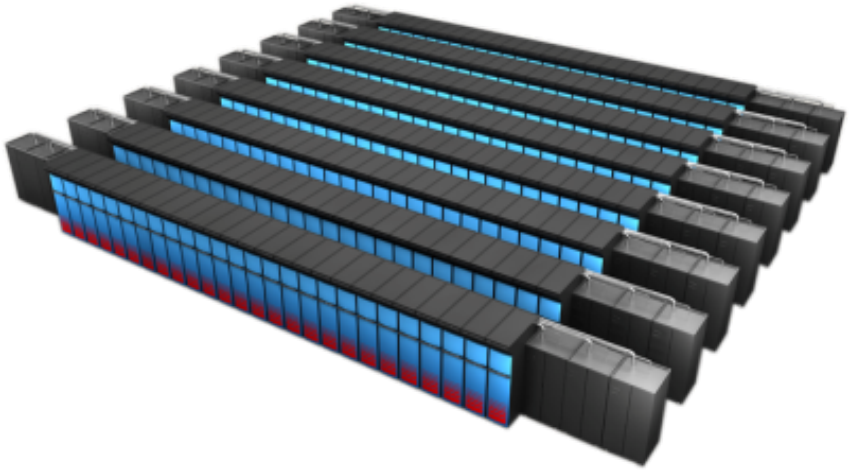
TITAN Application Readiness at ORNL



20 Years of Excellence in Computational Science

OLCF

OAK RIDGE LEADERSHIP COMPUTING FACILITY

1992–2012

Wayne Joubert

OLCF Center for Accelerated Application Readiness (CAAR)

Titan Users and Developers Workshop

Feb. 19-21 2013

Background

- We began planning for Titan in 2009
- At the time there were no large-scale GPU systems deployed anywhere
- Furthermore, OLCF had little previous institutional knowledge of GPUs other than scattered individuals
- However, the consensus was that codes will require restructuring for memory locality, threading and heterogeneity to get to exascale—we decided to do it now

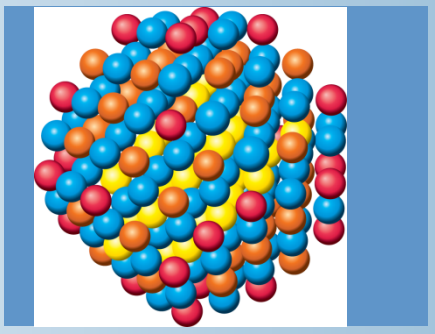
- Additionally, we didn't want a machine delivered that had no functioning application software
- Therefore we selected a small set of applications for early porting, to spearhead an effort to move codes to Titan
- We went through a process to selected a diverse set of codes to give broad coverage to represent use cases for our users
- At the same time we wanted to capture institutional knowledge as lessons learned for going forward

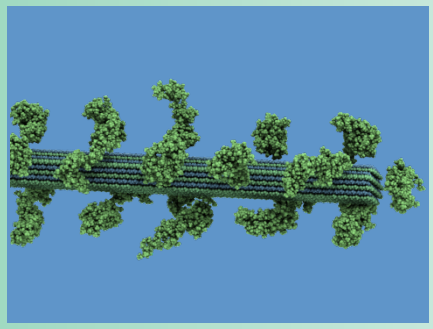
Criteria for selecting early readiness applications

Task	Description
Science	• Science results, impact, timeliness • Alignment with DOE and U.S. science mission • 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
User community (current and anticipated)	• Broad institutional and developer/user involvement • Good representation of current and anticipated INCITE workload
Preparation for steady state (“INCITE ready”) operations	• Mix of low (“straightforward”) and high (“hard”) risk porting and readiness requirements • Availability of OLCF liaison with adequate skills/experience match to application • Availability of key code development personnel to engage in and guide readiness activities

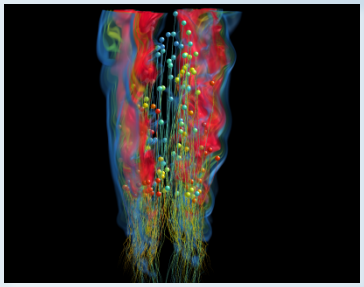
Center for Accelerated Application Readiness (CAAR)

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



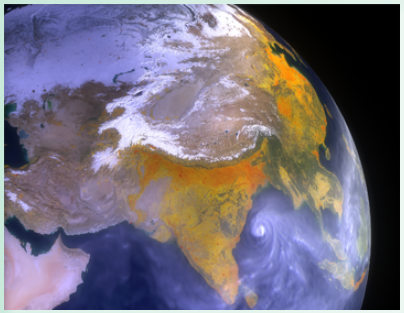


LAMMPS
A molecular description of membrane fusion, one of the most common ways for molecules to enter or exit living cells.

