3D 렌더링 어플리케이션에 특화된 오프로딩 기술에 대한 연구

양승준, 권용인, 이하윤, 권동현, 백윤흥
서울대학교 전기정보공학부
e-mail : sjyang@sor.snu.ac.kr

A Study on Offloading Techniques for 3D Rendering Applications

Seungjun Yang, Yongin Kwon, Hayoon Yi, Donghyun Kwon, Yunheung Paek
Dept. of Electrical and Computer Engineering

요 약

스마트폰 및 태블릿에서 동작하는 모바일 어플리케이션은 날이 갈수록 복잡하고 다양해지고 있다. 특히, 3D 게임과 같이 렌더링 연산을 주로 사용하는 어플리케이션은 많은 연산량을 필요로 하며 소모하는 전력 또한 매우 크다. 이러한 문제를 해결하기 위해 연산의 일부를 클라우드와 같은 강력한 외부 자원을 활용하여 처리하는 오프로딩 기술이 제안되었으나, 특정 어플리케이션이 아닌 일반적인 어플리케이션을 대상으로 한 연구들이 대부분이다. 본 논문에서는 3D 렌더링 어플리케이션을 보다 더 효율적으로 구동시키기 위하여 오프로딩 기술이 가져야 할 특성 및 구조에 대해 설명한다.

1. 서론

최근 들어 스마트폰 및 태플릿을 포함한 모바일 기기가 점점 더 넓고 다양한 분야에서 사용되면서, 이들 기기에서 동작하는 어플리케이션 또한 더욱 복잡하고 다양해지고 있다. 특히 기존에는 데스트탑 또는 전용 기기 등을 통해 소비되던 3D 비디오 게임이나 증강 현실, 실시간 비디오 인코딩과 같은 많은 연산량과 전력을 필요로 하는 어플리케이션들을 모바일에서 실행하는 경우가 늘어나는 추세이다. 안타깝게도, 모바일 어플리케이션들의 복잡도가 나날이 높아지는 데 비해 모바일 기기의 연산 능력이나 배터리 용량은 그 태생적 한계로 인해 상대적으로 더디게 향상되고 있다.

연산 오프로딩(Computation offloading)은 동작 중인 어플리케이션의 연산 일부를 다른 기기로 옮겨(오프로딩, offloading) 실행하는 기술로써, 모바일 기기의 부족한 성능을 클라우드와 같은 강력한 외부 자원을 이용하여 보완해준다. 이는 기존의 서버-클라이언트 혹은 씬 클라이언트 모델이 사용자 단말과 서버 간의 역할 분담을 고정시킨 것과 달리, 오프로딩 기술은 모바일 기기의 실행 환경을 실시간으로 측정하여 최적의 성능을 얻을 수 있는 분담을 동적으로 찾아내기에 가능하다.

지금까지 많은 연구들이[1][2] 연산 오프로딩 기술이 모바일 환경의 성능을 효과적으로 향상시킬 수 있음을 보여 왔지만, 이들 연구의 대부분은 특정한 어플리케이션이 아닌 일반적인 어플리케이션을 대상으로 한 것이다. 때문에 3D 게임과 같이 일반적인 어플리케이션과 상이한 특성을 가진 어플리케이션에 그대로 적용하기에는 무리가 따른다. 일반적인 어플리케이션과 비해 이들 어플리케이션은 매우 높은 응답 속도를 요구하며, 핵심 연산인 렌더링 연산은 일반적인 함수와는 확연히 다른 특성을 보인다. 이로 인해 일반적인 어플리케이션만을 고려한 기존의 오프로딩 기술을 3D 렌더링 어플리케이션에 곧바로 적용할 경우 충분한 성능 향상을 기대하기 힘들다.

