Oracle9*i* Database Release 2 on Linux: Red Hat Linux Advanced Server 2.1에 대한 성능, 신뢰성 및 관리 용이성 강화

Oracle 백서
2002년 6월

ORACLE®

Oracle9*i* Database Release 2 on Linux: Red Hat Linux Advanced Server 2.1에 대한 성능, 신뢰성 및 관리 용이성 강화

개요

Linux는 이제 전세계 중소기업 및 대기업을 위한 IT 애플리케이션의 대규모 설치가 이루어지는 시발점에 도달했다. 고객들이 비즈니스 크리티컬 애플리케이션을 Linux로 마이그레이션함에 따라 운영 체제에 대한 성능, 신뢰성 및 관리 용이성 개선에 대한 요구가 증가하고 있다. 여러 운영 체제 플랫폼을 설치하는데 대한 오랜 경험을 보유하고 있는 세계 최대 엔터프라이즈 소프트웨어 공급업체인 오라클은 Linux가 워크스테이션 환경에서 하이 엔드 서버 플랫폼으로 발전할 수 있도록 이끄는데 있어 독보적인 입지를 확보하고 있다.

오라클은 레드 햇(Red Hat)과 탄탄한 협력 관계를 구축해 왔으며 엔터프라이즈급 기능 및 미션 크리티컬 애플리케이션을 지원하는 운영 체제를 지원한다는 목표 하에 리눅스 커널을 개선하기 위해 협력해 왔다. 그 결과 오라클의 시장 주도적인 데이타베이스와 애플리케이션 서버에 맞춰 튜닝됐으며 레드 햇이 Advanced Server로 명명한 새로운 버전을 발표하게 됐다. 본 백서는 Red Hat Linux Advanced Server 2.1 상에 Oracle9*i* Database Release 2 (Oracle9*i*R2)를 설치하는 고객들이 누리게 될 새로운 기능 중 가장 중요한 기능들에 대해 다루고 있다.

Linux 아키텍처 상의 Oracle

Linux는 대부분의 UNIX 플랫폼과 유사한 프로그래밍 인터페이스 및 기능을 제공한다. Linux 상의 Oracle9*i* R2 아키텍처는 Solaris, HP-UX 및 Tru64 등 UNIX 기반 운영 체제 상의 Oracle 아키텍처와 매우 유사하다. 이는 지난 수 년간 기존 UNIX 플랫폼 상에서 실현됐던 성능 및 안정성이 Linux상의 Oracle 제품에서 그대로 실현된다는 사실을 보장하는 것이다.

프로세스 기반 모델

Linux 상의 Oracle9*i*R2는 Windows 상의 스레드 기반 아키텍처와는 달리 UNIX 플랫폼과 유사한 프로세스 기반 아키텍처를 보유하고 있다. Linux 상에서 Oracle은 프로세스를 사용함으로써 데이타베이스 라이터(database writer), 로그 라이터(log writer), 프로세스 모니터링(process monitoring) 등 백그라운드 작업은 물론 수신 클라이언트 커넥션 처리와 같은 포그라운드 작업을 수행한다. Windows의 경우 이들 모든 작업은 단일 프로세스 내

스레드로서 처리된다. 모든 스레드는 SGA, PGA 및 기타 메모리를 위한 동일한 3GB 주소 공간을 공유하기 때문에 스레드 기반 모델은 32 비트 인텔 펜티엄 기반 시스템의 경우 제한이 따르게 된다. 예를 들어 비교적 넓은 공간이 사용될 경우 모든 포그라운드 스레드가 동일한 3GB 풀로부터 메모리를 사용하기 때문에 가용 메모리에 문제가 생길 수 있다. 프로세스 모델의 경우 각 포그라운드 프로세스는 자체 고유 어드레스 공간을 보유하고 있기 때문에 사용할 수 있는 가용 메모리를 보다 많이 확보할 수 있게 된다. 각 프로세스는 최소한 8TB 사용자 주소 공간을 보유하고 있기 때문에 이는 64 비트 시스템에 대한 제약 사항으로 작용하지 않는다는 사실을 유념해야 할 것이다.

파일시스템 지원

Oracle9*i*R2는 Red Hat Linux Advanced Server 2.0 상의 표준 Linux 파일시스템 "ext 2"로 인증을 받았다. Oracle은 또한 원시 (포맷되지 않은) 디스크 파티션 상의 운영에 대해서도 인증을 받았다. 원시 파일은 파일시스템 오버헤드가 없기 때문에 I/O 성능이 향상된다는 이점을 보유하고 있다. 그러나 원시 파일은 관리가 훨씬 어렵기 때문에 하이 엔드 설치를 위해서만 사용되거나 또는 원시 파일을 요구하는 RAC을 사용할 경우에만 이용되고 있다.

64 비트 파일 I/O

Linux는 인텔 펜티엄 기반 서버와 같은 32 비트 플랫폼 상에서도 64 비트 파일 I/O를 지원한다. Oracle9*i*R2는 64 비트 파일 오프셋을 내부적으로 지원하기 때문에 데이타, 로그 및 컨트롤 파일에 대한 2GB 또는 4GB에 대한 제한이 없다. 데이타베이스 당 파일 수 제한 (64K), 파일 당 블록 수 제한 (4 백만) 그리고 최대 블록 크기 제한 (16KB)은 다른 Oracle 플랫폼에서도 공통적으로 적용된다. 이들 제약에 따라 데이타베이스 파일의 최대 크기는 64 GB이며 16K 블록을 보유한 데이타베이스의 최대 크기는 4 petabyte가 된다.

