<<OpenCL编程指南>>

图书基本信息

书名：<<OpenCL编程指南>>

13位ISBN编号：9787030349637

10位ISBN编号：7030349636

出版时间：2012-7

出版时间：科学出版社

作者：曼什

页数：603

字数：1003250

<<OpenCL编程指南>>

内容概要

新的OpenCL标准有助于充分利用CPU、GPU等处理器的丰富资源，已获得Apple、AMD、Intel、IBM等公司的认可，在服务器、嵌入式设备、高性能计算等领域有广阔的应用前景。

《OpenCL编程指南》由OpenCL的五大技术权威共同撰写，内容涵盖完整的规范。
在分析关键用户案例的基础上，说明了如何用OpenCL表示各类并行算法，并且提供了完整的API和OpenCLC语言的参考信息。
通过完整的案例学习和代码示例，讲解了编写复杂并行程序的方法，实现在众多不同设备间分解工作量，还介绍了OpenCL软件性能优化的要点。

《OpenCL编程指南》是第一本针对OpenCL1.1规范的全面、权威的实践指南，适合信息技术领域的研发人员和软件架构师阅读参考。

作者简介

无

<<OpenCL编程指南>>

书籍目录

ForewordPrefacePart I The OpenCL 1-1 Language and API1.An Introduction to OpenCLWhat Is OpenCL,or Why You Need This BookOur Many-Core Future:Heterogeneous PlatformsSoftware in a Many-Core WorldConceptual Foundations of OpenCLPlatform ModelExecution ModelMemory ModelProgramming ModelsOpenCL and GraphicsThe Contents of OpenCLPlatform APIRuntime APIKernel Programming LanguageOpenCL SummaryThe Embedded ProfileLearning OpenCL2.HelloWorld:An OpenCL ExampleBuilding the ExamplesPrerequisitesMac OS X and Code:BlocksMicrosoft Windows and Visual StudioLinux and EclipseHelloWorld ExampleChoosing an OpenCL Platform and Creating a ContextChoosing a Device and Creating a Command-QueueCreating and Building a Program ObjectCreating Kernel and Memory ObjectsExecuting a KernelChecking for Errors in OpenCL3.Platforms,Contexts,and DevicesOpenCL PlatformsOpenCL DevicesOpenCL Contexts4.Programming with OpenCL CWriting a Data-Parallel Kernel Using OpenCL CScalar Data TypesThe half Data TypeVector Data TypesVector LiteralsVector ComponentsOther Data TypesDerived TypesImplicit Type ConversionsUsual Arithmetic ConversionsExplicit CastsExplicit ConversionsReinterpreting Data as Another TypeVector OperatorsArithmetic OperatorsRelational and Equality OperatorsBitwise OperatorsLogical OperatorsConditional OperatorShift OperatorsUnary OperatorsAssignment OperatorQualifiersFunction QualifiersKernel Attribute QualifiersAddress Space QualifiersAccess QualifiersType QualifiersKeywordsPreprocessor Directives and MacrosPragma DirectivesMacrosRestrictions5.OpenCL C Built-In FunctionsWork-Item FunctionsMath FunctionsFloating-Point PragmasFloating-Point ConstantsRelative Error as ulpsInteger FunctionsCommon FunctionsGeometric FunctionsRelational FunctionsVector Data Load and Store FunctionsSynchronization FunctionsAsync Copy and Prefetch FunctionsAtomic FunctionsMiscellaneous Vector FunctionsImage Read and Write FunctionsReading from an ImageSamplersDetermining the Border ColorWriting to an ImageQuerying Image Information6.Programs and KernelsProgram and Kernel Object OverviewProgram ObjectsCreating and Building ProgramsProgram Build OptionsCreating Programs from BinariesManaging and Querying ProgramsKernel ObjectsCreating Kernel Objects and Setting Kernel ArgumentsThread SafetyManaging and Querying Kernels7.Buffers and Sub-BuffersMemory Objects,Buffers,and Sub-Buffers OverviewCreating Buffers and Sub-BuffersQuerying Buffers and Sub-BuffersReading,Writing,and Copying Buffers and Sub-BuffersMapping Buffers and Sub-Buffers8.Images and SamplersImage and Sampler Object OverviewCreating Image ObjectsImage FormatsQuerying for Image SupportCreating Sampler ObjectsOpenCL C Functions for Working with ImagesTransferring Image Objects9.EventsCommands,Queues,and Events OverviewEvents and Command-QueuesEvent ObjectsGenerating Events on the HostEvents Impacting Execution on the HostUsing Events for ProfilingEvents Inside KernelsEvents from Outside OpenCL10.Interoperability with OpenGLOpenCL/OpenGL Sharing OverviewQuerying for the OpenGL Sharing ExtensionInitializing an OpenCL Context for OpenGL InteroperabilityCreating OpenCL Buffers from OpenGL BuffersCreating OpenCL Image Objects from OpenGL TexturesQuerying Information about OpenGL ObjectsSynchronization between OpenGL and OpenCL11.Interoperability with Direct3DDirect3D/OpenCL Sharing OverviewInitializing an OpenCL Context for Direct3D InteroperabilityCreating OpenCL Memory Objects from Direct3D Buffers and TexturesAcquiring and Releasing Direct3D Objects in OpenCLProcessing a Direct3D Texture in OpenCLProcessing D3D Vertex Data in OpenCL12.C++ Wrapper APIC++ Wrapper API OverviewC++ Wrapper API ExceptionsVector Add Example Using the C++ Wrapper APIChoosing an OpenCL Platform and Creating a ContextChoosing a Device and Creating a Command-QueueCreating and Building a Program ObjectCreating Kernel and Memory ObjectsExecuting the Vector Add Kernel13.OpenCL Embedded ProfileOpenCL Profile Overview64-Bit IntegersImagesBuilt-In Atomic FunctionsMandated Minimum Single-Precision Floating-Point CapabilitiesDetermining the Profile Supported by a Device in an OpenCL C ProgramPart II OpenCL 1-1 Case Studies14.Image HistogramComputing an Image HistogramParallelizing the Image HistogramAdditional Optimizations to the Parallel Image HistogramComputing Histograms with Half-Float or Float Values for Each

