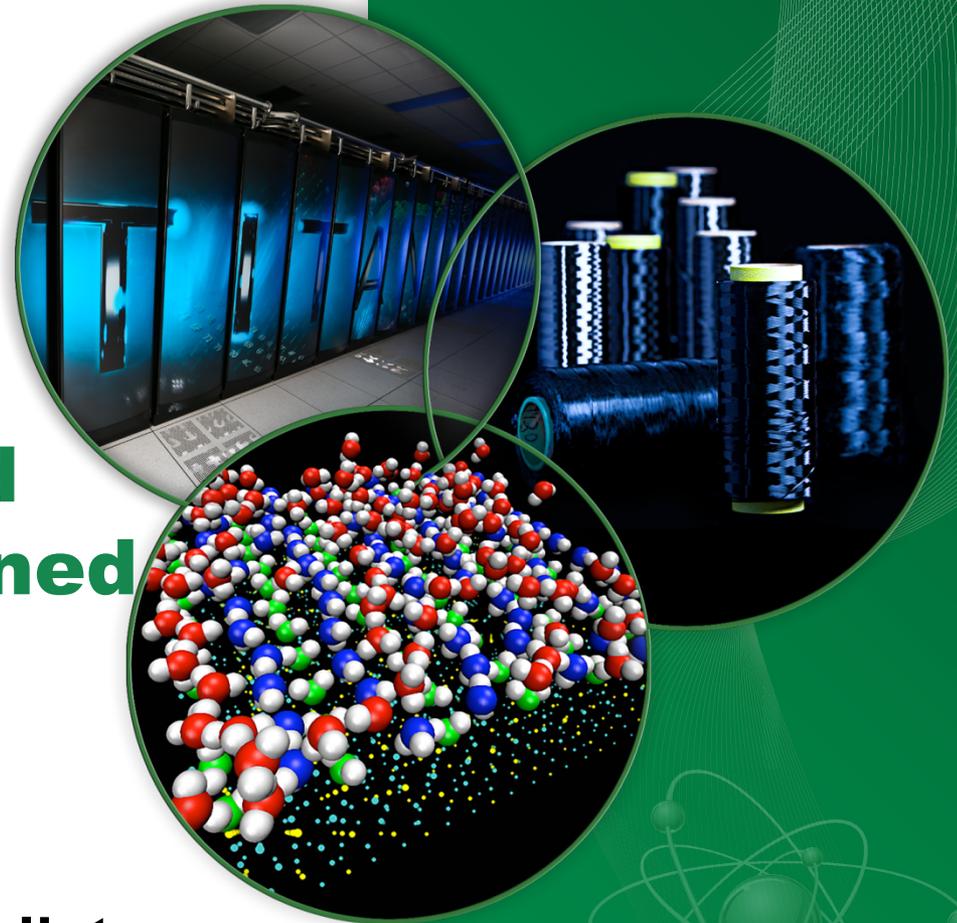


GPU Hackathons and CAAR: Lessons Learned

Fernanda Foertter

HPC User Assistance Specialist

ORNL is managed by UT-Battelle
for the US Department of Energy



 **OAK RIDGE**
National Laboratory

Two Tracks for Future Large Systems

Many Core

- 10's of thousands of nodes with millions of cores
- Homogeneous cores
- Multiple levels of memory – on package, DDR, and non-volatile
- Unlike prior generations, future products are likely to be self hosted

Hybrid Multi-Core

- CPU / GPU Hybrid systems
- Likely to have multiple CPUs and GPUs per node
- Small number of very fat nodes
- Expect data movement issues to be much easier than previous systems – coherent shared memory within a node
- Multiple levels of memory – on package, DDR, and non-volatile



Tianhe-2 (NUDT): TH-IVB-FEP
Intel Xeon E5-2692 12C 2.2 GHz
TH Express-2
Intel Xeon Phi 3151P



