Preparing the SUNDIALS Library for Heterogeneous Architectures

Cody J. Balos[1], David J. Gardner[1], Daniel R. Reynolds[2], and Carol S. Woodward[1]

[1] *Center for Applied Scientific Computing, LLNL*
[2] *Department of Mathematics, SMU*

P3HPC Forum 2020

Lawrence Livermore National Laboratory

CASC
Center for Applied Scientific Computing

SMU

LLNL-PRES-813805

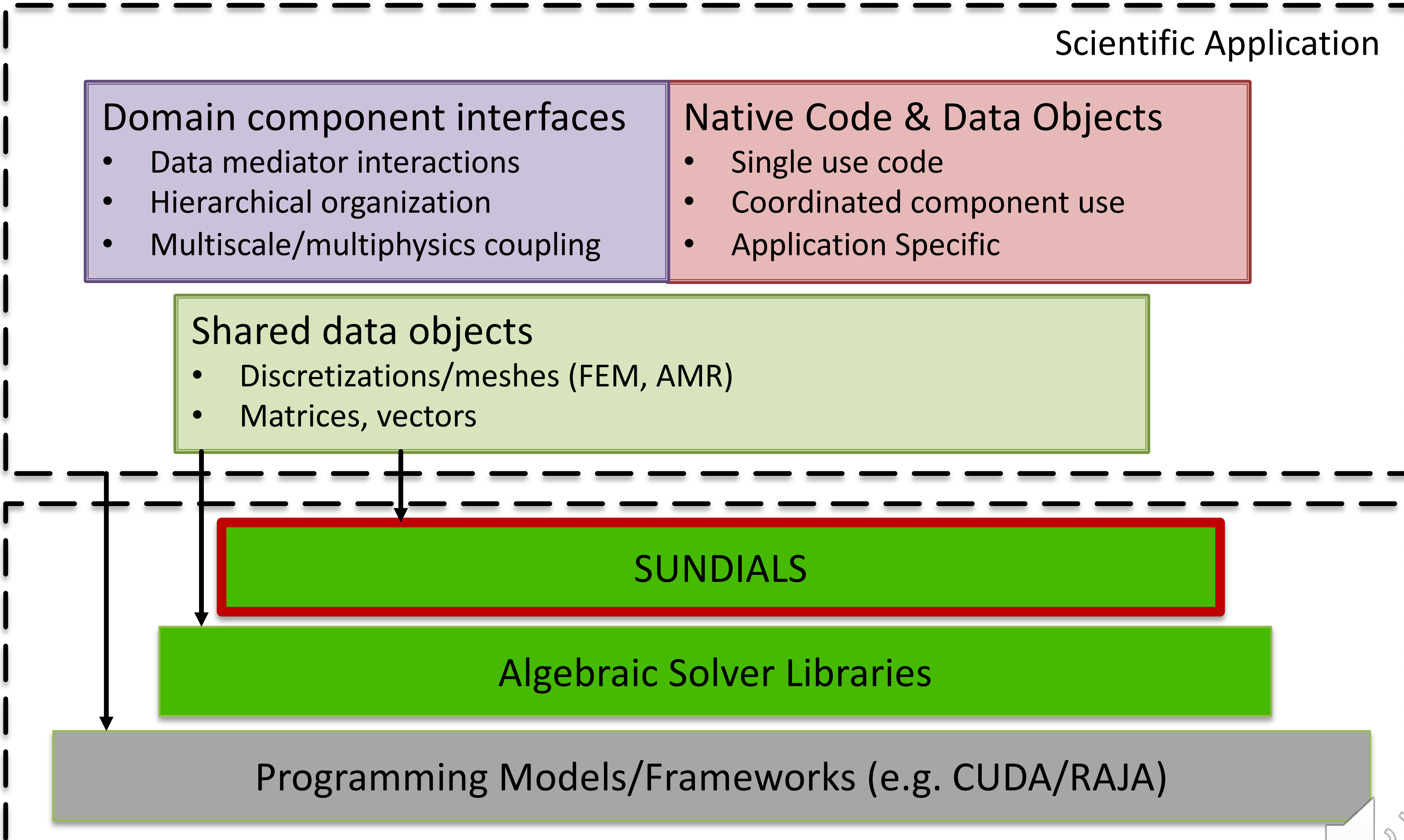
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

