

Programming Environ- ments and Libraries

Dominik Göddeke (TUDo)

Robert Strzodka (MPI), Christian Sigg (NVIDIA)

**38th SPEEDUP Workshop on High-Performance
Computing**

EPF Lausanne, Switzerland, September 7-8, 2009

http://www.speedup.ch/workshops/w38_2009.html

Overview

- **Programming environments**
 - NVIDIA CUDA
 - AMD Stream
 - BrookGPU / Brook+
 - RapidMind Platform
 - PGI Accelerator Compilers
- **Middleware and support libraries**
- **Math libraries**
- **Applications and application-specific libraries**

NVIDIA CUDA

- See presentations by Christian Sigg

AMD Stream

- **Libraries**
 - CORBA, ACML, ...
- **Compilers: Brook+ and RapidMind**
 - Target both GPUs and multicore CPUs
 - OpenCL in public beta (CPU version)
- **Compute Abstraction Layer (CAL)**
 - Formerly known as „close to the metal“
 - Very low level, but direct access to the hardware
- **Hardware: FireStream GPUs, HAL**

AMD Stream CAL

- **Design goals**

- Interact with the processing cores on the lowest level if needed
- Maintain forward compatibility
- Device-specific code generation
- Device management
- Multi-device support, resource management
- Kernel loading and execution (written in AMD IL – intermediate language, assembly-like)
- Small set of C routines and data types, e.g. to download IL code into command processor and to trigger the computation

- **Stream SDK and more information**

- <http://ati.amd.com/technology/streamcomputing/>

BrookGPU, Brook+

- **Initially developed at Stanford University**
 - GPU variant of the Brook stream programming language
 - <http://graphics.stanford.edu/projects/brook>
 - SIGGRAPH 2004 paper by Buck et al.
- **Stream programming on GPUs**
 - Compiler and runtime
 - C with stream extensions
 - Compiles to graphics API under the hood
- **Cross-platform**
 - Windows and Linux
 - Runtime environments for OpenGL and DirectX, ATI and NVIDIA
 - License: GPL (compiler) and BSD (runtime)

Streams and kernels

- **Streams**

- Collection of records requiring similar computation
 - Vertex positions, voxels, FEM cell, ...
- Provide data parallelism

- **Kernels**

- Functions applied to each element in stream
 - Transforms, PDE, ...
 - No dependencies between stream elements
- Encourage high arithmetic intensity
- Map, gather, reduce, (scatter)

BrookGPU, Brook+

- **BrookGPU no longer actively being developed**
- **AMD's Brook+**
 - Added backend and compiler support for IL/CAL
 - Added support for newer hardware features (scatter)
 - Part of the Stream SDK
 - Released to SourceForge,
<http://sourceforge.net/projects/brookplus/>