Titan (Cray): Cray XK7
AMD Opteron 6274 16C 2.2 GHz
Cray Gemini
NVIDIA K20x



Sequoia (IBM): BlueGene/Q
Power BQC 16C 1.6 GHz



K computer (Fujitsu)
SPARC64 VIII fx 2.0 GHz
Tofu



Mira (IBM): BlueGene/Q
PowerPC A2 16C 1.6 GHz



Piz Daint (Cray): Cray XC30
Intel Xeon E5-2670 8C 2.6 GHz
Cray Aries
NVIDIA K20x



Edison (Cray): Cray XC30
Intel Xeon E5-2695v2 12C 2.4 GHz
Aries

Cori at NERSC

- Self-hosted many-core system
- Intel/Cray
- 9300 single-socket nodes
- Intel® Xeon Phi™ Knights Landing (KNL)
- 16GB HBM, 64-128 GB DDR4
- Cray Aries Interconnect
- 28 PB Lustre file system @ 430 GB/s
- Target delivery date: June, 2016

Summit at OLCF

- Hybrid CPU/GPU system
- IBM/NVIDIA
- 3400 multi-socket nodes
- POWER9/Volta
- More than 512 GB coherent memory per node
- Mellanox EDR Interconnect
- Target delivery date: 2017

ALCF-3 at ALCF

- TBA
- Target delivery date: 2017-18

Accelerating your application

3 WAYS TO ACCELERATE APPLICATIONS

Applications

Libraries

“Drop-in”
Acceleration

Compiler
Directives

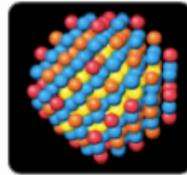
Easily Accelerate
Applications

Programming
Languages

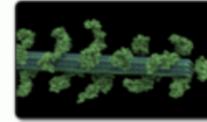
Maximum
Flexibility

Titan and CAAR

Center for Accelerated Application Readiness (CAAR)



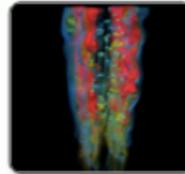
Material Science (WL-LSMS) Illuminating the role of material disorder, statistics, and fluctuations in nanoscale materials and systems.



Molecular (LAMMPS) A molecular description of soft materials, with applications in biotechnology, medicine and energy.

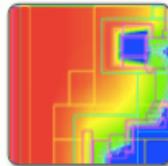
Combustion (S3D)

Understanding turbulent combustion through direct numerical simulation with complex chemistry.

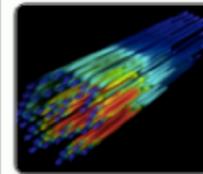


Climate Change (CAM-SE)

Answering questions about specific climate change adaptation and mitigation scenarios; realistically represent features like precipitation patterns / statistics and tropical storms.



Astrophysics (NRDF) Radiation transport – important in astrophysics, laser fusion, combustion, atmospheric dynamics, and medical imaging – computed on AMR grids.



Nuclear Energy (Denovo) Discrete ordinates radiation transport calculations that can be used in a variety of nuclear energy and technology applications.



CAAR Plan

- **Comprehensive team assigned to each app**
 - OLCF application lead
 - Cray engineer
 - NVIDIA developer
 - Other: other application developers, local tool/library developers, computational scientists
- **Single early-science problem targeted for each app**
 - Success on this problem is ultimate metric for success
- **Particular plan-of-attack different for each app**
 - WL-LSMS – dependent on accelerated ZGEMM
 - CAM-SE– pervasive and widespread custom acceleration required
- **Multiple acceleration methods explored**
 - WL-LSMS – CULA, MAGMA, custom ZGEMM
 - CAM-SE– CUDA, directives
 - Two-fold aim

CAAR Lessons Learned

CAAR: SElected Lessons Learned

- We estimate possibly 70-80% of developer time is spent in code restructuring, regardless of whether using CUDA / OpenCL / OpenACC / ...
- More available flops on the node should lead us to think of new science opportunities enabled—e.g., more DOF per grid cell
- We may need to look in unconventional places to get another ~30X thread parallelism that may be needed for exascale—e.g., parallelism in time

Enter Hackathons...



