Лекция 7: Использование графических процессоров для ускорения вычислений. Обзор сред программирования, библиотек и языков

Н.Д. Смирнова

Санкт-Петербургский государственный Политехнический университет

15.11.2011

Содержание

Мотивация

Что нужно решить прежде, чем реализовывать проект на GPU

- есть много времени для проб, ошибок + желание сделать сразу хорошо
  - изучаем языки программирования CUDA, OpenCL
  - переписываем существующий код и алгоритмы
- нет времени, есть желание попробовать быстро
  - используем директивный подход
  - или высокоуровневые библиотеки, пакеты для вычислений

Поговорим о

- Languages
- Middleware
- Math Applications
- Math Libraries

Содержание

Возможности

- CUDA (C/Fortran)
- OpenCL
- DirectCompute
- PGI Accelerator Compilers
- HMPP

CUDA

- модель программирования, архитектура GPU
- пути использования
  - высокий уровень: CUDA C + CUDA runtime API
  - низкий уровень: CUDA device API
- наиболее популярное средство
- зависимость от производителя графических карт
- кросс-платформенная: Linux, Window, MacOS
- изучаем тут http://nvidia.com/cudazone

OpenCL

- является открытым (бесплатным) "стандартом"
  - управляется организацией Khronos group
  - которая включает всех основных производителей железа
  - "кросс-производительный"стандарт
  - хочется, чтобы им больше занимались на практике
- C-подобный язык для kernel'ов
- низкоуровневый API для управления девайсами
  - похож на CUDA device API
  - дает портируемый код, требующий оптимизации для каждой платформы
- изучаем тут http://www.khronos.org/opencl/

DirectCompute

- стандарт для GPGPU на любых графических картах под Windows
- естественно предоставлен компанией Microsoft
- выпущен с DirectX11 для Windows7
- поддерживает все CUDA-карты (DX10+DX11) и карты от ATI
- низкоуровневая библиотека для управления графической картой и запуском kernel'ов
- определяет C-подобный язык на основе HLSL для kernel'ов
- изучаем тут http://archive.msdn.microsoft.com/DirectComputeLecture

Использование компиляторов - PGI Accelerator Compilers

- компилируют программы на C99/Fortran
  - теже makefiles, scripts, IDE, ...
- поддерживает только CUDA-GPU
- понимает дополнительные директивы (OpenMP-like), отмечающие участки кода для распараллеливания
- очень просто попробовать, минимум усилий
- стоит дополнительных денег
- изучаем тут http://www.pgroup.com/resources/accel.htm

HMPP (Hybrid Multicore Parallel Programming)

- HMPP - открытый стандарт
- компилирует программы на C/C++/Fortran
- распределяет вычисления между CPU и GPU
- дополнительные директивы (OpenMP-like), отмечающие участки программы для распараллеливания
- OpenHMPP [1]- реализация стандарта
  - все NVidia Tesla, ATI Firestream
  - Windows, Linux
  - стоит дополнительных денег
  - цель: упростить программирование для GPU
- изучаем стандарт тут http://www.openhmpp.org

[1] http://www.caps-entreprise.com/hmpp.html

Содержание

Возможности

- .NET
- Java
- PyCUDA
- Автоматическое распараллеливание кода
- Симуляторы и дизассемблеры

.NET, Java

.NET

- GPU.NET [2]
- CUDA.NET, OpenCL.NET [3]
- CAL.NET [4]
- .NET bindings для CUDA/CAL/OpenCL приложений
- идея: писать kernel'ы на языках C#, F#, VB.NET
- не надо переписывать код

Java

- jCUDA [5]

---

[2] http://www.tidepowerd.com/
[3] http://www.hoopoe-cloud.com/default.aspx
[4] http://gpgpu.org/tag/amd-cal
[5] http://www.jcuda.de/

PyCUDA, PyOpenCL

- функциональность CUDA/OpenCL в языке Phyton
- бесплатно с MIT-лицензией
- CUDA C код находится в строках
- предоставлено A. Klockner, Brown University / NYU
- изучаем тут http://mathema.tician.de/software/pycuda, http://mathema.tician.de/software/pyopencl
- существуют еще wrapper'ы для Perl, Lua, Ruby, IDL

Автоматическое распараллеливание

F2C-Acc - от Earth System Research Laboratory

- Fortran → CUDA C: source-to-source
- используются специальные директивы
- полученый код можно оптимизировать
- изучаем тут http://www.esrl.noaa.gov/gsd/ab/ac/F2C-ACC.html

Par4All

- только(?) для Linux
- компилятор автоматически распараллеливает и оптимизирует
- понимает C, Fortran
- изучаем тут http://www.par4all.org/

Симуляторы

GPUocelot

- симулирует CUDA-программы на AMD GPUs, x86 CPUs без перекомпиляции
- динамическая компиляция для разнородных систем
- изучаем тут http://code.google.com/p/gpuocelot/

Barra

- модульный функциональный симулятор для GPU
- симулирует CUDA-программы на уровне ассемблера
- может быть использован для оптимизации, профайлинга и дебага
- изучаем тут http://gpgpu.univ-perp.fr/index.php/Barra

