RADIUSS: Rapid Application Development via an Institutional Universal Software Stack

Lawrence Livermore National Laboratory
COMPUTING
EXPO 2020
September 30, 2020

radiuss

radiuss.llnl.gov

LLNL-PRES-814279
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

Build Tools

Build Tools

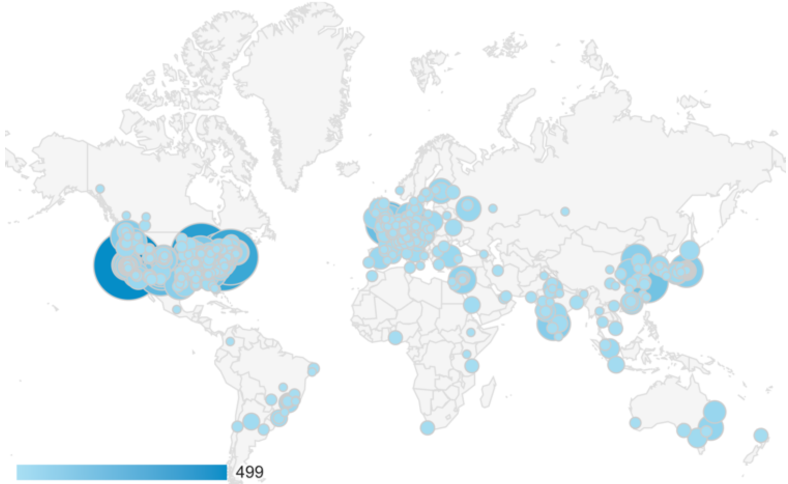
Technical Contact

Todd Gamblin

Project	Description	License	Maturity (years)	Website	Repository	Contact
Spack	A flexible package manager for HPC	Apache-2 or MIT	~7	spack.io	github.com/spack/spack	Todd Gamblin
BLT	A streamlined CMake build system foundation for HPC software	BSD	~2	llnl-blt.readthedocs.io	github.com/LLNL/blt	Chris White
Shroud	Easily create Fortran, C and Python interfaces for C or C++ libraries	BSD	~3.5	shroud.readthedocs.io	github.com/LLNL/shroud	Lee Taylor

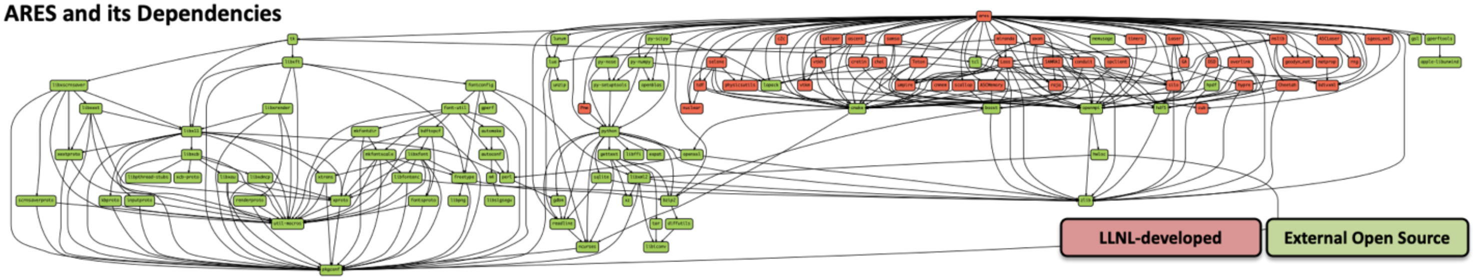
Spack
A flexible package manager for HPC

- **Automates complex builds**
  - Easily manage hundreds of dependencies, down to versions and build options
  - Easily test complex software with many compiler/MPI/BLAS combinations
- **Easily share and leverage others' work**
  - Leverage a library of 4,000+ community-maintained package recipes
  - Leverage others' internal/proprietary libraries with internal LLNL repositories
  - Allow other users and developers to easily use your software
- **Broad use inside and outside the laboratories**
  - ASC, LC, ENG, others at LLNL; codes at LANL, SNL, Fermi, ORNL, ANL, ECP
  - Nearly 3,000 worldwide users (per docs site), highly active community on GitHub



