Solaris 파일 시스템

파일 시스템의 성능(1)

지난 2회에 걸쳐 Solaris의 세 가지 일반적 파일 시스템, 즉 Solaris UFS 파일 시스템과 Veritas VxFS 파일 시스템, LSC QFS 파일 시스템의 주요 특징에 대해 알아보았다. 이번 호부터는 2회에 걸쳐 파일 시스템의 성능에 영향을 미치는 몇 가지 요인을 살펴보고, 파일 시스템의 구성 옵션을 다르게 설정했을 때 이들이 성능에 각각 어떻게 영향을 주는지 알아본다. 단 여기서는 파일 시스템의 캐싱만을 다룬다.

정리 · 김봉환 | 한국 썬 시스템엔지니어링 본부 과장

파일 시스템 성능

파일 시스템 성능은 전체 시스템 성능의 중요한 요소이며, 시스템에 부하를 발생시키는 애플리케이션의 특성에 의해 많이 좌우된다. 최적의 성능을 얻으려면 해당 파일 시스템 구성이 애플리케이션의 특성과 잘 매칭될 수 있도록 균형을 이뤄야 한다.

개발자들은 이미 애플리케이션이 파일 시스템을 통해 읽고 쓰는 방법에 대한 좋은 식견을 가지고 있겠지만, 애플리케이션 관리자들은 어떠한 형태의 I/O 프로파일이 파일 시스템에 적용되는지를 이해하고자 애플리케이션을 분석해야 한다. 애플리케이션을 완전히 이해하고 난 다음에는 주어진 스토리지 장비를 가장 효율적으로 사용할 수 있도록 파일 시스템 구성을 최적화시켜야 한다.

· 가능한 한 주어진 장비의 I/O 횟수를 최소화시킨다
· 소형 I/O들을 가능한 한 대형 I/O로 그룹화시킨다
· 디스크 검색을 위해 대기하는 데 소요되는 시간을 줄일 수 있도록 검색 패턴을 최적화시킨다
· 물리적 I/O를 줄일 수 있도록 데이터를 캐싱한다

워크로드(Workload) 프로파일에 대한 이해

파일 시스템을 구성하기 전에 먼저 파일 시스템을 사용하게 될 워크로드의 특성을 이해해야 한다. 간단한 애플리케이션 워크로드 프로파일을 먼저 살펴본 후 애플리케이션을 추적해, 주어진 애플리케이션에 어떠한 프로파일을 적용할지를 결정한다.

표 1에서 보는 바와 같이 워크로드 프로파일의 주요 특성들은 5개의 범주로 묶을 수 있다.

데이터 혹은 속성 집중적?

데이터 집중식 애플리케이션은 많은 파일을 생성하거나 삭제하지 않은 채 수많은 데이터가 집중된 형태인 반면 속성 집중식 애플리케이션은 수많은 파일을 생성하거나 삭제하지만 읽거나 쓰는 데이터는 아주 적은 애플리케이션이다. 데이터 집중식 워크로드의 한 예로 20GB의 파일을 생성하는 과학 연산 배치 프로그램을 들 수 있으며, 속성 집중식 워크로드의 예로는 1MB 미만의 수백개의 작은 파일을 생성 및 삭제, 저장하는 OA 파일 서버를 들 수 있다.

접근 형태

애플리케이션의 접근 형태는 수행 가능한 최적화 작업량에 따라 여러 가지 형태가 있다. 애플리케이션은 파일을 통해 순차적으로 읽기/쓰기를 할 수 있거나 랜덤하게 파일에 접근할 수 있다. 순차적 워크로드는 튜닝하기가 쉽다. 사용자는 I/O를 그룹화할 수 있으며, 주어진 스토리지 장비에 대한 I/O를 최적화시킬 수 있기 때문이다.

접근 형태의 다른 형식으로는 스트라이디드 접근으로, 과학 애플리케이션에서 볼 수 있다. 스트라이디드 방식은 작은 그룹(대략 1MB 정도)에서 순차적으로 작업을 걸며, 파일 내에서 검색하고 소량의 다른 순차적 블록을 기록한다. 이것은 파일 시스템을 튜닝할 때 가장 순차적인 형태로 보이지만, 랜덤한 워크로드와 비슷한 캐시 성격도 갖는다.

대역폭

애플리케이션의 대역폭은 애플리케이션이 가진 데이터의 양 및 그 파일에 따라 성격이 규정된다. 대부분의 경우, 대역폭은 스토리지 장비의 용량 계획에 유용하지만 애플리케이션이 사용하는 대역폭의 양에 영향을 받는 캐시 특성도 매우 중요하다.