Channel15.Sobel Edge Detection FilterWhat Is a Sobel Edge Detection Filter?Implementing the Sobel Filter as an OpenCL Kernel16.Parallelizing Dijkstra`s Single-Source Shortest-Path Graph AlgorithmGraph Data StructuresKernelsLeveraging Multiple Compute Devices17.Cloth Simulation in the Bullet Physics SDKAn Introduction to Cloth SimulationSimulating the Soft BodyExecuting the Simulation on the CPUChanges Necessary for Basic GPU ExecutionTwo-Layered BatchingOptimizing for SIMD Computation and Local MemoryAdding OpenGL Interoperation18.Simulating the Ocean with Fast Fourier TransformAn Overview of the Ocean ApplicationPhillips Spectrum GenerationAn OpenCL Discrete Fourier TransformDetermining 2D DecompositionUsing Local MemoryDetermining the Sub-Transform SizeDetermining the Work-Group SizeObtaining the Twiddle FactorsDetermining How Much Local Memory Is NeededAvoiding Local Memory Bank ConflictsUsing ImagesA Closer Look at the FFT KernelA Closer Look at the Transpose Kernel19.Optical FlowOptical Flow Problem OverviewSub-Pixel Accuracy with Hardware Linear InterpolationApplication of the Texture CacheUsing Local MemoryEarly Exit and Hardware SchedulingEfficient Visualization with OpenGL InteropPerformance20.Using OpenCL with PyOpenCLIntroducing PyOpenCLRunning the PyImageFilter2D ExamplePyImageFilter2D CodeContext and Command-Queue CreationLoading to an Image ObjectCreating and Building a ProgramSetting Kernel Arguments and Executing a KernelReading the Results21.Matrix Multiplication with OpenCLThe Basic Matrix Multiplication AlgorithmA Direct Translation into OpenCLIncreasing the Amount of Work per KernelOptimizing Memory Movement:Local MemoryPerformance Results and Optimizing the Original CPU Code22.Sparse Matrix-Vector MultiplicationSparse Matrix-Vector Multiplication(SpMV)AlgorithmDescription of This ImplementationTiled and Packetized Sparse Matrix RepresentationHeader StructureTiled and Packetized Sparse Matrix Design ConsiderationsOptional Team InformationTested Hardware Devices and ResultsAdditional Areas of OptimizationA.Summary of OpenCL 1.1The OpenCL Platform LayerContextsQuerying Platform Information and DevicesThe OpenCL RuntimeCommand-QueuesBuffer ObjectsCreate Buffer ObjectsRead,Write,and Copy Buffer ObjectsMap Buffer ObjectsManage Buffer ObjectsQuery Buffer ObjectsProgram ObjectsCreate Program ObjectsBuild Program ExecutableBuild OptionsQuery Program ObjectsUnload the OpenCL CompilerKernel and Event ObjectsCreate Kernel ObjectsKernel Arguments and Object QueriesExecute KernelsEvent ObjectsOut-of-Order Execution of Kernels and Memory Object CommandsProfiling OperationsFlush and FinishSupported Data TypesBuilt-In Scalar Data TypesBuilt-In Vector Data TypesOther Built-In Data TypesReserved Data TypesVector Component AddressingPreprocessor Directives and MacrosSpecify Type AttributesMath ConstantsWork-Item Built-In FunctionsInteger Built-In FunctionsCommon Built-In FunctionsMath Built-In FunctionsGeometric Built-In FunctionsRelational Built-In FunctionsVector Data Load/Store FunctionsAtomic FunctionsAsync Copies and Prefetch FunctionsSynchronization,Explicit Memory FenceMiscellaneous Vector Built-In FunctionsImage Read and Write Built-In FunctionsVector ComponentsVector Addressing EquivalenciesConversions and Type Casting ExamplesOperatorsAddress Space QualifiersFunction QualifiersImage ObjectsCreate Image ObjectsQuery List of Supported Image FormatsCopy between Image,Buffer ObjectsMap and Unmap Image ObjectsRead,Write,Copy Image ObjectsQuery Image ObjectsImage FormatsAccess QualifiersSampler ObjectsSampler Declaration FieldsOpenCL Device Architecture DiagramOpenCL/OpenGL Sharing APIsCL Buffer Objects>GL Buffer ObjectsCL Image Objects>GL TexturesCL Image Objects>GL RenderbuffersQuery InformationShare ObjectsCL Event Objects>GL Sync ObjectsCL Context>GL Context,SharegroupOpenCL/Direct3D 10 Sharing APIsIndex

