N-Tree 검색에 기반한 허프만 디코더의 최적 구현에 관한 연구

정종훈
삼성전자 디지털미디어 연구소
e-mail : jonghoon.jeong@samsung.com

An Efficient Huffman decoding method based on the N-Tree searching algorithm

Jong-Hoon Jeong
Samsung Electronics Digital Media R&D center

Abstract

This paper presents an efficient huffman decoding method based on the multiple branch technique. in the proposed search method, the internal node which does not contain a leaf node are removed for decrease the searching time and the memory consumption. The proposed search method gives 44% of improved in searching time and 34% of decreased in memory requirement compared to the binary search method.

I. 서론

멀티미디어 환경의 급속한 발전은 정보의 형태를 기존의 문자 및 음성위주의 형태에서 다채로운 화상 및 고품질 오디오 신호의 구성으로 변화시켜가고 있다. 이러한 기반에는 통신기술의 발달로 인한 전송속도의 증가뿐만 아니라, 기하급수적으로 증가하는 각종 멀티미디어 자료들을 효율적으로 전송 및 저장할 수 있도록 하는 각종 신호처리 기술들이 밑받침 되고 있다. 특히 최근의 휴대정보단말(PDA)과 같은 모바일 환경의 급속한 발달은 신호처리 과정에 있어 한정된 자원을 최대한 이용할 수 있도록 하는 하드웨어 및 소프트웨어의 구성을 요구하게 되었다.

이러한 신호처리 기술중 데이터 압축분야는 정보량을 유지하면서, 이에 소요되는 저장 공간을 감소시키는 기술이다. 이중 허프만 코딩 기법은 발생되는 데이터들의 확률분포에 따라 발생확률이 높은 신호에 짧은 길이의 부호를, 발생확률이 낮은 신호에 긴 부호를 할당하는 가변장 부호화 방식을 적용함으로써, 각 신호의 저장에 소요되는 평균적인 저장 공간을 감소시킬 수 있는 압축방식중의 하나이다. 허프만 부호화/복호화 기법은 압축 전/후의 데이터에 변화가 없는 비손실 압축 방법이다. 따라서 이를 수행함에 있어 부호화/복호화 과정에서 사용하는 저장 공간의 최소화와, 연산에 소요되는 시간의 단축이 주요 사항이라고 할 수 있다.[1][2]

이에 본 논문에서는 허프만 디코더를 구현함에 있어 기존의 이진트리 검색방식의 단점을 개선한 다중 검색 트리 방식을 제시하고, 이를 고품질 오디오 압축방식중의 하나인 Advanced Audio Coding (AAC)에 적용하고 관측을 수행하였다. 수행결과 기존의 이진트리 기반의 허프만 디코더에 비하여 약 44%의 평균 검색 횟수 단축 및, 약 34%의 저장 공간의 감소를 가져올 수 있었다. [3]

II. 관련 연구

본 절에서는 효율적인 허프만 디코더의 구현을 위하여, 기존의 이진트리 검색방식에 기반한 허프만 디코더의 구현원리 및 이에 대한 문제점을 기술하고, 본 논문에서 제시하고자 하는 N-tree 검색기법 디코더의 구현원리 및 개선사항에 대하여 설명하고자 한다.

2.1 허프만 부호화/복호화

허프만 부호화 방식은 데이터들의 발생확률에 기반하여 발생빈도가 높은 신호에 짧은 길이의 부호를, 발생빈도가 낮은 신호에 비교적 긴 길이의 부호를 할당함으로써 정보의 손실없이 저장 공간을 감소시킬 수 있는 압축 기법중의 하나이다. 부호화/복호화의 수행은

데이터들의 발생빈도에 기반하여 작성된 허프만 table을 이용하여 데이터의 압축 및 복원을 수행하게 된다. 표 1에서는 이러한 허프만 table의 예를 보여주고 있다.

표 1. 허프만 테이블의 예

index	codeword	length	data
0	0	1	A
1	100	3	B
2	101	3	C
3	110	3	D
4	1110	4	E
5	1111	4	F

표 1의 "data"항은 부호화 이전 및 복호화 이후의 데이터를 나타낸다. "codeword"항은 데이터의 압축후 생성된 부호 비트열로서 "data"항의 발생확률에 따라 서로 다른 길이의 부호를 가지게 된다. "length" 항은 "codeword"가 몇 비트의 길이를 가지는지를 나타내는 항목이다.