성능 개선

오라클은 데이타베이스 및 애플리케이션 서버에 이루어진 리눅스 커널의 성능 향상을 입증하기 위해 광범위한 성능 관련 테스트를 실시했다. 본 섹션은 Linux 상의 Oracle 제품의 이전 버전에 비해 상당한 성능 향상을 실현한 Red Hat Linux Advanced Server 2.1 상의 Oracle9*i*R2에서 지원하는 기능들을 설명하고 있다. 본 백서는 인텔 32 비트 (IA-32) 플랫폼 관련 성능 개선에 대해 기술하고 있지만 대부분의 정보는 아이테니움(Itanium) 2 프로세서 기반 인텔 64 비트 (IA-64) 플랫폼 상의 Oracle9*i*R2 및 Red Hat Linux Advanced Server 의 차기 버전에도 적용될 것이다.

I/O 서브시스템

Oracle database 워크로드 하에서 시스템의 I/O 처리 성능은 Advanced Server를 기반으로 할 경우 현격히 향상된다. I/O 서브시스템에서 이루어진 가장 중요한 기능 개선으로는 비동기식 I/O, 디스크 쓰기 실행 중 메모리 버퍼에 대한 다수의 복사 과정 제거, 커널 락에 대한 경합 감소 그리고 많은 IO 드라이버의 기능 개선 등을 들 수 있다.

비동기식 I/O

그 무엇보다도 가장 중요한 기능 개선 향상은 커널 내 비동기식 I/O (또는 논블록킹 I/O)라고 할 수 있다. Advanced Server에서 비동기식 I/O가 도입되기 전에는 프로세스는 디스크 I/O 요구를 순차적으로 제출했다. 각 I/O 요청은 요구가 완료될 때 까지 호출 프로세스가 진행되지 않도록 한다. 비동기식 I/O는 요청이 완료될 때 까지 기다릴 필요 없이 프로세스가 I/O 요청을 제출하도록 한다. 또한 이 구현은 Oracle 프로세스가 다수의 단일 I/O 요청이 아닌 단일 시스템 호출을 통해 다수의 I/O 요청을 디스크에 발행할 수 있도록 지원한다. 이는 2가지 방식으로 성능을 향상시키게 된다. 먼저, 프로세스는 커널이 처리할 수 있는 다수의 요구를 큐잉할 수 있기 때문에 커널은 요구 순서를 재지정하거나 또는 디스크에 인접한 개별 요구를 더 큰 요구 단위로 결합시켜 요구 수를 줄임으로써 디스크 활동을 최적화 할 수 있다.

본 백서는 Oracle9*i*R2 on Red Hat Linux Advanced Server 2.1의 여러 기능을 실행하기 위한 각 단계들을 개략적으로 기술함으로써 그 기술적 개념에 대한 이해를 돕는다는데 목적을 두고 있다. 최신 정보와 정확한 정보를 필요로 할 경우 관련 제품 설명서(Release Notes, 관리자용 설명서, 설치 가이드 등), Oracle Support 및 RedHat Support를 참조하면 된다.

기본적으로 Oracle9*i*R2는 비동기식 I/O지원이 비활성인 상태로 출하된다. 이는 이 기능을 지원하지 않는 기타 Linux 제품도 지원하기 위한 것이다. Red Hat Linux Advanced Server 2.1 상에 Oracle9*i*R2를 배치하는 경우 비동기식 I/O 기능을 활성화하기 위해 고객들은 제품 설명서에서 제시된 바와 같이 다음과 같은 단계에 따라 실행해야 한다.

1) cd to $ORACLE_HOME/rdbms/lib
   a) make-f ins_rdbms.mk async_on
   b) make-f ins_rdbms.mk ioracle
2) 비동기식 I/O 기능을 비활성화시켜야 하는 경우, cd to $ORACLE_HOME/rdbms/lib
   a) make-f ins_rdbms.mk async_off
   b) make-f ins_rdbms.mk ioracle
3) 원시 디바이스 경우 init.ora로 파라미터 설정:
   a) set 'disk_asynch_io=true' (기본값은 true)
4) 파일시스템 파일 경우 init.ora로 파라미터 설정:
   a) 반드시 모든 Oracle 데이타파일이 비동기식 I/O를 지원하는 파일시스템 상에 상주하도록 해야 한다. (예, ext2)

b) set ′disk_asynch_io=true′ (기본값은 true)

c) set ′filesystemio_options=asynch′

바운스 버퍼에 대한 복사 과정 제거

IA-32 시스템 상에서 물리적 메모리의 최초 1GB는 저 메모리라고 하며 1GB 이상의 물리적 메모리는 고 메모리라고 한다. Linux 커널의 이전 버전들은 애플리케이션이 하위 및 고 메모리 모두를 사용할 수 있는 경우에도 스토리지 장비로 전달된 데이타 버퍼가 물리 RAM의 저 메모리 영역에 위치하도록 요구했다. 저 메모리에 위치한 데이타 버퍼로부터의 I/O 요청은 메모리에 직접 액세스하게 된다(복사 과정 없이). 그러나 애플리케이션이 I/O 요청의 일부로서 고 메모리의 데이타 버퍼를 커널로 전달할 때 커널은 저 메모리에 임시 데이타 버퍼를 강제로 할당하고 데이타를 고 메모리의 애플리케이션 버퍼로 복사하거나 또는 그 역방향으로 복사하게 된다. "바운스 버퍼링"으로 일컬어지는 이 추가 데이타 복사 과정은 I/O 집약적인 애플리케이션의 성능을 크게 저하시키게 된다. 이는 할당된 대량의 바운스 버퍼가 메모리를 많이 소모할 뿐만 아니라 바운스 버퍼 복사로 인해 시스템 메모리 버스상에 로드를 가중시키기 때문이다. Red Hat Linux Advanced Server 2.1은 바운스 버퍼링을 수행해야 할 필요성을 대폭 줄일 뿐만 아니라 많은 경우 이를 완전히 제거한다.