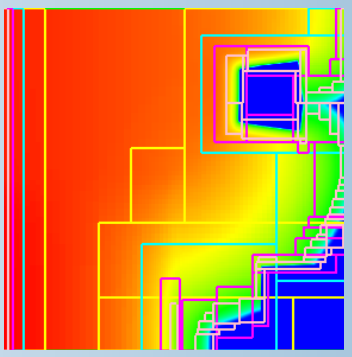


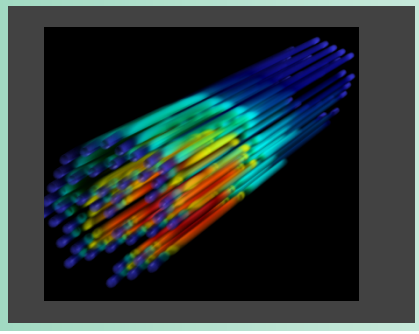
S3D
Understanding turbulent combustion through direct numerical simulation with complex chemistry.
.

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.

Slides courtesy Bronson Messer

Action plan for code porting

We developed a plan for porting these applications, which involved the following steps:

1. Multidisciplinary code team for each code – OLCF application lead, Cray engineer, NVIDIA developer, also cross-cutting support from tool and library developers
2. Early testbed hardware –white box GPU cluster “yona” for code development
3. Code inventory for each code to understand characteristics – application code structure, code suitability for GPU port, algorithm structure, data structures and data movement patterns. Also code execution profile – are there performance “hot spots” or is the profile “flat”
4. Develop parallelization approach for each application – ascertain which algorithm and code components to port to GPU, how to map work to GPU threads, how to manage data motion CPU-GPU and between GPU main memory and GPU caches/shared memory

Action plan for code porting (2)

5. Decide GPU programming model for port to GPU, e.g., CUDA for more close-to-the-metal programming, OpenACC for a higher abstraction level and a more incremental porting approach, OpenCL for portability advantages, or libraries when appropriate
6. Address code development issues – rewrite vs. refactor, managing portability to other platforms, incorporating GPU code into build system, relationship to the code repository main trunk
7. Representative test cases, e.g., early science problems, formulated as basis for evaluating code performance and setting priorities for code optimization. Also formulate comparison metric to measure success, e.g., time to solution on dual Interlagos Cray XE6 vs. Titan Cray XK7 Interlagos+Kepler

