제20회 한국정보처리학회 추계학술발표대회 논문집 제10권 제2호 (2003. 11)

가상 시스템 호출을 이용한 시간 측정 함수의 성능 향상

김재열, 손성훈, 정성인
한국전자통신연구원
e-mail :gauri, sonsh, sijung,@etri.re.kr

Enhanced Time Measurement Function Using Virtual System Call

Chei-yol Kim, Sung-hoon Son, Sung-in Jung
Electronics and Telecommunications Research Institute

요 약

최근의 인터넷의 급속한 발전에 힘입어 멀티미디어 데이터를 인터넷을 통해 서비스해 주는 스트리밍 서비스가 활성화가 되고 있다. 스트리밍 서버는 서비스를 요청한 클라이언트에게 일정한 데이터 전송률로 서비스를 하게 되는데, 이때 시간측정 함수를 매우 빈번히 사용하게 된다. 본 논문은 시간 측정함수를 가상 시스템 호출 기법을 이용해 성능을 향상 시킬 수 있음을 보여준다.

1. 서론

전세계에서 초고속 인터넷의 보급이 가장 빠른 나라로 손꼽히는 우리나라에서는, 이러한 인터넷의 보급으로 각종 미디어 데이터를 실시간으로 서비스 받을 수 있는 스트리밍 서비스가 많은 인기를 얻고 있다. 스트리밍 서버는 다수의 클라이언트에게 각각이 요청한 수준의 데이터를 실시간으로 보내주기 위하여 내부에서 시간을 확인하는 함수가 빈번히 사용된다. 이때 사용되는 것이 gettimeofday() 시스템 콜이다.

gettimeofday() 시스템 콜은 호출이 될 때마다 사용자 모드에서 커널 모드로 전환되는 문맥교환이 일어나게 된다. 이러한 문맥교환이 시스템에 상당한 부담으로 작용하게 된다는 것은 주지의 사실이다. 특히나 gettimeofday()와 같은 시스템 콜은 하는 일이 매우 작은 경우이다. 이렇게 실제 작업을 수행하는 시간은 짧으면서, 빈번히 불리어지게 되면 문맥교환에 의한 시스템의 부하는 더욱더 커지게 된다. 이러한 이유로 gettimeofday() 함수를 문맥교환 없이 수행할 수 있으면 gettimeofday() 함수 자체의 성능을 높일 수 있을 뿐만 아니라, 이를 자주 사용하는 응용프로그램의 부하도 크게 줄일 수 있다. 본 논문은 이렇게 빈번히 사용되는 gettimeofday() 시스템 콜을 문맥교환 없이 사용자 영역에서 수행할 수 있도록 가상 시스템 호출 기법을 이용하여 리눅스 커널 2.4.18-14 에 구현하여 시스템의 성능을 향상 시켰다.

2. 관련 연구

2.1. Enterprise Linux

인터넷과 E-비즈니스의 확산과 더불어 리눅스와 공

개 소프트웨어는 정보기술 분야에서 폭넓게 사용되고 있다. 공개 소프트웨어는 주로 인터넷 서버나 애플리케이션 서버로는 많이 사용되어 왔으나, 데이터베이스나 트랜잭션을 담당하는 back-end 서버로는 특정 기업의 제품(주로 유닉스 기반의)이 대부분 사용되어 왔다. 이에 나날이 증가하는 컴퓨팅 비용을 절감하기 위한 방편으로 많은 기업들이 공개 소프트웨어를 기업 환경에서 사용하기를 원하지만, 기존의 제품들과는 달리 공개 소프트웨어의 특성상 안정성과 확장성등 여러면에서 검증이 되지 않은 부분이 많아 기업들이 적극적으로 사용하기를 꺼리고 있다. 이러한 문제를 해결하기 위해 정보기술 분야의 선도 기업인 HP, IBM, NEC, Red Hat 등이 리눅스를 기업환경에서 사용할 수 있도록 하자는 의도로 컨소시움을 형성해 OSDL(Open Source Development Labs)[1]을 만들었다.

OSDL 에서는 공개 소스로 개발되어온 리눅스를 성능과 안전성이 중요한 기업형 컴퓨팅 시장에서도 사용할 수 있도록 개발하고 테스트하는 연구를 수행하고 있다.

OSDL 에서는 크게 통신장비에서 쓰일 수 있는 리눅스 운영체제를 연구하는 CGL(Carrier Grade Linux)부분과 데이터 센터급 서버 용도로 쓸 수 있도록 하기 위한 DCL(Data Center Linux) 부분으로 이루어 진다.

DCL 은 데이터 센터 서버의 특성상 확장성과 안정성 및 성능 향상에 초점을 맞추고 있다. 이러한 DCL 의 한 부분으로 가상 시스템 호출을 이용한 gettimeofday() 시스템 콜의 성능 향상 부분이 포함되어 있다[2].