SUNDIALS: SUite of Nonlinear and Differential / ALgebraic equation Solvers

- Software library of ODE and DAE time integrators and nonlinear solvers
  - Written in C with interfaces to Fortran
  - Modular implementation
  - Designed to be easily incorporated into existing codes
  - Freely available; BSD 3-Clause license; >27,000 downloads in 2019
  - Detailed user manuals and an active user community email list
- Consists of six packages
  - **CVODE, ARKODE:** Methods for Ordinary Differential Equations (ODEs)
  - **IDA:** Methods for Differential Algebraic Equations (DAEs)
  - **CVODES** and **IDAS:** Forward and adjoint sensitivity analysis variants
  - **KINSOL:** methods for nonlinear systems of algebraic equations

computing.llnl.gov/sundials

SUNDIALS’ Position in the Software Stack



Where is the Parallelism in SUNDIALS?

- In $O(n)$ vector operations, e.g. linear sum to compute nonlinear residual

$$\mathrm{F}(y^n) = \mathrm{y^n} - \gamma f(t_n, y^n) - a_n$$

- In $O(n^2)$ matrix and matrix-vector operations, e.g. matrix scale and add to compute system matrix from Jacobian of ODE/DAE right-hand side

$$A = I - \gamma \frac{\partial f}{\partial y}$$

- In (varying complexity) linear solves (iterative and direct)

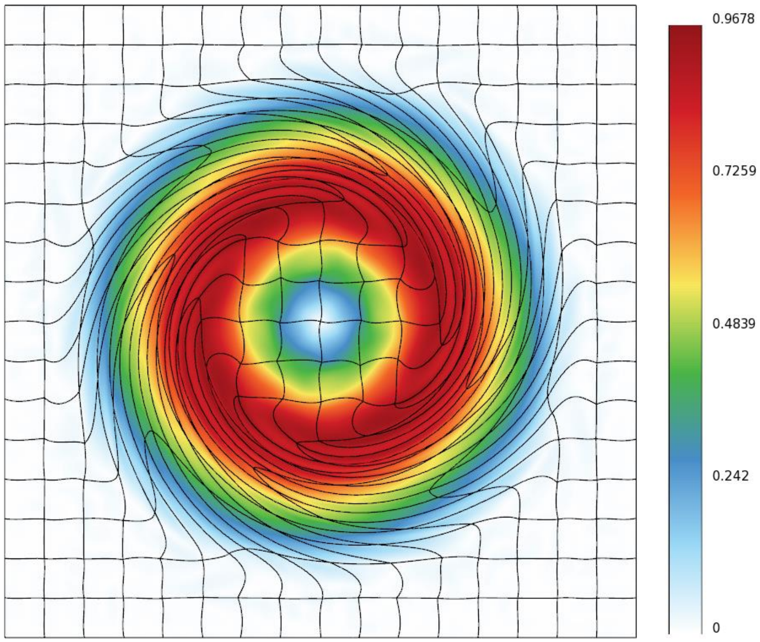
$$M\delta_m = -F(y^{n(m)})$$

- In (varying complexity) the evaluation of the ***user-provided*** right-hand side
  - User provides a function pointer to this and SUNDIALS calls it repeatedly
  - Typically, this is the most computationally expensive piece of the integration

A Couple of Very Different CPU+GPU Use Cases

(Case 1) SUNDIALS controls the main time-integration loop for the application, and a large ODE system is solved in a distributed manner.

- Use case example: Finite element applications
- Performance Strategy:
  - Keep data resident on the GPU
  - Optimize data operations for long vectors, and use iterative linear solvers
  - Application must perform function evaluations on the GPU to ensure data always resides on GPU

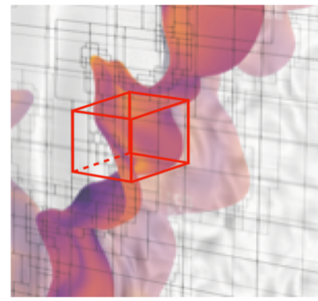


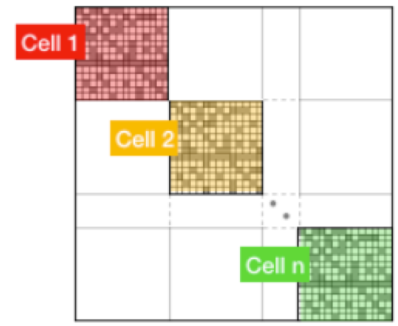
Gresho Vortex test problem in MFEM using SUNDIALS for the Lagrangian flow miniapp Laghos.

