챕터 1: ZFS 소개

시작하셨습니다. ZFS를 사용해 본 적이 없다면, 잠시 시간을 내어 테스트 시스템이나 가상 머신에 ZFS가 포함된 새로운 FreeBSD를 설치해 보세요. 암호화나 화려한 커스터마이징 옵션은 선택하지 마세요. 이 간단한 설치는 더 복잡한 설정으로 들어가기 전에 몇 가지 ZFS 기본 사항을 살펴볼 수 있는 기회를 제공합니다.

ZFS는 기존 파일 시스템과 볼륨 관리자의 기능을 결합한 것입니다. 따라서 개별 파일에 대한 권한과 어떤 파일이 어떤 디렉터리에 있는지부터 어떤 저장 장치가 어떤 용도로 사용되고 그 저장 장치가 어떻게 배열되는지 추적하는 것까지 모든 것을 처리할 수 있습니다. 시스템 관리자는 디스크와 파일을 정렬하도록 ZFS에 지시하지만, 그 아래의 전체 스토리지 스택은 ZFS가 관리합니다. 이 장에서는 기본 ZFS 설정으로 설치된 FreeBSD 10.1 호스트를 사용하여 ZFS 스택을 파일 시스템, 스토리지 풀 및 가상 장치의 세 가지 계층으로 분리합니다.

방향을 잡기 위해 스토리지 스택에서 가장 눈에 잘 띄는 부분부터 시작해서 아래로 내려갑니다. 각 계층이 서로 어떻게 결합되어 있는지 이해했다면, 이 책의 나머지 부분은 기초부터 시작하여 위로 올라갑니다.

ZFS 데이터 세트 (ZFS Datasets)

ZFS 파일시스템은 기존 파일시스템과 정확히 유사하지 않으므로 데이터 세트라고 부릅니다. 고전적인 유닉스 파일 시스템(UFS)과 그 파생물, 그리고 최신 BSD의 UFS2와 Linux의 extfs와 같은 유사 파일 시스템은 다양한 프로그램으로 파일 시스템을 관리합니다. **df(1), newfs(8), mount(8), umount(8), dump(8), restore(8)** 및 이와 유사한 명령어를 사용하는 데 익숙하실 것입니다. ZFS는 이러한 모든 기능을 zfs(8) 프로그램에 통합하여 ZFS 데이터세트를 생성, 삭제, 조회, 스핀들링할 수 있게 해줍니다.

zfslist로 기존 ZFS 데이터세트를 보는 것으로 시작하세요.

```
$ zfs list
NAME              USED  AVAIL  REFER  MOUNTPOINT
zroot            429M  13.0G    96K  none
zroot/ROOT          428M  13.0G    96K  none
zroot/ROOT/default  428M  13.0G   428M  /
zroot/tmp         104K  13.0G   104K
/tmp zroot/usr          428K  13.0G    96K  /usr
...
```

이것은 mount(8)과 df(1)의 출력을 결합한 것으로, UFS나 extfs를 관리해 본 사람이라면 누구나 익숙하게 볼 수 있을 것입니다.

각 데이터세트에는 이름이 있습니다. ZFS 데이터 세트 이름은 데이터 세트가 있는 ZFS storage polol, 즉 zpool로 시작합니다. 첫 번째 항목은 그냥 평범한 zroot라고 합니다. 이 항목은 다른 모든 것이 매달려 있는 풀의 root dataset를 나타냅니다.

다음 두 열은 사용된 공간과 사용 가능한 공간의 양을 보여줍니다. 풀 루트에는 429MB와 13GB의 여유 공간이 있습니다.

REFER 열은 ZFS에만 있는 특별한 열입니다. 이것은 데이터 집합에서 액세스할 수 있는 데이터의 양이며, 사용된 공간의 양과 반드시 같지는 않습니다. 스냅샷과 같은 일부 ZFS 기능은 서로 간에 데이터를 공유합니다. **zroot** 항목은 429MB를 "사용"했지만 96KB의 데이터만 참조합니다. 풀 전체에는 13GB의 여유 공간이 있지만, 이 특정 데이터 집합을 통해 96KB에 액세스할 수 있습니다. 그다지 많지 않은 양입니다. 나머지 공간은 이 데이터 세트의 자식들을 위해 사용됩니다. 6장에서는 ZFS 디스크 사용에 대해 자세히 설명합니다. 데이터 세트의 자식에는 이 책 전체에서 볼 수 있듯이 스냅샷, 볼륨 및 자식 데이터 세트가 포함됩니다.