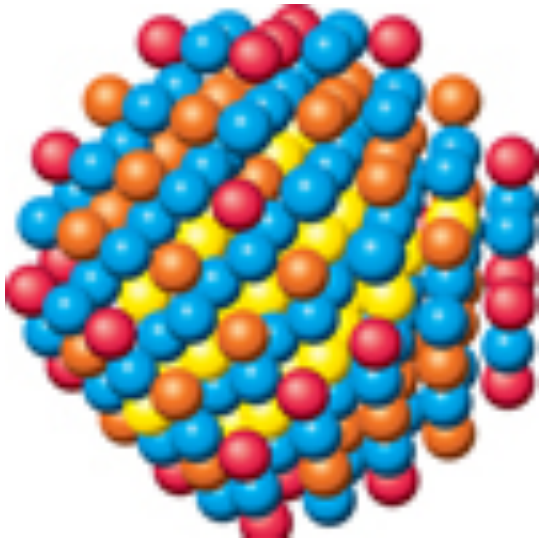
Application characteristics inventory

App	Science Area	Algorithm(s)	Grid type	Programming Language(s)	Compiler(s) supported	Approx. LOC	Communication Libraries	Math Libraries
CAM-SE	climate	spectral finite elements, dense & sparse linear algebra, particles	structured	F90	PGI, Lahey, IBM	500K	MPI	Trilinos
LAMMPS	Biology / materials	molecular dynamics, FFT, particles	N/A	C++	GNU, PGI, IBM, Intel	140K	MPI	FFTW
S3D	combustion	Navier-Stokes, finite diff, dense & sparse linear algebra, particles	structured	F77, F90	PGI	10K	MPI	None
Denovo	nuclear energy	wavefront sweep, GMRES	structured	C++, Fortran, Python	GNU, PGI, Cray, Intel	46K	MPI	Trilinos, LAPACK, SuperLU, Metis
WL-LSMS	nanoscience	density functional theory, Monte Carlo	N/A	F77, F90, C, C++	PGI, GNU	70K	MPI	LAPACK (`ZGEMM`, `ZGTRF`, `ZGTRS`)
NRDF	radiation transport	Non-equilibrium radiation diffusion equation	structured AMR	C++, C, F77	PGI, GNU, Intel	500K	MPI, SAMRAI	BLAS, PETSc, Hypre, SAMRSolvers

1. Wang-Landau LSMS

First principles, statistical mechanics of magnetic materials

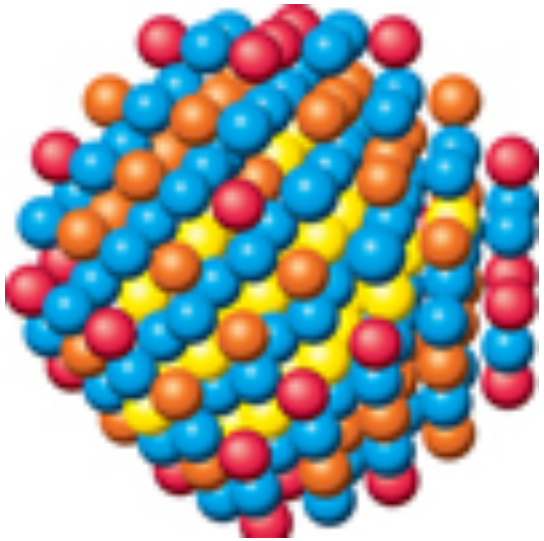
- **Purpose:** Compute the magnetic structure and thermodynamics of low-dimensional magnetic structures
- **Model:** Combines classical statistical mechanics (W-L) for atomic magnetic moment distributions with first-principles calculations (LSMS) of the associated energies.
- **Execution Structure:** Master node spawns many Monte Carlo “walkers” that do most of the work independently, results are occasionally combined on master
- **Execution Profile:** Very concentrated hot spot: most of the work is in the walkers: matrix inversion, and BLAS3 ZGEMMs, typical matrix sizes are 1200X1200, 1200X3600
- **Code Language:** C++ and F77
- **Lines of Code:** 70K



1. Wang-Landau LSMS

First principles, statistical mechanics of magnetic materials

- **Parallelization strategy :**
  - Compute ZGEMMs on GPU using cuBLAS library, use multiple streams via CPU OpenMP threads to saturate GPU
  - For LU factorization for matrix inversion, using cuBLAS, CULA, Cray libsci_acc or custom code (fastest)
  - Also moved matrix construction to GPU
  - Rewritten code LSMS_3 now allows multiple atoms per MPI rank, flexibility for more node parallelism, e.g., OpenMP threading, multiple GPU execution streams

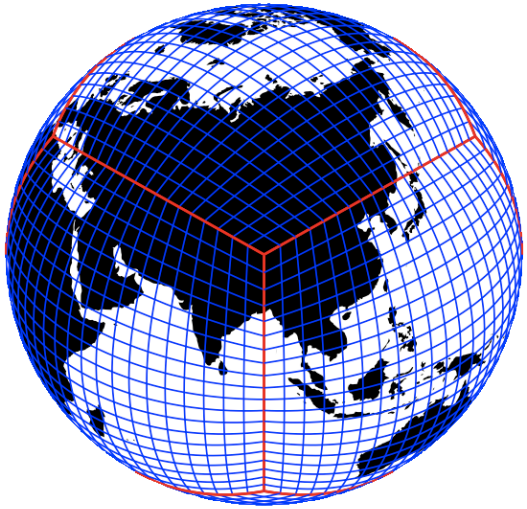


See also: Markus Eisenbach, http://www.olcf.ornl.gov/wp-content/uploads/2012/05/WL-LSMS-GPU_don1.pdf