io_request-lock에 대한 경합 감소

Linux 커널은 스핀 잠금을 사용해 여러 프로세스가 동시에 접속하는 커널 데이타 구조의 무결성을 유지시킨다. 커널의 이전 버전들은 전체 블록 장치 서브시스템에 대해 단일 스핀 잠금을 사용했다. 따라서 I/O를 수행하는 모든 프로세스들은 심지어 관계가 없는 장치에 대한 I/O를 수행할 경우에도 이 단일 자원을 이용하기 위해 경쟁할 수 밖에 없었으며 결국 전체 시스템 I/O 처리성능을 저하시키는 불필요한 병목현상을 일으켰다. Red Hat Linux Advanced Server 2.1은 블록 장치 서브시스템을 위해 정교하게 고안된 새로운 잠금 기법을 구현해 개별 블록 장치에 별도의 잠금 기능을 보유하도록 했다. 그 결과 과도한 데이타베이스 로드를 처리하는 다수의 I/O 컨트롤러를 보유한 SMP 시스템이 발휘하는 I/O 처리 성능을 크게 향상시킬 수 있게 된 것이다.

I/O 드라이버 최적화

Dell, HP/Compaq, EMC 등을 비롯한 협력 업체가 제공하는 서버 하드웨어에서 사용되는 I/O 카드용 드라이버 코드에 대한 많은 개선이 이루어졌다. 특히 이와 같은 개선을 통해 오라클의 성능 테스트 연구실에서 확인된 병목을 제거했을 뿐만 아니라 이들 드라이버가 앞에서 언급된 바운스 버퍼 및 io_request_lock 변경을 활용할 수 있도록 했다.

가상 메모리 서브시스템

Linux 상의 VM 서브시스템과 Oracle 간의 상호 작용에 대한 자세한 내용은 오라클 기술 백서 "Red Hat Linux Advanced Server 2.1의 Linux 가상 메모리 및 Oracle의 메모리 활용의 특성(Linux Virtual Memory in Red Hat Linux Advanced Server 2.1 and Oracle's Memory Usage Characteristics)"을 참조하면 된다.

VM(Virtual Memory) 서브시스템은 데이타베이스 성능에 매우 중요한 영향을 미치게 된다. Oracle 구조는 주로 모든 백그라운드 및 포그라운드 프로세스가 공유하는 메모리 영역인 SGA(Shared Global Area)에 따라 결정된다. 데이타베이스 블록 버퍼 캐시를 보유하고 있는 SGA의 크기는 Oracle 성능에 영향을 미치는 주요 튜닝 파라미터이다. SGA가 크면 메모리에 더 많은 데이타를 캐싱할 수 있기 때문에 다양한 데이타베이스 워크로드의 성능을 크게 향상시킬 수 있다. OS 페이지 크기 같은 다른 파라미터와 마찬가지로 SGA 크기는 VM 서브시스템에 따라 결정된다.

본 섹션에서 설명한 바와 같이 VM 서브시스템 기능이 향상되면 다양한 워크로드에서 성능 및 안정성을 높일 수 있기 때문에, VM의 장단점을 제대로 파악하는 것이 중요하다. 워크로드에 따라 데이타베이스 캐시 활용 특성이 다르기 때문에 한 가지 원칙에 따라 특정 상황에 가장 적합한 기능 향상 및 튜닝 파라미터 집합을 결정할 수는 없다. 최적의 시스템 구성을 결정하기 위해서는 실제 워크로드에 대한 통계를 수집해야 한다.

최고 4GB RAM을 장착한 시스템을 위한 대형 SGA

Red Hat Linux Advanced Server 2.1에는 2개의 커널이 함께 제공된다. 기본 "smp" 커널은 최대 4GB RAM을 지원하며 2 레벨 페이지 테이블을 사용한다. "enterprise" 커널은 최대 64GB RAM을 지원하며, 용량이 보다 큰 RAM에 액세스하기 위해서는 3 레벨 페이지 테이블이 필요하다. RAM 용량이 4GB 이하인 시스템의 경우, 메모리가 보다 작고 CPU가 2 레벨 페이지 테이블을 사용해야 하기 때문에 "smp" 커널을 사용할 것을 권장한다.

4GB RAM을 장착한 시스템에서 대부분의 Linux 배포판은 Oracle이 SGA에 1.7GB 정도의 주소 공간을 사용할 수 있도록 지원한다. Red Hat Linux Advanced Server 2.1은 /proc 파일 시스템에 조정 가능한 파라미터를 제공하는 최초의 Linux 배포판이다. 이러한 크기를 늘리기 위해서는 Oracle이 보다 낮은 SGA 베이스로 재연결 되어야 하며, Linux는 Oracle을 실행하는 프로세스에 맞춰 보다 낮춰진 매핑된 베이스를 보유하고 있어야 한다. 이를 위해서는 다음과 같이 2가지 단계가 필요하다.