마지막으로 파일 시스템 마운트 지점이 있습니다. **zroot** ZFS는 마운트되지 않았습니다.

두 번째 항목인 **zroot/ROOT**를 살펴보세요. 이것은 루트 파일시스템용으로 생성된 ZFS 데이터 세트입니다. **zroot** 풀과 마찬가지로 마운트되지 않습니다. 96KB의 데이터를 참조합니다. 이것은 루트 파일 시스템에서는 이상하게도 사용되지 않는 것으로 보입니다.

세 번째 항목인 **zroot/ROOT/default**는 현재 루트 파일시스템입니다. 이 파일은 428MB의 데이터를 사용하며, 유닉스 루트인 /에 마운트됩니다. 428MB라는 의미는 이 데이터 세트에 그 정도의 데이터가 있다는 뜻입니다.

ZFS가 이 데이터를 루트 파일 시스템에서 분리하는 이유는 무엇일까요? ZFS를 사용하면 여러 루트 파일시스템 중에서 쉽게 선택할 수 있습니다. 이 호스트는 FreeBSD 10.1을 실행하는데 일부 보안 업데이트를 적용하고 재부팅해야 한다고 가정해 보겠습니다. 운영 체제 패치를 적용할 때면 시스템 관리자는 항상 두려움과 희망이 뒤섞인 고민에 빠지게 됩니다. 잘 테스트된 업그레이드도 잘못되면 모든 사람의 하루를 망칠 수 있습니다. 하지만 ZFS를 사용하면 데이터 세트를 복제하고 스냅샷할 수 있습니다. FreeBSD 10.1-p1로 업그레이드할 때, **zroot/ROOT/10.1-p1**과 같은 새로운 데이터 세트를 생성하고 이를 루트 파티션으로 사용하도록 FreeBSD에 지시할 수 있습니다. **zroot/ROOT/default**를 마운트하지 않거나 **/oldroot**와 같은 다른 위치에 마운트합니다. 업그레이드가 제대로 진행되지 않으면 되돌리기가 쉽지 않습니다.

다음 데이터 세트인 **zroot/tmp**는 거의 비어 있습니다. 이 데이터셋은 **/tmp**에 마운트됩니다. 이 데이터 세트는 전통적인 임시 디렉터리입니다.

ZFS 파티션 및 속성 (ZFS partitions and properties)

ZFS에는 전통적인 파티션이 없습니다. 파티션은 디스크의 논리적 세분화로, 스토리지 장치에서 매우 특정한 LBA(논리 블록 주소)를 채웁니다. 파티션은 파티션에 있는 데이터를 인식하지 못합니다. 파티션을 변경한다는 것은 그 위에 있는 파일 시스템을 파괴하고 (아마도) 다시 구축하는 것을 의미합니다.

파티션이 없는 파일시스템을 처음 본 루카스는 스토리지를 어떻게 관리할지 궁금해졌습니다. 이는 추운 미시간의 긴 겨울이 지나고 몇 달 만에 처음으로 바깥에 나와 따뜻한 자연 공기를 느꼈을 때 느끼는 혼란과 비슷합니다. 혼란은 해방의 일부입니다. 파티션을 통해 스토리지를 관리하는 방법을 배웠던 것은 파티션이 좋거나 최고의 솔루션이기 때문이 아니라 해야만 했기 때문입니다. 파티션 없이 기존 파일 시스템을 실행하는 것은 좋지 않은 습관이지만, ZFS는 기존 파일 시스템이 아닙니다.

ZFS는 파일시스템과 하위 스토리지 계층을 긴밀하게 통합합니다. 즉, 필요에 따라 다양한 파일시스템 간에 스토리지 공간을 동적으로 분할할 수 있습니다. ZFS 파일시스템에 특정 크기 제한을 설정할 수 있지만, 데이터

세트에는 전통적인 크기가 없습니다. 풀에 파일을 위한 충분한 공간이 있는 경우 이를 사용할 수 있습니다. 이전에 **/var/log**에 제한된 양의 디스크 공간을 할당하여 광란의 로그가 디스크를 가득 채우는 것을 막았다면, 이제 ZFS 수준에서 이러한 제한을 설정해야 합니다.