2. CAM-SE

Community Atmosphere Model – Spectral Elements

- **Purpose:** Answer questions about climate change adaptation and mitigation scenarios; accurately represent regional-scale climate features of significant impact
- **Model:** Spectral element discretization with dynamical core (fluid dynamics + tracer transport) and other physics modules
- **Execution Structure:** Runge-Kutta explicit time stepping over 2-D logically unstructured cubed-sphere grid with vertical levels, at each time step dynamical core calculations, tracers, other physics.
- **Execution Profile:** Highly problem dependent. Targeted science case: tracer transport highest, then dynamical core, both employing vertical remap operation
- **Code Language:** F90
- **Lines of Code:** 500K

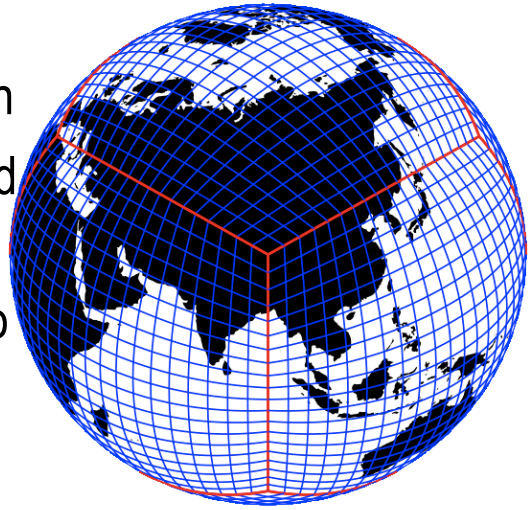


http://www-personal.umich.edu/~paullric/A_CubedSphere.png

2. CAM-SE

Community Atmosphere Model – Spectral Elements

- **Parallelization strategy :**
  - Tracers parallelized in straightforward data parallel fashion
  - For tracers used Mozart chemistry, more tracers, improved model realism, more GPU work
  - Vertical remap –tridiagonal solver replaced with splines, to expose more parallelism, 5X faster on CPU
  - Data structures were reworked – arrays of struct of arrays replaced with coalesced flat arrays
  - Separate element loops fused to improve granularity
  - Boundary exchange communications were optimized
  - Asynchronicity to overlap MPI and PCIe transfers using staging techniques
  - Used CUDA Fortran, will move to OpenACC for better integration with code trunk



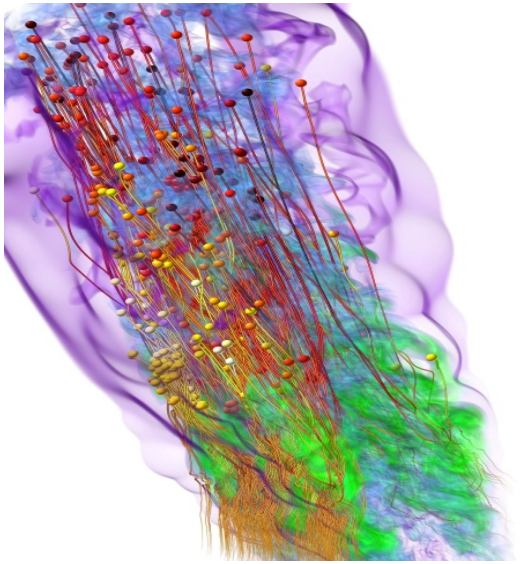
http://www-personal.umich.edu/~paullric/A_CubedSphere.png

See also: Matt Norman, http://www.olcf.ornl.gov/wp-content/uploads/2012/05/cray_workshop_2012_mrnorman-2.pdf
Jeff Larkin, http://www.slideshare.net/jefflarkin/progress-toward-accelerating-camse

