SHORE thread를 사용한 효과적인 다중 볼륨 접근 방법

안성수[0] 최윤수 진두석
한국과학기술정보연구원
(ssahn[0], armian, dsjin)@kisti.re.kr

Effective method of accessing multi volumes using SHORE thread

Sungsoo Ahn[0] Yunsoo Choi Duseok Jin
Dept. of Information System, Korea Institute of Science and Technology Information

요 약

사용자에게 서비스 할 데이터가 많을 경우 여러 볼륨에 저장해서 처리해야 할 경우가 발생한다. 볼륨이 여러 개일 경우 효과적이고 효율적인 접근 방법이 필요하다. 본 논문에서는 SHORE 저장 시스템을 이용할 경우에 효과적이고 효율적인 접근 방법을 알아보고자 한다. single thread, multi thread, multi process, socket을 이용한 접근 방법을 살펴보고 multi thread를 이용하는 방법이 가장 효율적인 것을 실험 결과를 통해서 보인다. SHORE thread는 CPU bound에 관련된 job이 많은 경우는 process를 사용했을 때에 비해 큰 효과가 없으나 I/O bound에 관련에 것일 경우는 multi process를 사용한 것과 비슷한 효과가 있음을 알 수 있다.

1. 서 론

SHORE[1]는 UW(University of Wisconsin - Madison)에 개발된 객체 저장 엔진으로 사용자에게 볼륨, 파일, 레코드, 인덱스, 트랜잭션 등의 객체를 제공한다. SHORE 저장 시스템을 가지고 응용할 수 있는 분야는 데이터베이스[2], 정보 검색 시스템[3], 멀티미디어 시스템의 저장 엔진으로 사용할 수 있다. 이러한 응용 분야에서 처리해야 할 데이터가 많을 경우 이를 하나의 볼륨에 저장하기보다 여러 볼륨에 데이터를 저장하여 사용자에게 빠른 데이터 접근을 허용해야 한다.

본 논문에서는 SHORE 저장 시스템을 사용하여 다중 볼륨을 접근해야 할 때 다중 CPU환경에서 효과적인 설계 방법을 살펴보고자 한다.

2. 본 론

실험은 볼륨의 레코드(record)와 엔트리(entry)를 스캔(scan)할 때 각 실험 방법의 시간을 측정하는 것으로 실험 방법의 효율성의 기준으로 삼았다. 실험 환경은 Pentium-III Xeon 700MHz 4 CPUs, 1024MB RAM, Linux 환경에서 SHORE Interim snapshot 2 버전을 사용했고 저장 시스템의 페이지 크기는 32KB를 사용했다.

하나의 볼륨에 1KB, 10KB, 20KB, 40KB, 1MB 레코드가 1000개, 2MB 레코드가 500개, 4MB 레코드가 250개 순차적으로 저장되어 있고 엔트리의 키는 한글 데이터 약 42만개, 엘리먼트는 12바이트의 정보가 저장되어 있다. 볼륨은 4개 존재하고 각 볼륨은 동일한 데이터를 가지고 있다. 레코드는 순차적으로 접근했고 이를 위해 물리적으로 레코드를 순차적으로 저장했다. 인덱스의 엔트리를 접근하는 방법은 범위를 설정하는 방법과 그렇지 않고 각 키를 개별적으로 접근하는 방법을 사용했고 엔트리는 벌크 로딩을 하여 저장했다.

볼륨의 레코드 및 엔트리를 접근하기 위해 사용된 방법은 단일 쓰레드(single-thread), 멀티 쓰레드(multi-thread), 멀티 프로세스(multi-process), 소켓(socket)이다. 위에서 사용된 쓰레드는 sthread package로 SHORE 저장 시스템에서 제공하는 쓰레드이다.

2.1에서 2.4는 레코드를 스캔하는 방법을 설명하고, 2.6는 인덱스의 엔트리를 스캔하는 방법을 설명한다.