 **OAK RIDGE** | OAK RIDGE
National Laboratory | LEADERSHIP
COMPUTING FACILITY

October 2014

 **NCSA**

April 2015



 **OAK RIDGE**
National Laboratory

Hackathon Overview

- Focuses on GPU
- Call for participation and review of apps
 - Teams of at least 2+ members (goal: 4-6)
 - Each team gets 2 mentors
 - Plan is for at least 4 teams.
 - 6 weeks of preparation
 - 5 days of live coding
- Goals
 - Learn to port applications to GPU
 - Port suffers no performance penalty
- Ancillary goals
 - New users, better apps, requirements gathering
 - quick on ramp

Schedule

- Prep
 - 6 week “familiarization” step
 - Goal: mentors get app, compile, profile
 - Identify candidate areas for porting
- Hackathon Week
 - Day 1: Login, profile app, get intro to OpenACC
 - Day 2-4: hack, scrum, hack, scrum
 - Day 5: Present on journey
- Post Hackathon
 - Will check on progress 8 weeks post event
 - Present findings at SC, GTC, CUG, ISC etc

Hackathon Findings: Overall

- Overwhelmingly positive
- “Days instead of months”
- 3.5 applications succeeded
 - 1 Team scaled to 8K nodes w/ 2.7x speed-up overall
 - 1 Team got a 20x speed up on a single intense function, ~2x speedup overall
 - 1 Team used a CUDA based library on 0.5 code, created a whole new app
 - 1 Team 50% improvement from original mini-app
- One team saw performance degradation
- One team had partial port, while other code couldn't inline

Findings: Compilers

- “Porting is straight forward, but time was spent on compilers.”
- Switching between compilers, more painful
- Compiling C++ very painful, plus more pain.

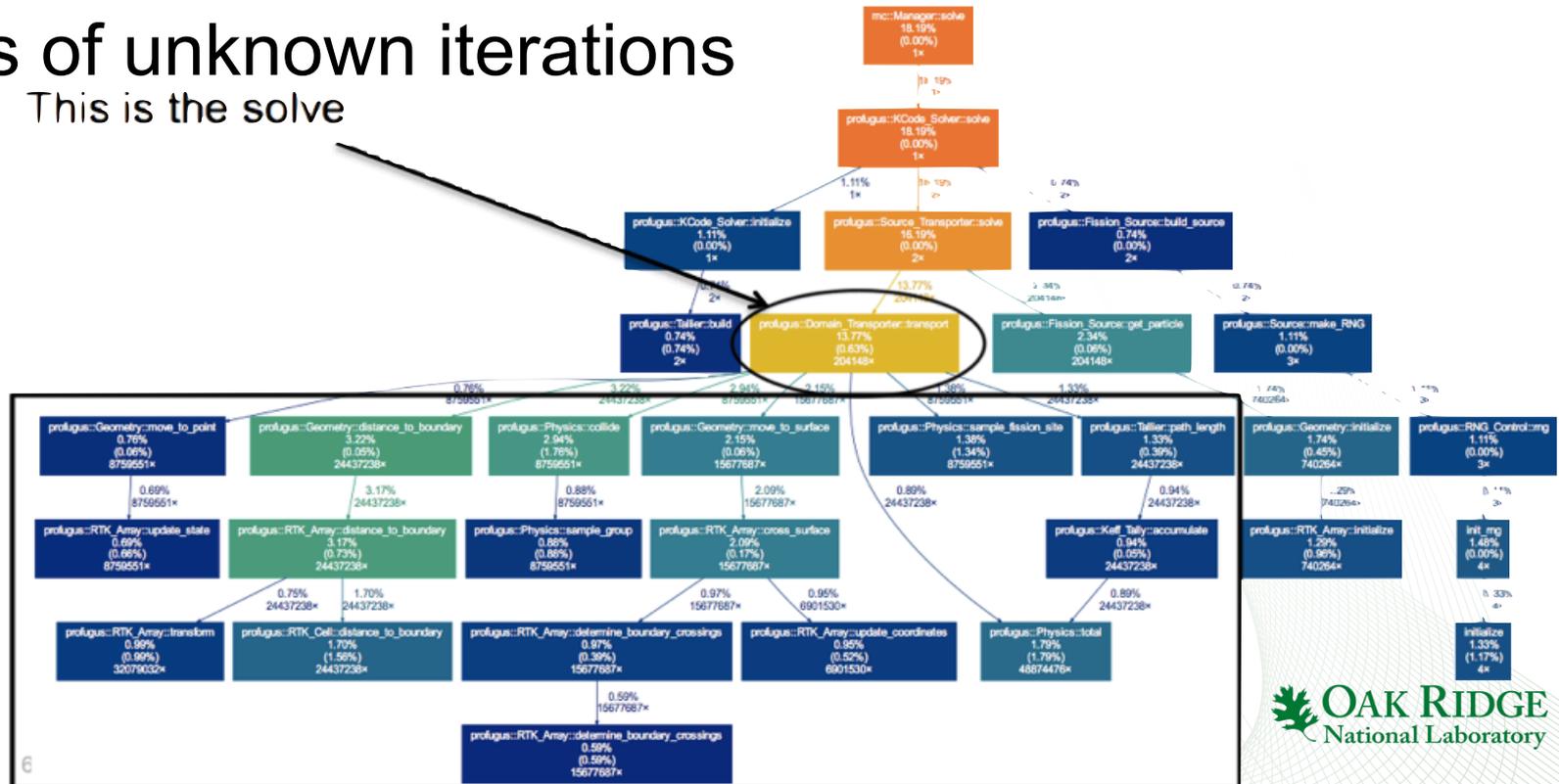
- But... lots of bug fixes! Made second hackathon a joy