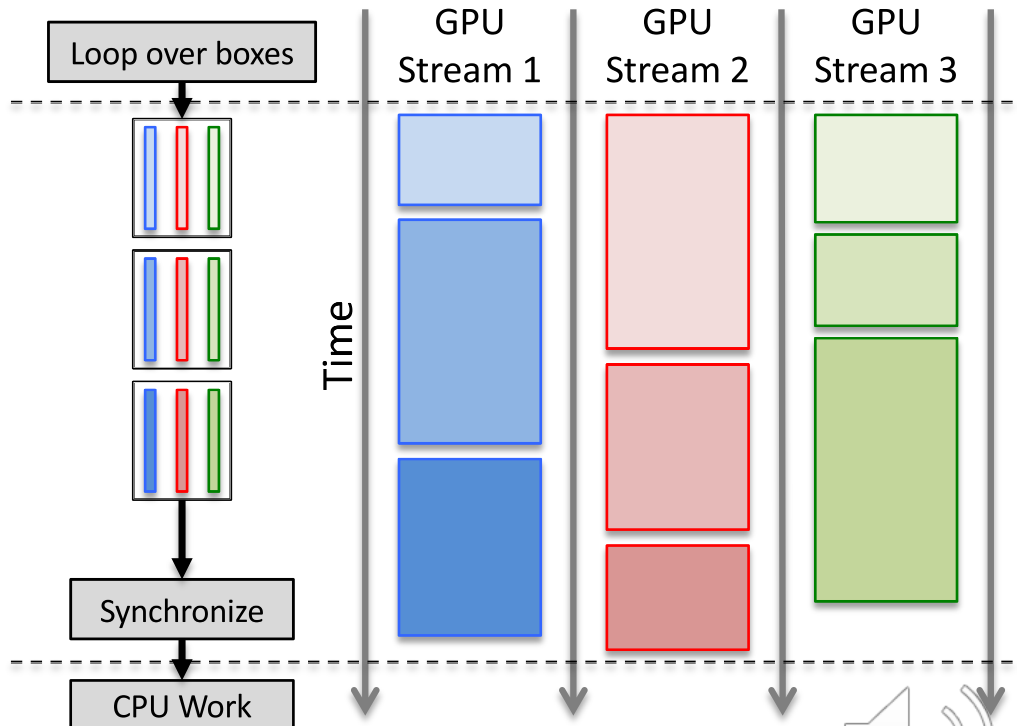
A Couple of Very Different CPU+GPU Use Cases

(Case 2) SUNDIALS is used as a local integrator for many small independent subsystems.

- Use case example: solving chemical kinetics per AMR grid cell
- Performance Strategy:
  - Group the cells and solve groups of subsystems as one large system
  - Solve multiple groups simultaneously in different CPU threads/GPU streams
  - Use linear solvers designed for block-diagonal linear systems







Considerations when preparing SUNDIALS for Heterogeneous Architectures

1. SUNDIALS’ position in the software stack requires interfacing in two directions
   - Application/Framework to SUNDIALS
   - SUNDIALS to libraries for algebraic solvers, parallelism libraries/frameworks, or architecture programming models

