동하고 또 사용자 공간에서 커널의 소켓 버퍼로 이동하면서 두번의 시스템 호출이 사용된다.

2.2 Splice 알고리즘 설계

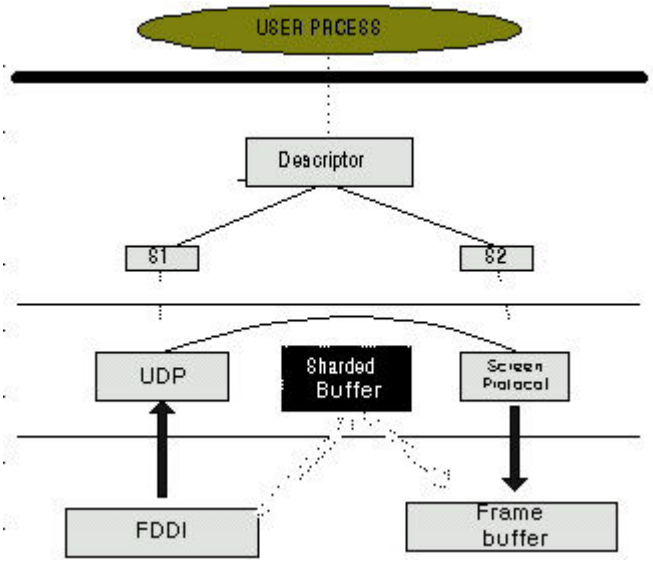


그림 2. Splice 알고리즘을 기반으로 한 메커니즘

Splice 알고리즘은 기술어를 사용하여 커널상에서 저장 공간과 네트워크 인데페이스 사이에서 데이터의 전달이 파일 기술어 f1, f2 을 참조함으로써 직접적으로 이루어진다. 이 방식은 버퍼 시스템을 제거하기 때문에 커널상에서의 버퍼(Shared burrer) 를 공유하여야 하며 커널상에서의 데이터 입력은 파이프과 같은 방식으로 데이터의 출력이 된다. 중요한 것은 응용프로그램은 Splice 알고리즘과 기존의 read/write() 시스템 호출 방식을 데이터의 흐름을 기반으로 하여 병행하여 사용해야 한다. 만약 중간단계의 프로세싱이 필요한 경우 기존의 방식인 read/write() 방식의 시스템 호출을 사용하고, 중간단계의 프로세싱이 필요하지 않을 경우는 Splice 시스템 호출을 사용해도 될 것이다.
불필요한 데이터의 복사를 줄이고 context switch 의 오버헤드를 줄임으로써 성능향상을 할 수 있다. 따라서 연속적인 멀티미디어를 기반으로 한 응용프로그램상에서는 일정한 시간을 기준으로 하여 전달하는 프레임의 수를 증가시킬 수 있고 연속적인 디스플레이 환경을 자연스럽게 처리해 준다. 버퍼 인터페이스를 제거함으로써, read / write() 의 동작을 하나의 시스템 호출로 모두 처리할 수 있다. 그리고 불필요한 컨텍스트 스위칭을 막기 때문에 CPU 자원을 절약할 수 있고, 가상 메모리에 대한 성능을 향상시킬 수 있다.

3 . One – Queue 디스크 스케줄링

3.1 다중큐 방식과 제한사항

멀티미디어 서버는 텍스트, 이미지, 비디어, 오디오 등의 다양한 종류의 데이터 형식을 저장한다. 각각의 데이터 형식은 고유의 마감시간과 우선순위를 가지고 있다. 오디오나 비디오 같은 연속적인 데이터는 고정된 일정한 시간안에 배달되어져야 한다는 마감시간의 제한을 가지고 있다. 만약 요청이 마감시간 전에 서비스가 되지 않으면, 디스플레이 하는 스테이션에서 연속적이지 못한 부자연스러운 디스플레이 환경을 한다. 이러한 중단과 지연을 총괄하여 손실이라 정의한다면 손실을 최소로 하기 위해서는 비연속적인 데이터의 처리는 비디오나 오디오 같은 연속적인 데이터에 비해 비교적 낮은 우선순위를 가진다. 그러나 높은 우선순위이 스케줄링 기법은 낮은 우선순위의 요청에 기아 현상을 일으키게 하는 주범이 된다.