- Next time: Start with code already running with compiler, correct versions of CUDA/Driver installed

Findings: Algorithms

- Deeply nested calls are an issue
 - Difficult to use with STL containers, math libs
 - Flattened calls
- Inlining failed after depth=2
- Loops of unknown iterations

This is the solve



Findings: Data

- Overwhelmingly, what prevented speedup was data movement , poor data locality.
- Data collisions from other threads were prevalent
- Deeply nested data types were hard. Dummy containers created as workaround.
- Converting 3D arrays to 1D

Findings: Other

- Not enough work
- Reorganizing so data lifetime is long enough.
- Not filling GPU with thousands of threads
- Reshaping of data required. Partially present errors
- Reshaping of function calls. Top down vs bottom up
- C++ --> C-like was a common theme
- Teams created “blackbox” CUDA based routines or used libs (AmdX)

- Users are driving the standard!

Future-proof your app...

- Use multiple compilers
- Production facilities favor stability over frequent updates
 - Older Drivers
 - Infrequent OS upgrades (affects supported standards)
 - Compiler versions
- Offer makefiles that are system specific
- Favor drop-in library dependencies over proprietary APIs
- Lifetime of machines increasing... average ~5+ years

OLCF Summit Key Software Components

- **System**

- Linux®
- IBM Elastic Storage (GPFS™)
- IBM Platform Computing™ (LSF)
- IBM Platform Cluster Manager™ (xCAT)



- **Programming Environment**

- Compilers supporting OpenMP, OpenACC, CUDA
 - IBM XL, PGI, LLVM, GNU, NVIDIA
- Libraries
 - IBM Engineering and Scientific Subroutine Library (ESSL)
 - FFTW, ScaLAPACK, PETSc, Trilinos, BLAS-1,-2,-3, NVBLAS
 - cuFFT, cuSPARSE, cuRAND, NPP, Thrust
- Debugging
 - Allinea DDT, IBM Parallel Environment Runtime Edition (pdb)
 - Cuda-gdb, Cuda-memcheck, valgrind, memcheck, helgrind, stacktrace
- Profiling
 - IBM Parallel Environment Developer Edition (HPC Toolkit)
 - VAMPIR, Tau, Open|Speedshop, nvprof, gprof, Rice HPCToolkit

Future

bit.ly/2015GPUHack

- More hackathons!
- More compilers!
- More compiler implementers!
- More teams!
- Bigger, better, greater!

GPU

[Hackathon]