허프만 복호화 과정은 그림 1에서 나타내는바와 같이 부호화된 비트열을 읽어 들이면서, 디코딩 테이블 상의 "codeword"항을 검색하면서 일치되는 비트열을 발견시 "data" 항목의 데이터로 복원하는 과정을 수행하게 된다.

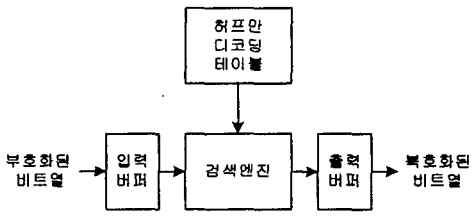

그림 1. 허프만 디코딩 과정

2.2 이진트리 검색방식에 기반한 허프만 디코더

허프만 디코딩의 수행은 매우 다양한 검색기법들이 존재하지만, 그중 이진트리 검색방식은 검색시간 면에서 매우 효율적인 방법으로 알려져 있다. 검색과정은 그림 2의 순서도 에서 나타내는바와 같이 읽어 들인 비트열의 값이 "0"인지 "1"인지에 따라 이동할 하위 노드의 위치를 결정하는 방식이다. [4]

이러한 이진트리 검색방식을 이용하는 예로서, 표 1에서 언급한 허프만 테이블을 검색하는 과정을 도식하면 그림 3과 같다.

이진트리 검색의 경우 각각의 노드에서 하위노드로의 분기는 항상 "0"과 "1"의 두 가지 경우만 발생하게 된다. 따라서 그림 3의 n1의 경우 "length"가 2인 "codeword"가 존재하지 않더라도, 다음 노드들과의 연결을 위하여 중간노드가 존재하여야 하므로, 불필요한 검색단계가 존재하게 되며, 또한 노드간의 연결 관계를 나타내기 위하여 추가적인 메모리를 소요하게 된다. 따라서 본 논문에서는 이러한 불필요한 중간노드를 제거하기 위하여 하나의 노드에서 N개 (N = 2^n)의 하위 노드를 가지는 다중분기 기법을 적용하여 검색단계를 감소시키고, 허프만 검색 테이블의 저장 공간을 절약할 수 있는 검색기법을 제시하고자 한다.

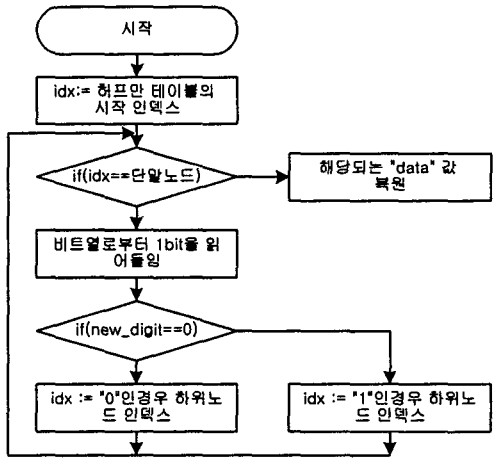

그림2. 이진트리 허프만 디코더의 검색과정

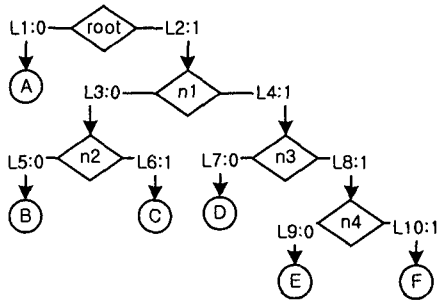

그림 3. 이진트리 기반 허프만 디코딩 트리 예

2.3 N-Tree 허프만 디코더

본 논문에서 제시하는 N-tree 허프만 디코딩 기법은 검색과정에서 불필요한 노드를 제거하는데 그 목적이 있다. 이진트리 허프만 디코더의 경우 하나의 노드에서 분기를 위하여 필요로 하는 비트의 수가 항상 1인데 반하여, N-tree 허프만 디코더에서는 한번에 1bit이상의 값을 이용하여 한번에 N개의 분기 (N=2^n, n:읽어 들인 비트수)를 수행함으로써 불필요 노드를 제거할 수 있도록 하였다. 그림 4에서는 표 1의 허프만 디

코딩 테이블을 이용하여 N-tree 허프만 디코딩 트리를 생성한 예를 보여주고 있다.

