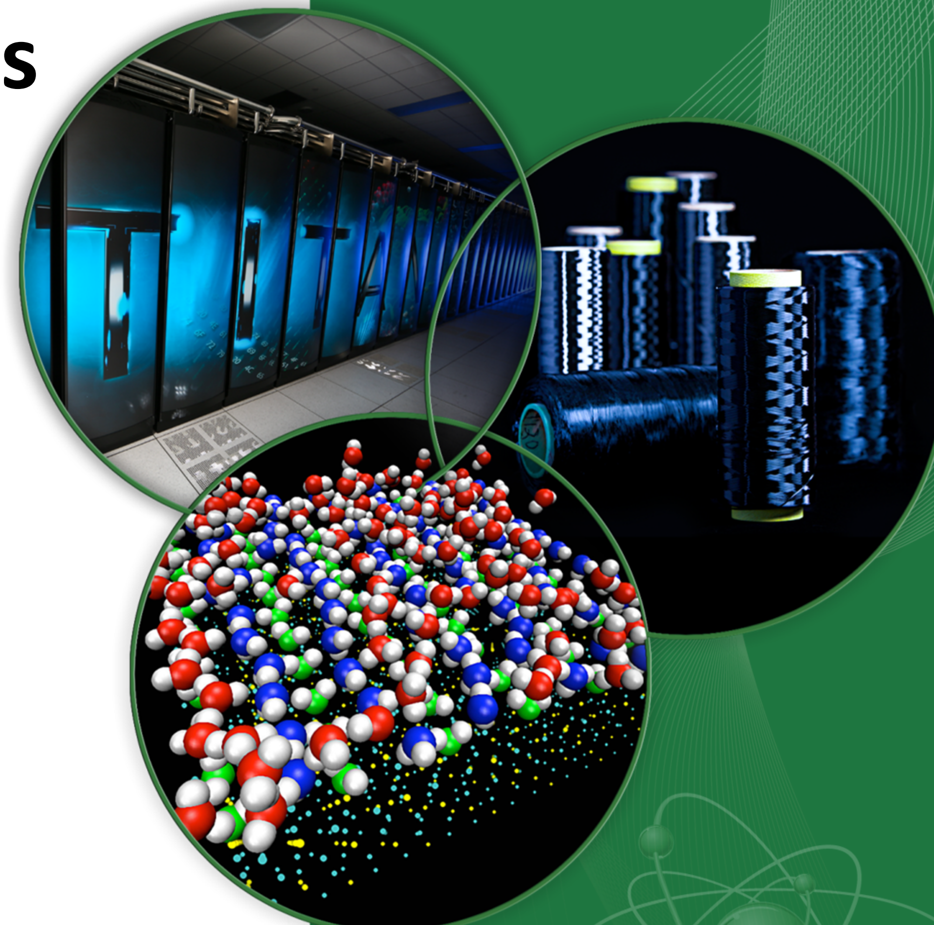
Center for Accelerated Application Readiness

Getting Applications Ready for the Next LCF Systems



Tjerk Straatsma

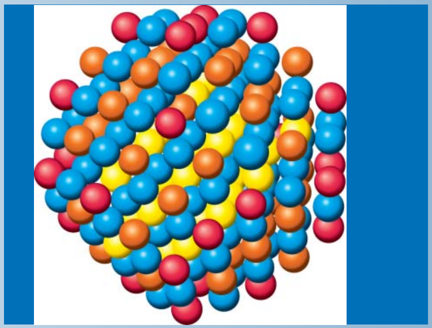
OLCF Scientific Computing Group

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

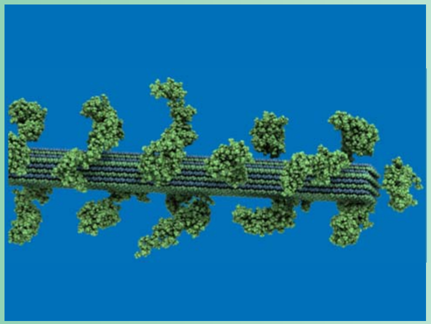
OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

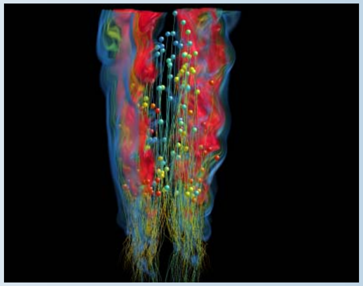
CAAR Applications in the OLCF-3 project (Titan)

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



LAMMPS
A molecular description of membrane fusion, one of the most common ways for molecules to enter or exit living cells.

