# Performance portability of numerical time integrators in SUNDIALS library

DOE Center of Excellence Performance Portability Meeting

Slaven Peles (LLNL), Johann Dahm (IBM),
John Loffeld (LLNL), and Carol S. Woodward (LLNL)

**Denver, CO, August 22, 2017**

LLNL-PRES- 736968

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

# Challenges

## Porting SUNDIALS code to heterogeneous hardware architectures

- Implement numerical integrators in a way that makes best use of heterogeneous hardware architecture
- Ensure performance portability when used standalone or within LLNL Software Stack.
- Develop code that can evolve along with the new hardware -- separate platform specific from algorithmic part (RAJA, Kokkos).
- Total cost of ownership:
  - How easy is it to deploy the code in new environments?
  - How easy is it to add new features?
  - What is the maintenance cost?

Maximizing performance is but one of several challenges that need to be addressed when moving to new architectures.

# SUNDIALS

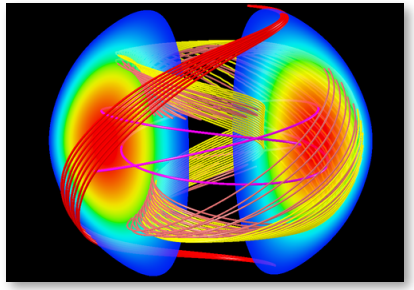
## Suite of state-of-the art numerical integrators and nonlinear solvers

- Forward looking, extensible object oriented design with simple and clean linear solver and vector interfaces.
- Designed to be incorporated into existing codes.
- Modular structure allows users to supply their own data structures.
- Scales well in simulations on over 500,000 cores.
- Supplied with serial, MPI and thread-parallel (OpenMP and Pthreads) structures, as well as *hypre* and PETSc vector interfaces.
- CMAKE support for configuration and build.
- Freely available, released under BSD license; Over >11,000 downloads in 2016.
- Modules and functionality:
  - ODE integrators: (CVODE) variable order and step stiff BDF and non-stiff Adams, (ARKode) variable step implicit, explicit, and additive Runge-Kutta for IMEX approaches.
  - DAE integrator: (IDA) variable order and step stiff BDF.
  - CVODES and IDAS include forward and adjoint sensitivity capabilities.
  - KINSOL nonlinear solver: Newton-Krylov and accelerated fixed point and Picard methods.

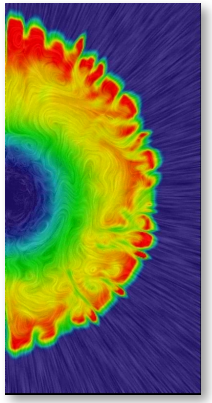
# SUNDIALS

## Used in industrial and academic applications worldwide

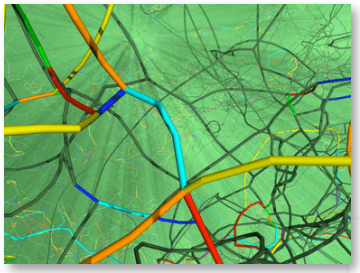
- Power grid modeling (RTE France, ISU)
- Simulation of clutches and power train parts (LuK GmbH & Co.)
- Electrical and heat generation within battery cells (CD-adapco)
- 3D parallel fusion (SMU, U. York, LLNL)
- Implicit hydrodynamics in core collapse supernova (Stony Brook)
- Dislocation dynamics (LLNL)
- Sensitivity analysis of chemically reacting flows (Sandia)
- Large-scale subsurface flows (CO Mines, LLNL)
- Optimization in simulation of energy-producing algae (NREL)
- Micromagnetic simulations (U. Southampton)



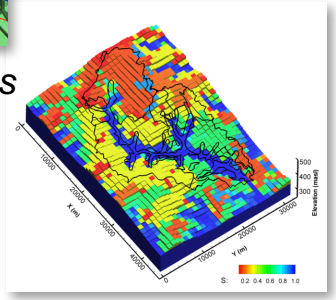
*Magnetic reconnection*



*Core collapse supernova*



*Dislocation dynamics*



*Subsurface flow*

# LLNL Software Stack

## Libraries currently being ported to heterogeneous architectures

**MFEM**: A free, lightweight, scalable C++ library for finite element methods.

**SUNDIALS**: Suite of state-of-the-art numerical integrators and nonlinear solvers.

***hypre***: A library for solving large, sparse linear systems of equations on massively parallel computers

The combined use of MFEM, *hypre* and SUNDIALS is critical for the efficient solution of a wide variety of transient PDEs, such as non-linear elasticity and magnetohydrodynamics.