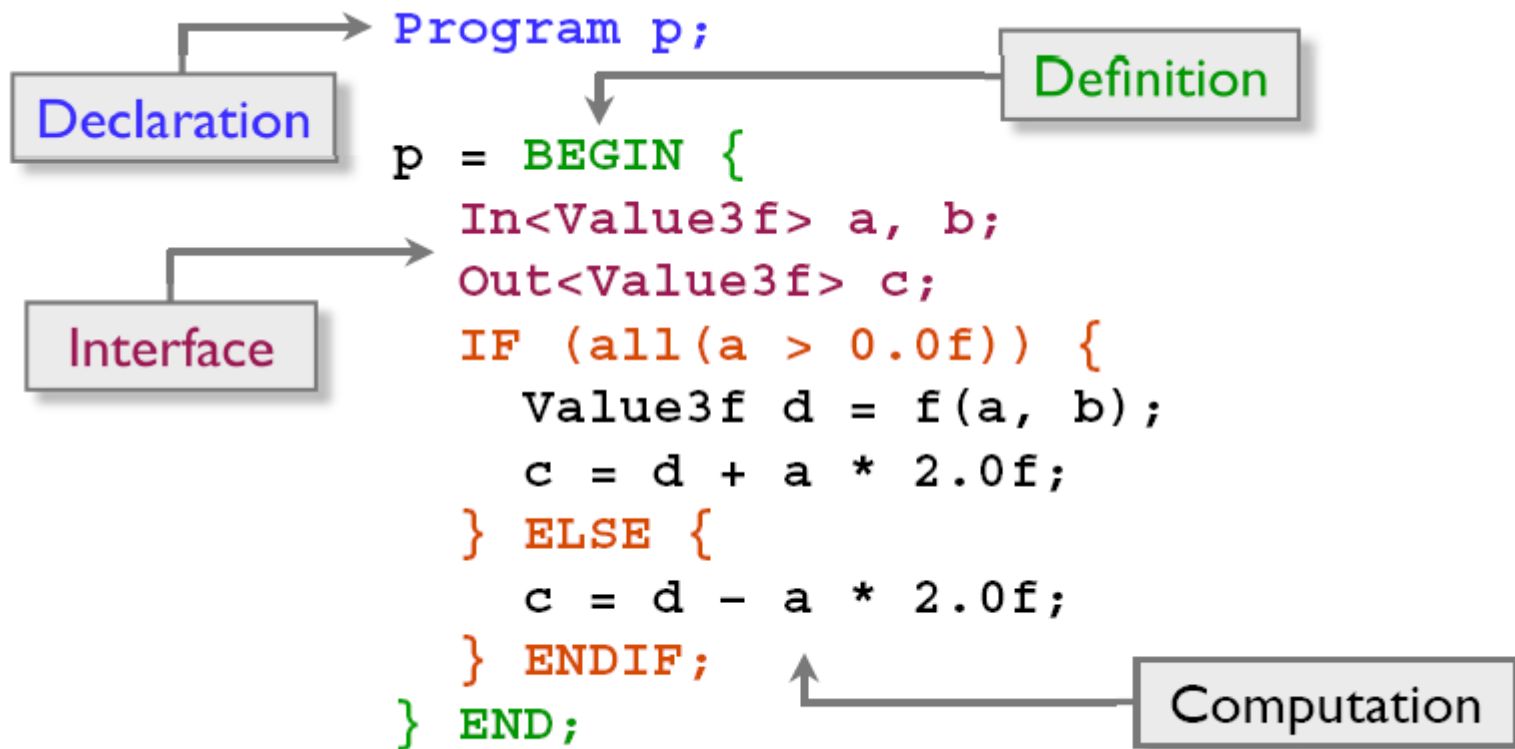
RapidMind Platform overview

- **Software development platform for both multicore and stream processors**
 - Multicore CPUs, Cell BE and GPUs
 - New: OpenCL
 - <http://www.rapidmind.net>
- **Embedded within ISO C++**
 - No changes to toolchain, compilers etc.
- **Portable code**
 - But exposes platform-specific functionality to allow fine-tuning if needed
- **Company acquired by Intel in late August**
 - RapidMind Platform continues to be available

RapidMind Platform features

- **SPMD data parallel programming model**
 - Data parallel arrays
 - Programs return new arrays
 - Programs may have control flow, may perform random reads from other arrays
 - Subarrays, ranges, ...
- **Collective operations**
 - Map, reduce, gather, scatter, ...

RapidMind Platform example



PGI Accelerator compilers

- **Portland Group PGI 9.0**

- Includes an Extended Preview of the PGI Accelerator Fortran and C99 compilers
- Supports x86_64 and CUDA-capable GPUs
- Linux systems
- <http://www.pgroup.com/resources/accel.htm>

- **Approach**

- Add OpenMP-style compiler directives to crucial code sections
- Recompile with appropriate compiler settings
- Compiler splits data and maps nested loops to GPUs

PGI Accelerator compilers

Example: Simple Fortran matrix multiplication

```
!$acc region
  do k = 1,n1
    do i = 1,n3
      c(i,k) = 0.0
      do j = 1,n2
        c(i,k) = c(i,k) + a(i,j) * b(j,k)
      enddo
    enddo
  enddo
!$acc end region
```

Disclaimer

- **All of the following is subjectively biased**
 - Based on news posted to gpgpu.org
 - If you miss your application, library, paper, submit it
 - <http://gpgpu.org/submit-news>
- **GPU computing is pretty widespread now**
 - Google „your field of research + CUDA“



- **Some slides courtesy of**
 - Mike Houston, Mark Harris, Stefanus DuToit, John Owens

