Live two-photon imaging was used to observe the dynamic features of angiogenic growth in an in vitro model of angiogenesis. A mosaic of four by five image stacks of a 3D vascularized construct was acquired every 2 hours over the course of two days. The resulting 4D dataset consisted of over 500 GB of data. This image shows a rendered FluoRender with an NVIDIA GeForce GTX TITAN graphics card.

Nitro: An Adaptive Code Variant Tuning Framework
Saurav Karandharan, Manu Shantharam, Mary Hall, Michael Garland*, Bryan Catanzaro - University of Utah and “NVIDIA Research

- Selects variant to execute at run-time based on input data characteristics
- Builds a statistical model that maps from input characteristics to variants
- C++ and Python interfaces to specify variants, features, constraints etc. and to customize the tuning process.

Fluorescent imaging was used to observe the dynamic features of angiogenic growth in an in vitro model of angiogenesis. A mosaic of four by five image stacks of a 3D vascularized construct was acquired every 2 hours over the course of two days. The resulting 4D dataset consisted of over 500 GB of data. This image shows a rendered FluoRender with an NVIDIA GeForce GTX TITAN graphics card.

Nitro: An Adaptive Code Variant Tuning Framework
Saurav Karandharan, Manu Shantharam, Mary Hall, Michael Garland*, Bryan Catanzaro - University of Utah and “NVIDIA Research

- Selects variant to execute at run-time based on input data characteristics
- Builds a statistical model that maps from input characteristics to variants
- C++ and Python interfaces to specify variants, features, constraints etc. and to customize the tuning process.

Dynamic Particle System on the GPU
Mark Kim, Guoning Chen and Charles Hansen

Extracting isosurfaces represented as high quality meshes from three-dimensional scalar fields is needed for many important applications, particularly visualization and numerical simulation. One recent advance for extracting high quality meshes for isosurface computation is based on a dynamic particle system. Unfortunately, this state-of-the-art particle placement technique requires a significant amount of time to produce a satisfactory mesh. To address this issue, we utilize the parallelism of the particle placement and combine it with the CUDA implementation, a parallel programming technique on the GPU, to significantly improve the performance. We have applied our GPU based particle placement to a number of data from bioengineering where particle system is frequently used to generate isosurface meshes for simulations. Our results show comparable quality to the meshes generated using conventional CPU based particle system with at least ten fold speed up for most data.

Performance evaluated on five high-performance GPU libraries.

Sparse Matrix-Vector Multiplication
- 6 Variants from CUSP, 5 Features
- Training Set Size: 54, Testing Set Size: 100; Drawn from UFL matrix collection

Linear Solvers and Preconditioners
- 6 (Solver,Preconditioner) combinations from CULA, 8 Features
- Training Set: 26, Testing Set: 100; Drawn from UFL matrix collection

Breadth-First Search
- 6 Variants from Back40Computing, 5 Features
- Training Set Size: 20; Testing Set: 100; Drawn from DIMACS10

Histogram
- 6 Variants from CUB, 3 Features
- Training Set: 200, Testing Set: 1291; Images from INRIA Holidays dataset

Parallel Sort
- 3 Variants from ModernGPU and CUB, 3 Features
- Training Set Size: 120, Testing Set Size: 600; Generated

The University of Utah Carbon Capture Multi-Disciplinary Simulation Center (CCMSC)
Phil J. Smith, Martin Berzins, Alan Humphrey

One of three large DOE NNSA PSAAP II Centers, CCMSC aims to use simulations at petascale and eventually exascale to facilitate the design of the next generation of clean coal boilers that will improve clean coal technologies for the generation of electric power.

Radiative Heat Transfer
The Uintah open source framework (www.unr.uta.edu) has been one of the first computational environments to deploy methods such as the Discrete Ordinates method (developed at LANL) for radiative heat transfer in CFD applications. Uintah is now taking advantage of the petascale hardware and the advances in Monte Carlo ray tracing technology to develop an efficient and scalable solution to radiative heat transfer. Our approach, Reverse Monte Carlo Ray Tracing (RMCRRT), lends itself to scalable parallelism because the intensities of each ray are mutually exclusive and amenable to domain decomposition. However, the all-to-all nature of radiation requires information about the entire computation to be available to each computational cell. To address this issue, we are currently developing scalable CPU and GPU multilevel RMRT algorithms that take advantage of increased resolution of each ray in the near field while using coarse grid information from the far field. This is accomplished by using multilevel structured AMR (below). Our prototype GPU implementations have shown to be an order of magnitude faster than the CPU counter part, while retaining the required accuracy.

A Fast Iterative Method for Solving the Eikonal Equation on Triangulated Surfaces
Zhisheng Fu, Woon-Ki Jeong, Yongsheng Pan, Robert M. Kirby, Ross T. Whitaker

Mesh Fast Iterative Method (meshFIM)
- An iterative computational technique to solve the Eikonal equation efficiently on parallel architectures.
- This method relies on a modifications of a label correcting method.
- The core elements for our meshFIM based method are:
  1. Upwind scheme: calculate the value at a vertex with the values of the solved vertices.
  2. Active list management: Active list contains the list which has wave front vertices. If an active patch is convergent, it is removed from the Active list and its neighbor patches are added to this list.
  3. Patch-based iteration: divide the whole mesh into patches to fit into GPU cores.
  4. Triangle-based Jacobi update: update all the triangles inside a patch concurrently with parallel threads and each thread updates values of the three triangle vertices.

Suitability for GPU
- Each vertex updates independently
- According to the algorithm, update operation can be completed concurrently
- Computing only depends on the neighbors of same facet at every time step

Result
- CPU: Intel i7 920, 2.66GHz, 8M cache
- GPU: NVIDIA GTX 275, 1.40GHz, 240 core
- We test running time (ms) for a CPU version of meshFIM to compare with GPU version on three different meshes:

Illustration of multilevel RMRT algorithm showing 3 levels of mesh refinement (red, blue, black), for the near field at the conjunction of three illustrative rays of radiation.