opencl 번역

1. Introduction

최신 프로세서 아키텍처는 성능 향상을위해 병렬 처리를 채택했습니다. 고정 된 전력 범위에서 높은 클럭 속도로 기술적 문제에 직면 한 현재 중앙 처리 장치 (CPU)는 다중 코어를 추가하여 성능을 향상시킵니다. 그래픽 처리 장치 (GPU)는 고정 기능 렌더링 장치에서 프로그래머블 병렬 프로세서로 진화했습니다. 오늘날의 컴퓨터 시스템은 고도의 병렬 CPU, GPU 및 다른 유형의 프로세서를 포함하기 때문에 소프트웨어 개발자가 이러한 기종 처리 플랫폼을 최대한 활용할 수 있도록하는 것이 중요합니다.

이기종 병렬 처리 플랫폼을위한 애플리케이션을 개발하는 것은 멀티 코어 CPU와 GPU에 대한 전통적인 프로그래밍 방식이 매우 다르기 때문에 어려운 일입니다. CPU 기반 병렬 프로그래밍 모델은 일반적으로 표준을 기반으로하지만 일반적으로 공유 주소 공간을 가정하며 벡터 작업을 포함하지 않습니다. 범용 GPU 프로그래밍 모델은 복잡한 메모리 계층 및 벡터 연산을 처리하지만 플랫폼, 벤더 또는 하드웨어에 따라 다릅니다. 이러한 제한으로 인해 개발자는 하나의 다중 플랫폼 소스 코드 기반에서 이기종 CPU, GPU 및 기타 유형의 프로세서의 컴퓨팅 성능에 액세스하는 것을 어렵게 만듭니다. 그 어느 때보 다 소프트웨어 개발자가 고성능 컴퓨팅 서버에서 데스크톱 컴퓨터 시스템, 핸드 헬드 장치에 이르기까지 다양한 병렬 CPU, GPU 및 DSP와 같은 다른 프로세서 및 Cell/B.E. 프로세서 등 다양한 이기종 프로세싱 플랫폼을 효과적으로 활용할 필요가 있습니다 .

OpenCL (Open Computing Language)은 CPU, GPU 및 기타 프로세서 전반에 걸친 범용 병렬 프로그래밍을위한 개방형 으로 로열티가없는 표준이다. 소프트웨어 개발자가 이러한 이기종 프로세싱 플랫폼의 성능에 이식 가능하고 효율적으로 액세스 할 수있게 해줍니다.

OpenCL는 임베디드 및 소비자 용 소프트웨어에서 HPC 솔루션에 이르는 다양한 애플리케이션을 저수준의 고성능 휴대용 추상화를 통해 지원합니다. 효율적인 close-to-the-metal programming interface 를 생성함으로써 OpenCL은 플랫폼 독립적 인 도구, middleware 및 layer of a parallel computing ecosystem of platform-independent tools (응용 프로그램의 병렬 컴퓨팅 에코 시스템의 기초 계층)을 형성합니다.
OpenCL은 일반 병렬 계산 알고리즘과 그래픽 렌더링 파이프 라인을 결합한 새로운 대화 형 그래픽 응용 프로그램에서 점점 더 중요한 역할을 수행하는 데 특히 적합합니다.

OpenCL은 이기종 프로세서에서 병렬 계산을 조정하는 API로 구성됩니다. 잘 알려진 계산 환경을 갖춘 교차 플랫폼 프로그래밍 언어입니다. OpenCL 표준 :

○데이터 및 작업 기반 병렬 프로그래밍 모델 모두 지원
○병렬 처리를위한 확장과 함께 ISO C99의 하위 집합을 활용합니다.
○IEEE 754를 기반으로 일관된 수치 요구 사항을 정의합니다.
○핸드 헬드 및 임베디드 장치에 대한 구성 프로파일을 정의합니다.
○OpenGL, OpenGL ES 및 기타 그래픽 API와의 효율적인 상호 운용

이 문서는 OpenCL의 기본 개념과 아키텍처에 대한 개요와 그 실행 모델, 메모리 모델 및 동기화 지원에 대한 자세한 설명으로 시작됩니다. 그런 다음 OpenCL 플랫폼 및 런타임 API에 대해 설명하고 OpenCL C 프로그래밍 언어에 대한 자세한 설명이 이어집니다.