2.1 단일 쓰레드 이용

이 방법은 하나의 메인 쓰레드(main thread)에서 각각의 볼륨의 레코드를 순차적으로 접근하는 방식이다. SHORE 저장 시스템은 각 볼륨이 개방(open)될 때 각 볼륨의 디스크 I/O를 처리하는 diskrw 프로세스가 생성되어 대기하고 폐쇄(close)될 경우 diskrw 프로세스는 종료(kill)된다. 레코드를 순차적으로 접근하기 때문에 첫 번째 볼륨의 레코드를 접근할 때, 즉 diskrw를 이용하여 레코드를 스캔할 때, 다른 볼륨들의 diskrw는 대기하고 있게 된다. 이 방법은 뒤에 서술하게 될 다른 방법에 비해 하나의 쓰레드만 존재하기 때문에 IPC[4] 및 통신 등을 사용하지 않는다.

2.2 멀티 쓰레드 이용

이 방법은 하나는 메인 쓰레드에서 각각의 볼륨을 접근하기 위해 일꾼 쓰레드(worker thread)를 생성(thread fork()), 실행하는 방법이다. 4개의 볼륨이 존재하기 때문에 4개의 일꾼 쓰레드를 생성하고 각각의 thread는 동시에 각 볼륨의 레코드를 스캔하게 된다. 이 방법은 단일 쓰레드를 이용할 경우에 비교해 각 볼륨의 diskrw가 동시에 실행되고 메인 쓰레드와 일꾼 쓰레드의 통신은 함수의 파라미터(parameter)를 사용한다. 일반적으로 UNIX fork()를 사용할 때 부모(parent)와 자식(child) 프로세스는 정보를 주고 받기 위해 PIPE 또는 IPC를 이용한다. SHORE 쓰레드를 이용할 경우 함수의 파라미터를 이용하면 더욱 빠르게 정보를 주고 받을 수 있다. 실험에서 이러한 시간은 제외하였다.

2.3 멀티 프로세스 이용

이 방법은 하나는 메인 쓰레드에서 각각의 볼륨을 접근하기 위해 자식 프로세스를 생성(UNIX fork())한 후 exec()하는 방법이다. 실험에서는 4개의 볼륨이 있기 때문에 4번의 UNIX fork()와 exec()가 실행된다. 이 방법은 하나의 프로세스에서 여러 쓰레드가 생성될 경우 SHORE 쓰레드가 다중 CPU를 지원하지 않기 때문에 하나의 CPU만 실행되고 다른 CPU는 실행되지 않는 것을 발견한 후 제안된 방법이다. 즉 프로세스를 여러 개 생성해 CPU를 동시에 사용해 디스크의 접근 속도를 빠르게 하려는 시도이다. 그러나 이 방법은 초기에 SHORE 저장 시스템을 초기화 시키는 시간이 오래 걸리는 단점이 있다. 또한 새롭게 exec()된 자식 프로세스는 부모 프로세스와 통신을 하기 위해서 PIPE 또는 IPC를 사용해야 한다.

2.4 소켓(Socket) 이용

이 방법은 각각의 볼륨에 서버를 두는 방식으로 클라이언트는 동시에 각 볼륨 서버에 요구(request)를 보내 각 서버의 데이터를 접근할 수 있다. 이 방법은 저장 시스템이 생성될 때 초기화와 메모리를 할당하고 볼륨을 개방하는 시간이 들기 때문에 이러한 과정을 한번만 하고 사용자의 데이터 접근이 있을 때 마다 바로 처리하는 방법이다. 서버는 클라이언트와 데이터를 보내기 위해 send(), recv()를 사용한다.

2.5 Record 스캔 분석

표1의 레코드 스캔 시간을 분석해 보면 소켓을 사용한 방법이 멀티 쓰레드, 멀티 프로세스보다 조금 빠르고 단일 쓰레드와는 많은 차이를 나타내고 있다.