Spack users worldwide

ARES and its Dependencies



@spackpm

BLT

A streamlined CMake build system foundation for HPC software

- **Simple macros for complex tasks**
  - Create libraries, executables, and tests
  - Manages compiler flags across multiple compiler families
  - Unifies complexities of external dependencies into one easy to remember name
- **Batteries included**
  - Example configurations for most LC/Linux/OSX/Windows system and compiler families
  - Built-in support for:
    - HPC programming models
    - Code health
    - Documentation generation
- **Open source**
  - Leveraged by ALE3D, Ascent, Axom, CHAI, Conduit, FloBat, GeosX, Kripke, LEOS, MSLIB, RAJA, RAJA Perf Suite, Umpire, VBF Shaft, VTK-h

HPC Programming Models

Code Health

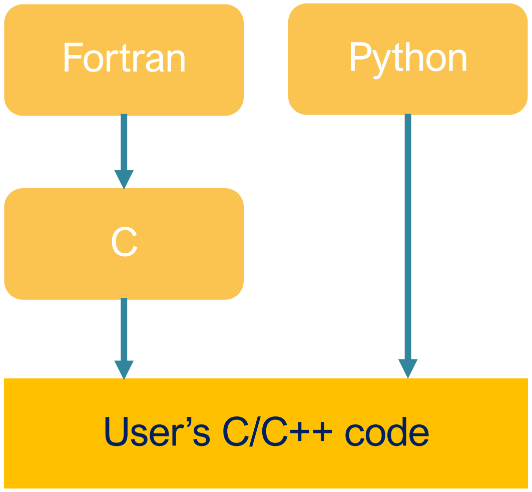
Fruit
Clang-query
Uncrustify
Astyle

Documentation

Shroud

Easily create Fortran, C, and Python interfaces for C or C++ libraries

- **Generate wrappers with an annotated description of the C++ API**
  - YAML input with C++ declarations for namespace, typedef, function, class, and struct
  - Annotations to provide semantic information: intent, dimension, ownership
  - Allows user control of generated names for functions and interfaces
  - Provides hooks to allow custom code to augment or replace generated wrapper
- **Creates a Fortran idiomatic interface**
  - Preserves object-oriented API
  - No need to be a Fortran expert to create Fortran wrapper
  - Uses C as lingua franca to access C++
- **Use the same YAML file to create a Python module**
  - Creates an extension module, no Python source code is created
  - Support for NumPy



Portable Execution and Memory Management

Portable Execution and Memory Management

Technical Contact

David Beckingsale

Project	Description	License	Maturity (years)	Website	Repository	Contact
RAJA	Loop-level abstractions to target machine-specific programming models and constructs	BSD	~5	software.llnl.gov/RAJA	github.com/LLNL/RAJA	Rich Hornung
CHAI	Optional add-on to RAJA for automating data motion between memory spaces	BSD	~4	software.llnl.gov/CHAI	github.com/LLNL/CHAI	David Beckingsale
Umpire	An application-focused API for memory management on NUMA & GPU architectures	MIT	~3	software.llnl.gov/Umpire	github.com/LLNL/Umpire	David Beckingsale
LvArray	Array classes for high-performance simulation software	BSD	~2	lvarray.readthedocs.io	github.com/GEOSX/LvArray	Ben Corbett

RAJA

Loop-level abstractions to target machine-specific programming models and constructs

- **Provides a portable API for loop execution**
- **Powerful “kernel” API to express nested, multi-dimensional loops**
- **Other portable features**
  - Reductions, scans, sorts, atomics, and multi-dimensional data views
- **Supports multiple back-end targets: OpenMP, CUDA, AMD, …**
- **Easy to integrate into existing applications**
  - Loop bodies remain generally unchanged
  - Can be adopted incrementally, one loop at a time
- **Open source**
  - Used by ASC and ATMD applications and libraries, and ECP projects: SAMRAI, MFEM, SUNDIALS, hypre, SW4, GEOS-X, ExaSGD, Alpine, etc.

```
for (int i = 0; i < N; ++i) {
  a[i] += c * b[i];
}
```

A simple C-style loop

```
forall<EXEC_POL>(RangeSegment(0, N),
  [=] (int i) {
    a[i] += c * b[i];
  }
);
```

Same loop using RAJA

Loop execution defined by “execution policy”: EXEC_POL can be seq_exec, openmp_exec, cuda_exec, etc.

