VOD 스트리밍 데이터를 위한 Consistency 알고리즘 개발

장승주*

*동의대학교 컴퓨터공학과

Development of Consistency Algorithm for VOD streaming Data

Seung-Ju Jang*

*Dept. of Computer Engineering, Dong-Eui University

e-mail : sjjang@deu.ac.kr

요 약

본 논문에서는 VOD에서 스트리밍 데이터를 효율적으로 서비스할 수 있는 Consistency 알고리즘을 제시하고 개발한다. 본 논문에서는 일반적인 계산 용도로 사용되는 barrier 메커니즘을 미디어 플레이 최소 단위인 (SH, GOP) 미디어 데이터에 서비스할 수 있도록 한다. VOD 시스템에서 RTP 패킷 데이터는 빠른 미디어 서비스 제공을 위하여 본 논문에서 제시한 consistency 알고리즘을 설계 및 구현하였다. 또한 이 알고리즘에 추가 기능으로 시간 동기화에 따라 공유메모리에 존재하는 멀티미디어 데이터의 순서화된 서비스를 보장하는 것이다. 그리고 예상 페이징 기법을 적용하여 효율적인 버퍼링 기능을 보장하는 것이다. 그리고 본 논문에서 제시한 알고리즘에 대한 성능 평가를 실시하였다.

ABSTRACT

This paper proposes and implements consistency algorithm that is serviced streaming data efficiently in VOD system. The media data is loaded into each node by Round Robin method. The barrier mechanism is changed into the minimum data factor(SH. GOP) of media data in this paper. In addition in order to fast media data service, the additional features are implemented in the consistency algorithm. Additional feature of the consistency algorithm is added. First, time synchronization algorithm is added the suggested consistency algorithm. Second, the prepaging mechanism supports efficient buffering service. I experimented the suggested consistency algorithm on two DSM systems.

키워드

VOD 시스템, Consistency 알고리즘, DSM(Distributed Shared Memory)

Ⅰ. 서론

초고속 통신망을 통한 인터넷의 발달은 많은 사회적 변화를 일으켰지만, 그 중에서도 제일 큰 변화를 가져온 것은 영상 산업 분야일 것이다. 현재 VOD(Video On Demand)에 대한 수요가 급증하여 다양한 VOD 서비스가 도입되고 있는 상황에서 데이터 및 사용자의 증가에 따른 문제점을 해결하는 것이 필수적이다.

본 논문은 분산 공유 메모리 방식의 프로그래밍을 이용하여 VOD 스트리밍에서 제공되는 MPEG 미디어 데이터를 분산처리 할 수 있는 Consistency 알고리즘을 설계한다. 이 알고리즘은 barrier 메커니즘을 이용하여 미디어 데이터에 대한 동기화 메커니즘을 적용함으로써 효율적인 데이터 일관성을 유지한다.

본 논문의 구성은 2장에서 관련 연구를 살펴보고, 3장에서는 Scope Consistency의 Barrier 메커니즘을 통한 미디어 데이터의 분산 처리 알고리즘을 제시하며, 4장에서는 DSM 과 관련한 실험을 설명한다. 마지막으로 5장에서 결론을 맺는다.

Ⅱ. 관련 연구

2.1 VOD 스트리밍

VOD와 AOD 같은 주문형 멀티미디어 서비스를 지원하는 대표적인 프로토콜로는 H.323, SIP, RTSP 등이 있는데, 이들 프로토콜들은 RTP(Real-time Transport Protocol)[1]를 멀티미

디어 데이터 전송 프로토콜로 채택하고 있다.

멀티미디어 컨텐츠를 수신하여 재생하기 위해서는 적절한 타이밍을 요구하는데, RTP는 이를 위해서 time stamping, sequence numbering과 같은 메커니즘을 제공한다. 이 메커니즘을 이용하여 서로 다른 스트림 데이터를 동기화시킨다.