I/O 크기

각 I/O의 크기는 파일 시스템이 어떻게 구성되는가에 많은 영향을 미친다. I/O 장비들은 적은 I/O에는 비교적 효율성이 떨어지므로, 주변의 적은 I/O를 그룹화해 스토리지 장비로 전송하는 파일 시스템을 사용할 수 있다. 랜덤한 워크로드 형태에서 I/O 크기는 파일 시스템의 블록 크기와 강력하게 상호 작용한다.

잠재적 민감성

몇몇 애플리케이션은 각 I/O가 수행되는 시간에 매우 민감하다. 예를 들어 DB는 DB의 로그 파일을 기록하는 데에 소요되는 시간에 매우 민감하다. 이러한 형태의 애플리케이션은 캐시 전략을 수립해 사용하면 상당히 유용하다.

데이터 집중적 순차적 워크로드

순차적 워크로드는 오름차순, 혹은 내림차순으로 순차적 파일 블록을 읽고 쓰는 반복적 I/O를 수행하는 것들이다. 대용량 데이터를 옮길 때 흔히 순차적 패턴을 볼 수 있다. 예를 들어 파일을 복사하고 대용량 파일을 읽거나 쓸 때가 이에 해당된다.

읽고 쓰거나 lseek 시스템 호출을 살펴보는 것을 통해 애플리케이션의 액세스 패턴의 특성을 알아보기 위해 truss 명령어를 사용할 수 있다. 다음에서는 프로세스 id 20432번인 애플리케이션이 발생시킨 시스템 호출을 추적하기 위해 truss 명령어를 사용했다.

```
# truss -p 20432

read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
read(3,  "\0\0\0\0\0\0\0\0\0\0\0\0" .., 512)      = 512
```

애플리케이션이 512바이트의 블록을 읽도록 결정하기 위해 읽기 시스템 호출로부터 독립변수를 사용한다. 애플리케이션이 순차적으

표 1. 애플리케이션 특성의 예

특성	값	설명
파일 액세스 프로파일	데이터 혹은 속성 집중적	애플리케이션이 다량의 소형 파일을 읽기/쓰기/생성/삭제하는가 혹은 기존 파일을 읽기/쓰기만 하는가?
액세스 패턴	랜덤/순차적/스트라이디드	읽기/쓰기가 랜덤한 순서인가 순차적 순서인가?
대역폭	MB/s	애플리케이션이 요구하는 대역폭은? 애플리케이션이 파일 시스템에서 읽거나 쓰기 작업을 할 때 데이터의 평균 속도 및 최고 속도는?
I/O 크기	바이트	요청된 일반적 I/O 크기는? 이것이 파일 시스템의 블록 크기와 매치되는가?
잠재적 민감성	ms	애플리케이션이 잠재적 읽기 및 쓰기 작업에 민감한가?

로 읽고 있음을 추론하는 각 읽기 작업간에는 lseek 시스템 호출이 없기 때문이다. 애플리케이션이 첫 번째 읽기를 요청하면 파일 시스템은 해당 파일에 대한 첫 번째 파일 시스템 블록을 읽게 되며 8KB의 chunk가 읽혀진다. 이 작업은 물리적인 디스크 읽기 작업을 요청하게 되는데, 이것은 수백ms 내에 이뤄진다. 두 번째 512바이트 읽기 작업은 아직 메모리에 남아 있는 8KB의 시스템 블록 내의 다음 512바이트를 간단히 읽어들이는 작업이며, 수백㎲ 내에 이뤄진다. 즉 각 8KB의 데이터 읽기를 할 경우, 한 번의 물리적 디스크 읽기 작업만을 보게 되는 것이다.

이것은 애플리케이션의 성능 향상에 많은 도움을 준다. 각 디스크 I/O는 수백ms 사이에 이뤄지기 때문에, 512바이트마다 모두 디스크 I/O를 위해 대기해야 한다면 애플리케이션은 디스크를 위해 대기하는 시간에 모든 것을 빼앗기게 될 것이다. 8KB 블록의 물리적 디스크를 읽는다는 것은 매 512바이트 읽기마다 디스크 I/O가 일어나는 것이 아니라 총 16개의 읽기를 한번의 디스크 I/O때 읽는다는 것을 의미하며, I/O를 위한 대기 시간에 소요되는 시간을 줄여준다.

각 파일 시스템 형태에서 사용되는 실제 알고리즘은 매우 다양하지만, 이들 모두 동일한 원칙을 따르고 있다. 이 알고리즘들은 최근의 액세스 패턴을 찾아서 대량의 블록을 우선적으로 읽는다. 먼저 읽어들이는 대량의 블록은 보통 구성이 가능하며, 종종 기본값은 최적의 성능을 제공하기에 충분할 정도로 크진 않다.