데이터 세트가 사용할 수 있는 공간의 양은 ZFS 속성의 한 예입니다. 예를 들어, **quota** 속성은 데이터 세트가 얼마나 커질 수 있는지를 제어하는 등 ZFS는 수십 개의 데이터 세트 속성을 지원합니다. ZFS 속성을 설정하려면 **zfs(8)**을 사용합니다.

```
$ zfs set quota=2G zroot/var/log
```

zfs get 명령으로 속성을 조회합니다.

```
$ zfs get quota zroot/var/log
NAME          PROPERTY  VALUE  SOURCE
zroot/var/log  quota      2G  local
```

zfs get all로 ZFS 데이터셋 이름과 데이터셋의 모든 속성을 볼 수 있습니다. 4장에서는 ZFS 속성을 자세히 살펴보고, 6장에서는 데이터 집합 크기 제한에 대해 설명합니다.

ZFS 제한 (ZFS Limits)

파일시스템에는 항상 최대 크기와 제한이 있었습니다. 우리가 모두 알고 있는 FAT 파일시스템은 최대 크기인 32MB를 극복하기 위해 여러 차례의 수정이 필요했고, 그 후 2GB, 4GB로 늘어났습니다. 요즘은 FAT32의 2TB 제한이 조금 비좁아 보이기 시작했습니다. UFS와 ext2/3/4fs도 비슷하게 임의적인 제한을 두고 있습니다. 이러한 제한은 파일시스템 개발자가 어딘가에 한도를 설정해야 했고, 향후 몇 년 동안은 괜찮을 것으로 예상되는 값을 선택했기 때문에 존재합니다. 그러나 인기 있는 파일시스템은 이러한 한계에 도달할 때까지 계속 사용되기 때문에 시스템 관리자는 이러한 한계에 반복적으로 대처해야 했습니다.

ZFS 옹호자들은 ZFS가 이러한 임의의 한계에 영향을 받지 않는다고 주장하지만, 이는 사실이 아닙니다. ZFS는 대부분의 값을 저장하는 데 128비트를 사용하기 때문에 오늘날 시스템 관리 업무를 하는 사람이라면 한계를 넘을 수 없을 정도로 높은 한계를 설정합니다. 하나의 디렉터리에는 각각 최대 16엑사바이트의 2^{48}개 파일을 저장할 수 있습니다. 단일 풀은 최대 256제타바이트, 즉 2^{78}바이트까지 가능합니다. 스토리지 풀에는 최대 2^{64}개의 장치가 포함될 수 있으며, 단일 호스트에는 최대 2^{64}개의 스토리지 풀이 있을 수 있습니다.

좋은 소식은 우리가 이러한 한계에 도달할 만큼 오래 살지는 않을 것이라는 점입니다. 나쁜 소식은 파일 시스템 간 마이그레이션에 대한 모든 전문 지식을 보유하고 있다는 것입니다. 기술이 ZFS의 한계에 부딪히면 파일시스템 간 마이그레이션에 익숙하지 않은 사람들은 어려움을 겪을 것입니다. 다행히도 연습을 위해 FAT/UFS/extfs 롤오버가 몇 가지 남아있을 것입니다.

스토리지 풀

ZFS는 디스크가 아닌 스토리지 풀을 사용합니다. 스토리지 풀은 기본 스토리지 공급자의 상단에 있는 추상화로, 물리적 매체와 그 위에 있는 사용자가 볼 수 있는 파일 시스템을 분리할 수 있게 해줍니다. **zpool(8)**을 사용해 시스템의 스토리지 풀을 보고 관리할 수 있습니다. 다음은 기본 FreeBSD 시스템의 풀입니다.

```
$ zpool status
  pool: zroot
state: ONLINE
  scan: none requested
config:

NAME       STATE   READ WRITE CKSUM
zroot      ONLINE     0     0     0
 gpt/zfs0 ONLINE      0     0     0

errors: No known data errors
```

먼저 풀의 이름과 상태가 표시됩니다. 시스템에는 둘 이상의 ZFS 풀이 있을 수 있으며, 수십 개의 하드 드라이브가 있는 대규모 시스템에는 여러 개의 풀이 있는 경우가 많습니다. 이 호스트에 여러 개의 스토리지 풀이 있는 경우 위의 샘플과 같이 각 풀이 별도의 설명으로 표시됩니다.

ZFS는 스토리지 풀에 대해 다양한 종류의 무결성 검사를 수행할 수 있습니다. **scan** 구문은 무결성 검사가 수행되고 있는지 여부와 가장 최근 검사 결과를 보여줍니다.