먼저, Oracle에 재연결함으로써 Oracle이 사용하는 SGA 베이스 주소를 낮춰야 한다. 현재, 출시되어 있는 Oracle 제품은 베이스 주소가 0x50000000으로 설정되어 있기 때문에 대부분의 Linux 제품에 설정되어 있는 기본값을 Oracle에서도 사용할 수 있다. 주소값을 낮추면 Oracle은 프로세스에서 보다 많은 주소 공간을 사용할 수 있지만, Advanced Server 2.1의 주소값을 변경해야만 이렇게 변경 및 재연결된 Oracle 바이너리 주소값이 유효하게 된다. 다음과 같은 단계를 통해 Oracle 내에서 SGA 베이스 주소를 낮출 수 있다.

- Oracle의 모든 인스턴스 종료
- cd $ORACLE_HOME/lib
- cp -a libserver9.a libserver9.a.org (백업 복사본 작성을 위해)
- cd $ORACLE_HOME/bin
- cp -a oracle oracle.org (백업 복사본 작성을 위해)

- cd $ORACLE_HOME/rdbms/lib
- genkms-s 0x15000000 〉ksms.s (SGA 베이스 주소값을 0x15000000로 낮춤)
- make-f ins_rdbms.mk ksms.o (새로운 SGA 베이스 주소로 컴파일)
- make-f ins_rdbms.mk ioracle (재연결)

그 다음, Linux 커널에 대한 매핑 베이스 주소값을 Oracle의 새로운 SGA 베이스 주소값 이하로 낮춰야 한다. Advanced Server 2.1은 각 프로세스에 대해 커널에 대한 매핑 베이스 주소값을 낮춰주는 in/proc 파라미터를 가지고 있다. in/proc 파라미터는 시스템 전반적인 파라미터가 아니라 프로세스 당 파라미터이다. 그러나 자식(child) 프로세스에 의해 상속되지는 않는다. 이 파라미터는 루트에서만 변경할 수 있다. 아래 단계에 따라 bash 터미널 세션에 대한 매핑 베이스 주소값을 낮출 수 있다. 낮춰진 매핑 베이스 주소값에 따라 세션이 변경되면 모든 Oracle 명령어에서 이 세션(윈도우)을 사용해야 하고, 따라서 Oracle 프로세스는 계승된(낮춰진) 매핑 베이스 주소값을 사용하게 된다.

- Oracle 인스턴스를 종료한다.
- 터미널 세션(Oracle 세션)을 개시한다.
- 두 번째 터미널 세션과 루트에 대한 su(루트 세션)를 개시한다.
- Oracle 세션에 대한 프로세스 id를 검색한다. 예를 들어, Oracle 세션에서 "echo $$"를 수행한다.
- Oracle 세션에 대한 매핑 베이스 주소값을 0x10000000으로 낮춘다. 루트 세션에서 268435456 〉/proc/〈pid〉/mapped_base를 에코(echo)한다. 이때, 〈pid〉는 앞에서 확인한 프로세스 id이다.
- shmmax 값을 늘리면 Oracle이 한 세그먼트에서 SGA를 할당하게 된다. 루트 세션에서 3000000000〉/proc/sys/kernel/shmmax를 에코한다.
- Oracle 터미널 세션에서 Oracle 인스턴스를 작동시키면 SGA가 낮춰진 주소값에서 시작된다. 따라서, Oracle은 더 많은 주소 공간을 사용할 수 있다.

이제 db_cache_size나 db_block_buffers의 init.ora 값을 늘려서 데이타베이스 버퍼 캐시 크기를 늘릴 수 있게 된다. 이러한 작업을 자동으로 수행하기 위한 명령, 제약 조건 및 스크립트는 Oracle Support 웹사이트에서 제공하고 있다.

최고 64GB RAM을 장착한 시스템 상의 VLM(Very Large Memory)

Oracle은 Advanced Server 2.1 커널을 통해 32-비트 인텔 플랫폼의 데이타베이스 버퍼 캐시에 4GB 이상의 메모리를 할당해서 사용한다. 오라클 설명서에서는 이 기능을 VLM이라고 명명하고 있다. Oracle9*i*R2 on Red Hat Linux Advanced Server 2.1를 통해 시스템에서 사용할 수 있는 RAM 메모리에 따라 SGA를 62GB라는 이론적 한계까지 확장할 수

위에서 설명한 바와 같이, 4GB 이상의 RAM에 액세스 하기 위해 사용되는 "enterprise" 커널은 3 레벨의 페이지 테이블을 필요로 하기 때문에 메모리 용량과 CPU 사용이 증가하게 된다. 한편, 대부분의 워크로드에 있어 추가 RAM에 액세스 할 수 있기 때문에 RAM이 4GB 이상인 시스템에서 확장성 및 성능을 크게 향상시킬 수 있다. 이 옵션을 사용하는 경우 장점과 단점을 모두 고려해야 한다는 점을 유념해야 한다.

있다. 실제 SGA 크기는 현재의 하드웨어 한계와 실제 구현 시 고려사항으로 인해 이론적 수치 보다 훨씬 작아질 수 있지만, VLM을 사용하지 않는 경우와 비교해 볼 때 수 배는 더 크다고 할 수 있다. VLM 모드에서 시스템을 작동하면 Linux에 대한 설정값을 일부 변경해야 하고 사용할 수 있는 기능 및 init.ora 파라미터가 제한된다.

