JPEG 시스템을 위한 효율적인 가상 메모리 관리 기법

채희중° • 이호석
호서대학교 컴퓨터 공학과

The efficient virtual memory management scheme for JPEG

Hui-Joong Chae° • Ho-Suk Lee
Department of Computer Engineering, Hoseo University

요 약

본 논문은 IJG JPEG 시스템에서 사용된 효율적인 메모리 관리 방법에 대해 소개한다. IJG JPEG 시스템의 메모리 관리 모듈은 여러 시스템에 독립적일 수 있도록 다양한 종류의 메모리 관리 모듈을 두고 있다. 각 메모리 관리 모듈은 메모리 객체를 small, large, virtual 의 3 가지로 구분하여 할당/해제하는데, 객체 단위로 할당/해제하지 않고 "pool"의 개념을 사용하여 한꺼번에 할당/해제한다. 메모리 관리 모듈은 부호화와 복호화 과정 모두에서 사용되며, 복잡한 포인터형 자료 구조를 사용하여 실제적인 데이터로 접근하는 방법을 사용하고 있다. 메모리 관리 모듈은 이미지를 처리하는 과정에서 매우 효율적인 성능을 나타내도록 설계되었다. 또한 가상 메모리를 지원하지 않는 운영체제에서도 가상 메모리를 관리할 수 있도록 가상 배열 관리 모듈이 제공된다.

1. 서론

IJG JPEG 시스템의 메모리 관리자는 이미지 처리 중에 발생하는 모든 메모리의 할당과 해제를 관리한다.[1][4] 이러한 메모리 관리 모듈들은 부호화 과정과 복호화 과정 양쪽에 동일하게 제공된다.[2] 할당될 객체들은 아직 사용되지 않은 메모리의 "pool"내에 한꺼번에 할당되어지고 사용된 후에는 한꺼번에 해제된다. 이런 방법으로 할당과 해제를 하는 이유는 메모리의 사용 효율을 증가시키고 malloc()/free()할때 보다 빠르게 할당/해제 할 수 있기 때문이다.

메모리 관리자가 사용하는 주요 자료 구조로서 가장 기본적인 단위는 JSAMPLE 과 JCOEF 가 있으며, 계층적 구조로서 상위의 자료구조가 이를 지시하고 있다.[2] JSAMPLE 은 제공되는 샘플 값이 8bits 일 경우에는 0~255 의 픽셀 값을 가지며, 데이터 타입은 "unsigned char"나 "char"로 정의된다. 그러나 샘플 값이 12bits 일 경우에는 0~4095 의 픽셀 값을 표현할 수 있으며, 데이터 타입은 "short"로 정의된다.[3]

입력된 이미지를 처리하기 위한 라이브러리의 최대 메모리 요구량은 이미지의 높이에 따라 결정되는 것이 아니라 폭에 따라 달라진다. 왜냐하면, 라이브러리는 일반적으로 이미지의 폭 만큼의 길이를 갖는 스트립 버퍼를 가지고 처리를 하기 때문이다. 이렇게 만들어진 스트립 버퍼 열들의 높이는 높지 않다.

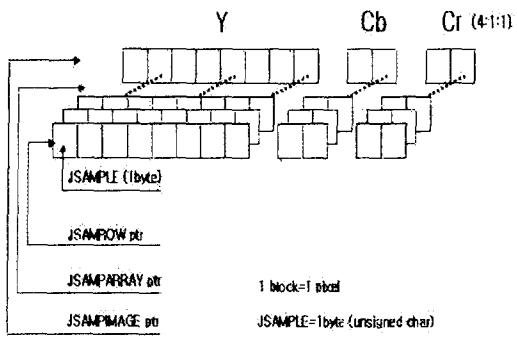
IJG JPEG 시스템에서 지원하는 특정 처리 모드 즉, 2-pass 양자화 같은 처리 모드는 스트립 버퍼가 아니라 이미지 전체를 담을 수 있는 대형의 버퍼를 필요로 한다[1]. 이와 같은 처리 모드일 경우에는 버퍼는 가상 배열로 처리한다. 메모리에는 현재 필요한 스트립만이 올려져 있고, 그 나머지는 임시 파일로 스왑된다. 가상 배열을 사용하는 메모리 관리자는 한 단계의 메모리 할당 루틴을 가지고 있지 않으며, 컨트롤 블럭을 할당하는 request() 루틴

과 공간 할당을 결정하고 단 한번의 호출로, 사용되는 모든 실제 버퍼를 할당하는 realize() 루틴으로 구성되어 있다.