Maintaining interoperability and performance portability of the software stack is more challenging on heterogeneous architectures.

# Interfacing SUNDIALS with other software

## Vector interface

- Specifies:
  - 3 constructors/destructors
  - 3 utility functions.
  - 9 streaming operators.
  - 10 reduction operators.
- Interaction with application data is carried out through these 19 operators.
- All are level-1 BLAS operators.
- Individual modules require only a subset of these operators.

## Linear solver interface

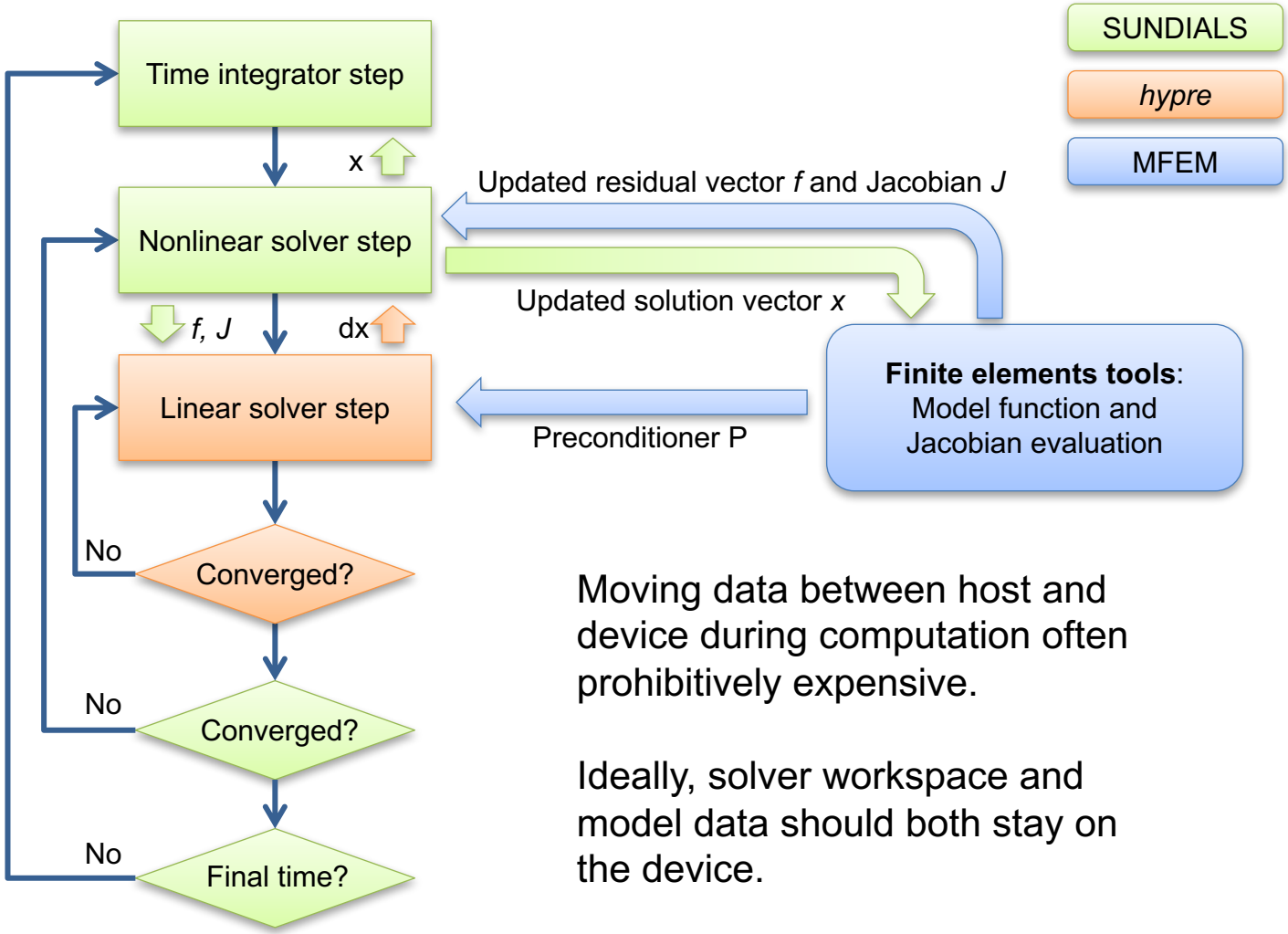
- Specifies following five functions: init, setup, solve, perf and free.
- SUNDIALS only requests linear solves at specific points. It is independent of linear solve strategy.
- Implementation of hypre linear solver interface is in progress.