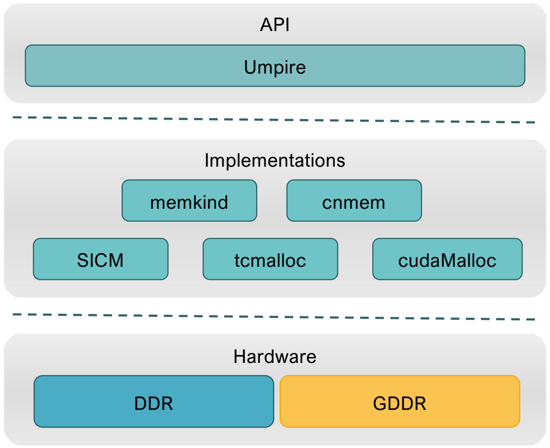
Umpire

An application-focused API for memory management on NUMA and GPU architectures

- **Simple and unified API to a wide range of memory resources:**
  - DDR
  - NVIDIA GPU memory
    - Constant memory
  - AMD GPU memory
  - NUMA support
- **Provides high-performance “strategies” for customizing data allocation:**
  - Memory pools, buffers, CUDA memory advice
- **“Operations” to copy, move, set data on any memory resource**
- **Open source**
  - Underpins CHAI
  - Used by LLNL ASC and ATDM applications, SW4, SAMRAI, MFEM

Umpire

```
auto allocator = rm.getAllocator(“DEVICE”);

double* data = allocator.allocate(1024);

allocator.deallocate(data);
```



API

Umpire

Implementations

memkind

cnmem

SICM

tcmalloc

cudaMalloc

Hardware

DDR

GDDR

CHAI

Optional add-on to RAJA for automating data transfers between memory spaces

- **Array-like object with automatic data migration**
- **Provides “unified memory” without any special system support**
- **Integrates with RAJA**
  - Could be used with other programming models
- **Uses Umpire, and behavior can be customized using different Umpire “Allocators”**
- **Open source**
  - Used in LLNL ASC applications
  - Works with Umpire & RAJA

```
chai::ManagedArray<double> data(100);

RAJA::forall<cuda_exec>(
  RangeSegment(0, 100), [=] (int i) {
    data[i] = i;
  }
);


RAJA::forall<seq_exec>(
  RangeSegment(0, 100), [=] (int i) {
    printf(“data[%g] = %f\n”,
           i, data[i]);
  }
);
```

CHAI arrays can be used on CPU or GPU, data migrates without user intervention

LvArray

Containers for use in high-performance simulation software

- **Containers**
  - A multi-dimensional array with a customizable memory layout and slicing.
  - A sorted unique list of values.
  - A jagged two-dimensional array.
  - A compressed row storage matrix and sparsity pattern.
- **All containers support customizable allocation behavior and work on device**
- **Integrates with RAJA and optionally CHAI**
- **Open source**
  - BSD license
  - Used by GEOSX ECP project

```
LvArray::Array<double,2,…> x(10, 11);

forall<POLICY1>(x.size(0),[x=x.toView()](int i)
{
  for(int j = 0; j < x.size(1); ++j )
    x(i, j) = foo(i, j);
} );

LvArray::Array<double,2,…> sums(x.size(0));
forall<POLICY2>(x.size(0),
[x=x.toViewConst(), sums=sums.toView()](int i)
{
  for(double value : x[i])
    sums[i] += value;
} );

sums.move(LvArray::MemorySpace::CPU);
std::cout << sums << std::endl;
```

When using CHAI `POLICY1` and `POLICY2` can be any RAJA policy and the data will migrate appropriately.