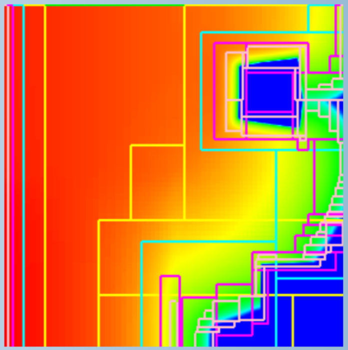


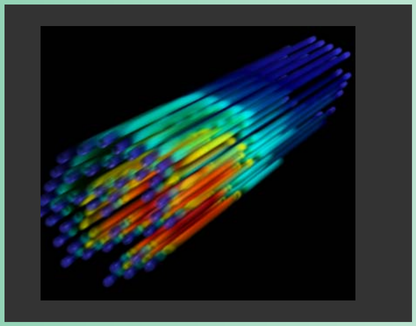


S3D
Understanding turbulent combustion through direct numerical simulation with complex chemistry.
.

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

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





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

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Best Practices

- Repeated themes in the code porting work:
  - finding more threadable work for the GPU
  - Improving memory access patterns
  - making GPU work (kernel calls) more coarse-grained if possible
  - making data on the GPU more persistent
  - overlapping data transfers with other work
- Helpful to use as much asynchronicity as possible, to extract performance (CPU, GPU, MPI, PCIe-2)
- Codes with unoptimized MPI communications may need prior work in order to improve performance before GPU speed improvements can be realized
- Some codes need to use multiple MPI tasks per node to access the GPU (e.g., via proxy)—others use 1 MPI task with OpenMP threads on the node
- Code changes that have global impact on the code are difficult to manage, e.g., data structure changes. An abstraction layer may help, e.g., C++ objects/templates
- Two common code modifications are:
  - Permuting loops to improve locality of memory reference
  - Fusing loops for coarser granularity of GPU kernel calls

Best Practices

- The difficulty level of the GPU port was in part determined by:
  - Structure of the algorithms—e.g., available parallelism, high computational intensity
  - Code execution profile—flat or hot spots
  - The code size (LOC)
- Since not all future code changes can be anticipated, it is difficult to avoid significant code revision for such an effort
- Tools (compilers, debuggers, profilers) were lacking early on in the project but are becoming more available and are improving in quality
- Debugging and profiling tools were useful in some cases (Allinea DT, CrayPat, Vampir, CUDA profiler)

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Best Practices

- Up to 1-3 person-years required to port each code
  - Takes work, but an unavoidable step required for exascale
  - Also pays off for other systems—the ported codes often run significantly faster CPU-only
- We estimate possibly 70-80% of developer time is spent in code restructuring, regardless of whether using CUDA / OpenCL / OpenACC / …
- Each code team must make its own choice of using CUDA vs. OpenCL vs. OpenACC, based on the specific case—may be different conclusion for each code
- Science codes are under active development—porting to GPU can be pursuing a "moving target," challenging to manage
- More available flops on the node should lead us to think of new science opportunities enabled
- 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

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

CAAR Selection Criteria – Personnel & Resources

Task	Description
Application Development Team	• Composition and commitment from development team • Integrate with active development • Process for including ported applications in distribution
Scientific Computing Group	• Adequate OLCF liaison skills and experience
Center of Excellence	• Engagement from Vendor Center of Excellence
Resources	• Access to current supercomputers • Access to early hardware

CAAR Selection Criteria - Technical

Task	Description
Science & Engineering	• Science results, impact, timeliness • Alignment with DOE and U.S. science mission • Application has broad user base • Broad coverage of science domains
Implementation (models, algorithms, software)	• Broad coverage of relevant programming models, environment, languages, implementations • Broad coverage of relevant algorithms and data structures (motifs) • Broad coverage of scientific library requirements
Performance Improvement Plan	• Measure of success is INCITE Computational Readiness • Performance benchmarks • Targets and clear plan for development • Plan for fault resiliency • Plan for good software practices
Portability Strategy	• Applications need to be portable to other architectures • Avoid duplicate technical work • Next systems are pre-exascale