3. S3D

Direct Numerical Simulation of Turbulent Combustion

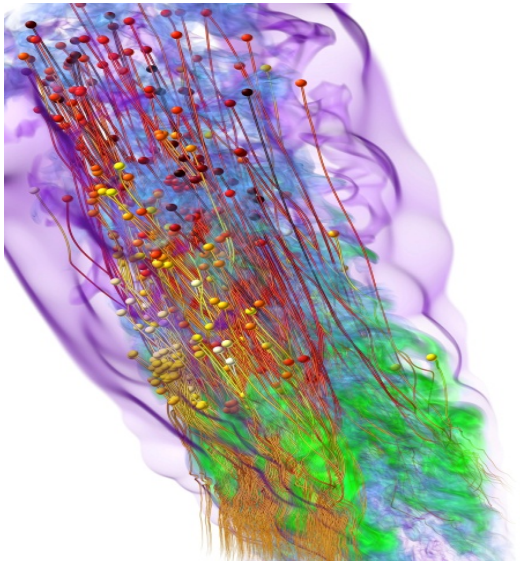
- **Purpose:** Provide fundamental insight into the chemistry-turbulence interaction of combustion processes
- **Model:** DNS Navier-Stokes simulation on Cartesian grid with particle tracking
- **Execution Structure:** 4th-order explicit Runge-Kutta time stepping over 3-D structured grid, 8th-order finite differences
- **Execution Profile:** 2-3 routines account for most of the runtime. Most prominent is reaction rates, the RHS and transport coefficients
- **Code Language:** F90
- **Lines of Code:** 10K



3. S3D

Direct Numerical Simulation of Turbulent Combustion

- **Parallelization strategy :**
  - Perform major code restructuring to move a 3-D grid loop up the call tree to expose coarser-grain parallelism
  - Port code from pure-MPI to hybrid, MPI-OpenMP, then ported kernels to GPU, then rewrite in OpenACC to run almost entirely on the GPU
  - Use compiler diagnostics to understand data movement
  - Identify data regions to scope arrays for efficient use of GPUs

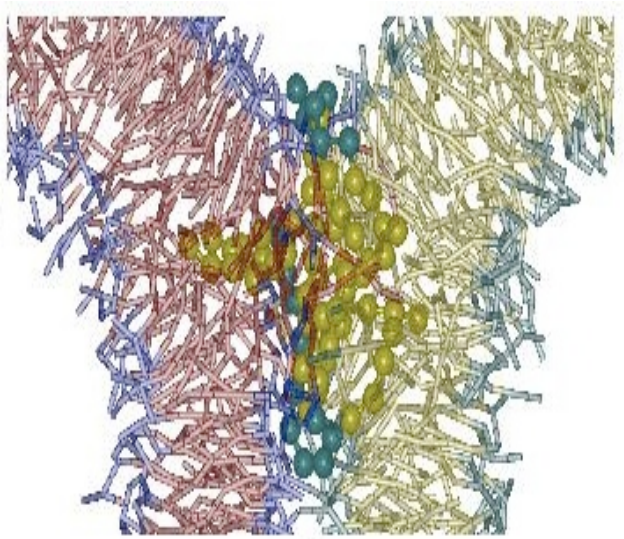


See also: John Levesque, http://www.olcf.ornl.gov/wp-content/training/CrayTech_XK6_2012/CTW_S3D_10_10.pdf
Ramanan Sankaran, http://www.olcf.ornl.gov/wp-content/uploads/2011/08/TitanSummit2011_Sankaran.pdf

4. LAMMPS

Large-scale, massively parallel molecular dynamics

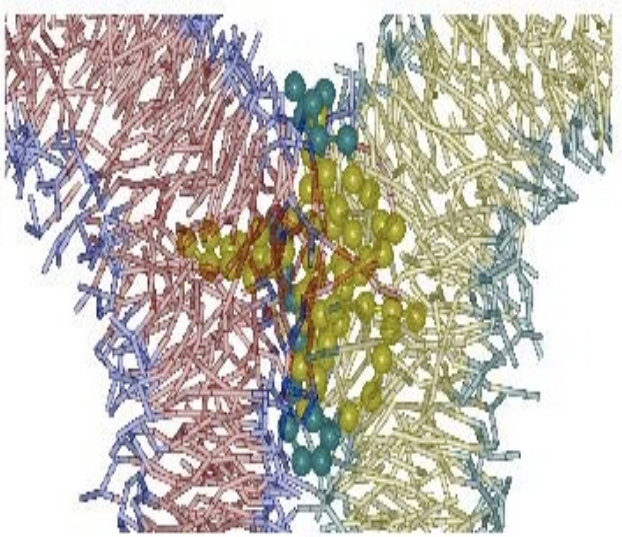
- **Purpose:** Provide understanding of molecular processes such as cellular membrane fusion
- **Model:** Classical N-body atomistic modeling with molecular dynamics
- **Execution Structure:** Forward stepping in time as particles move based on force field calculations
- **Execution Profile:** A large fraction of time is spent in short-range force calculations. Long-range force computations are chief barrier to high scalability
- **Code Language:** C++
- **Lines of Code:** 140K