April 20 - 24



July 6 - 10



CSCS

Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

October 19 - 23



OAK RIDGE
National Laboratory

OAK RIDGE
LEADERSHIP
COMPUTING FACILITY

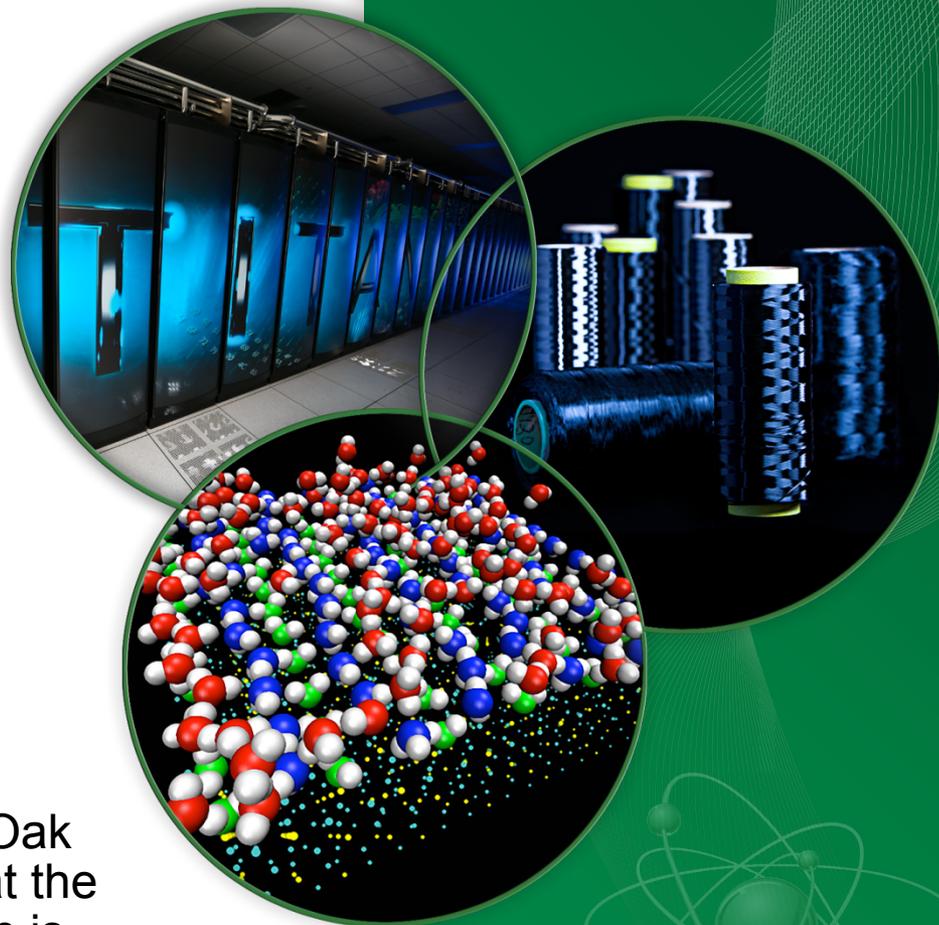
Congratulations!



foertterfs@ornl.gov

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

ORNL is managed by UT-Battelle
for the US Department of Energy



 OAK RIDGE
National Laboratory

Teams

Team	Application	Domain	Members	Institutions	OLCF
APPIGO	CICE, HYCOM, Wavewatch	Climate	4	LANL, Navy RL, UMiami	
NekACC	Nek500	CFD	2	UEdinburgh, KTH	Yes
Deep Learning	Internal + OS Package	NN, ML	6	ORNL, CDAG	Yes
Exnihilo	Exnihilo	Rad Transport	6	ORNL, NSED, UWisc	Yes
NekCEM	NekCEM	Comp E&M	3	ANL, KTH (Sweden)	Yes
iPIC3D	iPIC3D	Astrophysics	5	KU Leuven (Belgium)	

- Selection

- 17 teams applied
- 6 Teams selected
- Multiple apps
- Some have partial GPU
- Fortran, C, C++

- Other domains

- Encryption
- Materials
- Quantum Computing
- Reactor Simulation
- Benchmarking
- Urban Dynamics
- GIS/SAT
- Genomics

Thoughts on C++

- Treat library/user defined operators like first class. I.e. allow reductions using these functions.
- Important for `std::complex<>`
- Conditional deep copy: a class might contain data that references data owned by a different class:
 - Is that data already on the device? If not what data should be copied? The class that owns the data? But how does the compiler know?
- Can we support iterators? (At least a subset, like random access iterators...)

Case Study: Deep Learning

- Day 1: Start compiling and profiling
- Day 2: Integration of GA with Caffe on laptop
- Day 3: Successfully running and evolving neural network parameters
- Day 4: Successfully scaling to 64 nodes and identifying scaling issues with Caffe; profiling showed 80% of time spent running Caffe
- Caffe only compiled with GCC, prevented OpenACC implementation in GA wrapper.
- But, not enough work in GA to use OpenACC

Case Study: iPIC3D

- Array of pointers didn't work. Needed to trick compiler with dummy dimensions

```
type ***arr = new type**[sz1];
```

```
#pragma acc enter data create(arr[0:sz1])
```

```
#pragma acc enter data create(arr[0:sz1][0:1][0:1])
```