2. SUNDIALS parallelism is in vector and matrix operations, and linear solves

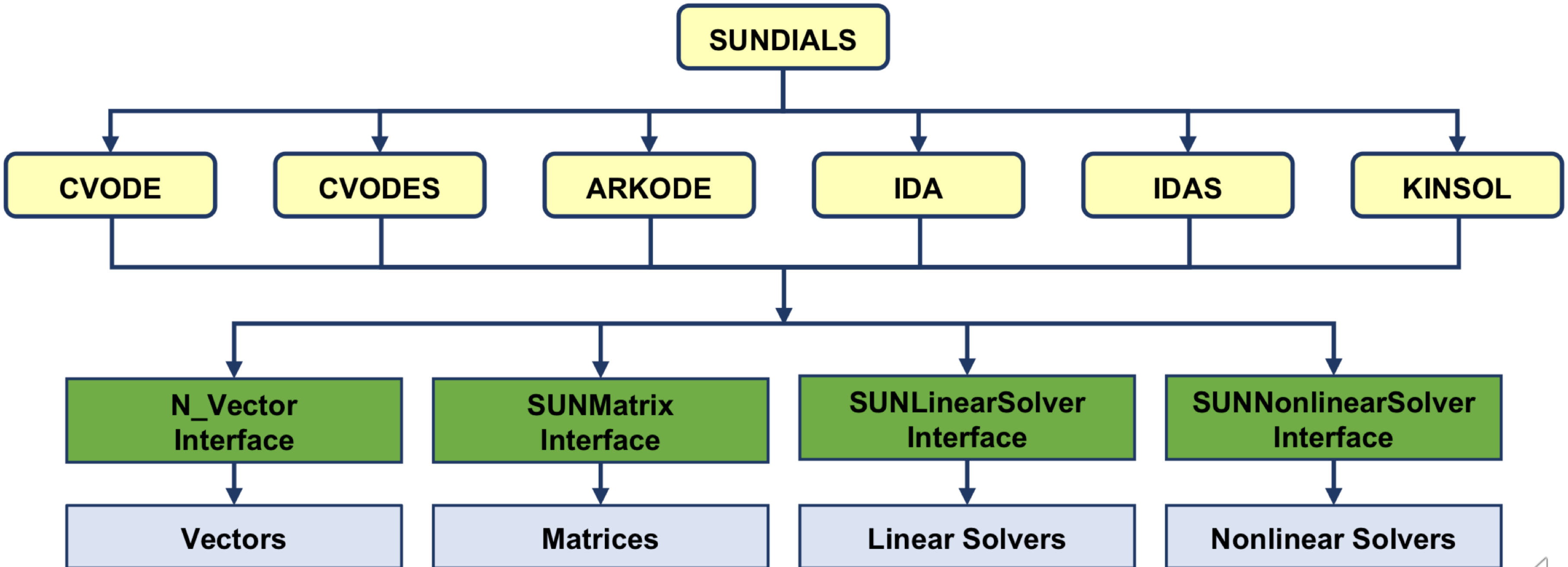
3. Must support very different use cases

4. Maintainability

Flexibility is Critical

SUNDIALS' Modular (Object Oriented) Design

- SUNDIALS always encapsulated data and vector operations with a vector data structure (N_Vector)
- In preparation for heterogeneous architectures (under the Exascale Computing Project) the SUNDIALS team has encapsulated nonlinear solvers, *linear solvers,* and *matrix operations*



SUNDIALS Ships with Data Structures that Support Heterogeneous Architectures

- We provide several N_Vector implementations for on-node parallelism:
  - OpenMP
  - OpenMP target offloading
  - Direct CUDA
  - Direct HIP (to be released in near future)
  - RAJA w/ CUDA backend (and HIP soon)
- A special MPI+X N_Vector takes any of the above and adds MPI parallelism
- The ManyVector N_Vector provides a mechanism for users to partition their simulation data among disparate computational resources