2. 본론

2.1 자료 구조

IJG JPEG 시스템에서 사용하는 픽셀 샘플의 자료 구조는 JSAMPLE, JSAMPROW, JSAMPARRAY, JSAMPIMAGE 가 있다. [그림 1] 은 픽셀 샘플에 대한 자료 구조이다.

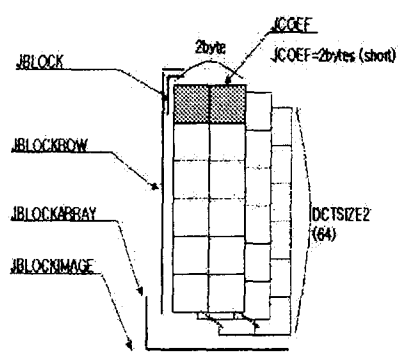


[그림 1] 픽셀 샘플의 자료 구조

JSAMPLE 은 각각의 픽셀 컴퍼넌트의 값이며, 샘플의 가장 기본이 되는 자료형이다. JSAMPROW 는 샘플들의 한 열에 대한 값을 가리키는 포인터이며, JSAMPARRY 는 각 열들의 리스트에 대한 포인터이다. 그리고 가장 큰 자료 구조인 JSAMPIMAGE 는 컬러 컴포넌트 배열에 대한 포인터이다.

자료구조로서 2 차원 배열을 사용하지 않고 각 열들마다 포인터를 사용함으로써 메모리를 적게 사용할 수 있는 효과를 얻을 수 있으며, 메모리를 할당할 때 각 열들의 폭을 알 필요가 없다. 그리고, 전체 할당 요소의 사이즈가 64Kb 이상의 큰 배열을 할당할 수 없는 malloc()을 사용하는 시스템에서도 그 보다 큰 배열을 할당할 수 있다. 그리고 열들의 형태를 지니고 있는 컴퍼넌트의 배열은 추가적인 복사 과정 없이 각기 다른 시간에 할당이 가능하다. 이것은 현재의 열로부터 이전의 열뿐만 아니라 그 다음의 열에도 접근이 가능해야 하는 smoothing 처리 과정에서 처리 속도 증대의 효과를 볼 수 있다. 하나의 픽셀은 N 개의 연속된 JSAMPLE 값들에 의해 표현된다. 그리고, 이미지에 대한 각 열들은 (컬러 컴포넌트의 갯수) x (이미지의 폭)의 길이를 갖는 배열이다. 이러한 방식으로 자료의 하나 또는 그 이상의 열들은 JSAMPARRAY 의 포인터로 표현될 수 있다.

IJG JPEG 시스템에서 사용하는 DCT 계수를 처리하기 위한 배열의 자료 구조로는 JCOEF, JBLOCK, JBLOCKROW, JBLOCKARRAY, JBLOCKIMAGE 가 있다. JCOEF 는 "short"로 선언되어 최소 16bit-signed integer 값을 갖는 하나의 DCT 계수를 저장하기 위한 배열의 기본적인 자료형이며, JBLOCK 는 계수들에 대한 8x8 블럭의 배열이다. JBLOCKROW 는 8x8 블럭에 대한 하나의 수평 열에 대한 포인터이며, JBLOCKARRAY 는 각 열들의 리스트에 대한 포인터이다. 마지막으로 가장 방대한 자료구조를 갖는 JBLOCKIMAGE 는 컬러 컴퍼넌트 배열의 리스트에 대한 포인터이다. [그림 2] 는 DCT 계수에 대한 배열의 자료 구조이다.[2]



[그림 2] DCT 계수의 자료 구조

JSAMPARRAY 와 JBLOCKARRY 자료구조는 실제 데이터에 대한 대형 객체와 각 열의 포인터에 대한 소형 객체의 조합이다. 편의성과 처리 속도를 위해 이러한 자료구조의 생성은 하나의 루틴을 통해 제공한다. 이와 유사하게 가상 메모리도 소형 컨트롤 블럭과 JSAMPARRAY 혹은 JBLOCKARRAY 작업 버퍼를 포함하고 있으며 이 모든 것은 단 한 번의 호출로 생성된다.

2.2 메모리 할당

메모리는 객체를 할당할 메모리의 크기에 따라 small, large, virtual 의 3 가지로 구분하여 할당한다.[1]

소형 객체로서 할당될 때는 일반적으로 할당되는 전체 메모리의 사이즈가 10Kb~20Kb 일 경우에 사용된다. 그리고 이러한 소형 객체는 언제 어디서든지 할당할 수 있으며, 이것에 의해 할당 되어진 메모리 사이즈는 realize() 루틴에서도 무시할 정도의 크기를 갖는다. 그리고 가상 배열에 대한 컨트롤 블록들은 소형 객체로 할당된다.