소켓을 이용한 접근 방법이 빠른 원인은 초기화의 과정을 한번만 하고 클라이언트의 요구가 있을 때 마다 바로 물리적인 디스크를 접근할 수 있기 때문이다.

표1. 레코드 스캔 시간

	ST	MT	MP	SO
1KB	0.75	0.72	0.21	0.18
10KB	1.28	1.03	0.61	0.30
20KB	2.32	1.50	1.03	0.73
40KB	3.27	2.00	1.62	0.86
1MB	1:51.49	1:22.82	1:22.91	1:21.19
2MB	2:16.78	1:27.19	1:25.93	1:25.22
4MB	2:37.06	2:12:25	2:12.98	2:11.63

ST: Single Thread
MT: Multi Thread
MP: Multi Process
SO: Socket

멀티 쓰레드를 이용한 접근 방법이 두 번째로 빠른 결과를 보이고 있다. 이것은 SHORE 쓰레드의 특징으로 쓰레드가 프로세스보다 fork()하는 시간이 적고 들고 적은 리소스를 사용하면서 디스크 I/O의 작업은 diskrw 프로세스에 위임하기 때문이다. sthread는 디스크 작업이 있을 경우 다른 쓰레드에 CPU를 양보(yield)해서 다른 쓰레드가 CPU를 사용할 수 있도록 한다. 디스크 I/O가 발생하는 동안 diskrw 프로세스가 CPU를 사용하고 이 쓰레드는 블록(block)된다. I/O작업이 끝나면 블록된 쓰레드는 일어나(wake up) 작업을 하게 된다. 이 방법은 디스크 I/O가 많을 경우 적은 시스템의 리소스를 사용하면서 효율적으로 작업을 할 수 있는 방법이다.

멀티 프로세스를 이용한 방법은 세 번째로 빠르다. 멀티 쓰레드보다 느린 원인 중의 하나는 일꾼 프로세스가 실행될 때 저장 시스템을 초기화하는 작업이 있기 때문이다. 디스크 I/O에서 이 방법은 CPU를 최대로 사용할 수 있는 장점이 있지만 디스크 I/O작업의 경우 멀티 쓰레드에 비해 큰 효과가 없는 것으로 나타났다. 그 이유는 SHORE 쓰레드는 디스크 I/O를 할 경우 diskrw 프로세스가 각 볼륨마다 존재하기 때문에 멀티 프로세스를 이용한 방법과 비슷한 효과를 가지기 때문이다.

단일 쓰레드로 레코드를 접근하는 방법은 가장 느리게 나타났다. 이것은 CPU가 여러 개 있는 것을 이용하지 못하고 CPU를 하나만 이용하는 방법이다. 이 실험은 다른 실험 방법과의 성능을 비교하며 사용될 수 있다.

위의 결과를 분석해 보면 멀티 프로세스와 소켓을 이용한 접근 방법은 CPU를 최대로 사용할 수 있으나 검색된 결과를 합치기 위해 IPC 또는 socket 통신을 이용해야 하는 특징이 있다. 그렇지만 멀티 쓰레드를 이용하여 볼륨을 접근하는 경우는 위의 두 방법에 비해 적은 시스템의 자원을 사용하면서 효과적으로 각 볼륨을 접근할 수 있다. 일꾼 쓰레드와 메인 쓰레드간의 통신은 함수 파라미터로 할 수 있어 소켓, 멀티 프로세스

방법보다 효율적이다. 다중 쓰레드를 이용한 방법이 I/O 작업이 많은 데이터베이스나 정보 검색 관리 시스템의 많은 양의 레코드와 포스팅 정보를 접근할 때 효율적인 방법임을 알 수 있다.