Overview

- **Programming environments**
- **Middleware and support libraries**
 - CUDPP
 - Thrust
 - .NET
 - Java
 - Autoparallelization tools
 - Close to the hardware
- **Math libraries**
- **Applications and application-specific libraries**

CUDPP

- **CUDPP: CUDA Data-Parallel Programming Primitives**
 - Includes optimised implementations of parallel
 - reduce,
 - sort (currently fastest parallel sort published)
 - scan (parallel prefix sum)
 - random number generation
 - Cooperation between UC Davis and NVIDIA, BSD license
 - <http://code.google.com/p/cudpp>

```
CUDPPConfiguration config;  
config.op = CUDPP_ADD;  
config.datatype = CUDPP_FLOAT;  
config.algorithm = CUDPP_SCAN;  
config.options = CUDPP_OPTION_FORWARD | CUDPP_OPTION_EXCLUSIVE;  
  
CUDPPHandle scanplan = 0;  
CUDPPResult result = cudppPlan(&scanplan, config, numElements, 1, 0);
```

```
// Run the scan  
cudppScan(scanplan, d_odata, d_idata, numElements);
```

Thrust

- **Thrust: A Template Library for CUDA Applications**

- Transformations, reductions, scans, sort, etc.
- Interface similar to the C++ STL (containers and iterators)
- Developed by NVIDIA, Apache 2.0 license
- <http://thrust.googlecode.com/>

```
#include <thrust/host_vector.h>
#include <thrust/device_vector.h>
#include <thrust/generate.h>
#include <thrust/sort.h>
#include <cstdlib>

int main(void)
{
    // generate random data on the host
    thrust::host_vector<int> h_vec(20);
    thrust::generate(h_vec.begin(), h_vec.end(), rand);

    // transfer to device and sort
    thrust::device_vector<int> d_vec = h_vec;
    thrust::sort(d_vec.begin(), d_vec.end());

    return 0;
}
```

.NET and JAVA

- **CUDA.NET, CAL.NET, OpenCL.NET**

- GASS GPU based Advanced Supercomputing Solutions
- .NET binding for CUDA/CAL/OpenCL applications
- .NET or Mono (Windows and Linux)
- Free-no-questions-asked license (at least for noncommercial use)
- <http://www.gass-ltd.co.il/en/products/>

- **jCUDA**

- By the same company
- JAVA bindings for CUDA

Close to the hardware

- **Barra**

- Sylvain Collange et al., Université de Perpignan
- Modular functional simulator of the G80 architecture
- <http://gpgpu.univ-perp.fr/index.php/Barra>

- **GPUocelot**

- Gregory Damos et al., Georgia Institute of Technology
- Low-level CUDA simulator, Valgrind-like memory checking
- <http://code.google.com/p/gpuocelot/>

- **decuda**

- Wladimir J. van der Laan, University of Groningen
- Disassembler for CUDA
- <http://www.cs.rug.nl/~wladimir/decuda/>

Overview

- **Programming environments**
- **Middleware and support libraries**
- **Math libraries**
 - ACML-GPU
 - CUBLAS and CUFFT
 - MAGMA
 - LAPACK
 - Matlab
 - Sparse Matrix Vector Multiply
 - ODEs, iterative solvers
- **Applications and application-specific libraries**

ACML-GPU

- **Stream-accelerated version of ACML**
- **Windows and Linux, Fortran and C**
- **Supported routines**
 - SGEMM and DGEMM
- **Standard ACML-BLAS interface (just re-link)**
- **Automatically switches between CPU and GPU (small problem sizes)**
- **Automatically scales across several GPUs (large problem sizes)**

CUBLAS and CUFFT

- **Interface modeled on BLAS and FFTW**
- **Three ways to use (similar for CUFFT)**
 - Explicit management of device memory and data transfers
 - `cublasMalloc()`, `cublasDgemm()`, `cublasGetMatrix()`
 - Implicit
 - "Thunking": copy-compute-readback in one call
 - From within standard CUDA device code
- **Fortran and C bindings**
- **Good subset of BLAS level 1, 2 and 3**
- **Requires CUDA-capable device**
 - GT200 and better for double
 - <http://www.nvidia.com/cuda>

