Porting Scientific codes to GPUs

- Common characteristics of scientific codes:
  - Usually in Fortran
  - FFT-based simulations involve all-to-all communication
  - High memory requirement

- Incompatibility with GPUs:
  - GPUs have small on-chip memory (~16GB max)
  - Various communication latencies

- Solutions for porting code:
  - Domain decomposition (regular or irregular)
  - Explicit properties of data and convolution kernel
  - Sampling pruning used so that domain results fit on GPU memory

Case study: MASSIF
- Hosts’ law simulation
- Partial Differential Equation solved by Green’s function
- FFT-based convolution and tensor contraction between rank-2 tensors and rank-4 Green’s function

MASSIF simulates Composite microstructure made up of grains

Proposed solution:
- Domain decomposition with grains as domains
- Domain-local FFT followed by convolution and tensor contraction
- Green’s function computed on-the-fly to avoid storage
- Adaptive sampling of dense convolution result to fit problem on GPU memory

Complex data mappings! How to get maximum performance on various platforms?

DOX-2
Source: ORNL

Challenges
- FFTW is de-facto standard interface for FFT
- Vendor libraries support the FFTW 3.x interface: Intel MKL, IBM ESSL, and AMD ACML (and others), Nvidia cuFFT, Cray LibSci/O克拉FT

Some issues:
- No native support for accelerators (GPUs, Xeon Phi, FPGA’s and SMIs)
- Parallel MPI version does not scale beyond 32 nodes
- No analogue to PACK for spectral method

Solution: Emerging interfaces like FFTX, extension of FFTW, enables algorithm specification as composition of sub-plans

Front end: Algorithm Specification

GPU

FFT, tensor contraction and sampling

Octree Sampling reduces storage on GPU

Back end: Code Optimization

FFTX is...
- Modernized FFTW-style interface
- Backwards compatible to FFTW 2.x and 3.x
- Small number of new features, familiar interface

Code generation backend using SPIRAL
- Library/application kernels are interpreted as specifications in DSL extract semantics from source code and lower library semantics
- Compilation and advanced performance optimization cross-over and cross library optimization, accelerator off-loading...
- Reference library implementation and bindings to vendor libraries

Ensemble platform and programming model definitions

Platform-aware formal program synthesis

Model: common abstraction + spaces of matching formulas

Automatically generated FFTW-like library components

DARPA Brass

Translating an OL expression into code

Constraint Solver Input:

C code:

Output =

Parser:

FFTW3.

Other FFT-based simulations

Hockey free-space convolution

NCSC Blue Waters

RUNX K computer

LANL Roadrunner

BlueGene/L

ANL BlueGene/Q

2012 HPC Challenge Class A Award (Most Productive System)

Future Plans + Other Applications

Future work: MASSIF

- Irregular domain decomposition
- Extension of adaptive sampling for irregular domains

Tessellated polytopes

Irregular shapes

Tesselated polytopes

LAPACK for spectral algorithms
- Define FFT as the analogue to BLAS
- Define class of numerical algorithms to be supported by SpectralPACK

PDE solver class defines host function, sparse in normrank space, signal processing

- Define SpectralPACK functions: circular convolutions, NUFFT, Poisson solver, free space convolution

Poisson’s equation in free space

Partial differential equation (PDE) Solution

Poisson’s equation, ψ is the Laplace operator

Solution: $G_\ell = \int \frac{e^{ikr} \phi(r)}{4\pi r} dr$ with Green’s function $G_\ell$

Green’s function kernel in frequency domain

References