GPULib, CUDA, GPGPU

GPULib IDL

그래픽 카드로 계산을 한다구요?

그래픽 카드라는 것이 아무래도 화면에 내용을 표출하기 위한 장치였던 것은 분명합니다. 맞습니다. 우리가 PC로 게임을 할 때 매우 중요한 부분을 차지하는 그 그래픽 카드 말입니다.
그런데, 그래픽 카드가 하는 일이라는 것이 그래픽 메모리에 있는 내용을 화면에 뿌려 주는 것이기도 하지만, 3D 구조물을 2차원 화면에 뿌려내기 위한 변환 계산도 담당합니다. 요즘의 3D 그래픽이나 게임이 워낙 사실 같은 그림을 그려내려다 보니, 처리해야 하는 계산량이 만만치 않게 되었습니다.
게임에서 등장 인물의 머리카락이 찰랑거리는 것을 묘사하고자 할 때, 머리카락을 5개의 노드로 처리하는 것과 20개의 노드로 처리하는 것은 사실감 면에서 많은 차이가 있습니다. "빠르게 많은 양의 계산을 수행하는 것!" 이것이 바로 좋은 그래픽 카드의 사명이 된 것입니다.

다각형 수를 두배로 올렸을 때 캐릭터의 수준을 비교
http://en.wikipedia.org/wiki/File:High_Definition_Half-Life_comparison.jpg)

하지만, 많은 계산을 하는 시뮬레이션 프로그래머들은 게임의 현실감이 아닌, 단번에 대량의 연산을 수행하는 그래픽 카드의 계산 능력에 주목하고 있었습니다.

GPGPU와 CUDA, GPULib

그래픽 처리 장치(Graphic Process Unit; GPU)는 본래 그래픽을 위해 태어났지만, 내부적으로 대량의 데이터를 일괄적으로 빠르게 계산하는 특성을 가지고 있습니다. 이 점을 이용하여 그래픽 프로세서를 그래픽 뿐 아닌 일반적인 계산에도 사용해 보고자 하는 시도를 General Purpose GPU(GPGPU)라고 합니다. 안타깝지만 당연하게도 GPU는 CPU와 매우 다릅니다. CPU는 태생이 여러 가지 종류의 데이터에 대해 여러 가지 계산을 할 수 있는 다재다능을 목적하고 있지만, GPU는 다각형의 3차원 변환을 목적하고 있습니다. 원래 GPU는 오직 이것만 하는 것입니다.
그래서 GPU를 이용하여 프로그래밍을 하기 위해서는 수행하고자 하는 계산을 모두 3차원 그래픽 변환인 것처럼 바꾸어 수행해야 합니다. 그래픽 언어에 익숙하지 않다면 불가능한 일입니다. 그래서 NVIDIA사에서는 자사의 그래픽 카드를 이용하는 경우, 좀 더 손쉽게 그래픽 카드로 연산을 할 수 있는 프로그래밍 인터페이스를 발표하게 되는데, 이것이 CUDA(Compute Unified Device Architecture)입니다. CUDA는 C/C++을 기반으로 하여 소스코드를 컴파일할 때, GPU가 할 일과 CPU가 할 일을 나누어 줍니다.
그래픽 카드가 얼마나 좋은지, CPU가 얼마나 좋은지에 따라 다르겠지만, GPU를 연산에 활용하여 수배~수십배의 속도 향상을 경험하였다는 보고나 논문들이 속속 나오고 있습니다.
IDL 프로그래머들은 C/C++을 직접 만지고 싶어하지 않습니다. 그래서 Tech-X 사에서는 CUDA를 이용하여 IDL이 GPU 가속 기능을 활용할 수 있는 IDL 명령어들을 라이브러리로 만들게 되었는데, 이것이 GPULib입니다.

지원되는 그래픽 카드

NVIDIA GeFORCE 계열 9000대 이상, 동급의 노트북용 그래픽에서도 지원은 합니다. 그래픽 카드마다 CUDA를 지원한다는 문구를 통해 확인할 수 있고, NVIDIA의 CUDA 사이트에서도 확인할 수 있습니다. 하지만 NVIDIA의 고성능 그래픽 카드인 Quadro 시리즈와, 그래픽 출력단자도 없는, 오직 계산 전용의 그래픽 카드 Tesla 시리즈를 이용하는 것이 가장 큰 효과를 볼 수 있습니다. 그래픽 카드의 연산 성능을 가늠하는 요소는 다음과 같습니다.
1) 스트림 프로세서 수 : CPU에서 코어가 몇 개인지와 유사한 개념입니다. Quadro FX5800이나 Tesla C1060은 240개입니다.
2) GPU 메모리 : 한번에 많은 데이터를 GPU로 옮기고 이 안에서 연산을 마무리하는 것이 효율적입니다. GeFORCE 계열이 주로 512~1Gb의 메모리인데 반해, 고사양 Quadro나 Tesla는 4Gb메모리를 제공합니다.
3) GPU의 Clock : 클럭이 높을수록 속도가 빠르겠지요.

요구되는 하드웨어

1) CUDA 지원 그래픽 카드

GeFORCE에서도 효과를 볼 수 있지만, 대용량 그래픽 메모리가 장착된 것이 좋습니다. Quadro나 Tesla가 권장됩니다.

NVIDIA Tesla. 그래픽 카드지만 모니터 아웃 단자가 없습니다.

2) 두 개 이상의 그래픽 카드 인터페이스(권장)

화면 출력용 그래픽 카드를 이용하여 계산도 할 수 있습니다만, 연산 전용과 화면 출력용으로 역할을 나누는 것이 더 좋습니다. 그래픽 카드를 두 개 이상 사용할 수 있는 메인보드가 권장됩니다.

