Exascale Monte Carlo R&D

Summer 2012

Larry Cox, Ryan Marcus

Overview

- **Exascale computing**
  - Different technologies
  - Getting there
- **High-performance proof-of-concept: MCMini**
  - Monte Carlo neutron transport
  - Features
  - Results
- **OpenCL toolkit: Oatmeal**
  - Purpose
  - Features

Exascale Computing

- **558 times faster than Roadrunner**
- **Long way off**
  - Intel predicts 2018
- **Lots of different technologies**
- **Some commonalities between all technologies**
  - Host / Device paradigm
  - Parallel

Exascale Computing: Technologies

- **NVidia CUDA**
  - The original GPGPU language
  - Proprietary: tied to NVidia devices
- **Khronos OpenCL**
  - A multi-platform massively parallel standard
  - Runs on all current HPC hardware
- **Intel MIC**
  - 4 x86 CPUs packed onto a PCI card
  - Runs current code (Fortran, C) with little modification
- **Advantages and disadvantages to each**

Exascale Computing: Technologies

Quality	CUDA	OpenCL	MIC
Runs everywhere	✗	✓	✗
Open standard	✗	✓	✗
Pre-existing code	✗	✗	✓
Large user base	✓	✓	✗
CPU support	✗	✓	✓

Quality	CUDA	OpenCL	MIC
Backer	NVidia	Apple, AMD, Intel	Intel
API Support	Good	Moderate	Full

Exascale Computing: Technologies

- **Hardware-agnostic code is very important**
  - Write once, run anywhere
  - Changing landscape of HPC means compatibility is important
- **Running current code is very attractive, BUT it won’t scale to the same level**
  - Memory constraints
  - Heavy instruction set
- **Open standard is key: code can’t depend on one company**
- **MCMini’s HPC technology: OpenCL**