OpenCL에서 작성된 예제 계산 사례 및 사례를 설명하는 몇 가지 예가 제공됩니다. 사양은 OpenCL 호환 구현이 지원해야하는 핵심 사양으로 나뉩니다. 핸드 헬드 및 임베디드 장치에 대한 OpenCL 컴플라이언스 요구 사항을 완화하는 핸드 헬드 / 임베디드 프로파일 OpenCL 사양의 이후 개정에서 핵심 사양으로 이동할 가능성이있는 옵션 확장 세트를 제공합니다.

2. Glossary

Application : 호스트에서 실행중인 프로그램과 OpenCL 장치의 조합입니다.

Blocking and Non-Blocking Enqueue API calls: 비 블로킹 enqueue API 호출은 명령 대기열에 명령을 놓고 즉시 호스트로 리턴합니다. 블로킹 모드 enqueue API 호출은 명령이 완료 될 때까지 호스트로 돌아가지 않습니다.

Barrier: 명령 대기열 장벽과 작업 그룹 장벽이라는 두 가지 유형의 장벽이 있습니다.

○ OpenCL API는 명령 대기열 장벽 명령을 대기열에 넣는 기능을 제공합니다.
barrier 명령은 명령 대기열안에 있는 대기열에 포함 된 모든 명령이 실행을 시작하기 전에 명령 대기열에 대해 이전 대기열에 있던 모든 명령이 실행을 완료했는지 확인합니다.

○OpenCL C 프로그래밍 언어는 내장 된 작업 그룹 장벽 기능을 제공합니다.
이 장벽 내장 함수는 장치에서 실행중인 커널이 커널을 실행하는 작업 그룹에서 작업 항목 간의 동기화를 수행하는 데 사용할 수 있습니다. 작업 그룹의 모든 작업 항목은 장벽을 넘어서 실행을 계속하기 전에 장벽 구조를 실행해야합니다.

Buffer Object: 선형 바이트 집합을 저장하는 메모리 객체입니다. 버퍼 객체는 장치에서 실행중인 커널의 포인터를 사용하여 액세스 할 수 있습니다. 버퍼 객체는 OpenCL API 호출을 사용하여 호스트에서 조작 할 수 있습니다. 버퍼 객체는 다음 정보를 캡슐화합니다.

○크기 (바이트).
○사용 정보 및 할당 할 영역을 설명하는 등록 정보.
○버퍼 데이터.

Command : 실행을 위해 명령 대기열에 제출되는 OpenCL 작업입니다. 예를 들어 OpenCL 명령은 컴퓨팅 장치에서 실행하기 위해 커널을 실행하고 메모리 개체를 조작합니다.

Command-queue : 특정 장치에서 실행될 명령을 보유하고있는 개체. 명령 대기열은 컨텍스트의 특정 장치에 작성됩니다. 명령 대기열에 대한 명령은 순서대로 대기열에 있지만 순서대로 또는 순서없이 실행될 수 있습니다. In-order Execution 및 Out-of-Order 실행을 참조하십시오.

Command-queue Barrier. 배리어를 참조하십시오.

Compute Device Memory : 컴퓨팅 장치에 연결된 하나 이상의 메모리를 나타냅니다.

Compute Unit: OpenCL 장치에는 하나 이상의 계산 단위가 있습니다. 작업 그룹은 단일 계산 단위에서 실행됩니다. 계산 단위는 하나 이상의 처리 요소와 로컬 메모리로 구성됩니다. 컴퓨팅 유닛은 또한 프로세싱 요소에 의해 액세스 될 수있는 전용 텍스처 필터 유닛을 포함 할 수있다.

동시성 (Concurrency) : 시스템 한 세트의 작업이 활성 상태를 유지하고 동시에 진행될 수있는 시스템의 속성. 프로그래머는 프로그램을 실행할 때 동시 실행을 활용하려면 문제의 동시성을 식별하고 소스 코드 내에서 노출 한 다음 동시성을 지원하는 표기법을 사용해야합니다.