Object oriented design and well defined interfaces simplify porting SUNDIALS to new platforms.

# Numerical simulation and data flow

## Use case: implicit integration scheme with iterative linear solver

Time integrator and nonlinear solver agnostic of vector data layout.




Numerical integrators and nonlinear solvers may invoke fairly complex step size control logic.

Moving data between host and device during computation often prohibitively expensive.

Ideally, solver workspace and model data should both stay on the device.

# Adapting SUNDIALS for execution on GPU

- Preliminary results show best performance is achieved when model evaluation and solver workspace are both in the device memory during computation. Moving entire or a part of the solver workspace prohibitively expensive.
- Developed vector kernels in CUDA for use on GPU-based hardware.
- Developed vector kernels using hardware abstraction layer RAJA as an alternative GPU-enabled implementation.
- Numerical integrator logic executed on the host; the integrator launches vector kernels on the device.
- Users will need to write CUDA kernels for their problem defining functions or code them in RAJA to realize benefit.

# Prototype GPU implementation

## SUNDIALS CUDA vector module

- The CUDA-based prototype was developed to understand requirements for running SUNDIALS on GPU-based architectures.
- The prototype uses standard SUNDIALS C-interface.

- Data layout class
  - Allocates memory on host and device.
  - Copies data to/from device
  - Provides pointer to partitioning class

```
template<class RealT, class IdxT>
class ThreadPartitioning {
  IdxT size_;
  IdxT mem_size_;
  RealT* h_vec_;
  RealT* d_vec_;
  bool ownPartitioning_;
  bool ownData_;
  ThreadPartitioning<RealT, IdxT>* part_;

  ...
};
```

- Thread partitioning class
  - Separate streaming and reduction thread partitionings.
  - Allocates data buffer for reduction kernels when needed
  - Hierarchical partitioning possible

```
template<class RealT, class IdxT>
class ThreadPartitioning {
  IdxT   block_;
  IdxT   grid_;
  IdxT   shMemSize_;
  RealT* d_buffer_;
  RealT* h_buffer_;
  IdxT   bufferSize_;

  ...
};
```

# Prototype GPU implementation

## SUNDIALS RAJA vector module

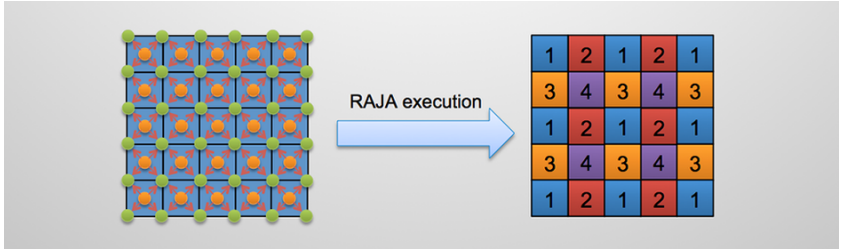
- Using RAJA hardware abstraction layer has a potential to significantly reduce code development and maintenance time compared to hand coded CUDA.
- The prototype uses standard SUNDIALS C-interface.