기존에 있던 다중 큐 알고리즘은 각각의 우선순위 클래스 별로 하나의 큐를 유지하고 우선순위에 의해 서비스를 제공하는 방식을 제공하였다. 각각의 우선순위를 가진 큐는 다른 방식에 의해 정렬되어 진다.

- 작업이 큐에 도착한 시간에 의해 정렬되어진다. (FIFO)
- 마감시간을 기준으로 정렬되어진다. (EDF)
- 디스크에 위치를 기준으로 정렬되어진다. (SCAN)
- 마감시간과 디스크의 위치를 기준으로 정렬되어진다. (SCAN-EDF)

이 알고리즘은 낮은 우선순위의 요청을 포기하는 대가로 높은 우선순위를 가진 요청을 처리하였다. 만약 Pi 의 우선순위가 Pj 보다 높을 때, Pj 의 우선순위의 큐로부터의 요청은 만약 Pi 의 우선순위의 큐안에서 임박한 요청이 없을 경우에 스케줄 되어진다. 일반적으로 이 알고리즘은 응답시간이나 손실율을 기준으로 측정한 최선의 성능향상의 보여주지만 낮은 우선순위를 가진 요청의 대가로 이루어진 것이기 때문에 전체적인 시스템의 성능향상은 떨어진다.

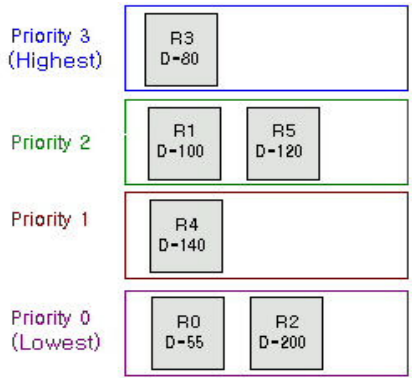


그림 3 다중 큐 방식의 스케줄링

그림 3 에서와 같이 우선순위를 0 에서 3 까지의 4 단계로 나누고, 3 은 높은 우선순위로 0 을 낮은 우선순위로 정의한다. 각각의 요청은(R0 - R5) 그들의 우선순위를 가진다. 이 환경에서 다른 마감시간과 우선순위를 가진 요청이 들어오고, 각각의 요청은 20msec 의 서비스 시간이 필요하다면, 낮은 우선순위를 가진 R0 는 55mec 의 마감시간을 가지고 있다. R0 는 R3 이 서비스가 다 이루어질때까지 서비스 되어지지 않는다. R1, R5, R4 는 모두 올바른 순서로 서비스 되어진다. 결국 우선순위가 낮은 R0 는 마감시간전에 서비스를 받을 수 없다. 그러나 R3 후에 R0 이 즉시 서비스를 받는다면 모든 요청을 마감시간안에 처리할 수 있다.

3.1 One-queue 스케줄링 알고리즘 설계

One - Queue 알고리즘은 높은 우선순위의 요청 뿐만아니라 낮은 우선순위의 서비스 할수 있는 방법이 제한되어야 한다. 먼저 탐색 시간을 최소화하고 디스크 처리율을 극대화 시키기 위해서 큐는 SCAN 알고리즘을 통하여 처리한다. 만약 새로운 작업이 요청된다면 새로운 알고리즘은 세가지의 요소를 ⓐ 우선순위 ⓑ 디스크 헤드의 움직임 ⓒ 마감시간 을 고려하여 서비스의 순서를 정한다. 하나의 SCAN 싸이클에 속하는 요청은 디스크 헤드의 위치에 관계하여 오름차순이나 내림차순으로 정렬한다. 스케줄링 큐가 만약 모든 요청을 마감시간을 어기지 않고 모두 처리한 상태를 유효하다고 정의하면, 큐는 초기에 유효한 상태이다. 만약 새로운 요청을 받아 Rnew 가 Ri 와 Ri+1 사이에 삽입된다면, 큐의 상태를 여전이 유효한 상태로 유지하기 위해서 다시 정렬을 해야한다. 만약 Rnew 의 삽입이 유효하지 않은 상태가 된다면, (적어도 하나이상이 마감시간을 지키지 못한다면) 알고리즘은 유효한 상태를 회복시키기 위해서 일련의 단계를 가지고 있어야 하며 다음과 같은 원칙에 의해 정렬이 되어야 한다.