대형 객체로서 할당될 때는 할당되는 메모리의 사이즈가 수 10Kb~수 100Kb 일 경우에 사용된다. 의미적으로는 소형 객체와 동일한 동작을 한다. 소형 객체와 다른 점이라면 MS-DOS 시스템에서 대형 객체는 FAR 포인터에 의해 참조되지만, 소형 객체는 NEAR 포인터에 의해 참조된다. 그리고 대형 객체는 size_t 에 의해 허용된 사이즈를 초과하여 할당할 수 없다.[1][4]

가상 배열은 JSAMPLE 혹은 JBLOCK 의 방대한 2 차원 배열이다. 일반적으로 전체 입력 이미지를 한꺼번에 처리할 수 있을 만큼 큰 메모리이며, 메모리 관리자는 이러한 배열에 대해 스트립 단위의 접근을 제공한다. 가상 메모리를 제공하지 않는 시스템에서는 배열의 나머지 부분을 임시 파일로 스왑 시킨다.

2.3 가상 배열의 관리

입력 이미지에 대한 일반적인 처리로서 스트립 버퍼를 이용한 일반적인 배열의 처리 루틴이 사용된다. 반면에 2-pass 양자화와 같은 특별한 처리 모드로 처리할 경우에는 전체 이미지를 수용할 수 있는 크기의 대형 버퍼를 사용하는데 이러한 경우에는 가상 배열을 사용한다.[3][4] 가상 배열의 할당은 한 단계로 이루어지지 않으며, 컨트롤 블록을 할당하는 request() 루틴과 할당될 메모리 공간을 결정하고 모든 실제 버퍼를 생성하는 realize() 루틴을 통해 단계적으로 할당된다.

request_virt_array() 루틴은 한번에 접근할 수 있는 전체 이미지의 사이즈와 최대 열들의 개수를 파악한다. request() 루틴은 in-memory 버퍼를 생성하는 것이 아니라 컨트롤 블럭을 생성한다. 앞에서 request() 루틴을 통해 필요로 하는 메모리의 전체 할당 공간을 파악했으므로 realize_virt_array() 루틴에서는 메모리를 공평하게 할당할 수 있다.

access_virt_array() 루틴은 접근 가능한 특정한 스트립 영역을 생성하며, 만약 새로 접근하는 열들에 대해 pre-zeroing 요구가 있다면 그것을 수행할 수 있다.

3. 결론

본 논문에서는 IJG JPEG 시스템의 메모리 관리 방법에 대해 연구하였다. IJG 의 JPEG 소프트웨어는 다양한 시스템에 독립적일 수 있도록 설계되었기 때문에 여러 종류의 메모리 관리 모듈을 두어 메모리를 관리하고 있다. 이와 같은 메모리 관리 모듈들은 부호기와 복호기 모두에 동일하게 적용되고 있으며, 각 객체에 대해 small, large, virtual 등의 3 가지로 구분하여 메모리의 할당과 해제를 한다. 또한 자료 구조로서 복잡한 포인터구조를 사용하여 실제 이미지 데이터로의 접근을 허용하고 있다. IJG JPEG 은 가상 메모리를 지원하지 않는 운영체제에서도 가상 메모리를 사용할 수 있는 장치를 제공하고 있다.

이와 같은 IJG JPEG 시스템에서 사용된 효율적인 메모리 관리 방법은 향후 MPEG-2 시스템을 개발하는데 있어서 메모리 관리 방법으로의 적용이 기대된다.

4. 참고문헌

[1] Thomas G. Lane, 'IJG(Independent JPEG Group)'s JPEG Library: Version 6a', Independent JPEG Group, 1996.

[2] CCITT Recommendation T.81: PART 1 "Information Technology: Digital Compression and Coding Of Continuous-Tone Still Images: Requirements and Guidelines", ITU(International Telecommunication Union), 1993.

[3] William B. Pennebaker and Joan L. Mitchell, "JPEG Still Image Data Compression Standard", Van Nostrand Reinhold, 1993.

[4] 채희중, 이호석. "IJG JPEG 부호기의 구조와 동작", 한국정보과학회 99 가을학술발표논문집 제 26 권 2 호 p262-264, 한국정보과학회, 1999.10.