MAGMA

- **Dense linear algebra on heterogeneous CPU+GPU systems**
 - GPU backend requires CUDA
- **Idea and goal**
 - „To address the complex challenges of the emerging hybrid environments, optimal software solutions will themselves have to hybridize, combining the strengths of different algorithms within a single framework.“
 - Keep all resources in the system equally busy by automatic scheduling and load balancing
- **Small subset of BLAS2 and 3 implemented, continuously expanded**
- **Innovative Computing Lab (UTK), University of Coimbra (PT), UC Berkeley, University of Denver**
 - <http://icl.cs.utk.edu/magma/index.html>

LAPACK

- **CULA by EM Photonics**
- **Basic edition**
 - LU, QR and singular value decomposition, least squares
 - Limited to single precision
 - Public beta (preview release) available now
- **Premium edition**
 - More functionality, double precision, support
 - To be released at NVIDIA GTC'09 (Sep 30)
- **Commercial edition**
 - Redistribution rights, direct support
- **<http://www.culatools.com/>**

Matlab

- **Jacket**

- <http://www.accelereyes.com>

```
>> G = gones( 3 );    % Create a GPU matrix
>> G = fft( G );     % Perform a GPU FFT
>> G = G * G;        % GPU Matrix Multiply
>> C = double( G );  % Bring back to CPU
```

- **Libra**

- <http://www.gpusystems.com>

- **GPUmat**

- <http://www.gp-you.org>

- **All essentially based on operator overloading**

- Easy to use
- Feature sets vary
- Commercial licenses (trials available)

SpMV

- **Rajesh Bordawekar and Muthu Manikandan Baskaran (IBM)**
 - Linux, CUDA
 - <http://www.alphaworks.ibm.com/tech/spmv4gpu>
- **Nathan Bell and Michael Garland (NVIDIA)**
 - Linux and Windows, CUDA
 - <http://forums.nvidia.com/index.php?showtopic=83825>
 - Upcoming Supercomputing 2009 paper
- **Luc Buatois, Guillaume Caumon and Bruno Lévy (Inria, France)**
 - Block-CSR for symmetric sparse matrices
 - Windows (Linux possible), Stream and CUDA
 - http://alice.loria.fr/publications/papers/2008/CNC/Buatois_et_al_CNC.pdf

Misc. Math libraries

- **CULSODA**

- Livermore LSODA ODE solver for CUDA
- <http://code.google.com/p/culsoda/>

- **Sparse iterative solvers**

- Acceleware (commercial license)
- <http://www.acceleware.com/index.cfm/our-products/linear-algebra-solvers/>

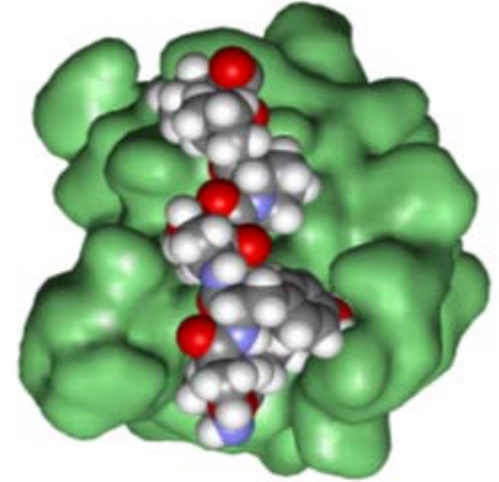
Overview

- **Programming environments**
- **Middleware and support libraries**
- **Math libraries**
- **Applications and application-specific libraries**
 - Molecular dynamics
 - Electromagnetic and acoustic waves
 - Computer vision
 - Computational statistics
 - Computational finance

Applications: MD

- **Folding@Home and OpenMM**

- Vijay Pande et al., Stanford University
- F@H: AMD, CUDA and PS3 clients
- OpenMM: CUDA
- <http://simtk.org/home/openmm>
- <http://folding.stanford.edu>



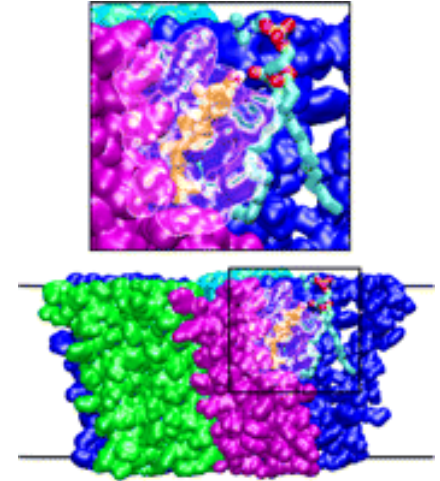
- **HOOMD-blue!**

- Highly Optimized Object-oriented Many-particle Dynamics - Blue Edition!
- Formerly known as hoomd
- Joshua Anderson et al., University of Michigan, Ames Lab
- <http://codeblue.umich.edu/hoomd-blue>