상수 메모리 (Constant Memory) : 전역 메모리의 한 영역으로, 커널을 실행하는 동안 일정하게 유지됩니다. 호스트는 상수 메모리에 배치 된 메모리 객체를 할당하고 초기화합니다.

Context: 커널이 실행되는 환경과 동기화 및 메모리 관리가 정의되는 도메인입니다. 컨텍스트에는 장치 세트, 해당 장치에 액세스 할 수있는 메모리, 해당 메모리 속성 및 커널 실행 또는 메모리 개체 작업을 예약하는 데 사용되는 하나 이상의 명령 대기열이 포함됩니다.

Data Parallel Programming Model:전통적으로 이 용어는 데이터 구조 집합 내의 여러 요소에 적용되는 단일 프로그램의 명령으로 동시성을 표현하는 프로그래밍 모델을 나타냅니다. 이 용어는 OpenCL에서 단일 프로그램 명령어 집합이 인덱스의 추상 도메인 내의 각 지점에 동시에 적용되는 모델을 나타 내기 위해 일반화되었습니다.

Device: 장치는 계산 단위 모음입니다. 명령 대기열은 장치에 명령을 대기 행렬에 넣는 데 사용됩니다. 명령의 예로는 커널 실행 또는 메모리 오브젝트 읽기 및 쓰기가 있습니다. OpenCL 장치는 일반적으로 GPU, 멀티 코어 CPU 및 DSP와 같은 다른 프로세서 및 Cell/B.E. 프로세서에 해당합니다.

Event Object: 이벤트 객체는 명령과 같은 작업의 상태를 캡슐화합니다. 컨텍스트에서 작업을 동기화하는 데 사용할 수 있습니다.

Event Wait List: 이벤트 대기 목록은 특정 명령 실행이 시작될 때를 제어하는 데 사용할 수있는 이벤트 개체의 목록입니다.

프레임 워크 (Framework) : 소프트웨어 개발 및 실행을 지원하는 컴포넌트 세트를 포함하는 소프트웨어 시스템. 프레임 워크는 일반적으로 라이브러리, API, 런타임 시스템, 컴파일러 등을 포함합니다.

Global ID: 글로벌 ID는 작업 항목을 고유하게 식별하는 데 사용되며 커널을 실행할 때 지정된 전역 작업 항목 수에서 파생됩니다. 글로벌 ID는 (0, 0, ... 0)에서 시작하는 N 차원 값입니다. 지역 ID를 참조하십시오.

Global Memory: 컨텍스트에서 실행되는 모든 작업 항목에 액세스 할 수있는 메모리 영역입니다. 읽기, 쓰기 및 맵과 같은 명령을 사용하여 호스트에 액세스 할 수 있습니다.

Handle: OpenCL에서 할당 한 객체를 참조하는 불투명 유형입니다. 객체에 대한 모든 작업은 해당 객체의 핸들을 참조하여 발생합니다.

Host: 호스트는 OpenCL API를 사용하여 컨텍스트와 상호 작용합니다.

Host pointer: 호스트의 가상 주소 공간에있는 메모리에 대한 포인터입니다.

Illegal : 명시 적으로 허용되지 않고 OpenCL에서 오류로보고되는 시스템의 동작입니다.

Image Object: 2 차원 또는 3 차원 구조 배열을 저장하는 메모리 객체입니다.
이미지 데이터는 읽기 및 쓰기 기능으로 만 액세스 할 수 있습니다. 읽기 함수는 샘플러를 사용합니다.

image 객체는 다음 정보를 캡슐화합니다.
○이미지의 크기.
○이미지의 각 요소에 대한 설명입니다.
○사용 정보 및 할당 할 영역을 설명하는 등록 정보.
○이미지 데이터.

이미지의 요소는 사전 정의 된 이미지 형식 목록에서 선택됩니다.

Implementation Defined: OpenCL의 준수 구현간에 명시 적으로 허용되는 동작입니다. OpenCL 구현자는 구현 정의 동작을 문서화해야합니다.

In-order Execution: 명령 대기열의 명령이 제출 순서대로 실행되어 각 명령이 완료되기 전에 실행되어 다음 명령이 시작되는 OpenCL 실행 모델입니다. out-of-order 실행을 참조하십시오.