2.2. LTP(Linux Test Project)

LTP[3]는 SGI, IBM, OSDL, Bull, Wipro Technologies 등의 기업들이 리눅스의 신뢰성, 견인성, 안정성을 테스트 하기 위해 만든 프로젝트이다. LTP 는 리눅스 커널 자체와 커널에 관련된 항목들을 테스트 한다. 여기에는 시스템 콜 테스트도 포함이 된다. 본 논문에서 구현된 gettimeofday() 시스템 콜도 LTP 를 통하여 테스트를 하였으며, 테스트 결과는 8 절에서 보여준다.

3. 가상 시스템 호출

일반적인 시스템 콜은 사용자 영역의 코드가 커널 영역의 코드를 수행하기 위한 한 가지 방법이다.

그림 1 은 일반적인 시스템 함수가 사용자 영역에서 libc 라이브러리를 통해서 수행되는 과정을 보여준다. 일반적인 시스템 콜을 수행하기 위해서는 사용자 영역에서 커널 영역으로의 문맥교환이 필수적이다. 이러한 문맥교환을 방지하는 하나의 방법으로 제안된 것이 가상 시스템 콜 기법이다.

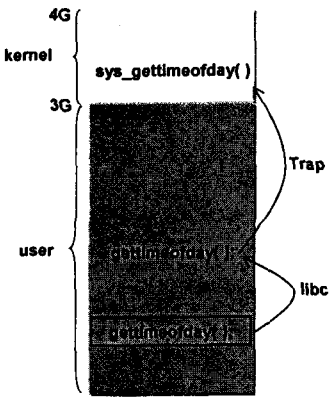

그림 1. General System call

가상 시스템 콜 기법은 커널 메모리 영역의 특정 부분을 사용자 영역이 읽고 수행할 수 있도록 컴파일 시에 지정해 줌으로서 가능하다. 리눅스에서는 0xffffd000 영역 이후의 커널 메모리 영역을 가상시스템 콜로 사용할 수 있도록 정의하고 있다. 커널 컴파일 시에 지정한 0xffffd000 이후의 메모리 영역에 사용자가 지정한 코드와 데이터를 위치시켜 놓고 해당 메모리 영역을 사용자 모드의 프로세스가 접근할 수 있도록 해 줌으로써 가상 시스템 호출을 가능하게 한다.

본 논문에서는 그림 2 에서와 같이 이러한 가상 시스템 호출 기법을 gettimeofday() 함수에 적용하여 리눅스 커널에 구현하였다.

그림 2 에서 보는 바와 같이 일반적인 gettimeofday()함수는 glibc 라이브러리를 통해서 커널 영역의 sys_gettimeofday() 시스템 콜을 호출한다. 이에 반하여 향상된 gettimeofday() 함수는 glibc 라이브러리를 거치지 않고 컴파일 시에 미리 지정한 커널 메모리 영역의 gettimeofday 함수로 점프하여 바로 커널 영역에서 sys_gettimeofday() 시스템 콜이 하는 일을 수행하고 결과값을 사용자에게 전달함으로써 문맥교환을 없애고 gettimeofday() 함수의 수행시간을 급격히 줄일 수 있다.

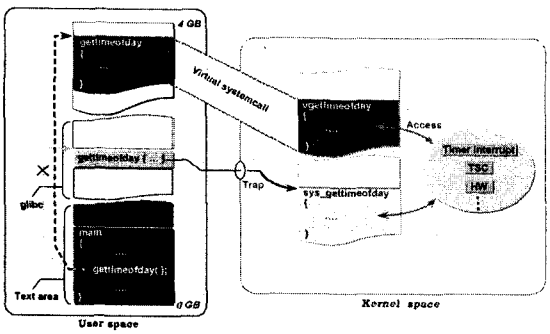

그림 2. Enhanced gettimeofday()

그림 2 에서와 같이 gettimeofday()함수는 CPU 에 있는 TSC(Time Stamp Counter)를 읽어서 현재의 시간을 계산한다. TSC 에 대해서는 다음 절에서 설명할 것이다.

4. Time Stamp Couter(TSC)

TSC 는 시스템이 부팅 될 때부터 프로세서의 시스템 클럭에 따라 증가하는 카운터로서 본 논문의 향상된 gettimeofday() 함수가 구현된 인텔 계열의 프로세서에서는 이를 읽을 수 있는 인스트럭션을 제공한다.

TSC 는 64 비트 크기를 가지며 프로세서의 클럭에 따라 하나씩 증가하며 시스템이 reset 될 때 마다 다시 "0"으로 바뀐다.

TSC 에 접근하기 위해서 사용자는 RDTSC(Read Time-Stamp Couter)라는 인스트럭션을 사용할 수 있고 읽혀진 카운터 값은 EDX 와 EAX 레지스터에 나누어져서 기록된다. 이 때 읽혀진 TSC 값은 시간이 아니라 단지 사이클일 뿐이므로 시스템이 부팅된 이후의 지나간 시간을 계산하기 위해서는 읽은 TSC counter 값을 아래와 같이 프로세서의 주파수로 나누어 줘야 한다.

#seconds = (# TSC) / (frequency of CPU)

이렇게 나온 결과값을 가지고 부팅시의 날짜와 시간을 비교하면 현재의 날짜와 시간을 계산할 수 있다.