미리 읽기는 순차적 성능에 도움을 준다

그런데 16개의 512바이트 블록 데이터를 읽기 위해 디스크 I/O가 대기해야 한다는 것은 비효율적인 것이다. 이러한 512바이트 블록을 처리하는 데에는 단지 수백㎲밖에 소요되지 않지만, 디스크 내의 다음 8KB 블록을 읽기 위해 대기하는 데 10ms나 소요됨을 알기 때문이다. 이러한 관점에서 계산해보면 이 작업은 10분간 여행하기 위해 버스를 타고, 내려서 다시 10분간을 여행하기 위해 6시간동안 버스를 기다리는 것과 동일한 비율이다.

파일 시스템은 이러한 문제를 간단히 해결해버릴 만큼 영리하다. 액세스 패턴은 반복적이므로, 파일 시스템은 사용자가 순차적인 순서로 읽으면 사용자가 다음 블록을 읽을 것을 미리 예측할 수 있다. 대부분의 파일 시스템은 이러한 작업을 수행하는 알고리즘을 구현하고 있는데, 이것을 '미리 읽기' 알고리즘이라고 한다. 미리 읽기 알고리즘은 요청된 현재의 블록을 검색해 파일이 순차적으로 읽혀지는 것을 알아낸다. 그후 이미 요청된 마지막 블록을 이것과 비교한다. 만일 이들이 서로 인접해 있다면, 파일 시스템은 현재의 블록을 읽어들인 것처럼 다음 몇 개의 블록을 먼저 읽기 시작한다. 그 후 읽기 작업을 해야 할 다음 블록으로 되돌아와 있으므로, 이를 읽기 위해 정지하거나 기다릴 필요가 없는 것이다. 앞서의 예로 돌아가서 생각해보면 미리 전화를 걸어 다음 번에 탈 버스를 예약해서 우리가 버스에서 내리면 타야 할 다른 버스가 대기하고 있기 때문에 기다릴 필요가 없는 것과 마찬가지다.

각 파일 시스템 형태에서 사용되는 실제 알고리즘은 매우 다양하지만, 이들 모두 동일한 원칙을 따르고 있다. 이 알고리즘들은 최근의 액세스 패턴을 찾아서 대량의 블록을 우선적으로 읽는다. 먼저 읽어들이는 대량의 블록은 보통 구성이 가능하며, 종종 기본값은 최적의 성능을 제공하기에 충분할 정도로 크진 않다.

UFS 파일 시스템 미리 읽기

UFS 파일 시스템은 마지막으로 읽은 기록을 추적함으로써 언제 미리 읽기를 구현할지 결정한다. 만일 마지막 읽기 작업과 현재의 읽기 작업이 순차적이라면 파일 시스템 블록의 그 다음 번을 미리 읽게 된다. UFS 미리 읽기에서 실행하게 될 여러 가지 범주는 다음과 같다.

- 읽어들인 마지막 파일 시스템은 현재 파일과 순차적이어야 한다.
- 동시에 파일을 읽는 프로세스는 하나뿐이어야 한다(다른 프로세스가 파일을 읽게 되면 파일에 대한 순차적 접근 패턴이 깨진다).
- 읽을 파일 블록은 디스크 상에 순차적으로 배열되어 있어야 한다.
- 이 파일은 시스템 호출을 읽고 쓰는 것을 통해 읽고 쓰여져야 하며, 메모리에 매핑된 파일들은 UFS 미리 읽기를 사용하지 않는다.

UFS 파일 시스템은 먼저 미리 읽기를 하는 블록의 양을 나타내는 클러스터의 크기 개념을 사용한다. 기본값은 Solaris 2.6 버전까지는 7×8KB 블록(56KB)이며(56KB가 구형 I/O 버스 시스템의 최대 DMA 크기였음), Solaris 2.6에서는 기본값이 주어진 장비가 지원하는 최고의 전송 크기로 변경되어 대부분의 스토리지 장비에서 16×8KB 블록(128KB)이 되었다.

미리 읽기의 기본값은 때때로 적절하지 못한 수치일 수도 있으므로, 최적의 읽기 속도를 허용할 수 있도록 크게 세팅되어야 한다. 미리 읽기 클러스터의 크기 역시 최신 I/O 시스템의 이점을 살릴 수 있는 고성능 순차적 액세스에 알맞게 매우 크게 세팅되어야 한다. 최신 I/O 시스템은 큰 대역폭을 갖고 있지만, 각 I/O의 비용은 여전히 고려 대상이므로, 결론적으로 사용자들은 각 I/O의 양을 최소화시킬 수 있도록 클러스터의 크기를 최대한 크게 선택하고자 하는 것이다.

다음의 스토리지 장비에 대해 보고된 I/O의 평균치를 살펴보면 512바이트의 읽기 사례에 대한 기본 성능을 살펴볼 수 있다.