4. LAMMPS

Large-scale, massively parallel molecular dynamics

- **Parallelization strategy :**
  - Port short-range force calculations and other calculations to GPU
  - For long-range forces, replace communication-expensive 3-D FFT with MSM, a multigrid-like algorithm with scalable communication

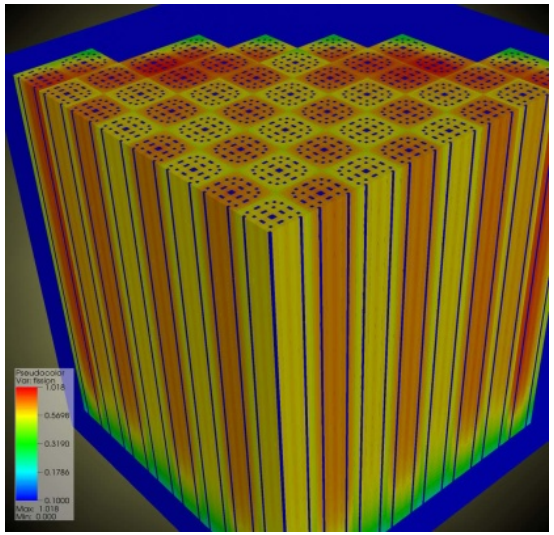


See also: Mike Brown, http://www.olcf.ornl.gov/wp-content/uploads/2012/05/brown_cray_tech_12.pdf

5. DENOVO

3D Neutron Transport for Nuclear Reactor Design

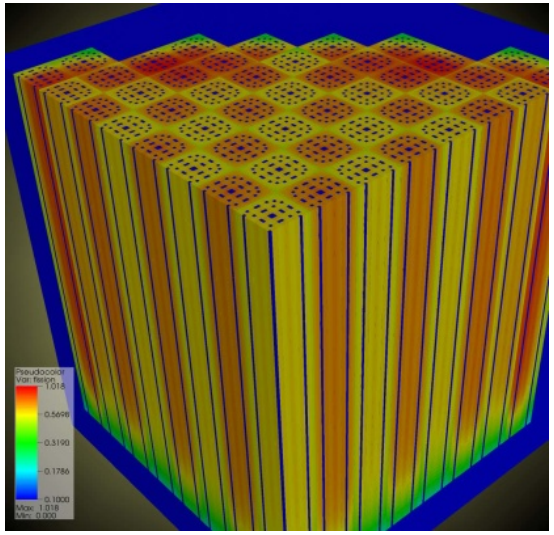
- **Purpose:** Model radiation transport for reactor safety and nuclear forensics
- **Model:** Linear Boltzmann transport, discrete ordinates method
- **Execution Structure:** Arnoldi eigenvalue solver, inner GMRES loop, matrix-vector product contains a 3-D sweep operation
- **Execution Profile:** Nearly all of the runtime is spent in the 3-D sweep code. The next most expensive part is GMRES
- **Code Language:** C++
- **Lines of Code:** 46K



5. DENOVO

3D Neutron Transport for Nuclear Reactor Design

- **Parallelization strategy :**
  - Restructure Denovo to provide to provide another axis of parallelism (energy-group) for better cross-node scaling and GPU threading
  - KBA sweep algorithm ported to the GPU, exploiting multiple problem dimensions to get enough threads
  - Permute loops to optimize for memory locality
  - Use CUDA to extract high performance
  - Trilinos/GMRES used GPU solves



See also: Wayne Joubert, http://www.olcf.ornl.gov/wp-content/uploads/2012/05/CrayTechWorkshop2012-Denovo-WJ.pdf

Performance results:

How Effective are GPUs on Scalable Applications?