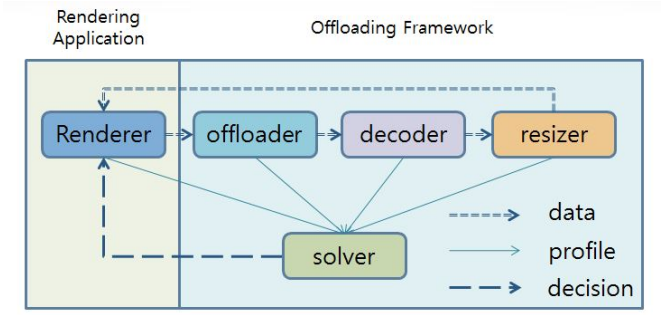
본 논문에서는 이러한 문제를 해결하여 3D 렌더링 어플리케이션을 보다 더 효율적으로 구동시키기 위한 오프로딩 기술의 특성 및 구조를 연구한다. 또한 이러한 특성을 더욱 발전시키기 위해 필요한 요소들에 대하여 고찰한다.

2. 오프로딩 프레임워크

다른 어플리케이션과 3D 렌더링 어플리케이션을 구분하는 가장 큰 특징은 바로 핵심 연산인 렌더링이다. 이들 어플리케이션은 렌더링 연산을 통해 주어진 입력 및 내부 데이터를 이용하여 사용자에게 보여줄 이미지를 만들어내며 대부분의 실행 시간을 이러한 이미지를 만들어 내는 데에 사용한다. 때문에 3D 렌더링 어플리케이션에 오프로딩 기술을 효과적으로 적용하기 위해서는, 가장 많은 실행 시간을 차지하는 렌더링 연산을 클라우드로 보내는 동시에 그 결과물인 많은 양의 이미지를 효과적으로 모바일로 되돌려주는 것이 필수적이다.

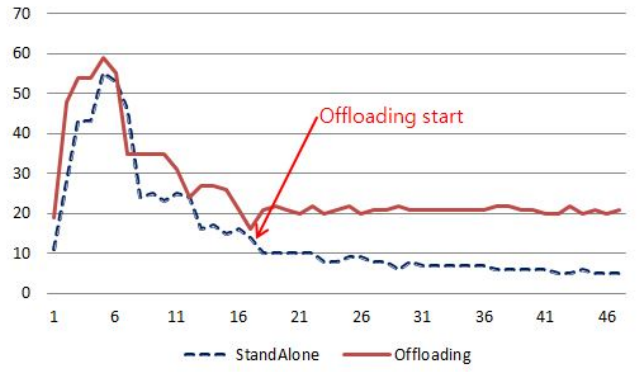
이러한 문제를 해결할 수 있는 방법 중 하나는 클라우드에서 생성된 이미지를 압축하여 모바일로 전송하고 모바일에서는 이를 해제하여 사용자에게 보여주는 것이다. 이러한 방법은 클라우드 게이밍 등의 기

존 연구에서 이미 사용되고 있는 것으로[3][4], 오프로딩 기술에 적용될 경우 클라우드 및 모바일 간의 데이터 전송량을 효과적으로 줄임으로써 오프로딩의 효율을 크게 향상시킬 수 있다. 다음 그림은 이러한 기법이 추가된 오프로딩 프레임워크의 구조를 간단하게 나타낸 것이다.



(그림 1) 오프로딩 프레임워크

통신이나 계산기(solver)와 같이 기존의 오프로딩 연구에서 필수적으로 포함되었던 기능 이외에도 이미지 압축해제를 위한 디코더와 해상도 조절을 위한 변환기(resizer) 등을 추가함으로써 렌더링 연산을 효과적으로 오프로딩 할 수 있게 된다. 다음 그래프는 이러한 오프로딩 구조를 3D 게임에[5] 적용하였을 때와 그렇지 않았을 때의 1 초당 프레임 수(Frame per second, FPS)를 비교한 것이다.