- Data layout class
  - Allocates memory on host and device.
  - Copies data to/from device

```
template<class RealT, class IdxT>
class ThreadPartitioning {
  IdxT size_;
  IdxT mem_size_;
  RealT* h_vec_;
  RealT* d_vec_;
  bool ownData_;

  ...
};
```

- Thread partitioning
  - Handled by RAJA; hidden from user.
  - The same code can produce CUDA or OpenMP shared memory parallelization
  - Hierarchical partitioning possible

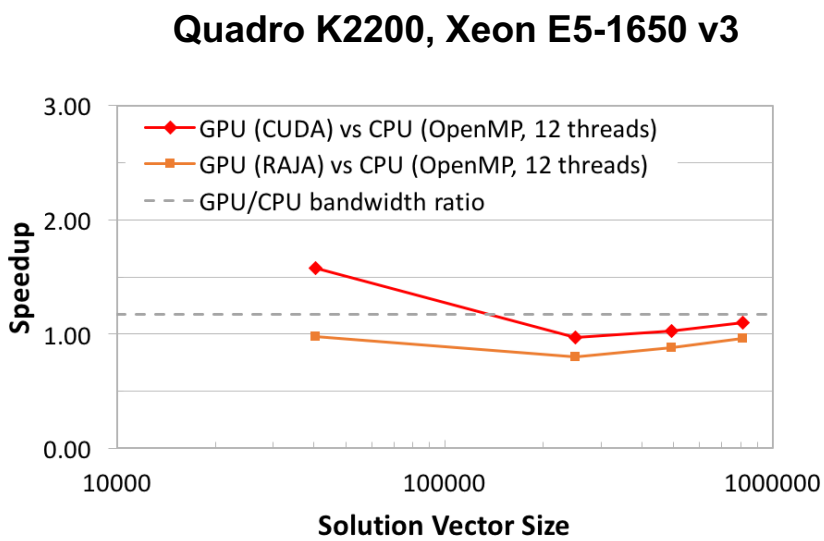


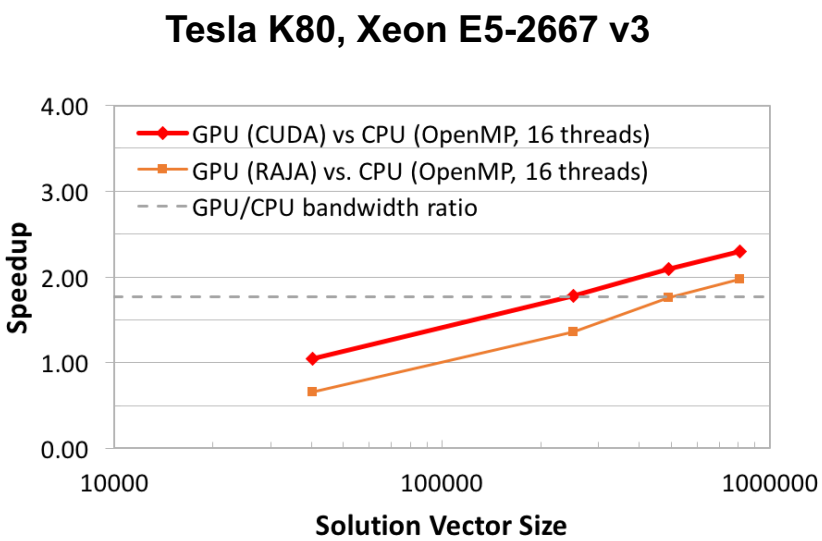
# Performance test on GPU architectures

## Advection-diffusion-reaction system integrated with CVODE

- We test a simple 2-D advection-diffusion-reaction system discretized on a unit square with standard 5-point stencil:

$$\frac{\partial u}{\partial t} = 0.01\nabla \cdot \nabla u + 10\nabla u + 100u(u+10)(1-u) \text{ on } \Omega, \qquad -\nabla u \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega$$