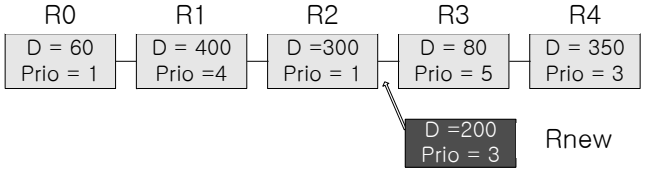
① Rnew 의 삽입은 높은 우선순위를 가진 다른 요청들의 마감시간을 놓치지 말아야 한다.

② 마감시간을 못 지킨 요청의 수를 최소화 시켜야 한다.

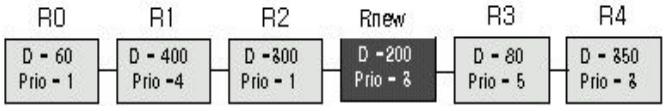
③ 유효한 상태를 복구하는데 있어서, SCAN 순서를 혼합하든 새로운 SCAN 순서를 만들간에 대역폭의 효율성을 저하해서는 안된다.

새로운 요청이 도착했을 때, 만약 Rj (Rj 가 이전에 있던 요청이거나 새로운 요청일 경우)의 마감시간을 놓쳤을 때, 먼저의 위치에서 큐의 마지막으로 이동시킨다. 그러나 이동시킨 요청은 이동시킴으로써 마감시간을 지키지 못할 수 있다. 그러므로 이동할 요청을 선택할 때에는 낮은 우선순위를 가진 요청을 선택해야 한다. 그리고 만약 새로운 요청이 낮은 우선순위를 가지고 있다면 그것을 큐의 마지막으로 이동시킴으로 문제를 해결할 수 있다. 이동하는 작업을 선택하는데 있어서는 낮은 우선순위의 순서대로, 같은 우선순위를

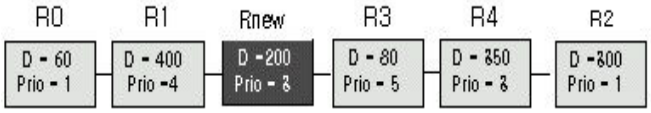
가진 요청들은 마감시간이 넉넉한 것부터 선택한다. 선택된 요청은 스케줄이 유효한 상태로 될 때까지 큐의 마지막으로 옮겨진다. 따라서 이것들은 마감시간의 여유가 있으며 낮은 우선순위를 가진 요청들이다. 밀려난 요청은 마감시간을 놓치기 쉽다. 만약 요청이 마감시간을 놓칠 경우, 낮은 우선순위의 작업이 큐의 마지막으로 밀려난다.



A) Rnew3 이 들어오기 전



B) Rnew 이 들어온 후 (R3 이 마감시간을 놓침)



C) 재배열

그림 4. One-Queue 방식의 스케줄링

그림 4 에서 알수 있듯이 하나의 큐에는 우선순위가 다른 몇 개의 요청을 있다. 높은 값은 높은 우선순위를 의미하고, D(Deadline)는 각 요청들의 마감시간을 의미한다. 각 요청은 20msec 의 정해진 서비스 타임을 사용한다. A) 상태는 새로운 요청의 삽입이 이루어지기 전이며 B) 상태는 우선순위가 3 이고 D 가 200 인 SCAN 알고리즘에 따라 R3 앞에 삽입된 경우이다. 이때 R3 은 마감시간을 놓치게 된다. 그러므로 큐의 순서는 재배열해야 한다. R3 앞의 다수의 요청들 사이에서 우선순위가 가장 낮고 마감시간에 여유가 있는 요청을 찾아 선택한다. C) 와 같이 원칙에 따라 R2 가 선택되어 지고 R2 는 큐의 가장 마지막으로 이동하게 된다.