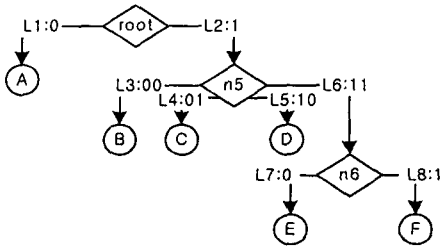

그림 4. N-tree 기반 허프만 디코딩 트리 예

그림 3에서 "data" B, C, D를 검색하기 위해서는 반드시 노드 n1을 경유하여 n2또는 n3으로 이동하는 단계가 필요로 하지만, 그림 4의 경우 노드 n5에서 다중 분기를 이용함으로써 검색단계를 단축시킬 수 있도록 하였다.

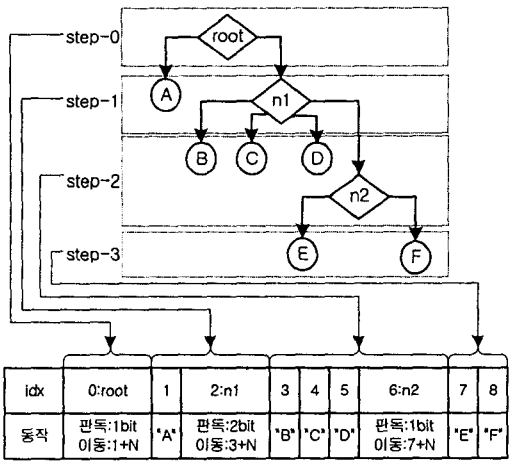

그림 5. 검색 테이블의 생성과정

하위 노드로 이동하기 위한 검색 테이블상의 인덱스의 결정과정에서는 일반적인 트리 검색에서 존재하는 비교/분기 구문을 제거하고 수치연산 기법을 도입함으로서 분기구문에서 발생하는 pipeline-stall 문제를 감소시킴으로써 검색 속도의 향상을 가져올 수 있도록 하였다. 즉 읽어 들인 n-bit의 값이 생성하는 N값($0 \sim 2^n - 1$)과 하위 노드중의 하나의 노드위치와의 덧셈 연산을 통하여 신속한 이동 위치를 판단할 수 있게 된다. 그림 5에서는 이러한 검색 테이블의 생성과정을 보여준다. [5]

III. 실험 결과

본 절에서는 기존의 이진트리 검색방식에 기반한 허프만 디코더와 비교하여, 본 논문에서 제시하는 N-Tree 검색기법 허프만 디코더의 검색속도 및 소요 메모리에 대하여 측정하였다.

3.1 실험환경

본 논문에서 제시하는 N-Tree 검색방식에 기반한 허프만 디코더의 성능을 측정하기 위하여, 고품질 오디오 코덱중의 하나인 MPEG-2/4 AAC 디코더의 허프만 디코더 부분을 사용하여 평균 검색횟수 및 메모리 사용량을 측정하였다. 비교 대상으로는 이진트리 검색방식에 기반한 허프만 디코더를 선정하였다. 테스트를 수행하기 위한 샘플 파일로는 MPEG-2/4 AAC의 성능 측정기준으로 사용되는 conformance test bitstream중 16개의 파일을 선정하여 측정을 실시하였다. 디코더의 작성은 C언어를 이용하여 작성하였으며, Microsoft Visual C++ 6.0 컴파일러의 Release build 옵션을 선택하여 Pentium-4 PC상에서 시뮬레이션을 수행하였다. [6][7]

3.2 평균 검색횟수 측정

본 실험에서는 허프만 부호화된 비트열로부터 원래 신호를 복원하기까지, 몇 번의 비교 구문을 수행하는가에 대한 평균적인 검색 횟수를 측정하였다.