MCMini

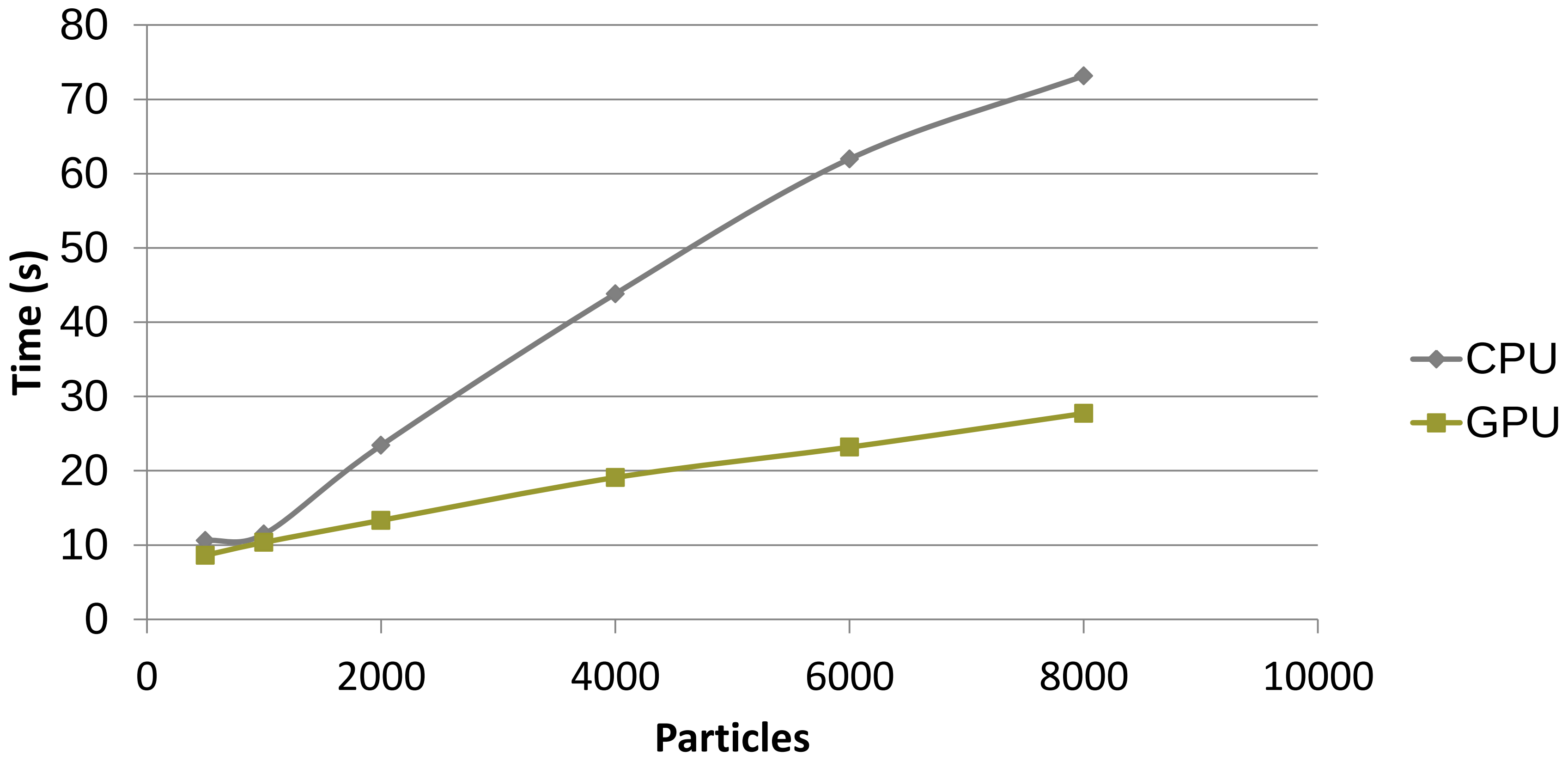
- **A Monte Carlo “mini-app”**
- **Proof of concept**
- **Performance capable and performance driven**
  - Written in C
- **Basic, Newtonian physics**
  - Algorithmically similar to “real” physics
- **Support for three reaction types: scatter, fission, and absorption**
- **Reads LNK3DNT meshes (MCNP geometry compatible)**
- **Multiple OpenCL devices (CPU, GPU)**
- **Multiple nodes**

MCMini: Capabilities

- **MCMini can handle a lot of particles on really big geometries**
- **Trace and attenuation code is result-similar to MCNP**
- **Very capable (4th generation particles calculated)**