2.2 JIAJIA

JIAJIA는 scope consistency를 이용하여 분산 공유메모리 시스템에서 테이타 처리를 보장한다. JIAJIA에서는 세 가지 동기화 기법을 제공하는데, lock, barrier와 조건 변수가 있다. 그 중 barrier는 모든 프로세스들의 처리가 완료될 때까지 어떠한 프로세스로부터의 처리를 금지함으로써 전역적인 동기화 기법을 제공한다[2].

2.3 Consistency Model

DSM(Distributed Shared Memory)은 병렬화(parallelism) 컴퓨팅을 지원한다. DSM 방식으로 구현하게 되는 스트리밍 서버는 한 대의 서버를 통해 제공되는 스트리밍 서비스를 여러 대의 서버로 병렬 처리하여 클라이언트에게 스트리밍 서비스를 제공해줌으로써 좀 더 향상된 성능의 서비스를 제공해 줄 수 있다.

DSM은 요구되어지는 다른 프로세서로부터의 메모리 내용을 가져올 수 있어야 한다. 이것은 다른 물리적인 메모리와 같이 공유된 메모리의 다중 복사를 초래한다. DSM은 이들 다른 복사들의 일관성(consistency)을 유지해야 하며, 공유 메모리에 접근한 프로세서는 올바른 결과를 반환해야 한다. 이러한 작업은 Memory Consistency Model이 담당한다[3, 8].

2.3.1 Sequential Consistency

공유 메모리 다중 프로세서를 위해 고안된 가장 일반적인 메모리 일관성 모델은 Sequential Consistency이다.[5] 다음 표 1.은 Sequential consistency에서 같은 네트워크를 구성하는 모든 프로세서간에 데이터 처리 방법을 나타낸 것이다 [4].

표 1. Sequential consistency 데이터 처리 방법 예제

순서 / 프로세서	1	2	3	4	5
p1	W(x)1				
p2			W(y)2		
p3		R(y)2		R(x)0	R(x)1

표 1.은 Sequential consistency 모델을 사용함으로써 각 프로세스에 대한 변수의 상태와 공유 메모리에 접근하는데 순서를 가지고 접근하는 것을 보여주고 있다. 프로그램을 수행하는데 있어 순서가 정해져 있다.

2.3.2 Eager Release Consistency(ERC)

ERC 모델은 write 접근 정보가 다음 접근 때의 복사로 인한 지연을 줄이기 위하여 release point에 모든 공유 복사가 제공되며, Release는 모든 cacher들로부터 인정을 받을 때까지 block된다. 접근 실패시, 메시지는 페이지를 위한 디렉토리 관리자로 보내진다.

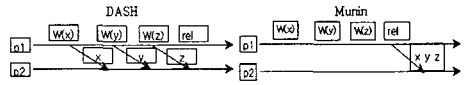

그림 1. DASH와 Munin의 메시지 전송 방법

그림 1.의 DASH는 remote memory writes pipeline처리와 release point에서의 동기화를 하며, Munin에서는 remote memory writes 기록과 하나의 메세지로 같은 목적지를 가지는 모든 write들의 병합 메세지 교환을 감소시킨다[5].

2.3.3 Lazy Release Consistency(LRC)

LRC 모델의 consistency message는 단지 마지막 releaser와 새로운 acquirer 사이에서만 나타난다[6].

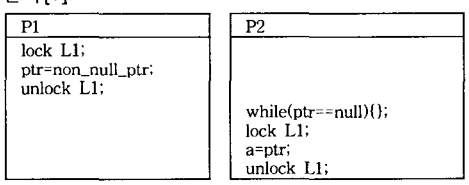

그림 2. LRC와 ERC의 알고리즘

그림 2.은 ERC와 LRC에서 볼 수 있는 동작으로 새로운 non-null pointer가 P1에 의해 unlock(release)전에 P2로 전달되는 것을 보장한다. LRC일 경우, P2가 lock을 실행할 때까지 P1은 쓰기를 하지 않는다. while loop에서 read가 있기 전에나 synchronization으로 분류하기 전에, 적절한 acquire synchronization이 위치해야 한다.