Disassembler

decuda

- для CUDA-программ
- были проблемы с CUDA 3.x [6]
- берем и изучаем тут https://github.com/laanwj/decuda/wiki/

[6] http://forums.nvidia.com/index.php?s=bdc089f074a26c2ff6e406ddd8a95f1b&show

Содержание

Где встречается

- Matlab
- Mathematica
- LabView, PETSc, Trilinos, OpenFoam,

Matlab

Matlab Parallel Computing Toolbox (PCT) [7]

- набор инструментов для параллельных вычислений от Matlab
- позволяет распараллелить исполнение Matlab-вычислений без знания CUDA и MPI
- поддерживаются только CUDA-карты
- возможно использование нескольких GPU одновременно
- советы можно найти тут http://www.mathworks.com/discovery/matlab-gpu.html

Matlab Distributed Computing Server

- позволяет запускть PCT-приложение на кластере

[7] http://www.mathworks.com/products/parallel-computing/

Jacket / AccelerEyes

- компилирует Matlab-код для CUDA-карт
- высоко-уровневый интерфейс
- поддержка нескольких GPU одновременно
- понимает CUDA/OpenCL kernel'ы
- сравнение с Matlab PCT можно посмотреть тут http://www.accelereyes.com/products/compare
- ориентирован на скорость вычислений

Methematica

- GPGPU начинается с Mathematica 7.0
- высоко-уровневый интерфейс
- набор специальный GPU-функций
- линейная алгебра, обработка изображений, преобразования Фурье могу выполняться на GPU
- взаимодействие с CUDA/OpenCL кодом
- CUDALink, OpenCLLink позволяют использовать соответствующие интерфейсы
- читаем подробности тут http://reference.wolfram.com/mathematica/guide/GPUComputing.html

Кто еще?

- LabVIEW [8]
  - графический язык программирования
  - предоставляет CUDA интерфейс для CUDA-карт
- PETSc [9]
  - решает системы уравнений в частных производных
  - поддерживает выполнение некоторых вычислений на GPU
  - позволяет смешивать MPI/threading, MPI/GPU
- Trilinos [10]
  - позволяет смешивать MPI/threading, MPI/GPU
- OpenFoam - OpenFoam FEM Package
  - http://openfoamspeedit.sourceforge.net

[8] http://www.labview.ru/labview/
[9] http://www.mcs.anl.gov/petsc/
[10] http://trilinos.sandia.gov/

Содержание

NVIDIA

CUSparse

- CUSparse
  - векторно-матричные операции на разреженных данных
- CUBLAS
  - плотная линейная алгебра
- CUFFT
  - быстрые преобразования Фурье
- CUSP [11]
  - высокоуровневый интерфейс
  - операции над разреженными матрицами,
  - решение разреженных систем линейных уравнений

[11] http://code.google.com/p/cusp-library

AMD

- APPML [12]
  - поддерживает BLAS, FFT
  - генерацию случайных чисел
  - написана на OpenCL
  - выросла из ACML-GPU

[12] http://developer.amd.com/libraries/appmathlibs/Pages/default.aspx

CUDPP

- написано на CUDA C
- поддержка prefix sum (scan), segmented scan, parallel reductions
- графы, деревья
- radix sort
- генерация случайных чисел
- векторно-матричные манипуляции с разреженными данными
- читаем тут http://code.google.com/p/cudpp

Thrust

- библиотека C++ template'ов
- похоже на STL (контейнеры, итераторы)
- исключаются ошибки работы с памятью
- алгоритмы
  - сортировки
  - редукция
  - скан,...
- читаем тут http://code.google.com/p/thrust

CUDA math.h, CURAND и NPP

CUDA math.h

- реализация простейшей мат. функциональности от NVIDIA
- C99 совместима + дополнительные функции

CURAND

- генератор случайных чисел
- псевдорандом- и квазирандом- генерация

NPP

- обработка изображений

MAGMA

- LAPACK для сложных систем из CPU+GPUs
- разреженные векторно-матричные вычисления
- вещественная и комплексная арифметика
- float, double
- реализована большая часть курса численных методов
- смотрим тут http://icl.cs.utk.edu/magma/

OpenCurrent

- уравнения в частных производных на структурированных сетках
- обучающие инструменты
- отдельные классы для разных проблем, например уравнения Навье-Стокса
- поддержка нескольких GPU одновременно
- смотрим тут http://code.google.com/p/opencurrent/

Еще

- ViennaCL [13]
- LAMA [14]
- ImpLAtoolbox
  - содержит HiFlow3 - пакет для метода конечных элементов
  - предоставляет интерфейсы x86 multi-core CPUs (OpenMP, IMKL, ATLAS), NVIDIA GPUs (CUDA, CUBLAS), AMD/ATI GPUs(OpenCL)
- FEAST [15]
  - метод конечных элементов
  - механика твердого тела
  - гидродинамика

[13] http://viennacl.sourceforge.net/
[14] http://www.libama.org/
[15] http://www.feast.tu-dortmund.de/

ПРОДОЛЖЕНИЕ СЛЕДУЕТ...