OLCF-3 Early Science Codes

Very early performance measurements on Titan

	XK7 (w/ K20x) vs. XE6	Cray XK7: K20x GPU plus AMD 6274 CPU Cray XE6: Dual AMD 6274 and no GPU Cray XK6 w/o GPU: Single AMD 6274, no GPU
Application	**Performance Ratio**	**Comments**
S3D	1.8	• Turbulent combustion • **6% of Jaguar workload**
Denovo sweep	3.8	• Sweep kernel of 3D neutron transport for nuclear reactors • **2% of Jaguar workload**
LAMMPS	7.4* (mixed precision)	• High-performance molecular dynamics • **1% of Jaguar workload**
WL-LSMS	3.8	• Statistical mechanics of magnetic materials • **2% of Jaguar workload** • 2009 Gordon Bell Winner
CAM-SE	1.8* (estimate)	• Community atmosphere model • **1% of Jaguar workload**

Slide courtesy Buddy Bland

Lessons Learned

- 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

Lessons Learned

- 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

Lessons Learned

- 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)

Lessons Learned

- 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

Lessons Learned

- 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 (Denovo 2X, CAM-SE >1.7X)
- 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

Lessons Learned

- 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—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

Acknowledgements

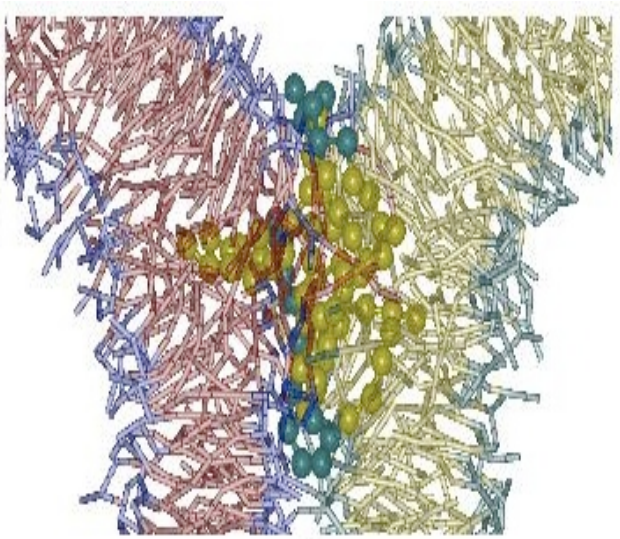
Bronson Messer, Mike Brown, Matt Norman, Markus Eisenbach, Ramanan Sankaran

Supplementary slides

4. LAMMPS

Large-scale, massively parallel molecular dynamics

- **Parallelization strategy :**
  - Port short-range force calculations, neighbor list calculations and parts of long range force calculations to GPU. Apply one or more threads per atom
  - Split work between CPU and GPU to use all available resources, overlapping work when possible
  - Make extensive use of CUDA streams
  - Multiple MPI tasks access GPU
  - Use Geryon middleware library to be able to target CUDA and OpenCL in same code
  - For long-range forces, replace communication-expensive 3-D FFT with MSM, a multigrid-like algorithm

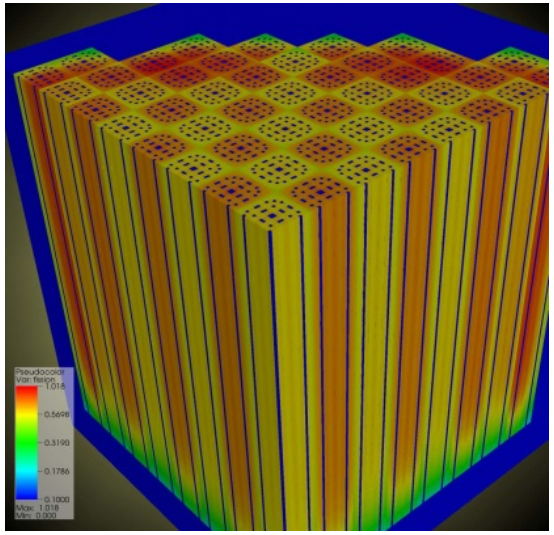


See also: Mike Brown, http://www.olcf.ornl.gov/wp-content/uploads/2012/05/brown_cray_tech_12.pdf

5. DENOVO

3D Neutron Transport for Nuclear Reactor Design

- **Parallelization strategy :**
  - Restructure Denovo to provide another axis of parallelism (energy-group) for better cross-node scaling and GPU threading
  - KBA sweep algorithm ported to the GPU, exploiting multiple problem dimensions to get enough threads
  - Trilinos/GMRES used for GPU solves



See also: Wayne Joubert, http://www.olcf.ornl.gov/wp-content/uploads/2012/05/CrayTechWorkshop2012-Denovo-WJ.pdf