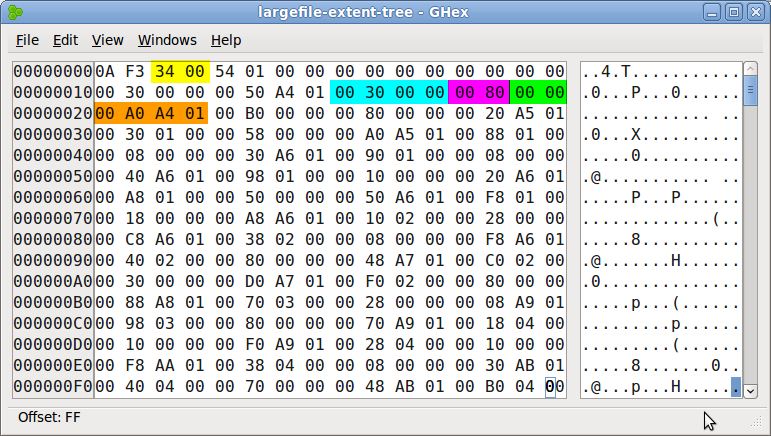
파일 사이즈가 큰 Extent tree를 보면서 설명하지 못하는 부분에 대한 이메일을 받았다.

결과를 재현하기 위해서 4gb에 달하는 사이즈가 큰 파일을 만들었다.

파트 1에 논의한 대로라면 extent를 저장하기 위한 16bit의 필드가 존재한다.

그러나, 그 해당 필드의 상위비트는 사전 할당된 범위를 표시하기 위해 예약되어 있어서, 한 extent에서 32k의 block 만 가질 수 있다. 한 block 크기를 4K라고 가정할 때, 우리는 한 extent에서 128MB(32K \*4K) 의 데이터만 가질 수 있다는 뜻이다. 4GB 파일은 적어도 32개의 extent가 필요하고, 심지어 32K의 연속된 Bolck의 32개를 사용하는 것을 찾을 수 있다는 것을 가정한다.Extents 수가 32개 보다 많을 가능성이 있고, 그중 일부는 128MB를 전부다 사용하고 있지는 않을 것이다.

4GB 파일을 만든 후, 파트 3에 설명 된 기술을 사용하여 파일의 extent tree 구조를 디코딩하고 파일의 실제 extent 를 보유하고있는 데이터 block을 찾았습니다.



실0 extent header에서 extent 수 필드( 노란색)를 보면 실0 52(0x0034)extent를 사용하고 있음을 알 수 있습니다. 그러나 가장 흥미로운 것은 2번째 하이라이트 된 extent 구조 입니다. 이구조를 decode하면, extent가 논리적으로 0x00003000으로 시작하고 (파일시작에서 12288block)과 물리적 block 0x000001A4A000(block 번호 27577080)시작한다.

그러나 가장 놀라웠던 것은 다름이 아닌 extent크기ㅏ 0x8000이라는 점입니다. 이진수로, 그것은 high bit set 와 낮은 15 비트 모두 0 인 16 비트 값입니다. 왜냐하면, EXT4에서 사전 할당된 범위를 표시하기 위해 high bit 사용되므로 0 byte는사전 할당된 범위를 의미합니다. 이건 말이 전혀 되지 않는데 무슨 일이 일어나고 있는 걸까??

Amit은 Andreas의 패치에서 먼저 오류를 병합하여 초기화 된 extent가 전체 32768 block이되도록합니다. 초기화되지 않은 범위는 32767 block으로 제한됩니다. Amit은 이에 대한 의견을 추가하고 내일 업데이트 패치를 준비 할 것입니다.

두 번째 링크는 위의 참고에서 언급 한 코드 / 코멘트, 특히 다음과 같습니다

여기서 무슨 일이 일어나고 있는지 이해하려면 아직 다루지 않은 EXT 파일 시스템의 측면을 검토하는 것이 도움이 됩니다. EXT와 같은 전통적인 Unix 파일 시스템에서 block과 inode는 파일 시스템에서 순차적 인 block 그룹으로 배열됩니다. block 그룹의 block 수는 일반적으로 파일 시스템에서 block 크기의 8 배입니다. 왜냐하면 각 block 그룹의 시작 부분에는 block 그룹의 각 block이 할당되었는지 여부를 추적하는 block 비트 맵이라는 파일 시스템 메타 데이터 구조를 보유하는 단일 block이 있기 때문입니다. block 비트 맵의 ​​각 비트가 block 그룹의 block을 추적하는 경우 단일 block 비트 맵 block을 사용하여 추적 할 수 있는 대부분의 block은 4K block 크기의 일반 파일 시스템에서 8x (block 크기) 또는 32K block입니다.

block 비트 맵과 inode 용으로 예약 된 block 및 inode 비트 맵 (EXT는 일반적으로 block 그룹 (btw)의 4 개 block마다 하나의 inode를 할당 함)과 함께 super block 및 기타 파일 시스템 메타 데이터의 사본이 일반적으로 데이터가 저장되기 직전에 저장됩니다. block 그룹의 block. 이 파일 시스템 메타 데이터는 32K 개 이상의 연속 데이터 block을 EXT 파일 시스템에서 절대로 찾을 수 없음을 의미하며, 해당 block 그룹이 현재 사용되지 않는 경우에만 해당됩니다.

이제 EXT4의 extent 크기 필드와 관련하여 생각해 봅시다. 이 필드는 16 비트 값이지만 높은 비트는 예약되어 있습니다. 즉, extent에는 최대 2 ^ 16-1 block 또는 32767 block (단일 block 그룹의 block 수보다 정확히 한 block 미만) 만 포함될 수 있습니다. 이것은 낭비입니다.

이제 extent크기 의 상위 비트가 예약 된 이유는 무엇입니까? 따라서 파일 시스템은 특정 extent를 "초기화되지 않았지만 예약 된"것으로 표시 할 수 있습니다. 이 "사전 할당"전략을 통해 EXT4는 향후 파일에 해당 block이 필요할 것으로 생각되는 경우 다른 파일이 특정 block을 사용하지 못하도록하여 증가하는 파일을 조각화하지 않아도됩니다. 소년의 바지에 옷자락을 두는 것처럼 아이가 자라면서 나중에 실망 할 수 있습니다.

그러나 extent가 전체 block 그룹을 채울 수 있도록 EXT4 개발자는 더러운 트릭을 수행했습니다. extent 크기가 0x8000이면 일반적으로 "0 block이없는 초기화되지 않은 extent"를 의미합니다. 그러나 왜 0 block을 미리 할당합니까? 아무 소용이 없다. 따라서 EXT4 개발자는 0x8000 값이 block 그룹의 전체 32K block의 할당 된 범위를 의미하는 특별한 경우를 추가했습니다.

High bit set를 가진 다른 모든 값은 범위가 다른 15 비트에 의해 길이가 결정되는 사전 할당되었지만 초기화되지 않은 범위를 의미합니다. 그러나 다시 block 그룹의 최대 block 수보다 2 ^ 16-1 block 또는 32767 block까지만 사전 할당 할 수있게되었습니다. 주의 깊게 읽으면 위에서 언급 한 주석과 코드가 우리에게 말하려고하는 것입니다.

짧은 대답은 extent 크기 값 0x8000은 32K 블록 길이의 할당 된 extent를 의미한다는 것입니다. 높은 비트가 설정되지 않으므로 더 작은 값도 할당 된 extent가됩니다. 0x8000보다 큰 값은 길이가 값의 하위 15 비트로 지정된 사전 할당 된 extent입니다.