Application CS Infrastructure

Application CS Infrastructure

Technical Contact

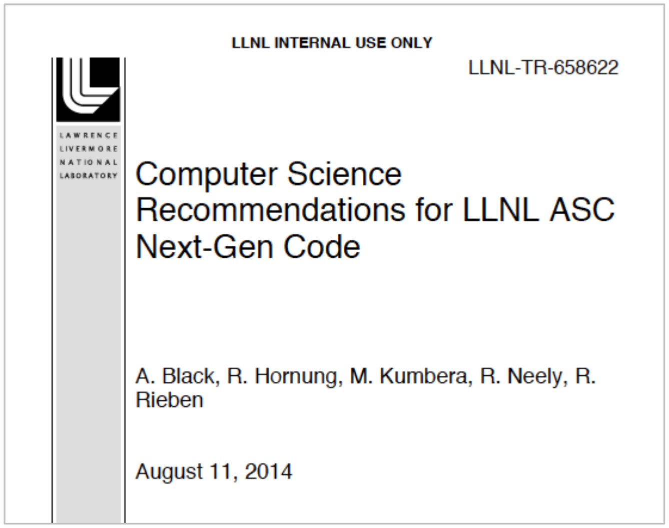
Rich Hornung

Project	Description	License	Maturity (years)	Website	Repository	Contact
Axom	Flexible software infrastructure for the development of multi-physics applications and computational tools	BSD	~5	software.llnl.gov/axom	github.com/LLNL/axom	Rich Hornung

Please direct detailed technical questions to the Axom developer team:
axom-dev@llnl.gov

Application CS Infrastructure (Axom)

- **Motivated by LLNL ASC next-generation code planning**
  - Core infrastructure for the LLNL ATDM code
  - Used across the LLNL ASC code portfolio
- **The report (at right) contains 50 recommendations spanning**
  - Software architecture and design
  - Software processes and tools
  - Software sharing and integration
  - Performance and portability
  - Co-design, external interactions, research
- **In development for 5+ years**
- **Open source**



LLNL INTERNAL USE ONLY

LLNL-TR-658622

LAWRENCE LIVERMORE NATIONAL LABORATORY

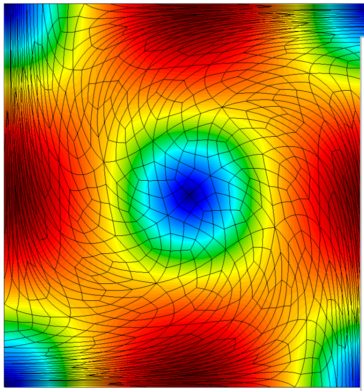
Computer Science Recommendations for LLNL ASC Next-Gen Code

A. Black, R. Hornung, M. Kumbera, R. Neely, R. Rieben

August 11, 2014

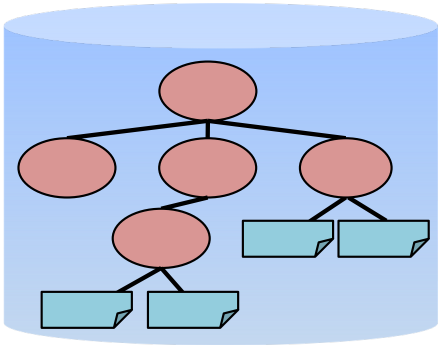
Application CS Infrastructure (Axom)

Mesh-aware data schema

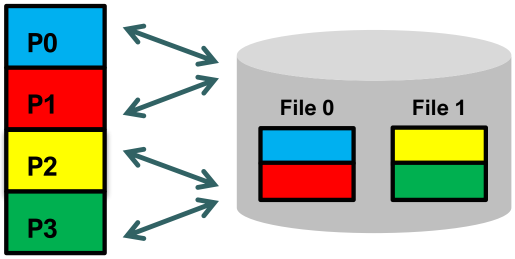


```
“coordsets”:  {
       “coords”:  {
           “type”:
“explicit”
           “values”:  {
              “x”:
[double],
              “y”:
[double]
           }
       }
    },
    “topologies”: {
       …
    …
```