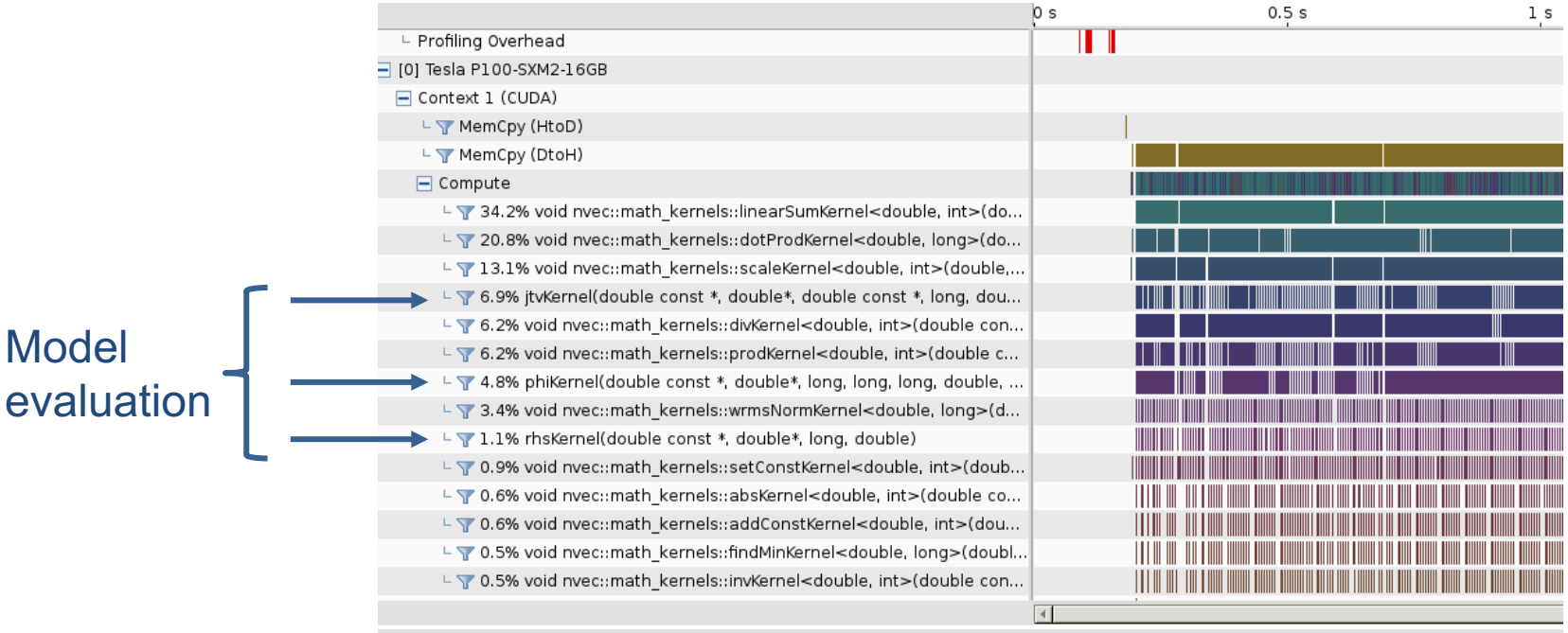




- RAJA adds a small performance overhead that decreases with vector size.

# Code profiling

## Advection-diffusion-reaction system integrated with CVODE

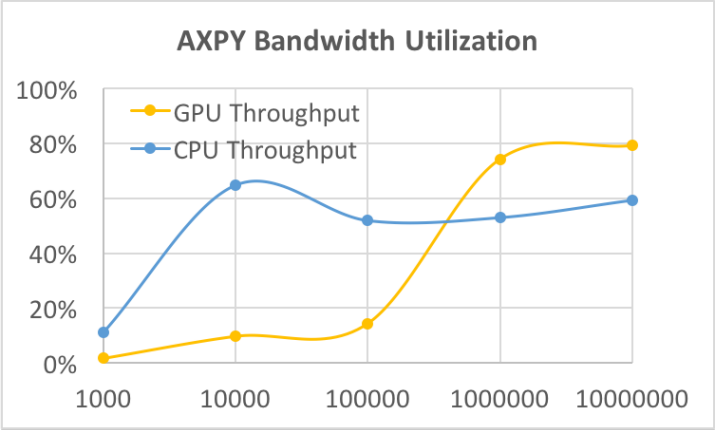
- 79% of processing time is taken by 6 vector kernels (2 reductions).
- 20% is spent on model evaluation and 1% on everything else.
- Overall performance can be predicted from testing results for a representative streaming and reduction kernels, respectively.