<<OpenCL编程指南>>

章节摘录

版权页： 插图： The solution to this problem is for the program object to be built from source at runtime.The host program defines the devices within the context.Only at that point is it possible to know how to compile the program source code to create the code for the kernels.As for the source code itself,OpenCL is quite flexible about the form.In many cases,it is a regular string either statically defined in the host program,loaded from a file at runtime,or dynamically generated inside the host program. Our context now includes OpenCL devices and a program object from which the kernels are pulled for execution.Next we consider how the kernels interact with memory.The detailed memory model used by OpenCL will be described later.For the sake of our discussion of the context,we need to understand how the OpenCL memory works only at a high level.The crux of the matter is that on a heterogeneous platform,there are often multiple address spaces to manage.The host has the familiar address space expected on a CPU platform,but the devices may have a range of different memory architectures.To deal with this situation,OpenCL introduces the idea of memory objects.These are explicitly defined on the host and explicitly moved between the host and the OpenCL devices.This does put an extra burden on the programmer,but it lets us support a much wider range of platforms.We now understand the context within an OpenCL application.The context is the OpenCL devices,program objects,kernels,and memory objects that a kernel uses when it executes.Now we can move on to how the host program issues commands to the OpenCL devices. Command-Queues The interaction between the host and the OpenCL devices occurs through commands posted by the host to the command-queue.These commands wait in the command-queue until they execute on the OpenCL device.A command-queue is created by the host and attached to a single OpenCL device after the context has been defined.

<<OpenCL编程指南>>

编辑推荐

《国外信息科学与技术优秀图书系列:OpenCL编程指南(英文版)》针对最新的OpenCL1.1规范进行编写。
由OpenCL技术领域的五大权威共同撰写，内容全面，涵盖完整的规范。
提供大量的用户案例和代码示例，详尽完整的API和OpenCL C语言参考，具有很强的实用价值和参考价值。

版权说明