16개의 MPEG-2/4 AAC 테스트 파일을 이용하여 허프만 디코딩에 필요로 하는 검색횟수를 측정한 결과, 이진트리 기반 허프만 디코더의 경우 평균 5.21회의 검색을 반복한 뒤 결과 값을 출력하는데 비하여, 본 논문에서 제시하는 다중 트리구조 허프만 디코더를 적용할시 평균 2.90회의 검색으로 검색을 완료함으로써 검색횟수 측면에서 약 44.2%의 성능향상을 확인할 수 있었다. 표 2에서는 평균 검색횟수에 대한 측정결과를 나타내고 있다.

3.3 소요 메모리 측정

임베디드 환경과 같은 메모리 사용량이 제한되는 환경의 경우 소요 메모리의 최소화는 매우 중요한 문제라고 할 수 있다. 따라서 본 측정에서는 MPEG-2/4 AAC의 허프만 테이블을 N-tree검색방식으로 구현할 경우 소요되는 메모리를 측정하였다. 기존의 이진트리 검색방식의 경우 2712word (1word=16bit)의 메모리를 소요하는 반면에 N-tree의 경우 1824word를 소요함으로써 약 33.92%의 메모리 사용량을 감소되었다.

표 2. 허프만 디코더의 평균 검색횟수

테스트파일	평균 검색횟수		
	이진트리	N트리	성능개선비율
L1_32000.aac	5.35	2.92	45.4%
L1_44100.aac	5.17	2.97	42.5%
L2_32000.aac	5.36	2.96	44.8%
L2_44100.aac	5.20	3.03	41.8%
L3_32000.aac	5.51	2.90	47.5%
L3_44100.aac	5.36	2.96	44.7%
L4_32000.aac	5.33	2.92	45.1%
L4_44100.aac	5.14	2.97	42.3%
L5_32000.aac	5.54	2.88	47.9%
L5_44100.aac	5.36	2.92	45.5%
L10_32000.aac	5.34	2.92	45.2%
L10_44100.aac	5.15	2.97	42.3%
L11_32000.aac	5.34	2.92	45.2%
L11_44100.aac	5.15	2.97	42.3%
SIN2_32000.aac	4.29	2.59	35.6%
SIN2_44100.aac	4.74	2.66	43.9%
평균	5.21	2.90	44.2%

표 3. 허프만 검색테이블의 소요 메모리

테이블	테이블	저장공간 사용량 (단위:word)		
번호	크기	이진트리	N트리	감소량
0	121	241	163	78
1	81	161	99	62
2	81	161	113	48
3	81	161	119	42
4	81	161	107	54
5	81	161	109	52
6	81	161	109	52
7	64	127	87	40
8	64	127	93	34
9	169	337	229	108
10	169	337	229	108
11	289	577	367	210
합계	1362	2712	1824	888

V. 결론 및 향후 연구 방향

본 논문에서는 데이터 압축의 한 기법으로서 널리 사용되는 허프만 디코딩을 효율적으로 수행할 수 있는 N-tree 검색기법에 의한 허프만 디코딩 알고리즘을 제시하였다. 이 방법은 기존의 이진트리에 의한 허프만 디코더에서 존재하는 불필요 검색단계를 제거함으로써, 검색속도의 향상 및 검색 테이블을 저장하기 위한 공간의 절약을 가져올 수 있었다.

참고문헌

[1] A.Gersho, "Advanced in speech and audio compression", proc IEEE, vol 82, 99. 901~918, June 1994 Digital speech

[2] Nick B.Body and donald G.Bailey, "Efficient representation and decoding of static huffman code tables in a very low bit rate environment", IEEE, 1998

[3] Maja Bystrom and Susanna Kaiser, "Soft decoding of variable-length codes", IEEE, 2000

[4] Jae Ho Jeong, Yougn Seo park, and Hyun Wook Park, "a fase variable-length decoder using separation", IEEE transactions on circuit and systems for video technoloty, vol. 10, No. 6, August 2000

[5] J.H.Jeong and T.G.Chang, An Efficient search of binary tree for huffman decoding based on numeric interpretation of codewords", ISCAPS, October 2001

[6] ISO/IEC JTC1/SC29/WG11 N1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding, AAC)"

[7] http://www.mpeg.org