 # iostat -x 5

device	r/s	w/s	kr/s	kw/s	wait	actv	svc_t	%w	%b
fd0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
sd6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
ssd11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
ssd12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
ssd13	**49.0**	0.0	**6272.0**	0.0	0.0	3.7	73.7	0	93
ssd15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
ssd16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0
ssd17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0	0

iostat 명령어를 이용해 우리는 ssd113 디스크에 초당 49회의 읽기 작업을 수행시켰으며, 이는 평균 초당 6,272KB에 달한다. 만일 전송률을 초당 I/O 횟수로 나눈다면 평균 전송량이 128KB임을 유추할 수 있다. 이를 통해 기본값 128KB의 클러스터 크기는 512바이트의 읽기 요청을 128KB의 그룹으로 묶고 있음을 알 수 있다.

mkfs 명령어에 현재의 파일 시스템 패러미터를 보여주는 -m 옵션을 사용하면 UFS 파일 시스템의 클러스터 크기를 살펴볼 수 있다. 미리 읽기 크기나 클러스터 크기는 maxcontig 패러미터를 사용해 나타낼 수 있다.

 # mkfs -m /dev/rdsk/c1t2d3s0 3105360 **UFS**
mkfs -F ufs -o nsect=80,ntrack=19,bsize=8192,fragsize=1024,
cgsize=32,free=4,rps=90,nbpi=4136,opt=t,apc=0,gap=0,nrpos=8, **VxFS**
maxcontig=16 /dev/rdsk/c1t2d3s0 3105360

이 파일 시스템의 경우 클러스터의 크기가 8,192바이트로 된 16개의 블록 혹은 128KB임을 알 수 있다. 몇몇 파일 시스템에서는 fstyp -v 명령어를 이용할 수도 있다.

 # fstyp -v /dev/dsk/c1t4d0s2 : head -n 20 **UFS**
ufs

magic	11954	format	dynamic	time	Sun May 23 16:16:40 1999		
sblkno	16	cblkno	24	iblkno	32	dblkno	400
sbsize	2048	cgsize	5120	cgoffset	40	cgmask	0xffffffe0
ncg	86	size	2077080	blocks	2044038		
bsize	8192	shift	13	mask	0xffffe000		
fsize	1024	shift	10	mask	0xfffffc00		
frag	8	shift	3	fsbtodb	1		
minfree	3%	maxbpg	2048	optim	time		
maxcontig	**16**	rotdelay	0ms	rps	90		
csaddr	400	cssize	2048	shift	9	mask	0xfffffe00
ntrak	19	nsect	80	spc	1520	ncyl	2733
cpg	32	bpg	3040	fpg	24320	ipg	2944
nindir	2048	inopb	64	nspf	2		
nbfree	190695	ndir	5902	nifree	247279	nffree	304
cgrotor	31	fmod	0	ronly	0		

fs_reclaim is not set
file system state is valid, fsclean is 2

미리 읽기 클러스터 크기 선택

상식적으로 볼 때 최대한의 대역폭을 얻기 위해 초당 200번 이상의 I/O 동작을 할 수 있는 기기는 없다. 이러한 법칙을 이용해 우리는 먼저 읽어야 할 파일 시스템에 대한 최적의 클러스터 크기를 알아낼 수 있고, 초당 I/O 동작의 횟수를 유지해 호스트 CPU 시간을 절약할 수 있다. 운영 체계가 많은 SCSI 요청을 발생시킬 필요가 없기 때문이다.

iostat 명령어와 함께 보여준 앞서의 예제에는 7,200rpm의 4GB 디스크가 최대한의 대역폭을 얻는 데에는 단지 초당 49회의 I/O 동작만이 필요하다고 되어 있다. 오늘날 볼 수 있는 10,000rpm 디스크는 초당 20MB의 용량을 가지고 있으며, 기본 클러스터 크기도 128k에 달하고 있다. 디스크들의 하드웨어 레이드 컨트롤러나 소프트웨어 레이드 스트라이프와 같은 최신형 스토리지 장비는 보다 빠른 전송 속도를 제공한다. 초당 50~100MB의 전송률을 자랑하는 최신형 스토리지 장비를 흔히 볼 수 있을 것이다. 이러한 장비에 파일 시스템을 장착할 경우, 효율적으로 미리 읽기 성능을 발휘하기 위해서는 클러스터 크기를 다른 값으로 사용해야 한다.

초당 100MB를 제공하는 Sun StorEdge A5200 광 스토리지 어레이의 경우, 초당 200회의 동작을 이용하려면 클러스터 크기를 512k로 해야만 이 성능을 최대한 사용할 수있다.

마치며

데이터 집중적 워크로드와 관련된 추가적인 몇 가지 사항들은 다음 호에서 계속 살펴보겠다.