3) PCI Express x16 2.0

GPU 연산에서 상당한 시간이 메인메모리와 GPU 메모리의 데이터 전송에 소요됩니다. 현존하는 가장 빠른 인터페이스는 PCI Express x16 2.0이며 최신의 메인보드들이 대부분 채택합니다. PCI Express x16 1.0은 통신 속도가 조금 느려집니다.

4) 대용량의 파워 서플라이

고성능 Quadro나 Tesla라면 그래픽 카드 한 장 당, 넉넉히 200W 정도가 추가로 필요합니다.

5) 대용량 메인 메모리

효율적인 자료 흐름을 위해, GPU 메모리의 2배 이상 메인 메모리가 권장됩니다.

요구되는 소프트웨어

IDL에서 GPU를 고속 연산에 활용하는 GPULib은 CUDA를 기반으로 만들어진 IDL 명령어들의 모음입니다. 그러므로 GPULib을 사용하기 위해서는 CUDA를 반드시 설치하여야 합니다. CUDA 설치 이후, GPULib을 설치합니다.

1) CUDA 설치

http://kr.nvidia.com/object/cuda_get_kr.html

에서 다운로드 및 설치 문서를 받을 수 있으며, 무료로 제공됩니다.

2) GPULib 설치

http://www.txcorp.com/products/GPULib

에서 다운로드 및 설치 문서를 받을 수 있습니다. 교육기관에서는 무료로 사용할 수 있으며, 그 외에는 평가판을 사용해 보고 구입을 결정할 수 있습니다.

GPULib의 사용

Tech-X사의 GPULib 홈페이지에서 Documentation 페이지를 보면 IDL에서 GPULib 명령문들을 어떻게 사용하는지 확인할 수 있습니다. 거의 모든 명령어가 프로시저와 함수의 두 가지 형태 중 편리한 쪽을 선택하여 사용할 수 있게 제공됩니다.

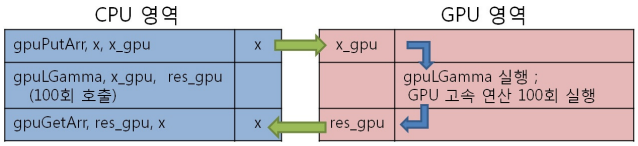
기본적인 사용 순서는 다음과 같습니다.

1) GPULib을 사용하기 위한 시동(gpuINIT)
2) CPU에서 처리할 일을 수행(일반 IDL)
3) 데이터를 GPU 메모리로 전송(gpuPutARR)
4) GPU 내에서 필요한 연산을 수행
   (gpuSin, gpuFFT, gpuCongrid, gpuInterpolate 등)
5) GPU 메모리에서 IDL의 메모리로 데이터 가져 오기
   (gpuGetARR)
6) CPU에서 처리할 일을 수행(일반 IDL)
7) 가시화(일반 IDL)
8) GPU 메모리 해제(gpuFree)

위 과정 중, 계산 시간과 상관 없는 3)과 5)에서 많은 시간이 소요된다는 점을 기억해 두고, 가능한 3)과 5) 과정을 한번만 수행할 수 있도록 프로그램을 만드는 것이 효율을 높이는 방법입니다.

간단한 예제 코드

```
pro gputest1
 gpuinit    ;GPULib 시동
 nx=1000000L & niter=100
 x=randomu(seed, nx)
 t=systime(1)
 gpuPutArr, x, x_gpu      ;GPU 메모리로 데이터 전송
 for i=0, niter do res_gpu=gpuLGamma(x_gpu) ;GPU 연산
 x=gpuGetArr(res_gpu)    ;메인 메모리로 데이터 복원
 gputime=systime(1)-t
 print, 'GPU Time = ', gputime
end
```



gpuLGamma 함수는 IDL의 LnGamma 함수에 대응하는 GPULib 함수입니다. 동일한 연산을 수행할 경우 몇 개의 CPU와 GPU에서 측정한 실행 속도는 다음과 같습니다.

Pentium D 3.0Ghz	Core2Duo T8100 2.1Ghz	Quad-Core Xeon 3.4Ghz 2개(8Core)	NVIDIA Quadro FX570
RAM 2Gb	RAM 2Gb	RAM 16Gb	VRAM 256Mb
35초	12초	1.6초	0.5초

100만개 요소에 대해 ln(Γ)를 100회 연산하는 소요 시간

GPU 연산의 한계

GPU는 본래 화면에 출력할 내용을 계산하기 위해 설계가 되었기 때문에, 단정밀도 실수형(Single Float) 정밀도면 충분했으며 이로인해 4byte Single Float 형이 가장 효율적으로 계산됩니다. Tesla 카드의 경우 배정밀도 실수형 연산을 지원하지만, 속도가 느립니다. 이는 차차 장비의 발전과 함께 해결될 문제입니다.

메인 메모리와 GPU 메모리간의 데이터 전송을 가능한 적게 수행해야 합니다.

결과적으로 GPU 연산이 좋은 효과(속도 향상)를 내는 분야가 존재하며, IDL과 GPULib 에서는 이미지(시그널) 프로세싱, 행렬 연산 등에서 좋은 효과를 내는 것으로 알려져 있습니다.

참고 사이트

1) http://kr.nvidia.com/object/cuda_home_kr.html
   CUDA를 이용하여 수행되는 연구 사례를 비롯한 CUDA의 많은 정보를 볼 수 있습니다.
2) http://www.txcorp.com/products/GPULIB
   Tech-X 사의 제품중 GPULib에 대한 소개와 문서, 소프트웨어를 다운로드 받을 수 있습니다.