2.3.4 Entry Consistency

Entry consistency 모델은 lock에서 exclusive와 non-exclusive 접근으로 구별된다. 주어진 lock과 함께 연관된 변수들을 write하기 위해서 프로세스는 lock을 소유해야 하고 exclusive 모드에서 대응하는 lock을 얻어야 한다. non-exclusive 모드에서 다중 프로세스들은 같은 lock을 얻을 수는 있지만 연관된 변수들에 대해 단지 read만 가능하다 [7].

2.3.5 Scope Consistency(SCC)

ScC 모델[8]에서는 scope 개념을 같은 lock을 사용하는 모든 임계영역들로 정의된다. scope는 acquire 에서 opened되고 release에서 closed된다.

III. 제안하는 Consistency 알고리즘

3.1 공유 메모리를 이용한 VOD 스트리밍 데이터 처리 방식

논문에서 제안하는 알고리즘을 구현하기 위한 데이타 처리 방식은 하나의 호스트에 한 프래그먼트 크기의 공유메모리를 할당한다.

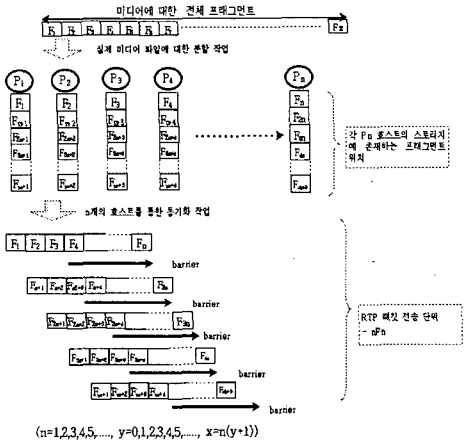

그림 3. Barrier 메커니즘을 통한 미디어 파일의 데이타 처리 과정

그림 3.에서는 공유메모리의 미디어 데이터에 barrier 메커니즘을 적용하여 RTP 패킷으로 전송하게 되는 과정을 보여준다. 미디어 파일을 구성하는 프래그먼트들은 F1부터 Fx로 구성한다. 이들 프래그먼트들은 라운드로빈 방식으로 각 호스트의 스토리지에 저장된다. 공유 메모리의 프래그먼트들은 barrier 메커니즘을 통하여 제어한다. barrier 메커니즘은 공유 메모리에 대한 사용이 모든 호스트에서 완료될 때까지 대기하게 되고, 공유 메모리의 사용이 완료되면 공유메모리 데이터 값을 서비스 할 수 있다.

3.2 미디어 파일 처리를 위한 동기화 알고리즘

미디어 데이타를 서비스하기 위한 프래그먼트에 대한 동기화 알고리즘은 다음 알고리즘 1.과 같다.

```
미디오 데이터(MPEG)에 대한 (SH, GOP)를 기본 단위로 하여
각 노드에 분산하여 저장 (SH,GOP)의 크기 → SH_GOP
각 호스트의 분산 처리를 위해 비디오 서버들( N 대 )을 연결
각 호스트의 프래그먼트 처리를 위한 공유 메모리 설정
if( 클라이언트로부터 미디어 파일에 대한 요청이 발생하면 ){
   while( 미디어 파일에 대한 프래그먼트를 다 읽을 때까지 반복 ){
     AP플래그 활성화에 따른 코드 시작 위치
       if( 각 호스트의 pid가 n 이면 ) {
          AP플래그 비활성화
          n 번째 호스트의 저장 노드에 지정된 seek point의
          프래그먼트 읽기 공유 메모리에 읽어 온 프래그먼트를
          저장 → 프래그먼트가 저장되는 위치 결정
          ⇒ 공유메모리의 주소:((SH_GOP×n)의 주소)에 저장
          n 번째 호스트의 저장 노드에 다음 프래그먼트의
          seek point 지정
          if(공유 메모리가 full이면) {
            공유메모리 데이터의 순번 지정
            이전 순번 데이터가 서비스 중이면 현재 순번 대기
            공유메모리의 데이터를  RTP 패킷 전송 단위로 선정
            if ( pid가 0 이면 ) {
              추후에 생성되는 순번의 공유 데이터에 대한 서비스
              방지 서비스될 때 AP플래그 활성화와 선정된
              데이터를 서비스
            }
          }
          공유 메모리에 N개의 프래그먼트 저장을 위한 동기화(barrier)
       }
   }
}
```