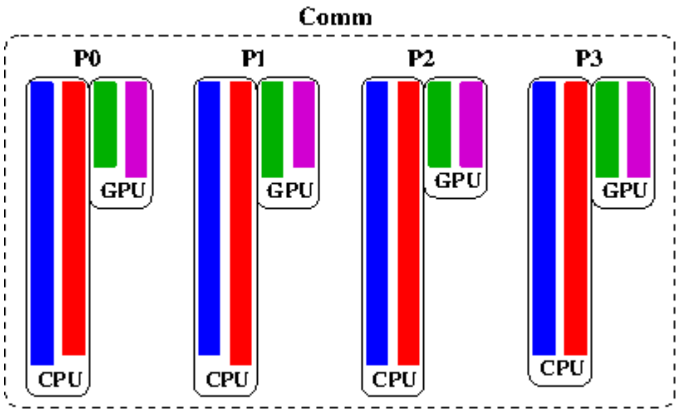


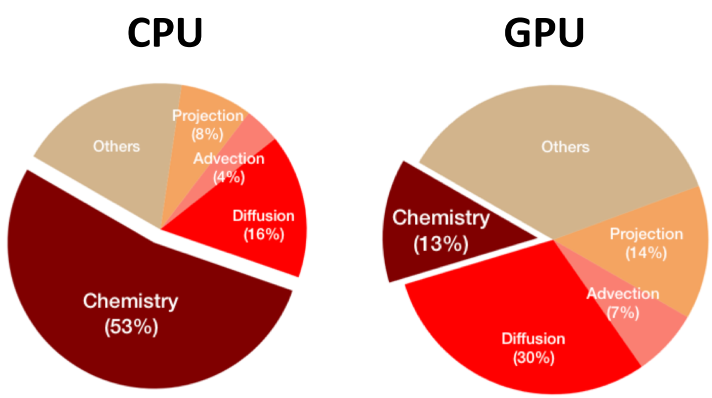
Figure 1, ManyVector use case for multi-rate or data partitioning, allowing for each vector to utilize distinct processing elements within the same node (e.g. red/blue on CPU and green/magenta on GPU) or for collective communications to be combined to minimize latency overhead (e.g., during Gram-Schmidt orthogonalization within linear or nonlinear solvers).

SUNDIALS Ships with Data Structures that Support Heterogeneous Architectures

- Also provide a SUNMatrix interface to cuSPARSE sparse matrix and a SUNLinearSolver interface to cuSOLVER
- Will be developing interfaces to more linear solver libraries to support more architectures and paradigms (e.g. DPC++)
- Alternatively, users can supply their own implementations of the N_Vector, SUNMatrix, SUNLinearSolver, and SUNNonlinearSolver interfaces
- Having all these different implementations does raise maintainability concerns, but…
  - Our object-oriented design insulates the core and most complex parts of SUNDIALS, i.e. the integrators
  - Use cases of SUNDIALS are so diverse that the benefit of having many out-of-the-box options is important while applications are rapidly changing and adapting to the heterogeneous architecture landscape

Performance Results on GPUs (Use Case 2)

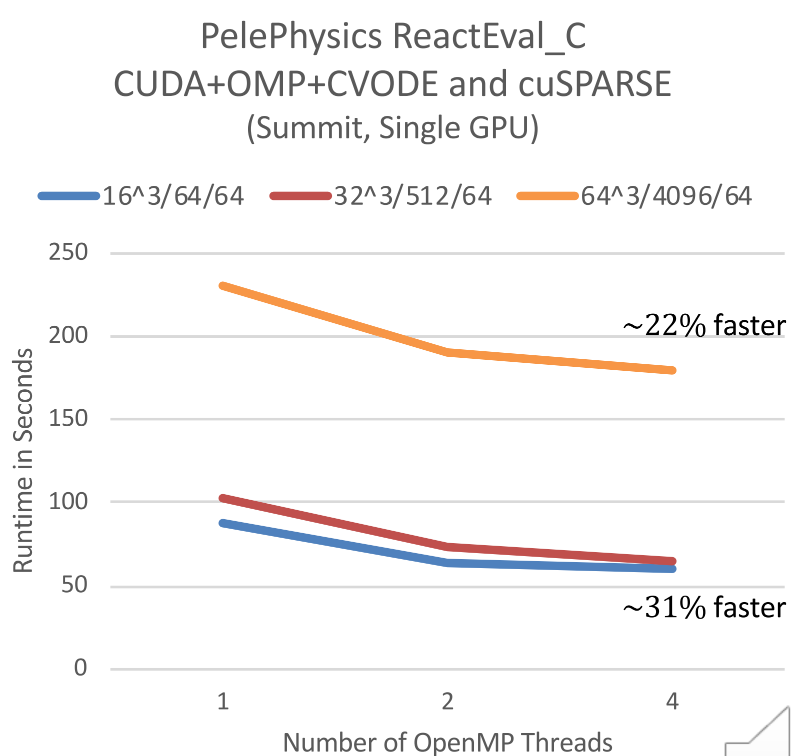
- **PeleLM:** Combustion application that uses AMReX
- **3D Flame Sheet** (CORI-GPU): using CVODE with GPUs makes chemistry ~10x faster on full application



- Moving to CVODE on the GPU reduced chemistry time from 53% to 13% of full flame sheet application run time.

Figures and results courtesy of Anne Felden and AMReX Team

- **PelePhysics ReactEval_C:** Chemistry only test code for PeleLM
- Threading + GPU streams can increase utilization w/o increasing the grid size



Thank You!

EXASCALE COMPUTING PROJECT

CASC

Center for Applied Scientific Computing

SMU®

Lawrence Livermore National Laboratory