Linux OS 수정: 관리자는 "루트"로서 인메모리(in-memory) 파일 시스템을 데이타베이스 버퍼 캐시에 사용할 수 있는 메모리 용량에 해당하거나 그 보다 더 큰 /dev/shm에 마운트 해야 한다. 인메모리 파일 시스템이 이미 마운트 되어 있을 때는 용량이 초과되기 전까지 이를 사용할 수 있다. 파일 시스템이 마운트 되어 있지 않은 경우에는 아래와 비슷한 마운트 명령어가 수행된다.

mount-t shm shmfs-o nr_blocks=2097152/dev/shm

이 명령어는 8GB에 달하는 /dev/shm에 shmfs 파일 시스템을 생성해준다. nr_block 각각은 4096 바이트이다. 확장 버퍼 캐시 기능을 활성화 함으로써 Oracle을 작동하기 시작한 경우에는 Oracle의 버퍼 캐시에 해당하는 /dev/shm에 파일이 생성된다.

init.ora 변경 내용: init.ora 파일에서 'use_indirect_data_buffers=true'를 설정하면 확장 버퍼 캐시 기능이 활성화 된다. 이를 통해 버퍼 캐시 크기를 더 크게 지정할 수 있다. 그러나, 이 기능은 최신 동적 캐시 파라미터와는 호환되지 않기 때문에, 확장 캐시 기능이 활성화 되어 있을 때는 다음과 같은 파라미터를 사용할 수 없다.

- db_cache_size
- db_2k_cache_size
- db_4k_cache_size
- db_8k_cache_size
- db_16k_cache_size
- db_32k_cache_size

확장 캐시 기능을 사용할 경우에는 db_block_buffers를 통해 데이타베이스 캐시 크기를 지정해야 한다. 확장 캐시 기능과 기타 제한 기능에 대한 자세한 내용은 Oracle9*i*R2 설명서를 참조하기를 바란다.

VLM 모드(init.org 파라미터 'use_indirect_data_buffers=true')를 통해 이미 대형 버퍼 캐시를 가지고 있는 경우에는 Oracle SGA 베이스 주소값과 Linux에 대한 매핑 베이스

주소값(앞 섹션 참조)을 낮춰서 간접 버퍼 윈도우 크기를 늘릴 수 있다. 따라서, 간접 윈도우 크기가 늘어나면 Oracle 주소 공간에 간접 버퍼를 매핑할 필요가 없기 때문에 특정 조건 하에서 성능을 다소 향상시킬 수 있다. 간접 윈도우의 기본 크기는 512MB이다. 크기를 늘리기 위해서는 환경 변수 VLM_WINDOW_SIZE를 Oracle 인스턴스를 작동하기 전의 윈도우 크기(바이트)로 설정해야 한다. 예를 들어, VLM_WINDOW_SIZE=1073741824를 익스포트해서 간접 윈도우 크기를 1GB로 설정할 수 있다. 모든 값은 64KB 단위로 설정되어야 한다.

주:

- 버퍼 캐시 크기(또는 간접 윈도우 크기)가 너무 크면 Oracle을 실행시킬 때 연결 오류가 발생할 수도 있다.
- SGA 베이스 주소값이 낮은데도 /proc/〈pid〉/mapped_base 값을 낮추지 않은 상태에서 Oracle 바이너리 주소값을 사용하면 ORA-3113 오류나 연결 오류 같이 예상치 못한 결과가 나타날 수 있다.
- shmmax 값을 늘리지 않으면 실행 시 연결 오류가 발생할 수 있다.

대형 페이지

IA-32는 4KB 페이지와 "대형 페이지"를 모두 지원한다. PSE(Page Size Extension) 기능이 활성화 되어 있는 정상 모드(Linux 4GB 커널)의 경우, "대형 페이지"는 4MB의 크기를 뜻하며 CPU는 2 레벨 페이지 테이블을 사용한다. PSE가 활성화 되어 있는 PAE(Physical Address Extension) 모드(Linux 64GB 커널)에서는 "대형 페이지"는 2MB의 크기를 뜻하며 CPU는 3 레벨 페이지 테이블을 사용한다.

Linux 커널에서 일반적인 페이지 프레임 크기는 4KB이다. Advanced Server 2.1은 Oracle9*i* R2가 SGA 할당을 위해 대형 페이지(2MB나 4MB)를 사용할 수 있도록 지원한다. SGA를 위해 대형 페이지를 사용하면 다음과 같이 성능을 향상시킬 수 있다.

- IA-32 프로세서는 2 레벨 또는 3 레벨 페이지 테이블을 사용해 4KB 페이지 내의 메모리를 처리함으로써 가상 주소를 물리적 주소로 매핑하게 된다. 내부적으로 TLB(Translation Look-aside Buffer) 엔트리를 사용해 이렇게 번역된 주소를 캐싱할 수 있기 때문에, CPU는 물리 주소를 찾기 위해 모든 메모리에 있는 페이지 테이블에 액세스할 필요가 없다. 프로세서에는 TLB 엔트리가 매우 적기 때문에 Oracle 같이 대용량 메모리에 액세스하는 애플리케이션은 TLB 액세스 실패율(miss rate)이 높아질 수 있다. 프로세서는 대형 페이지 기능을 통해 4MB(모드에 따라서는 2MB) 페이지의 메모리를 처리할 수 있다. 이 경우, TLB의 PTE(Page Table Entry) 엔트리는 4KB가 아닌 4MB의 메모리(1024배 확장된 메모리)를 처리하게 된다. 주소 A, A+4KB, A+8KB, A+4MB에 액세스 하면 TLB 액세스에 실패하는 경우가 없기 때문에 액세스 실패율은 크게 줄어든다.
- 하나의 4MB PTE가 1024개의 4KB PTE를 처리하기 때문에 페이지 테이블 크기는 크게 줄어들고, 따라서 메모리 활용도가 높아진다.
- 대형 페이지는 교체 아웃(swap out) 되지 않으며 이는 전체 db_block_buffers가