알고리즘 1. 미디어 파일 처리를 위한 동기화 알고리즘

알고리즘 1.은 제안한 Consistency 알고리즘에 대한 의사코드이다. 이 알고리즘은, 분산 공유 메모리 방식의 프로그래밍을 지원하는 JIAJIA 라이브러리를 이용하여 고안된 알고리즘이다. 각 호스트의 저장노드에는 (SH, GOP)를 묶어 프래그먼트 단위로 결정하고, 공유메모리의 사이즈를 결정하는 문제에 있어서 사용될 값으로 SH_GOP로 정의하였다.

3.3 미디어 파일 전송을 위한 시간 동기화

여러 대의 VOD 서버로부터 형성된 공유 메모리의 미디어 데이터는 시간적인 관계에 있어서 동기화가 필요하다.

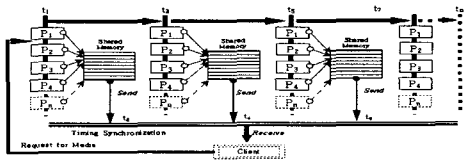

그림 4. 시간 동기화와 예상 페이징 기법을 적용한 공유메모리에서의 미디어 데이터 읽기 과정

그림 4.는 시간(tx)에 따라 각 호스트로부터 공유메모리에 읽어들인 미디어 데이터의 프래그먼트들을 어떻게 처리할 것인지에 대해 언급하였다. 최초 t1일 때, 클라이언트가 미디어 파일을 요청하게 되면 3.1절에서 언급한 barrier 메커니즘을 통하여 각 호스트로부터 읽어들인 프래그먼트들을 공유 메모리에 저장한다. t_{2n}(n=1,2,3,...일 때)이면, 시간 동기화가 적용되어 현재 공유메모리에 형성되었던 데이터만 클라이언트에게 서비스를 제공할 수 있도록 하며, t_{2n+1}이면 예상 페이징 기법을 적용하여 공유메모리에 각 호스트로부터 t_{2n-1}일 때 읽어들인 프래그먼트의 다음 프래그먼트들을 P_1부터 P_n까지 barrier 메커니즘을 통해 읽어들여 공유메모리에 다시 저장할 수 있도록 한다. 다음에 읽혀지게 될 미디어 데이터를 미리 공유메모리에 적재하게 함으로써 프로세스의 전체적인 실행시간을 줄일 수 있고 미디어 데이터의 처리속도가 향상된다. 그리고, $t_{2n}<x<t_{2(n+1)}$의 x 시간동안에는 t_{2n-1} 시간 때 공유 메모리에 제공된 미디어 데이터를 클라이언트에게 보여준다.

IV. 실험

실험은 두 대의 리눅스 서버를 사용하였다. 해당 서버는 RedHat Linux 8.0 버전을 이용하였고 gcc 버전은 3.2.2 이다.

본 실험은 MPEG 프래그먼트 처리 단위를 GOP로 가정하고, 이 GOP 데이터의 평균적인 크기는 28kbyte 정도이다. 실험을 수행할 때 실험하기 위한 프로그램을 작성하였다. 실험을 위한 시뮬레이션 프로그램 구조는 다음 그림 5.와 같다.