Kernel: 커널은 프로그램에서 선언되고 OpenCL 장치에서 실행되는 함수입니다. 커널은 프로그램에 정의 된 모든 함수에 적용되는 __kernel 한정자로 식별됩니다.

Kernel Object: 커널 객체는 프로그램에서 선언 된 특정 __kernel 함수와 __kernel 함수를 실행할 때 사용할 인수 값을 캡슐화합니다

Local ID: 로컬 ID는 커널을 실행중인 주어진 작업 그룹 내에서 고유 한 작업 항목 ID를 지정합니다. 지역 ID는 (0, 0, ... 0)에서 시작하는 N 차원 값입니다. 글로벌 ID를 참조하십시오.

Local Memory: 작업 그룹과 연관되고 해당 작업 그룹의 작업 항목 만 액세스 할 수있는 메모리 영역입니다.

마커 (Marker) : 명령 대기열에 대기중인 명령. 명령 대기열의 마커 앞에 대기중인 모든 명령에 태그를다는 데 사용할 수 있습니다. marker 명령은 응용 프로그램이 마커 이벤트에 대한 대기를 대기시키기 위해 사용할 수있는 이벤트를 반환합니다. 즉 마커 명령이 완료되기 전에 대기 된 모든 명령을 기다립니다.

Memory Objects: 메모리 객체는 전역 메모리의 참조 카운트 영역에 대한 핸들입니다. 또한 버퍼 객체 및 이미지 객체를 참조하십시오.

Memory Regions (or Pools): OpenCL의 고유 한 주소 공간. OpenCL은 논리적으로 구별되는 것으로 취급하지만 메모리 영역은 실제 메모리에서 겹칠 수 있습니다. 메모리 영역은 private, local, constant 및 global로 표시됩니다.

Object: 객체는 OpenCL API에 의해 조작 될 수있는 자원의 추상 표현입니다. 예제에는 프로그램 개체, 커널 개체 및 메모리 개체가 포함됩니다.

Out-of-Order Execution : 작업 큐에 배치 된 명령이 이벤트 대기 목록 및 명령 대기열 장벽에 의해 부과 된 제약 조건과 일치하는 순서로 실행을 시작하고 완료 할 수있는 실행 모델입니다. 순서 실행을 참조하십시오.

Platform: 호스트에 OpenCL 프레임 워크가 관리하는 장치 모음이 포함되어있어 응용 프로그램이 자원을 공유하고 플랫폼의 장치에서 커널을 실행할 수 있습니다.

Private Memory: 작업 항목 전용 메모리 영역. 한 작업 항목의 개인 메모리에 정의 된 변수는 다른 작업 항목에 표시되지 않습니다.

Processing Element: 가상 스칼라 프로세서. 작업 항목은 하나 이상의 처리 요소에서 실행될 수 있습니다.

Program: OpenCL 프로그램은 일련의 커널로 구성됩니다. 프로그램에는 __kernel 함수 및 상수 데이터에 의해 호출되는 보조 함수가 포함될 수도 있습니다

Program Object: 프로그램 개체는 다음 정보를 캡슐화합니다.

○연관된 컨텍스트에 대한 참조입니다.

○프로그램 소스 또는 바이너리.

○최신으로 성공적으로 빌드 된 프로그램 실행 파일, 프로그램 실행 파일이 빌드 된 장치 목록, 사용 된 빌드 옵션 및 빌드 로그.

○현재 접속 된 커널 오브젝트의 수.

Reference Count: OpenCL 개체의 수명은 참조 수 (개체에 대한 참조 수의 내부 수)에 의해 결정됩니다. OpenCL에서 객체를 만들면 참조 카운트가 1로 설정됩니다.
이후에 적절한 retain API (예 : clRetainContext,clRetainCommandQueue)를 호출하면 참조 횟수가 증가합니다. 마지막 개정 날짜로의 호출 : 6/1/11 Page 18 적절한 릴리스

API (예 : clReleaseContext, clReleaseCommandQueue)는 참조 횟수를 감소시킵니다. 참조 횟수가 0이되면 개체의 리소스는 OpenCL에 의해 할당이 해제됩니다.