물리적 메모리 내에 잠금된다는 것을 의미한다. 따라서 버퍼 성능의 향상을 실현할 수 있게 된다. 커널이 이러한 페이지의 교체 아웃을 고려할 필요가 없기 때문에 시스템 성능의 향상을 실현하게 된다. 페이지에 대해 교체 공간이 미리 할당되어 있지 않기 때문에 더 많은 교체 공간을 사용할 수 있으며 페이지 캐시의 복잡성을 완화시킬 수 있게 된다.

다음과 같은 방법으로 대형 페이지를 활성화시키게 된다.

- 커널 부트 옵션에서 bigpage=〈size〉MB를 부트 로더 파일(예: /etc/lilo.conf)에 추가한다. 이 때, 시스템에 해당되는 MB 값이 페이지 크기가 된다.
- 값 2를 포함하도록 /proc/sys/kernel/shm_use-bigpages 파일을 설정한다. 대형 페이지가 아닌 경우에는 0을 사용할 수 있고, sysV 공유 메모리(shmfs가 아닌)를 사용하는 대형 페이지의 경우에는 1을 사용할 수 있다.

다음과 같은 공식에 따라 시스템의 대형 페이지 최대값을 계산할 수 있다.
대형 페이지의 최대값 = HighTotal/1024*0.8 MB (HighTotal = /proc/meminfo에서 얻은 Kbyte 값)
메모리의 20%는 커널 관리(bookkeeping)에 할당된다고 가정한다. 예를 들어, 8GB RAM을 갖춘 시스템의 경우 HighTotal은 7208944 KB이고, 따라서 대형 페이지 최대값은 약 5631 MB가 된다. 대형 페이지 값이 너무 높게 설정되면 사용자 커넥션에 사용할 수 있는 메모리가 줄어들게 된다.

사용자 수가 많아질수록 해당 대형 페이지의 값은 줄어든다. 사용자 커넥션의 최대 수와 각 사용자 커넥션에 사용되는 메모리 양을 예측해서 다음과 같이 정확한 대형 페이지 값을 계산할 수 있다.
대형 페이지 = (HighTotal – 최대 사용자 커넥션에 필요한 메모리(KB)) / 1024*0.8 MB

프로세스 스케쥴러(Process Scheduler)

프로세스 스케쥴러는 CPU에 대한 프로세스 액세스 제어를 담당한다. Advanced Server 2.1은 새로운 스케쥴러를 사용함으로써 기존 설계의 많은 한계를 해결했다. 다음은 2002년 1월 3일에 있었던 Linux 커널 메일링 리스트에 대한 잉고 몰나르(Ingo Molnar)의 설명을 토대로 향상된 기능을 요약해 놓은 것이다.

- 0(1) 고정 시간 스케쥴링 알고리즘

- 새로운 스케줄러는 대형 runqueue에 대해 잠금(lock)을 수행하는 대신 CPU 단위로 runqueue 및 잠금 기능을 구현한다. 인터로킹(interlocking) 없이 별도의 CPU에서 작업 스케줄링을 병렬 수행할 수 있기 때문에 SMP 확장성이 향상된다.
- 기존 스케줄러에서는 과중한 로드로 인해 CPU 간의 프로세스 이동이 많았다. 새로운 스케줄러는 전역 동기화(global synchronization)나 재계산 대신, CPU 단위로 분할된 시간을 분배하는 등 CPU 친화성(affinity)을 높였다.

또한, 새로운 스케줄러는 스레드가 일정 시간 동안 캐시 핫(cache hot) 상태에 있을 경우 스레드를 스케줄링하는 대신 스레드에서 캐시가 차지하는 공간을 추적하는 등 캐시 친화력을 높였다. 이로써 프로세스에 대한 캐시 적중률(cache-hit rate)을 향상시켰다.

신뢰성 개선

신뢰성 및 안정성을 향상시키기 위해 Oracle9*i*R2 on Advanced Server 2.1는 자원 부족을 해결할 수 있는 내구성을 향상시키는 것을 목표로 광범위한 워크로드 상에서 테스트됐다. 강력한 엔터프라이즈급 운영 시스템은 피크 사용 시간의 폭증하는 요구를 원활하게 처리할 수 있어야 한다. 또한 엔터프라이즈급 시스템 상의 성능은 사용자 로드가 시스템 용량을 초과하면서 점진적으로 저하된다. 오라클과 레드햇은 전체 소프트웨어 스택(데이타베이스 및 커널)에서 "취약 지점"을 파악해 내 높은 사용자 로드 하에서 안정성을 향상시킬 수 있는 많은 개선 기능을 추가했다. 이러한 기능 개선은 IO, 메모리 관리, 네트워킹 및 프로세스 스케줄링의 영역에서 이뤄졌다. Oracle9*i*R2와 Advanced Server 커널 모두는 단일 노드 뿐만 아니라 RAC 멀티노드 구성에서 다수의 데이타베이스 사용자들을 지원할 수 있도록 강화됐다. 예를 들면 Oracle Applications 11*i*와 같은 워크로드 성능은 동시 사용자의 수가 시스템 용량을 넘어서게 되면 점진적으로 저하된다. 본 섹션에서는 신뢰성을 높이고 단계적인 성능 저하를 지원하는 여러 주요 기능 개선에 대해 설명하고 있다.