제한한 알고리즘의 개요는 다음과 같다.

⑴ 새로운 서비스가 요청된다면, SCAN 순서에 따라 삽입시킨다.

⑵ 만약 마감시간을 어기는 요청이 없을 경우에는 알고리즘이 끝낸다.

⑶ 만약 요청 Rj(큐에 이미 있던 요청이거나 새로운 요청일 경우 모두 다)가 마감시간을 넘길 경우,

· Rj 보다 선행하여 실행되는 요청들중에서 가장 낮은 우선순위를 가진 요청을 찾아서 큐의 마지막으로 이동시킨다.

· 다수의 후보 요청들이 있다면, 그중에서 가장 여유가 있는 마감시간을 가진 요청을 선택한다.

· 영향을 받은 요청의 서비스 시간을 계산한다.

· 만약 마감시간안에 서비스를 받지 못할 경우, 마감시간을 지키지 못했음을 선언하고 큐에서 제거한다.

⑷ ⑵단계로 돌아간다.

4.결론 및 향후 연구 과제

본 논문에서는 저장공간과 네트워크 인터페이스 사이에 직접적으로 데이터를 전달함으로써 데이터의 복사와 사용자 공간과 커널 공간사이의 시스템 호출을 줄여 서버 입장에서 다수의 사용자에게 좀더 빠르고 효율적인 서비스를 제공하도록 Splice 알고리즘과 하나의 큐에 다수의 요청을 SCAN 순서에 의해 정렬하고 마감시간과 우선순위를 고려하여 만약 마감시간을 지키지 못하는 요청이 있을 경우 사용자의 요구를 만족시켜주는 One-Queue 알고리즘을 제한하였다. 다중큐 방식은 각각의 우선순위 레벨을 위한 각각의 큐를 가지고 높은 우선순위를 위한 큐가 비어있을 경우에만 낮은 우선순위의 작업을 서비스 해준다. 따라서 이 방식은 높은 우선순위의 요청을 처리하는 데는 만족하지만 상대적으로 낮은 우선순위를 가지는 작업을 처리하는데 있어서 기아 현상을 야기한다. 따라서 이 논문에서 제시한 알고리즘은 하나의 큐를 두고 세가지의 요소를 만족시키면서 서비스를 제공하기 때문에 낮은 우선순위의 요청을 처리하는 수도 많아지고 많은 양의 서비스를 처리하기 때문에 성능향상에도 이바지 할 수 있다.

향후 연구과제로는 Splice 알고리즘과 One-Queue 알고리즘을 리눅스 환경에서 구현함으로 확실한 성능향상이 있음을 보이는 것이다.

참고문헌

[1] Ibrahim Kamel and T. Niranjan, A Novel Deadline Driven-Disk Scheduling Algorithm for Multi-Priority Multimedia Objects, 1999

[2] Milind M.Buddhikot, Enhancement to 4.4 BSD UNIX for Efficient Networked Multimedia in Project MARS, 1994

[3] Kevin Fall, Improving Continuous-Media Playback Performance with In-Kernel Data Paths, 1995

[4] Robert k. Abbott and Hector Garcia-Molina, Scheduling I/O Requests with Deadlines : a Performance Evalution, LEEE, 1990

[5] T.N. Hiranjan , Implementation and Evalution of a Multimedia File System, IEEE, 1997

[6] Y.Rompogiannakis and G.Herjes, Disk Scheduling for Mixed-Media Workloads in a Multimedia Server, ACM Multimedia '98, Bristol. Uk, 1998

[7] Ravi Wijayaratne and A.L. Harasimha Reddy, Integrated QOS management for disk I/O, IEEE, 1999