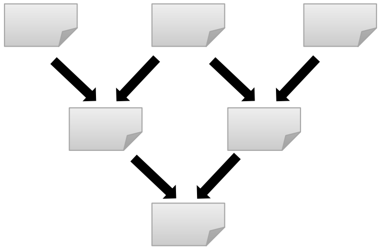
Hierarchical key-value in-memory datastore



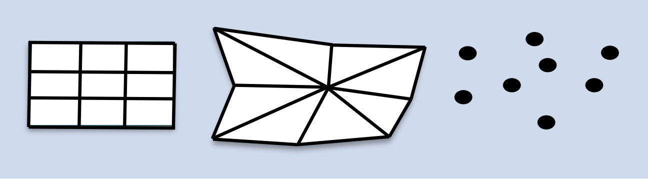
Parallel file I/O & burst buffer support



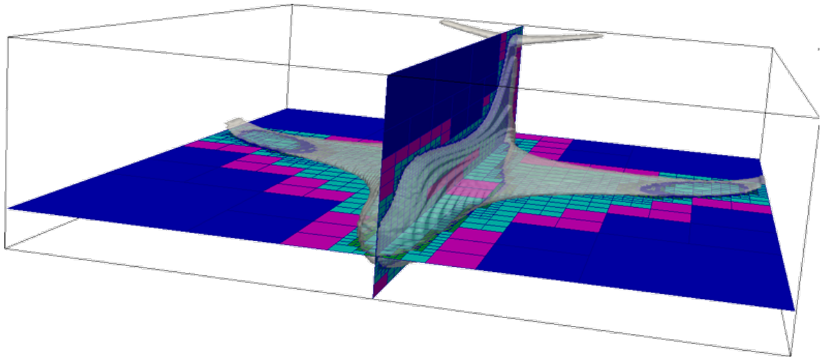
Unified inter/intra-package message logging & parallel filtering



Mesh data model

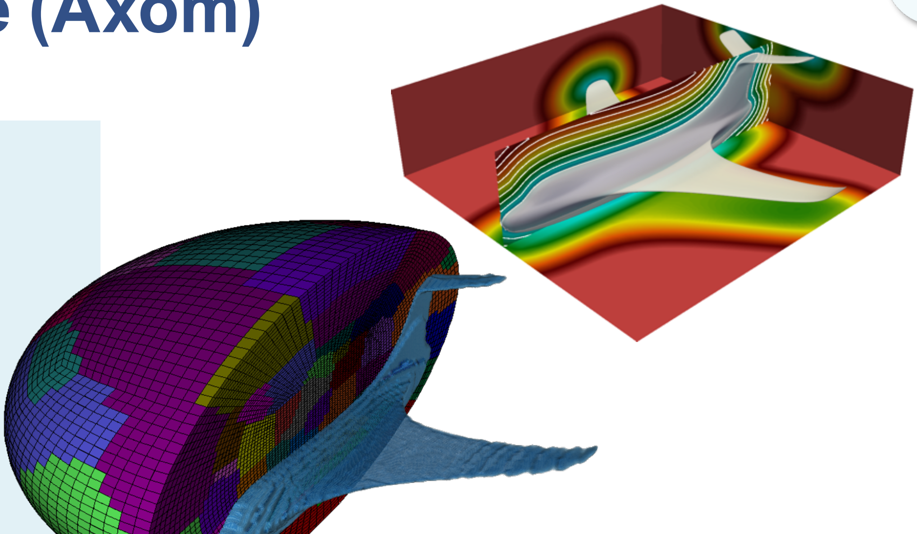


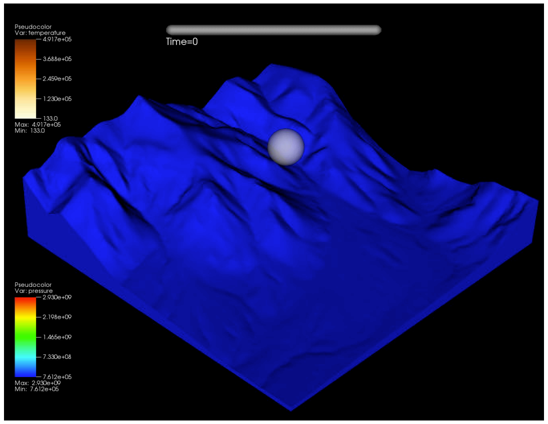
Surface queries & spatial acceleration data structures