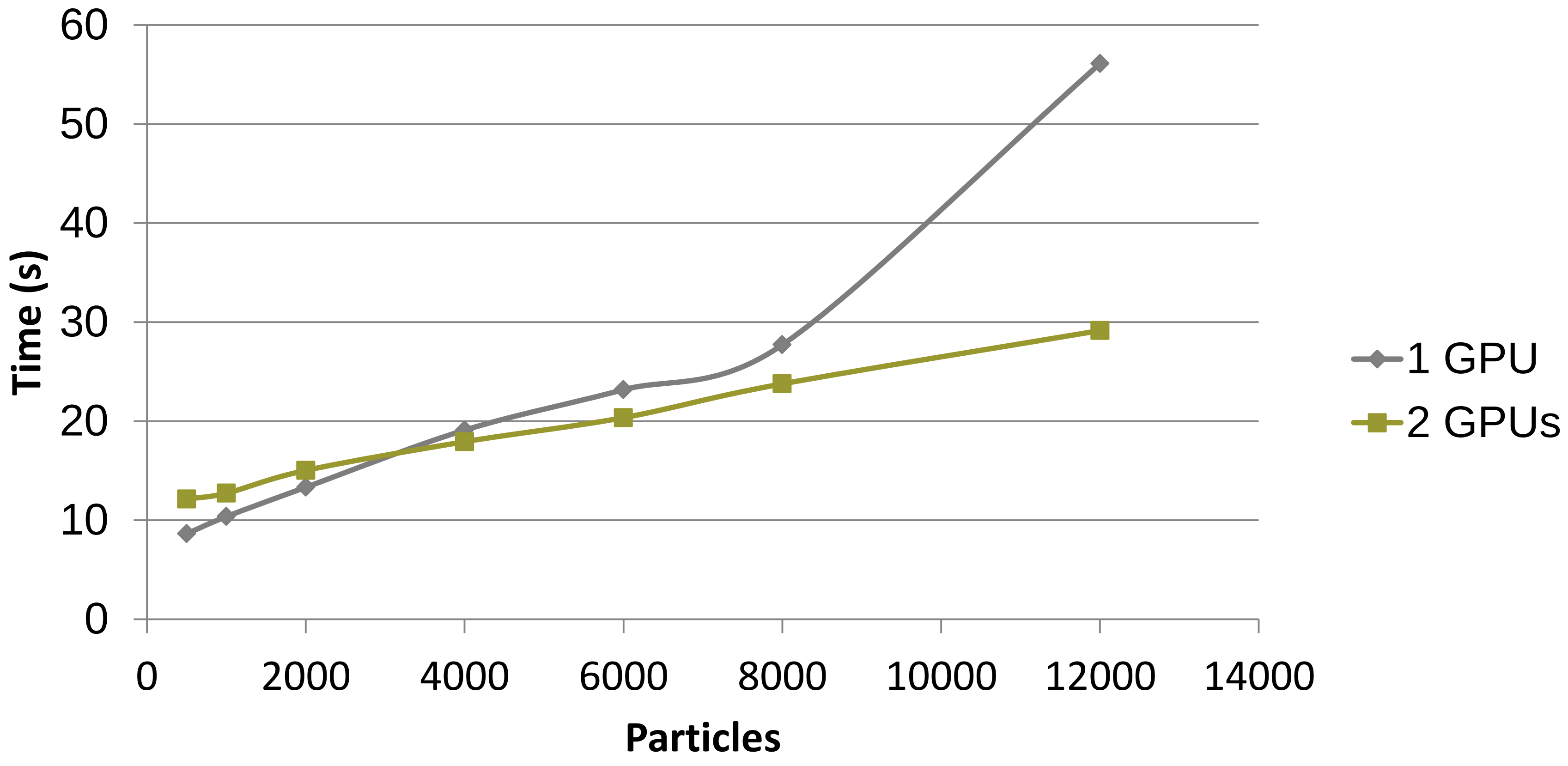
Nodes	GPUs	Mesh cells	Particles	Time (m)
1	2	1000^3	10^5	0.5
2	4	2000^3	10^5	1.02
4	8	3000^3	10^5	2.8
8	16	10000^3	10^5	6.0

MCMini Scaling: CPU vs GPU (Cypress)