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

Action plan CAAR application porting

1. Multidisciplinary partnership for each code – Code development team, OLCF application lead, Vendor Center of Excellence, cross-cutting support from tool and library developers
2. Resource Allocations
   a. OLCF and other DOE/SC facilities
   b. Access to early testbed hardware
3. Development of common parallelization approach – Code and performance portability, Avoidance of duplicate efforts
4. Common training of all Application Readiness teams
5. Application Development
   a. Code analysis & benchmarking to understand application characteristics: code structure, code suitability for architecture port, algorithm structure, data structures and data movement patterns, code execution characteristics ("hot spots" or "flat" profile)
   b. Develop parallelization approach to determine the algorithm and code components to port, how to map algorithmic parallelism to architectural features, how to manage data motion
   c. Decide on programming model such as compiler directives, libraries, explicit coding models
   d. Address code development issues – rewrite vs. refactor, managing portability, managing inclusion in main code repository
6. Development of Early Science Project, i.e., challenging science problem that demonstrates the performance and scientific impact of the developed application port

Tentative time-line CAAR application porting

1. Oct/Nov 2014: Call for CAAR applications
2. Feb/Mar 2015: Selection of CAAR application teams
3. Mar/Apr 2015: CAAR application training event
4. Apr/May 2015: CAAR application teams start
5. Jun 2016: CAAR review and Go/No-Go
6. Sep 2017: Call for Early Science projects
7. Oct/Nov 2017: Selection Early Science projects
8. Dec 2017: Early Science projects start
9. Jun 2018 – Jun 2019: Early Science project period

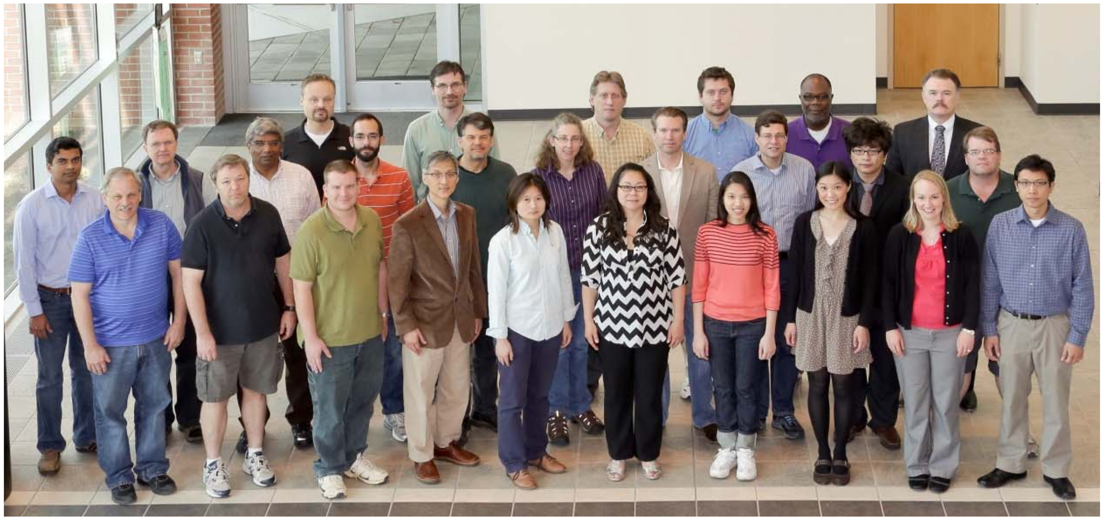
OLCF Scientific Computing Group

<u>**22 Staff and Postdoctoral Fellows**</u>

- **Computational Science Expertise**
  - Astrophysics
  - Biophysics
  - Chemical Physics
  - Climate Sciences
  - Combustion
  - Earth Sciences
  - Fluid Dynamics
  - Materials Science
  - Mathematics
  - Mechanical Engineering
  - Nuclear Physics
  - Turbulence

- **Visualization Expertise**
  - Data Analytics & Visualization
  - EVEREST Visualization Laboratory

- **Data Analytics Expertise**
  - ADIOS I/O
  - Data management and workflow



Questions & Discussion