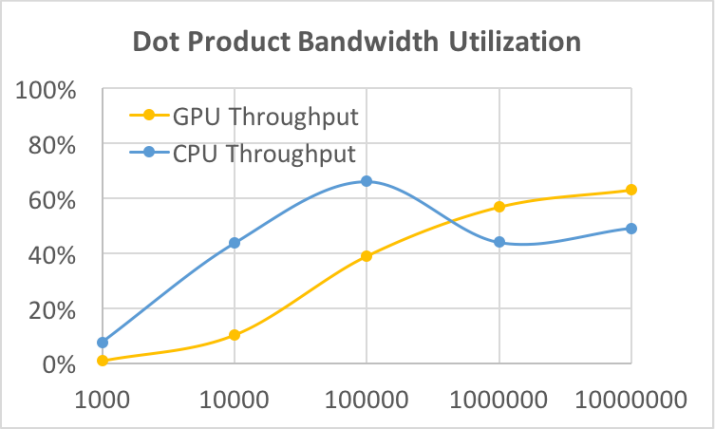


# SUNDIALS Bandwidth utilization

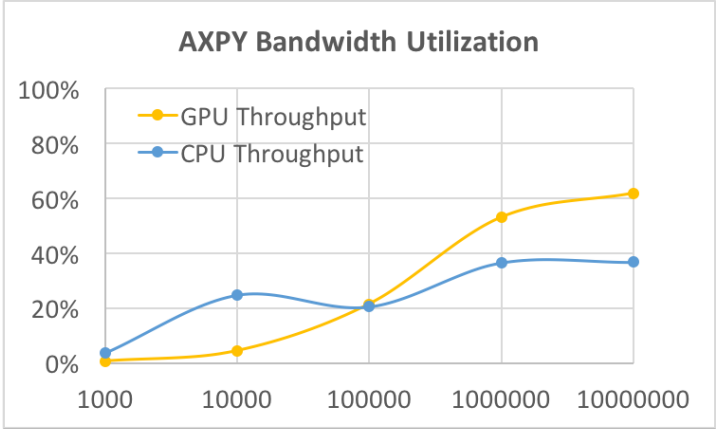
## Dot product and AXPY kernels make ~55% of the test case runtime

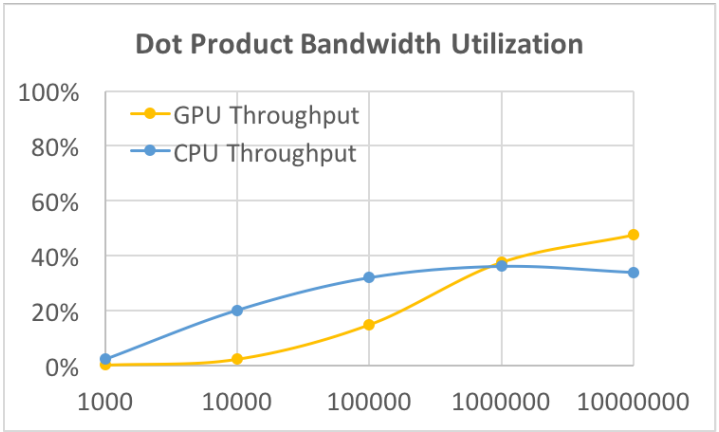
**Quadro K2200, Xeon E5-1650 v3**




**Tesla K80, Xeon E5-2667 v3**



# LLNL Software Stack

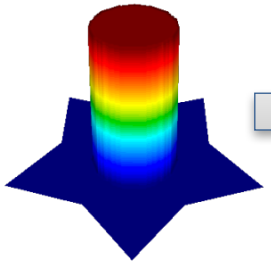
## Test case description

- We test nonlinear heat equation:

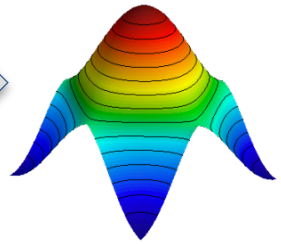
$$\frac{du}{dt} = \nabla \cdot (\kappa + \alpha u)\nabla u \quad \text{in } \Omega \text{ and } \nabla u \cdot \boldsymbol{n}_{\partial\Omega} = 0$$

- Finite element model is created in MFEM and integrated by SUNDIALS module CVODE using explicit Adams-Bashforth scheme.