Applications: MD

- **NAMD+VMD**

- University of Illinois at Urbana-Champaign
- Jim Phillips, John Stone et al.
- Executes on GPU clusters (CUDA)
- <http://www.ks.uiuc.edu/Research/namd>



- **ACEMD**

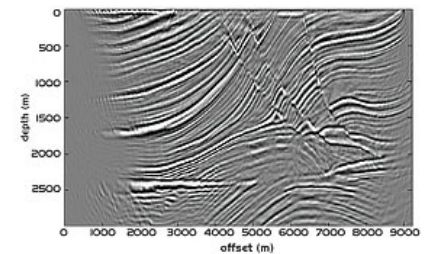
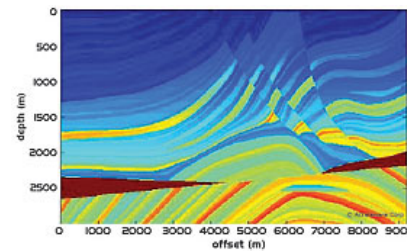
- Universitat Pompeu Fabra (Barcelona), Barcelona Biomedical Research Park, Imperial College London
- CUDA, supports multiple GPUs
- <http://www.multiscalelab.org/acemd>

- **All pretty much open source license**

Applications: Waves

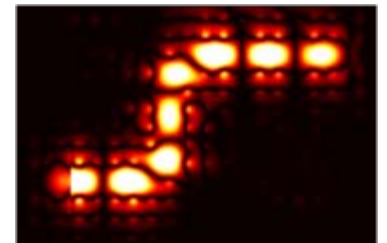
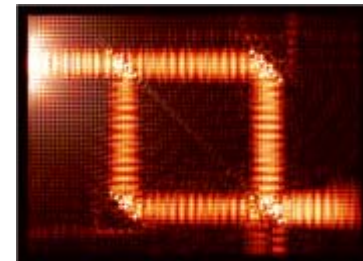
- **Acceleware**

- FDTD electromagnetic solvers
- Reverse time migration for seismic imaging
- Commercial
- <http://www.acceleware.com>



- **EM Photonics**

- FDTD electromagnetic wave propagation
- GPU and custom FPGA solutions
- Commercial
- <http://www.emphotonics.com>



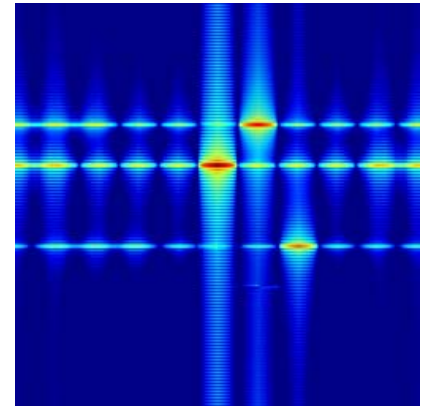
Applications: Computer vision

- **GPU VSIPL**

- GPU implementation of the Vector Signal Image Processing Library
- Supports 565 VSIPL functions
- Georgia Tech Research Institute
- Windows and Linux, binary-only static library, separate redistributable license
- <http://gpu-vsimpl.gtri.gatech.edu/>

- **GpuCV**

- OpenCV-like interface
- Open source, CeCILL-B license (BSD-like)
- <http://picoforge.int-evry.fr/projects/gpucv>



Applications: Statistics and finance

- **Computational statistics**

- R+GPU
- University of Michigan
- Reasonable subset of R functions
- Easy to use, just prefix "gpu" to function name
- GPL
- <http://brainarray.mbni.med.umich.edu/brainarray/rgpgpu/>

- **Computational finance**

- OPLib: Elementary Pricing Functions in CUDA/OpenCL and OpenMP
- Claudio Albanese, LUISS University, Rome
- GPL
- <http://www.level3finance.com/>