고 메모리 PTE 패치

이전 섹션에서 설명된 바와 같이 IA-32 시스템에서 지원되는 최초 1GB의 물리적 메모리는 "저 메모리"라고 알려져 있으며 1GB 이상의 물리적 메모리는 "고 메모리"라고 불린다. 기존 커널에서 Linux는 1GB로 제한된 저 메모리에서만 PTE(Page Table Entries)을 할당할 수 있다. 대형 메모리 및 다수의 프로세스를 사용하는 Oracle9*i*R2와 같은 애플리케이션의 경우 PTE의 전체 크기는 크다. 데이타베이스로 연결되는 사용자의 수가 많아질수록 커널은 PTE의 공간을 모두 소비하게 되며, 여유 메모리 및 스왑 공간이 존재한다 해도 시스템은 중단되거나 충돌하게 된다.

고 메모리 PTE 패치는 VM이 고 메모리를 사용해 PTE를 할당하도록 지원한다. 데이타베이스로 연결하는 사용자의 수가 증가되고 프로세스가 추가적으로 생성될 경우 PTE를 저장하는 영역은 고 메모리로 오버플로우되어 기존 커널에서 지원했던 사용자 수 보다 3배 5배 많은 사용자 수를 지원할 수 있게 된다. 또한 시스템 용량을 초과해 사용자의 수가 증가되는 경우 데이타베이스 응답 시간은 점차 느려지게 되며 결국 더 이상의 데이타베이스 사용자 커넥션을 허용하지 않게 된다. 따라서 Advanced Server 커널은 이전 커널의 경우 앞서 설명된 바와 같은 충돌 또는 중단을 발생시키지 않는다.

Oracle9*i*R2 RAC 확장성 개선

Oracle9*i* 및 이전 버전의 경우 RAC Cluster Manager는 TCP/IP 프로토콜을 사용하여 클러스터의 노드 간에 커뮤니케이션했다. 각 Cluster Manager 프로세스의 TCP 커넥션 수는 제품 사용자 수와 노드 수에 비례해 증가됐다. 다수의 노드(예, 8개 이상의 노드)로 구성되어 있는 클러스터에서 프로세서 당 TCP/IP 커넥션의 수가 제한될 경우 클러스터 전반에서 동시 데이타베이스 사용자 커넥션의 전체 수가 제한됐다.

Oracle9*i*R2의 Cluster Manager는 UDP 프로토콜을 사용해 클러스터의 노드 간에 커뮤니케이션한다. UDP는 커넥션 리스(connection-less) 프로토콜로서 프로세스 당 최대 TCP/IP 커넥션의 수가 제한되더라도 이러한 경우에 영향을 미치지 않는다. 따라서 다수의 노드로 구성된 Linux RAC 클러스터는 Oracle on Linux의 이전 버전 보다 많은 수의 동시 사용자를 지원할 수 있게 된다.

관리 용이성 개선

오라클과 레드햇은 긴밀한 협력을 통해 Linux 기반의 오라클 제품을 위해 사용 용이성 및 지원 능력을 향상시킬 수 있도록 관리 용이성 개선을 추진하고 있다. 현재 이들 작업의 상당 부분이 진행 중인 상황에서 본 섹션에서는 Oracle9*i*R2 on Red Hat Linux Advanced Server 2.1 버전 출시와 함께 제공되는 새로운 일부 강화 기능에 대해 설명하고 있다.

lsraid 유틸리티

소프트웨어 RAID 스토리지 관리를 위해 오라클이 개발한 "lsraid" 유틸리티를 한 예로 들 수 있다. 다음은 Advanced Server 2.1의 "메인(man) 페이지"에서 lsraid를 설명한 부분이다.

lsraid는 Linux md 디바이스를 질의하는 프로그램으로서 복합 디바이스와 이에 속해있는 블록 디바이스를 설명할 수 있다. 또한 이 프로그램은 /etc/raidtab 구성 파일에 포함될 수 있는 md 디바이스를 나타낼 수 있다.

lsraid는 온라인 및 오프라인 디바이스에서 운영될 수 있다. 커널 인터페이스를 통해 온라인 디바이스를 읽고 이에 대한 정보를 제공할 수 있다. 디바이스가 오프라인일 경우 lsraid는 md 디바이스에 포함된 모든 블록 디바이스를 확인할 수 있으며 영속적인 md superblock 를 읽어 정보를 확인할 수 있다.

네트워크 콘솔 및 덤프 장애 퍼실리티

Advanced Server 2.1은 레드햇 최초의 "덤프 장애" 퍼실리티와 함께 번들로 제공된다. 네트워크 콘솔 기능은 Linux 장애 서명 메시지를 포함한 모든 커널 메시지를 네트워크를 통해 중앙 집중화된 서버로 로깅할 수 있도록 지원한다. 이 툴은 시스템 및 커널 로그에 대한 일관적이면서 중앙집중화된 뷰를 제공하여 고객 사이트의 문제를 신속하게 해결할 수 있도록 한다. 자세한 내용은 레드햇 웹사이트, http://www.redhat.com/support/wpapers/redhat/netdump/index.html 의 "레드햇의 네트워크 콘솔 및 덤프 장애 퍼실리티(Red Hat, Inc.′s Network Console and Crash Dump Facility)" 백서를 참조하면 된다.