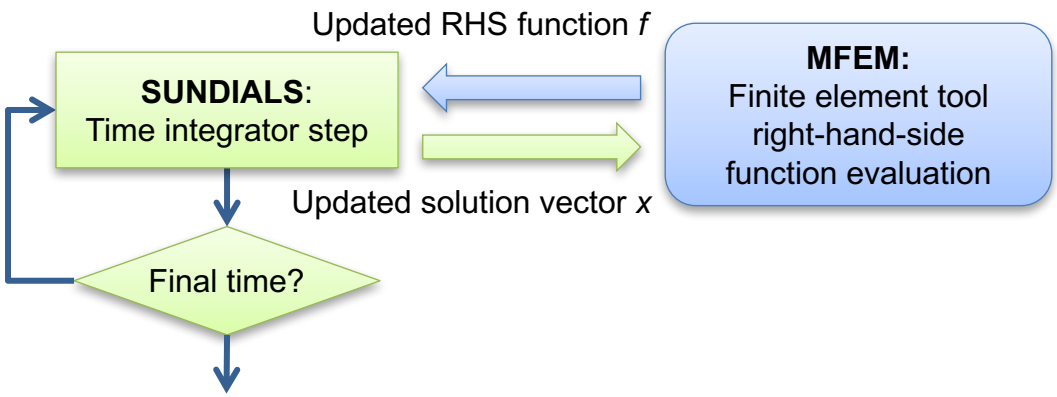
**Initial condition**



**Solution at t = 0.5**



**Explicit integration schematics with SUNDIALS and MFEM**



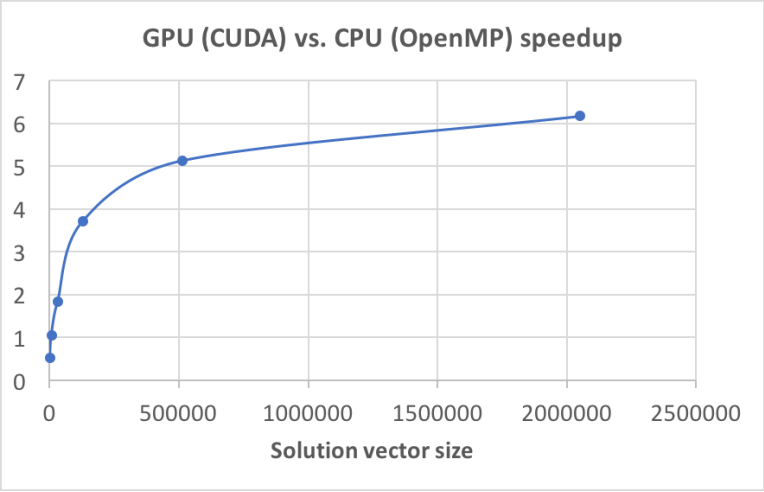
# LLNL Software Stack

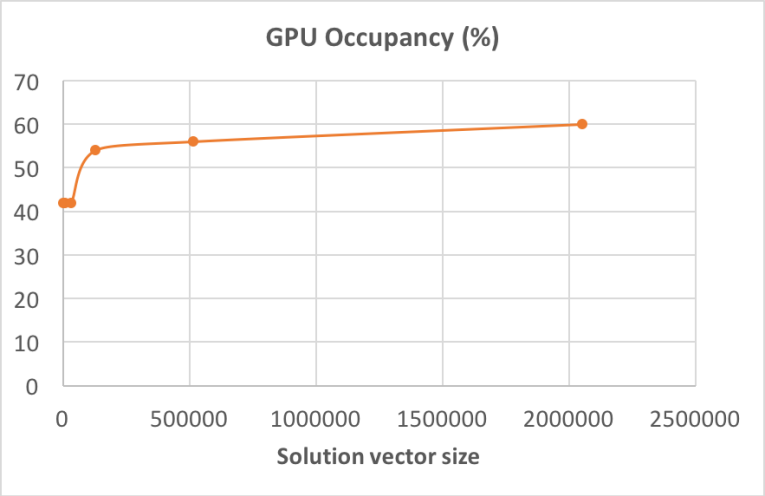
## SUNDIALS-MFEM combined performance

- We test nonlinear heat equation:

$$\frac{du}{dt} = \nabla \cdot (\kappa + \alpha u)\nabla u \quad \text{in } \Omega \text{ and } \nabla u \cdot \boldsymbol{n}_{\partial\Omega} = 0$$

- Solution vector size increased by refining the mesh and using higher order stencils.
- Performance testing shows significant speedup when running the test on the GPU vs. CPU, and good GPU device utilization.