Relaxed Consistency (Relaxed Consistency) : 장벽이나 다른 명시적인 동기화 지점을 제외하고는 다른 작업 항목이나 명령에서 볼 수있는 메모리의 내용이 다를 수있는 메모리 일관성 모델입니다.

Resource: OpenCL에 의해 정의 된 객체의 클래스. 자원의 인스턴스는 객체입니다. 가장 일반적인 리소스는 컨텍스트, 명령 대기열, 프로그램 개체, 커널 개체 및 메모리 개체입니다. 전산 자원은 프로그램 카운터를 발전시키는 활동에 참여하는 하드웨어 요소입니다. 예를 들면 호스트, 장치, 계산 단위 및 처리 요소가 포함됩니다.

Retain, Release : OpenCL 객체를 사용하여 참조 카운트를 증가 (유지) 및 감소 (해제)하는 동작입니다. 이것은이 객체를 사용하는 모든 인스턴스가 완료되기 전에 시스템이 객체를 제거하지 않도록하는 기능을 유지하는 책입니다. 참조 횟수를 참조하십시오.

샘플러 (Sampler) : 커널에서 이미지를 읽을 때 이미지를 샘플링하는 방법을 설명하는 객체. 이미지 읽기 함수는 샘플러를 인수로 사용합니다. 샘플러는, 이미지 주소 지정 모드, 즉 범위 외의 이미지 좌표의 처리 방법, 필터 모드, 및 입력 이미지 좌표가 정규화 또는 비 표준화 된 값 일지 어떨지를 지정합니다.

SIMD : 단일 명령어 다중 데이터. 커널이 각각 자체 데이터와 공유 프로그램 카운터를 가진 다중 처리 요소에서 동시에 실행되는 프로그래밍 모델. 모든 처리 요소는 엄격하게 동일한 명령어 세트를 실행합니다.

SPMD : 단일 프로그램 다중 데이터. 커널이 각각 자체 데이터와 자체 프로그램 카운터를 가진 다중 처리 요소에서 동시에 실행되는 프로그래밍 모델. 따라서 모든 계산 리소스가 동일한 커널을 실행하지만 자신의 명령 카운터를 유지하고 커널의 분기로 인해 실제 명령 시퀀스는 처리 요소 세트 전체에서 상당히 다를 수 있습니다.

작업 병렬 프로그래밍 모델 (Task Parallel Programming Model) : 작업이 하나의 작업 그룹 (크기가 하나 인 그룹)에서 실행되는 여러 동시 작업의 관점에서 계산이 표시되는 프로그래밍 모델입니다. 동시 작업은 다른 커널을 실행할 수 있습니다.

Thread-safe: OpenCL API 호출은 OpenCL에 의해 관리되는 내부 상태가 여러 호스트 스레드에 의해 동시에 호출 될 때 일관성이 유지되면 스레드로부터 안전하다고 간주됩니다. 스레드로부터 안전한 OpenCL API 호출은 응용 프로그램이 여러 호스트 스레드에서 이러한 함수를 호출 할 수있게합니다. 즉, 호스트 스레드간에 상호 배제를 구현할 필요가 없으므로 다시 침입자가 안전합니다. 최종 수정 날짜 : 2011 년 6 월 1 일

Undefined : OpenCL API 호출의 동작, OpenCL에서 명시 적으로 정의하지 않은 커널 내에서 사용되는 기본 제공 함수 또는 커널 실행. 부적합한 구현체는 OpenCL에서 정의되지 않은 구조체가 발생할 때 발생하는 것을 지정하지 않아도됩니다.

Work-group: 단일 계산 단위에서 실행되는 관련 작업 항목 모음입니다. 그룹의 작업 항목은 동일한 커널을 실행하고 로컬 메모리 및 작업 그룹 장벽을 공유합니다.

Work-group Barrier. 배리어를 참조하십시오.

작업 항목 (Work-item) : 명령에 의해 장치에서 호출 된 커널의 병렬 실행 모음 중 하나입니다. 작업 항목은 계산 단위에서 실행되는 작업 그룹의 일부로 하나 이상의 처리 요소에 의해 실행됩니다. 작업 항목은 해당 글로벌 ID 및 로컬 ID로 컬렉션 내의 다른 실행과 구별됩니다.