5. Library overriding

그림 2 에서와 같이 일반적인 gettimeofday()는 glibc 라이브러리를 통하여 커널의 sys_gettimeofday()시스템 콜을 호출하게 된다. 이것을 본 논문에서 구현한 가상 시스템 콜을 이용하게 하기 위한 첫번째 방법은 glibc 라이브러리를 수정해 기존의 sys_gettimeofday() 함수를 요청하지 않고 커널 영역에 구현한 gettimeofday()로 점프하도록 하는 것이다. 이 방법을 사용하면 glibc 내에 포함된 많은 함수들이 바뀌지 않는데도 불구하고 libc 라이브러리 자체를 수정하고 다시 컴파일 및 설치해야 한다. 이러한 추가적인 작업없이 glibc 라이브러리내에서 수정한 gettimeofday() 함수만을 바꿀 수 있는 방법이 library overriding 기법이다. 이 overriding 기법을 사용하면 libc 에 일체의 수정없이 gettimeofday() 함수에 대해서만 수정된 vgettimeofday() 함수를 호출할 수 있다.

라이브러리 overriding 을 하기 위해서는 새로운 내용이 적용된 함수들로 오브젝트 파일을 생성하고, LD_PRELOAD 환경변수에 해당 파일을 지정해 주면 응용프로그램은 새로운 시스템 콜을 호출하게 된다.

6. 실험환경

- CPU : Pentium IV 1.3Ghz
- Main Memory : 768 MB Rambus D-RAM

- 운영체제 : Linux Redhat 8.0
- kernel : 2.4.18-14

7. 성능 실험

개선된 gettimeofday() 함수의 성능을 측정하기 위하여 연속으로 백만번의 gettimeofday() 함수를 호출하고 이의 평균시간을 계산하였다.

각각에 대해서 4 번씩 수행한 결과가 표 1 이다.

표 1. 향상된 gettimeofday() 의 성능

일반 시스템 콜을 사용한 gettimeofday(us)		가상 시스템 콜을 사용한 gettimeofday(us)	
1.628733	1.626490	0.328322	0.329448
1.626632	1.626350	0.328545	0.328290

표 1 의 실험결과를 보면 기존의 시스템 콜을 사용한 gettimeofday()의 결과는 한번 수행하는 평균시간이 1.627 us 정도가 걸렸으며, 가상 시스템 콜을 사용한 gettimeofday()는 0.329 us 정도가 걸려 대략 5 배 정도의 성능 차이가 난다. 이러한 성능 차이는 문맥교환이 발생하지 않음으로 해서 생기는 것이라고 볼 수 있다.

8. LTP 테스트

gettimeofday() 시스템 콜의 LTP 테스트 결과는 아래와 같다.

```
gettimeofday01   1   PASS : gettimeofday(2) set the errno EFAULT correctly
gettimeoday02    0   INFO : checking if gettimeofday is monotonous takes 30s
gettimeofday02   1   PASS : gettimeofday monotonous in 30 seconds
```

그림 3. LTP 테스트 결과

그림 3 과 같이 LTP 에서는 gettimeofday() 함수에 대해서 2 가지의 테스트를 수행한다. 첫 번째 테스트는 gettimeofday()에 인자로 넘겨주는 포인터에 잘못된 값을 주었을 때 EFAULT 에러가 반환되는지를 테스트하는 것이고, 두번째는 30 초 동안 gettimeofday()를 호출해 반환하는 시간이 정상적으로 흐르고 있는 지를 확인한다. 그림 3 에서와 같이 두 가지 테스트에 대해서 모두 PASS 판정을 받아 LTP 테스트를 통과하였다.

9. 결론

사용자 수준의 응용프로그램이 커널의 코드를 수행하기 위한 하나의 방법이 시스템 콜 기법이다. 이러한 시스템 콜을 사용함으로써 커널의 코드를 사용자가 직접적으로 접근할 수 없도록 하여 커널의 코드를 보호할 수 있다. 하지만, 가상 시스템 호출 기법을 사용하면 커널 영역을 사용자 수준의 프로그램이 접근할 수 있게 되므로 이와 같은 보호 장치가 없어지게 되므로 보안 측면에서 취약한 시스템이 될 수도 있다. 이러한 점 때문에 가상 시스템 호출 기법을 많은 분야에 대해서 사용할 수 없다. 하지만, 본 논문에서 다룬 gettimeofday() 에서는 단순히 TSC 를 읽어서 계산하는 일만 수행함으로써 가상 시스템 호출을 적용할 수 있었다. 이러한 가상 시스템 호출 기법을 적절히 활용할 수 있다면 문맥교환없는 커널영역 접근이 가능해 좋은 성능을 내는 시스템 콜을 구현할 수 있을 것이다.

참고문헌

[1] www.osdl.org

[2] Data Center Linux – Roadmap : www.osdl.org/docs/dcl_roadmap.pdf

[3] Linux Test Project : ltp.sourceforge.net