클러스터 관리

Oracle Cluster Manager는 Oracle RAC 아키텍처의 컴포넌트이다. Oracle Cluster Manager는 클러스터 멤버십 및 노드 모니터링 서비스를 구현한다. Cluster Manager 기능에서 중요한 부분은 클러스터에서 장애가 발생된 노드를 정확히 피해갈 수 있도록 지원하는 것으로 Linux 와치도그(watchdog) 디바이스를 통해 재설정된 하드웨어 노드를 통해 구현된다. 오라클은 와치도그 코드를 여러 측면에서 향상시켜 RAC on Linux의 관리 기능을 강화했다.

예를 들면 와치도그 타이머 마진(margin)은 와치도그 모듈 로딩 중 지정된 soft_margin 파라미터에 의해 정의된다. 하지만 이전 커널에서는 런타임에 커널에서 soft_margin의 값을 가져오는 메커니즘이 지원되지 않았다. 사용자는 soft_margin을 미러링했던 Cluster Manager에 대해 부가적인 파라미터를 명시적으로 설정해야만 했다. 오라클은 Advanced Server 2.1의 코드를 향상시켜 Cluster Manager가 런타임에 ioctl 콜을 통해 커널에서 soft_margin 값을 가져 올 수 있도록 지원한다.

Oracle9*i*R2 on Advanced Server 2.1의 전체 I/O 펜싱(fencing) 솔루션은 Oracle on Linux의 이전 버전에서 구현된 I/O 펜싱 기능 보다 뛰어나다. 예를 들면 기본 구성은 "셧다운 중지" 명령어를 해당 노드의 데이타베이스 인스턴스로 발행한 다음 하드웨어 재설정을 요구하지 않도록 하는 것이다.

64 비트 지원

64 비트 아이테니엄 기반 시스템의 발표을 통해 Linux는 인텔 플랫폼 상에서 현저한 성능 및 확장성 향상을 이루어냈다. 오라클은 Oracle8*i* Release 8.1.7이후 IA-64 상의 데이타베이스 소프트웨어의 개발자 버전을 발표했다. Oracle9*i*R2 on IA-64의 현업 버전은 인텔 아이테니엄 2 프로세서를 토대로 한 시스템과 함께 출시될 예정이다. Oracle on IA-64는 다수의 사용자들까지 지원할 수 있도록 확장됐을 뿐만 아니라 대규모 SGA 할당을 지원하고 32 비트 Linux의 데이타베이스 보다 더 높은 CPU 및 I/O 용량을 제공하게 된다.

오라클은 레드햇, 인텔 및 OEM 협력업체들과 긴밀히 협력해 아이테니엄 2 기반 시스템에서 모든 향상 기능을 적극 활용할 수 있도록 지원하고 있다. Red Hat Linux Advanced Server 2.1 on IA-64는 32 비트 버전에서의 모든 기능 향상 뿐만 아니라 아이테니엄 2 기반 시스템에 특화된 다양한 강화 기능을 지원하게 될 것이다.

결론

오라클은 Linux 지원을 위해 부단한 노력을 경주하고 있다. Linux 상의 오라클 플랫폼을 더욱 향상시키기 위해 오라클은 레드햇과 협력해 하이엔드 성능, 신뢰성 및 관리 용이성을 실현하는 기능 개선을 정의 및 개발하고 있다. Red Hat Linux Advanced Server 2.1에서 제공되는 새로운 기능을 통해 Oracle on Linux는 Linux 및 상용 하드웨어 플랫폼 상에 설치하는데 따른 비용 이점을 유지하는 동시에 엔터프라이즈급 성능 및 안정성을 제공하고 있다.

ORACLE®

한국오라클(주)
서울특별시 강남구 삼성동 144-17
삼화빌딩
대표전화 : 2194-8000
FAX : 2194-8001

한국오라클교육센타
서울특별시 영등포구 여의도동 23-10
SK증권빌딩 11층(사무실)
19 · 20층(강의실)
대표전화 : 3779-4000
FAX : 3779-4100 1

대전사무소
대전광역시 서구 둔산동 929번지
대전둔산사학연금회관 18층
대표전화 : (042)483-4131 2
FAX : (042)483-4133

대구사무소
대구광역시 동구 신천동 111번지
영남타워빌딩 9층
대표전화 : (053)741-4513 4
FAX : (053)741-4515

부산사무소
부산광역시 동구 초량동 1211 7
정암빌딩 8층
대표전화 : (051)465-9996
FAX : (051)465-9958

울산사무소
울산광역시 남구 달동 1319-15번지
정우빌딩 3층
대표전화 : (052)267-4262
FAX : (052)267-4267

광주사무소
광주광역시 서구 양동 60-37
금호생명빌딩 8층
대표전화 : (062)350-0131
FAX : (062)350-0130

고객에게 완전하고 효과적인
정보관리 솔루션을 제공하기 위하여
오라클사는 전 세계 145개국에서
제품, 기술지원, 교육 및
컨설팅 서비스를
제공하고 있습니다.

http://www.oracle.com
http://www.oracle.com/kr

제품구입문의

수신자부담 전화번호 : 00368-440-0051 수신자부담 팩스번호 : 00368-440-0062 E-Mail문의 : oracleisd_kr@oracle.com