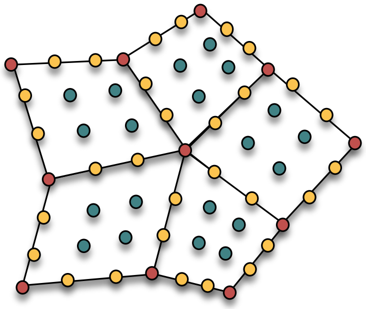


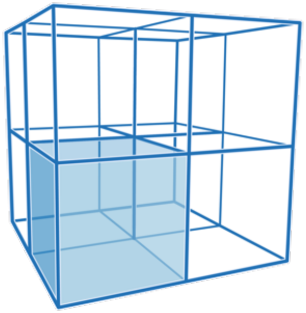
Application CS Infrastructure (Axom)

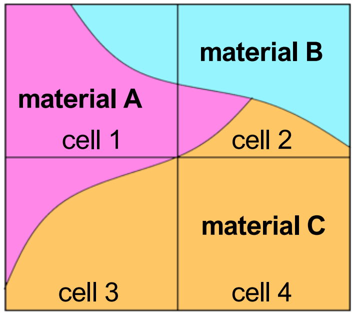
- **Examples of Axom application support**
  - Centralized, hierarchical simulation data management
  - Parallel file I/O for checkpoint-restart and visualization
  - Access to in-situ visualization and analysis tools
  - Shaping in arbitrary, complex material geometries
  - Immersed boundaries, interfaces
  - Building blocks for particle-based algorithms
  - Integrated cross-package parallel message logging











Application CS Infrastructure (Axom)

Axom Component	Description
Sidre	In-core hierarchical key-value data management, plus parallel file I/O (restart, viz. files), support for heterogeneous memory systems, etc.
Quest	Spatial point/surface queries; in-out, signed distance, point containment, point-in-cell, etc.
Primal	Geometric primitives (point, vector, triangle, etc.) and operations (distance, intersection, closest point, etc.)
Spin	Spatial acceleration data structures; octree, kd-tree, R-tree, BVH, etc.
Mint	Mesh data model; structured, unstructured, particles.
Slam	Set, relation, map abstractions.
Slic/Lumberjack	Unified/shared inter-package message streams, parallel logging, and filtering.

All Axom components provide native interfaces for C++, C, and Fortran (Python in the works).

Math + Physics Libraries

Math + Physics Libraries

Technical Contact

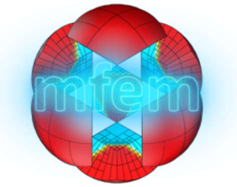
Tzanio Kolev

Project	Description	License	Maturity (years)	Website	Repository	Contact
MFEM	Unstructured high-order finite element library	BSD	~15	mfem.org	github.com/mfem	Tzanio Kolev
hypre	Preconditioners and solvers for large-scale matrices	Apache-2 or MIT	~20	www.llnl.gov/casc/hypre	github.com/hypre-space	Rob Falgout
SUNDIALS	Nonlinear and differential/algebraic equation solvers	BSD	~20	www.llnl.gov/casc/sundials	github.com/LLNL/sundials	Carol Woodward
SAMRAI	Structured Adaptive Mesh Refinement framework	LGPL-2.1	~20	computation.llnl.gov/projects/samrai	github.com/LLNL/SAMRAI	Noah Elliott
XBraid	Lightweight support for multigrid Parallel-in-Time	LGPL-2.1	~5	www.llnl.gov/casc/xbraid	github.com/xbraid	Rob Falgout