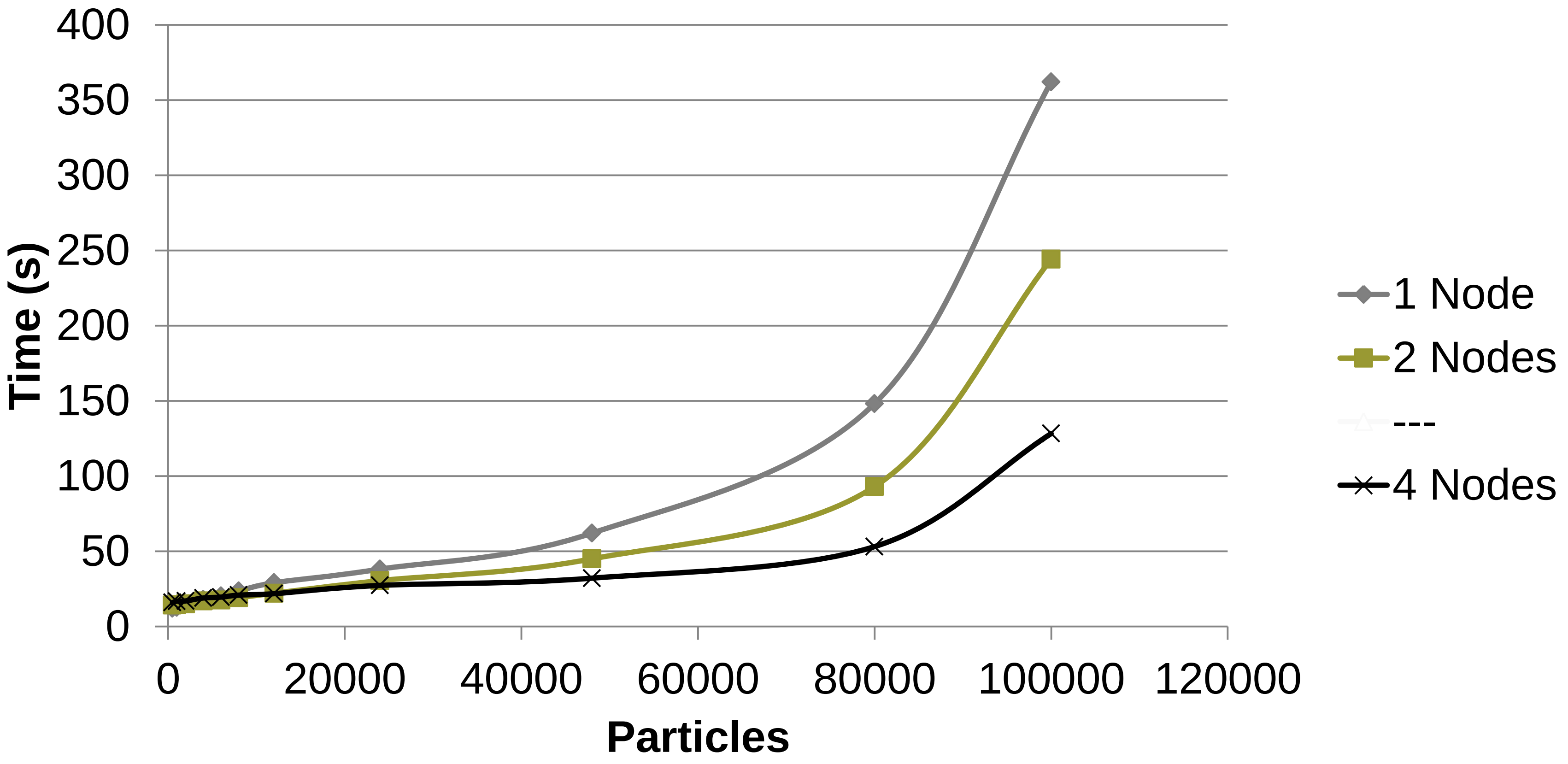


UNCLASSIFIED

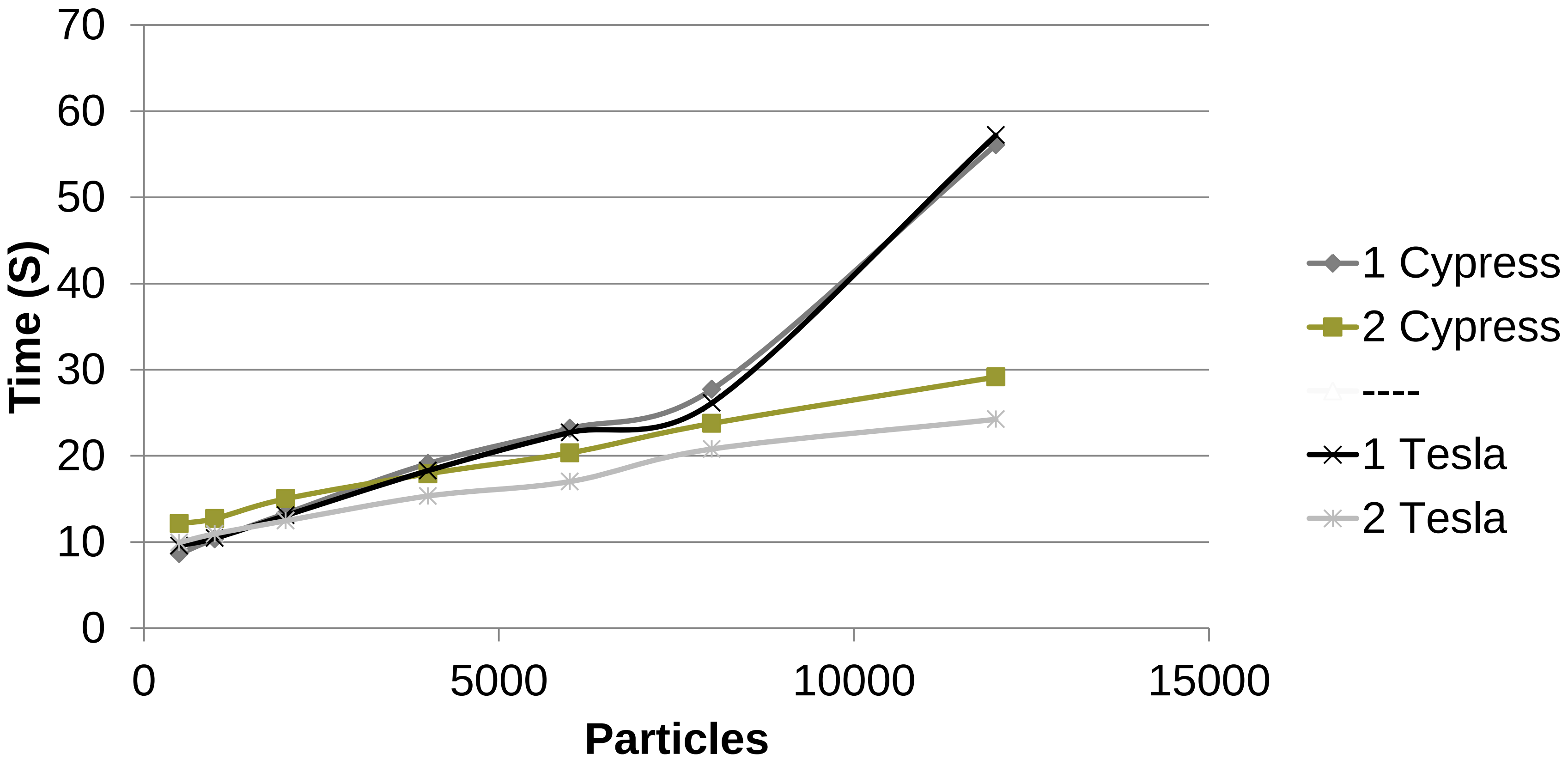
MCMini Scaling: 1 GPU vs 2 GPUs (Cypress)



MCMini Scaling: Nodes



MCMini Scaling: Cypress vs Tesla (C2070)



MCMini / MCNP comparison

- **Neutron flux calculations**
  - “Comparable”
  - Similarly scaling
- **Time comparison is inherently unfair**
  - MCNP is doing far more than MCMini
- **Geometries that take a long time in MCNP generally requires minutes or seconds in MCMini**