2.6 Entry 스캔 및 분석

표2. Entry 스캔 시간

	ST	MT	MP	SO
개별	3:53.93	1:00.16	59.28	58.75
범위	1:05.47	1:06.59	16.81	16.27

표2의 실험 결과를 보면 인덱스의 엔트리를 개별적으로 접근할 때는 소켓, 멀티 프로세스을 이용한 방법이 비슷하고 멀티 쓰레드를 이용한 방법이 약간 느리고 단일 쓰레드를 이용하는 방법이 가장 느리게 나타나고 있다. 42만개의 엔트리를 하나 하나 스캔할 때 디스크 I/O가 발생한다. 레코드를 스캔할 때와 같이 멀티 쓰레드를 이용하는 것이 시스템의 자원을 적게 사용하면서 효과적인 방법임을 알 수 있다.

그러나 전체 범위 설정을 한 후 엔트리를 접근 할 경우는 큰 차이가 있음을 알 수 있다. 이것은 SHORE 저장 시스템의 특징으로 개별적으로 접근 할 경우 내부적으로 사용되는 객체(scan_index_i)를 초기화할 때 이미 그 결과를 알 수 있지만 범위를 설정한 경우는 처음 것 이후는 다음 결과를 메모리에서 찾아야 하는 과정을 거치기 때문이다. 즉, 범위 설정을 한 후 엔트리를 접근하는 경우는 I/O bound가 아닌 CPU bound 작업이 된다. 따라서 멀티 프로세스 및 소켓을 이용한 방법이 빠른 효과를 나타내고 있는데 이것은 멀티 프로세스 및 소켓은 개별적인 프로세스가 여분의 CPU를 사용할 수 있기 때문이다. 또한 SHORE 쓰레드의 문맥 변환(context switch)가 발생하기 때문에 단일 쓰레드를 사용한 것에 비해 멀티 쓰레드를 사용한 것이 약간 느리게 나타나는 것으로 분석된다.

SHORE 멀티 쓰레드를 이용할 경우 각 쓰레드가 여분의 CPU를 이용하지 못하기 때문에 범위 설정의 작업이 많을 때 효과적으로 처리하는 방법이 요구된다. 위의 실험은 약 42만개의 키를 전체 범위 설정한 후 엔트리를 접근한 방법이다. 스캔 범위가 좁은 경우 각 방법의 성능이 큰 차이는 없겠지만 범위가 넓을 경우 시스템의 성능에 큰 영향을 미치게 된다. 따라서 멀티 쓰레드를 사용하여 많은 엔트리를 스캔할 경우 PIPE 또는 IPC를 이용하여 엔트리를 접근하는 효과적인 방법이 필요하다.

3. 결 론

다중 CPU상황에서 SHORE 쓰레드를 이용하여 분산된 볼륨의 레코드, 개별 엔트리를 접근할 경우 멀티 쓰레드를 이용하여 접근하는 방식이 멀티 프로세스 또는 소켓을 이용한 것보다 효율적인 것을 위의 실험을 통해 알 수 있었다. 그렇지만 인덱스의 스캔 범위가 넓을 경우는 소켓, 멀티 프로세스가 더 좋은 성능을 보이고 있다.

위의 실험 결과를 기초로 다중 볼륨을 이용하는 데이터베이스 서버 또는 정보 검색 시스템의 서버를 효율적으로 설계, 구현할 수 있으리라 생각된다.

SHORE 저장 시스템은 트랜잭션 인터페이스를 제공한다. 위 실험에서는 SHORE의 기본적인 인터페이스를 적용하고 있다. 내부적으로 디스크를 접근할 때 트랜잭션에 의한 잠금 기법(locking)을 제공하는데 이 부분의 효과적인 사용 방법을 분석하고 적용해서 시스템을 설계할 때 적용해서 더욱 안정적이고 빠른 서비스를 사용자에게 제공하는 것이다.

참고 문헌

[1] The Shore Project Group, Computer Sciences Department, UW-Madison, Storage Manager Architecture. June 29, 1999

[2]Raghu Ramakrishnan, Database Management Systems, McGraw-Hill, 1998

[3]정보 시스템 연구실, 한국과학기술정보연구원, KRISTAL-2000 사용자 매뉴얼, 2002

[4] John Shapley Gray, Interprocess Commmunications in UNIX, Prentice-Hall, 1998