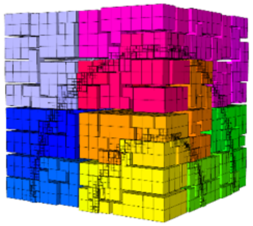
MFEM

Lightweight, scalable C++ library for finite element methods

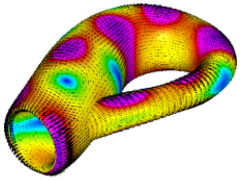
- **Supports arbitrary high-order discretizations and meshes for a wide variety of applications**
- **Flexible discretizations on unstructured grids**
  - Triangular, quadrilateral, tetrahedral and hexahedral meshes.
  - Local conforming and non-conforming refinement.
  - Bilinear/linear forms for variety of methods: Galerkin, DG, DPG, …
- **High-order and scalable**
  - Arbitrary-order H1, H(curl), H(div)- and L2 elements. Arbitrary order curvilinear meshes.
  - MPI scalable to millions of cores and GPU-accelerated. Enables application development on wide variety of platforms: from laptops to exascale machines.
- **Built-in solvers and visualization**
  - Integrated with: HYPRE, RAJA, UMPIRE, SUNDIALS, PETSc, SUPERLU, …
  - Accurate and flexible visualization with VisIt and GLVis
- **Open source**
  - BSD license with thousands of downloads/year worldwide.
  - Available on GitHub. Part of ECP's CEED co-design center.



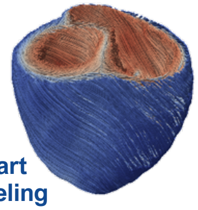
High-order curved elements



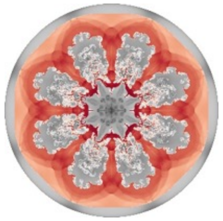
Parallel non-conforming AMR



Surface meshes



Heart modeling



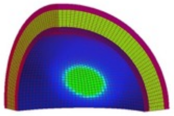
Compressible flow ALE simulations

Hypre

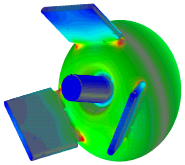
Highly scalable multilevel solvers and preconditioners

- **Conceptual linear system interfaces**
  - Provides natural “views” of the linear system: structured, semi-structured, finite element, linear algebraic
  - Enables more efficient data storage schemes and kernels
- **Scalable preconditioners and solvers**
  - Structured and unstructured algebraic multigrid (including constant coefficient)
  - Maxwell solvers, H-div solvers, and more
  - Demonstrated scalability beyond 1M cores
- **Integrated with other math libraries**
  - SUNDIALS, PETSc, Trilinos
- **Unique, user-friendly interfaces**
- **Open source**
  - Used worldwide in a vast range of applications
  - Available on GitHub, Apache-2 or MIT license