(그림 2) 시간에 따른 FPS 변화

시간에 따라 오브젝트의 수가 늘어나면서 FPS 값이 점점 떨어지는 단독 실행(Standalone)과 달리 오프로딩을 적용하였을 경우 17~18 초부터 오프로딩이 시작되면서 20 FPS 정도의 성능을 꾸준히 보여주는 것을 알 수 있다. 단독 실행의 경우 오브젝트의 수가 많아지면서 렌더링 연산에 걸리는 시간이 점점 길어져 성능이 떨어지게 되지만, 오프로딩을 적용하면 렌더링을 클라우드에서 처리함으로써 오브젝트 수가 많아지더라도 균일한 성능을 내는 것이 가능해진다. 또한 오프로딩 이후 20 FPS 의 성능을 얻을 수 있었던 것은 클라우드에서 생성된 이미지를 앞서 설명한 디코더 및 변환기 등을 이용하여 효율적으로 처리할 수 있었기 때문이다. 만약 이러한 기법을 적용하지 않고 클라우드에서 생성된 이미지를 그대로 모바일로 전송했다면 그 과도한 크기로 인해 많은 통신 오버헤드가 발생하여 원하는 만큼의 성능 향상을 이루기 힘들게 된다.

결과적으로 오프로딩을 통하여 렌더링 어플리케이션의 성능을 향상시키기 위해서는 클라우드에서 생성된 이미지를 효과적으로 모바일로 가져올 수 기법이 반드시 필요하며, 이러한 기법이 적용되었을 경우 렌더링 어플리케이션의 성능이 실제로 향상될 수 있음을 알 수 있다.

3. 향후 연구

앞으로의 연구에서는 보다 더 효율적으로 오프로딩 여부를 결정할 수 있는 계산기에 대한 연구와 모바일 및 클라우드에서의 렌더링 성능을 효과적으로 예측하여 계산기로 하여금 보다 더 정확한 결정을 내릴 수 있도록 하는 성능 예측 도구에 대한 연구를 진행하고자 한다. 또한 H.264 와 같이 단순한 이미지 압축해제가 아닌 동영상 코덱을 사용하여 스트리밍 방식을 적용하였을 때 성능에 어떤 차이가 있는지도 연구하고자 한다.

4. 결론

오프로딩을 통해 모바일에서 동작하는 3D 렌더링 어플리케이션의 성능을 효과적으로 향상시키기 위해서는 핵심 연산인 렌더링 연산의 특성을 이해하고 이에 특화된 오프로딩 기법을 적용하는 것이 필요하다. 본 논문에서는 기존의 오프로딩 기법에 이미지 압축해제 및 해상도 조절을 위한 모듈을 추가하여 클라우드에서 생성된 이미지를 효율적으로 되돌려 받을 수 있는 방법에 대해 설명하였다. 또한 이러한 방법을 통해 오프로딩에 필요한 오버헤드를 줄임으로써 효과적으로 3D 렌더링 어플리케이션의 성능을 향상시킬 수 있음을 보였다.

5. Acknowledgement

본 연구는 IDEC, 교육과학기술부/한국과학재단 우수연구센터 육성사업(과제번호 2012-0000470), 미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신) [No. 10047212, 1kB 이하 암호문 간의 연산을 지원하는 동형 암호 원천 기술 개발 및 응용 연구] 및 중소기업청에서 지원하는 2012 년도 산학연공동기술개발사업(No. C0019562), 의 지원을 받아 수행하였습니다.

참고문헌

[1] B.-G.. Chun and et al, “CloneCloud: elastic execution between mobile device and cloud” in EuroSys’11, 2011
[2] E. Cuervo and et al, “MAUI: making smartphones last longer with code offload” in MobiSys’10, 2010
[3] S.Wang and S. Dey, “Rendering adaptation to address communication and computation constraints in cloud mobile gaming” in IEEE Globecom’10, 2010
[4] S.Wang and S. Dey, “Adaptive mobile cloud computing

to enable rich mobile multimedia applications" in IEEE Transactions on Multimedia, 2013
[5] libGDX, open source Java game development framework, libgdx.badlogicgames.com/