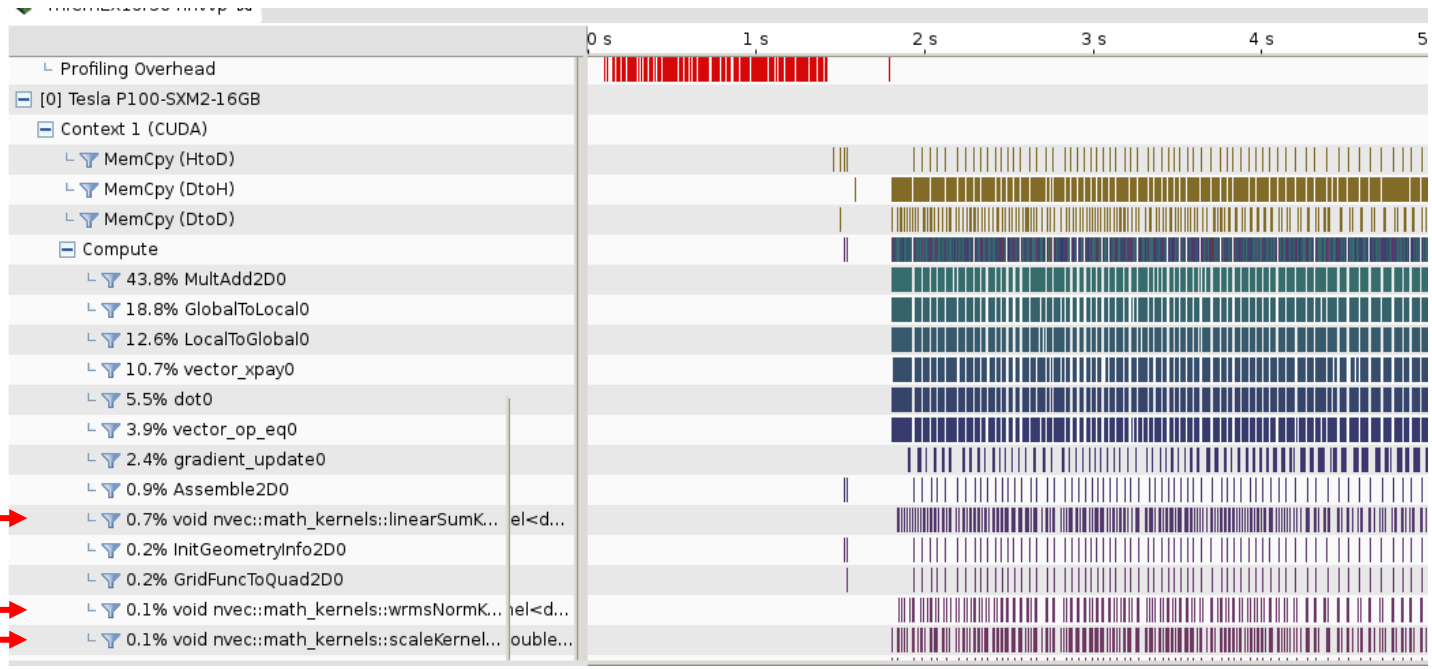


# LLNL Software Stack

## Profiling results

- Except for very simple problems SUNDIALS vector kernels make up only a small fraction of total execution time (~1% in this test case).
- Numerical integrator reduction kernels cannot be run on stream parallel with other computations.




SUNDIALS kernels

# Conclusions

- Moving data between host and device still most expensive action during numerical integration. Device memory appears to be the main performance limiting factor.
- Numerical integrators take small fraction of the overall computation time. The main reason to run numerical integration on the device is to minimize device to host communication.
- Numerical integrator reduction operators cannot be run on streams parallel to other computations. They are potential computational bottlenecks. Parallel in time integration may address this issue.