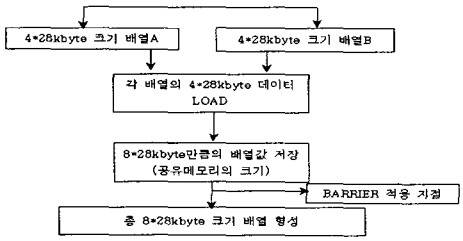

그림 5. 공유메모리를 8*28kbyte로 설정하였을 경우 배열을 합치는 프로그램 구조

그림 5.은 공유메모리의 크기를 배열A와 배열B의 크기를 모두 합친 8*28kbyte로 설정하였다. 이것은 일반적으로 병렬 프로그래밍을 할 때 작성할 수 있는 프로그램 형태로 barrier를 적용하는 시점이 공유메모리에 존재하게 될 모든 데이터의 업데이트가 완료되고 난 다음 마지막 지점에 한번만 사용한다. 다음 표 2.에서 보는바와 같이 두 배열을 합치는데 소요되는 시간이 단축된다.

표 2. 공유 메모리의 크기 설정에 따른 전체 배열 형성 시간(시간단위:초)

공유메모리의 크기 / 소요 시간	2*24kbyte	8*24kbyte
두 배열을 합치는데 소요되는 시간	0.114(∓0.002)	0.059(∓0.001)

표 2.는 그림 5.에서 두 배열을 합치는데 소요되는 시간을 측정한 결과이다. 적용되는 barrier의 횟수는 총 4회이다. local1배열과 local2배열이 가지는 값은 다음 표 3.과 같다.

표 3. local1배열과 local2배열이 가지는 값

배열 종류 / 배열 위치	local1[]	local2[]
0~6999	0~6999	7000~13999
7000~13999	14000~20999	21000~27999
14000~20999	28000~34999	35000~41999
21000~27999	42000~48999	49000~55999

표 3.의 local1과 local2 배열의 데이터 값은 7000 단위로 데이터 값을 읽는다. 즉, 2*7000 크기를 가지는 공유메모리의 total 배열에 배열 local1, local2순으로 7000개의 데이터를 순차적으로 읽은 다음 0~13999 크기를 가지는 total 배열에 데이터 값을 전달하고, 총 4회 반복하여 local1과 local2의 전체 데이터 값을 전달한다.
그림 5.에서 barrier는 공유메모리의 데이터가 최종적으로 변경이 완료되는 마지막 시점에만 적용되어 barrier로 인해 발생하는 부하가 적다.

V. 결론

본 논문에서는 VOD 스트리밍 데이터를 처리하기 위한 방안으로 멀티미디어 데이터를 분산 공유 메모리 방식으로 처리하는 Consistency 알고리즘을 설계 및 구현하였다. 그리고 본 논문에서 제시한 알고리즘에 대한 성능 평가를 실시하였다. 논문의 실험 결과 본 논문에서 제시하는 consistency알고리즘의 구현 시 barrier단위를 가능하면 최소화시키는 것이 성능 향상에 좋음을 알 수 있다.

참고 문헌

[1] H. Schulzrinne, et al., "RTP: A Transport Protocol for Real-Time Application", RFC 1889, Jan. 1996

[2] W. Hu, W. Shi, Z. Tang, M.Rasit. Eskicioglu "JIAJIA User's Manual", June 3, 1998.

[3] The JUMP Software DSM Software, http://www.srg.csis.hku.hk/srg/html/jump.htm

[4] Sarita V. Adve, Kourosh Gharachorloo, "Shared Memory Consistency Models: A Tutorial", WRL Reasearch Report, July 1995

[5] Leslie Lamport, "How to make a multiprocessor computer that correctly executes multiprocess programs", IEEE Transactions on Computer, C-28(9):690~691, September 1979.

[6] W. Hu, W. Shi, Z. Tang, and M. Li, "A Lock-based Cache Coherence Protocol for Scope Consistency", Journal of Computer Science and Technology, Vol. 13, No. 2

[7] B.N. Bershad and M.J. Zekauskas. "Midway: Shared Memory Parallel Programming with Entry Consistency for Distributed Memory Multiprocessors". Technical Report CMU-CS-91-170, Carnegie Mellon University, September 1991.

[8] L.Iftode, J.P. Singh and K.Li. "Scope Consistency: A Bridge between Release Consistency and Entry Consistency". In Proc. of the 8th Annual ACM Sym. on Parallel Algorithms and Architectures, June 1996.