풀 목록의 마지막 부분에는 풀에 있는 가상 장치의 레이아웃이 표시됩니다.

가상 장치 (Virtual Devices)

스토리지 풀에는 하나 이상의 가상 장치, 즉 VDEV가 포함되어 있습니다. VDEV는 기존 RAID 장치와 유사합니다. 시스템 관리자가 실제로는 여러 개의 작은 디스크라는 것을 알고 있지만 파일 시스템 계층에는 하나의 큰 장치로 표시됩니다. 가상 장치를 사용하면 특정 장치를 특정 역할에 할당할 수 있습니다. VDEV를 사용하면 필요에 따라 물리적 스토리지를 배치할 수 있습니다.

가상 디바이스는 ZFS의 모든 마법이 일어나는 곳입니다. RAID 방식의 중복성을 위해 풀을 배열할 수 있습니다. 공급자를 전용 읽기 및 쓰기 캐시로 사용하여 가상 장치의 성능을 향상시킬 수 있습니다. 2장에서는 가상 장치에 대해 더 자세히 다룹니다.

ZFS의 데이터 중복성 및 자동 오류 수정도 VDEV 수준에서 이루어집니다. ZFS의 모든 데이터는 무결성 검증을 위해 체크섬 처리됩니다. 풀에 충분한 중복성이 있는 경우, ZFS는 자가 복구 기능을 제공합니다. 풀에 중복성이 부족하다면 적어도 데이터가 손상되었다는 것을 알 수 있고 백업에서 복원할 수 있습니다.

> ZFS는 백업의 필요성을 없애지 않습니다. 백업을 없애는 유일한 방법은 절대적인 무관심뿐입니다.

풀의 상태를 표시하는 **zpool status** 명령은 해당 풀의 가상 장치도 표시합니다. 이전 섹션의 예시를 보세요. 이 매우 간단한 풀인 zroot에는 단일 스토리지 공급자인 **/dev/gpt/zfs0**이 포함되어 있습니다. 이 공급자는 디스크가 아닌 GPT 파티션입니다. 2장에서 설명한 대로 ZFS는 모든 종류의 기본 스토리지를 사용할 수 있습니다. GPT 파티션을 사용하는 것이 매우 일반적이지만, 다른 옵션으로는 전체 디스크, 파일 및 기타 GEOM 공급자가 있습니다. FreeBSD는 암호화와 같은 기능을 지원하기 위해 GEOM 공급자를 사용합니다.

블록과 이노드 (Blocks and Inodes)

전통적인 파일시스템은 거의 항상 데이터를 저장하기 위해 다양한 데이터 블록을 사용하고 이러한 블록의 내용을 인덱스 노드로 매핑합니다. BSD의 UFS와 Linux의 extfs는 이러한 블록과 이노드를 호출합니다. Microsoft의 FAT 파일시스템에도 데이터 저장 블록과 인덱스 노드가 있습니다.

이러한 파일시스템과 마찬가지로 ZFS는 인덱스 블록과 데이터 블록을 사용합니다. 그러나 이전 파일시스템과 달리 ZFS는 필요에 따라 인덱스 노드를 생성합니다. ZFS는 가능할 때마다 데이터에 맞는 크기로 스토리지 블록을 생성합니다. 가변 크기 블록이 생성 가능한 모든 파일에 항상 맞는 것은 아니지만, 기존 파일시스템보다 확실히 더 유연합니다.

UFS 슈퍼블록과 달리 동적으로 생성된 인덱스 블록은 디스크의 알려진 위치에 배치할 수 없습니다. ZFS는 인덱스 블록이 손상될 가능성에 어떻게 대처할 수 있나요? ZFS는 중요한 인덱스 블록의 복사본을 알고리즘적으로 예측 가능한 위치에 여러 개 저장합니다. 이러한 블록은 디스크의 여러 위치에 복제됩니다. 3장에서는 ZFS 블록, 우버블록, 동일 블록, 트랜잭션 그룹 등에 대해 설명합니다.

이제 ZFS의 기본을 알았으니, 이 책의 나머지 부분에서는 수백 가지의 세부 사항을 설명합니다. 스택의 가장 아래쪽인 가상 장치부터 시작하겠습니다.

Revision #2
Created 21 January 2024 03:08:23 by 고기만두 (MeatDumpling)
Updated 21 January 2024 03:37:05 by 고기만두 (MeatDumpling)