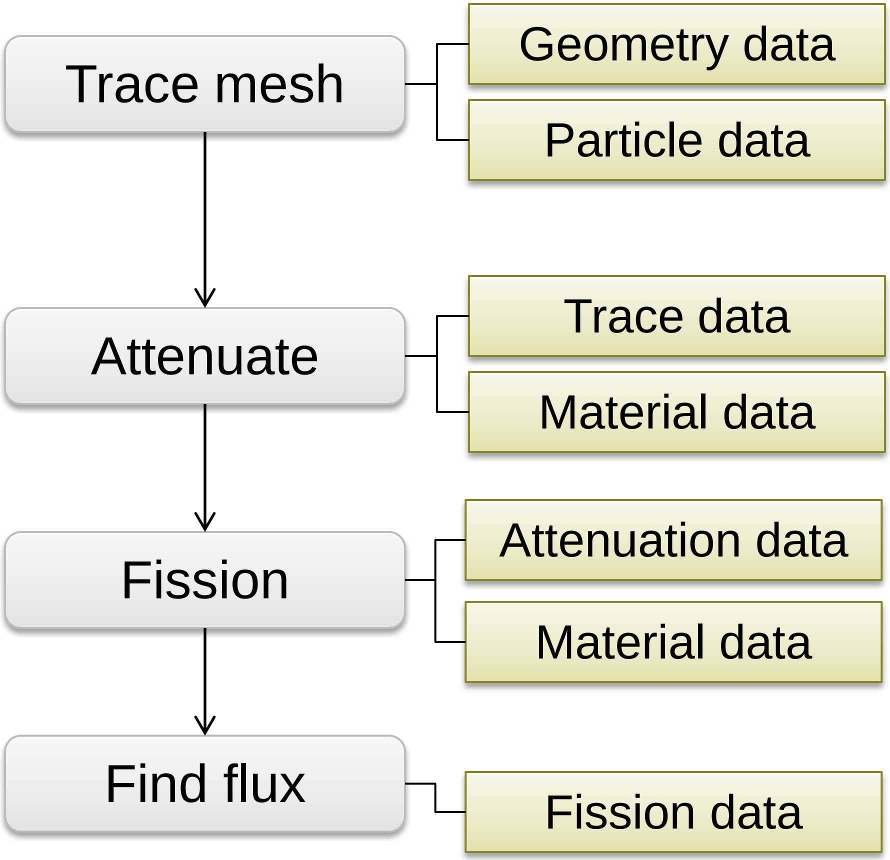
MCMini: Results

- **OpenCL is a viable HPC tool**
  - Scaling factors
  - General performance
- **OpenCL code can be easily designed to run on both CPUs and GPUs**
- **Monte Carlo style tasks are very well fitted to OpenCL and GPGPU computing**
- **Host/device paradigm introduces some interesting complications**

Oatmeal: Purpose

- **Traditionally, you either have enough RAM, or you don’t.**
- **With GPGPUs**
  - Large host memory pool (500GB on some Darwin nodes)
  - Limited device memory pool (1 – 5 GB)
  - Device can’t do computations on data not in device memory
- **Memory can be moved around between the host and the device**
  - Costly
- **OpenCL code is broken down into kernels**
  - Minimal task unit: the smallest logical division of a task

Oatmeal: Purpose



- **Steps require different data**
  - Data required by a step is that step's "context"
- **Context switching can be expensive**
  - Transferring data takes time
- **We want to minimize data transfer**
  - But we can't run out of memory
- **We could maintain a minimum context**
- **Or we can calculate an optimal context**

Oatmeal: Features

- Oatmeal is the OpenCL AuTomatic Memory Allocation Library
- A C++ framework for calculating optimal context switches
- Takes in a dynamic graph, executes the program
- Can run kernels over multiple devices and automatically reduce data
- Can run host-based tasks in parallel
- Can be used to profile memory usage and identify hotspots
  - Both memory and computational
- Can determine if bandwidth costs exceed computational benefit
  - And then run code on the CPU

Conclusions

- **Despite driver issues, OpenCL seems like a good, hardware agnostic tool**
- **MCMini demonstrates the possibility for GPGPU-based Monte Carlo methods**
  - Shows great scaling for HPC application
  - Algorithmic equivalence
- **Oatmeal provides a flexible framework to aid in the development of scientific OpenCL codes**