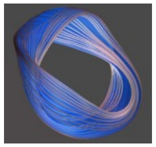
hypre high performance preconditioners



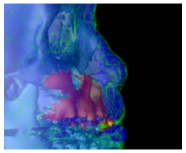
Elasticity / plasticity



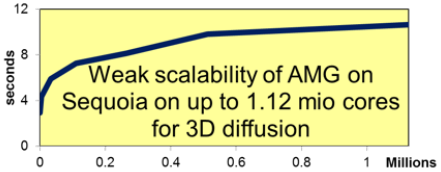
Electro-magnetics



Magneto-hydrodynamics



Facial surgery

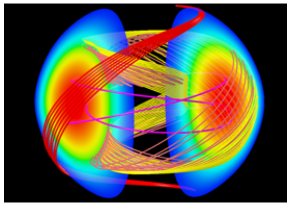


Weak scalability of AMG on Sequoia on up to 1.12 mio cores for 3D diffusion

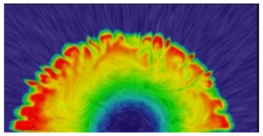
SUNDIALS

Adaptive time integrators for ODEs and DAEs and efficient nonlinear solvers

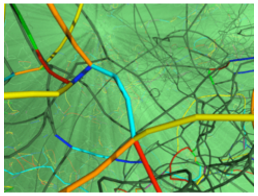
- **ODE integrators:**
  - CVODE(S): variable order and step BDF (stiff) and Adams (non-stiff)
  - ARKode: variable step implicit, explicit, and additive IMEX Runge-Kutta
- **DAE integrators**: IDA(S) - variable order and step BDF integrators
- **Sensitivity analysis (SA):** CVODES and IDAS provide forward and adjoint SA
- **Nonlinear solvers:** KINSOL - Newton-Krylov, Picard, and accelerated fixed point
- **Modular design**
  - Written in C with interfaces to Fortran
  - Users can supply own data structures and solvers
  - Optional use structures: serial, MPI, threaded, CUDA, RAJA, hypre, & PETSc
  - Encapsulated parallelism
- **Open source**
  - Freely available (BSD License) from LLNL site, GitHub, and Spack
  - CMake-based portable build system
  - Can be used from MFEM, PETSc, and deal.II
- **Supported by extensive documentation, a sundials-users email list, and an active user community**
- **Used by thousands worldwide in applications from research and industry**



Magnetic reconnection



Core collapse super-nova

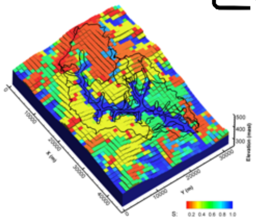


Dislocation dynamics

FASTMATH

ECP

xSDK



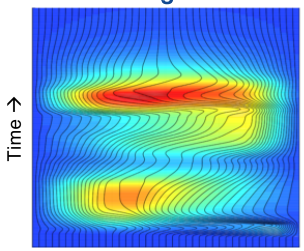
Subsurface flow

XBraid

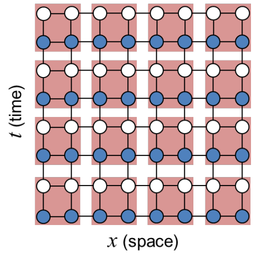
Parallel-in-time multigrid solver software

- **Speeds up existing application codes by creating concurrency in the time dimension**
- **Unique non-intrusive approach**
  - Builds as much as possible on existing codes and technologies
  - Converges to same solution as sequential code
- **Demonstrated effectiveness and potential**
  - Tech: Implicit, explicit, multistep, multistage, adaptivity in time and space, moving meshes, spatial coarsening, low storage approach
  - Apps: Linear/nonlinear diffusion, fluids (shocks), power grid (discontinuities), elasticity, optimization, …
  - Codes: Strand2D, Cart3D, LifeV, CHeart, GridDyn, …
- **Leverages spatial multigrid research and experience**
  - Extensive work developing scalable multigrid methods in hypre
- **Open source**
  - Available on GitHub, LGPL-2.1

Moving Mesh

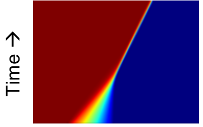


Multigrid in time

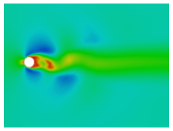


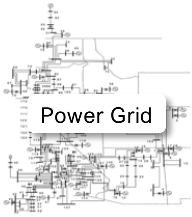
Parallelize space and time
Store several time steps

Inviscid Burgers



Navier-Stokes



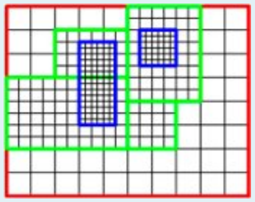


Up to 50x speedup on some problems (so far)

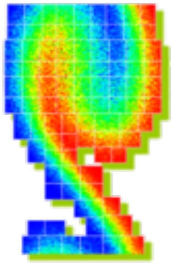
SAMRAI

Structured adaptive mesh refinement applications infrastructure

- **Object-oriented library, scalable and flexible for use in many applications**
- **Full support of AMR infrastructure**
  - Multi-level dynamic gridding of AMR mesh
  - Transparent parallel communication (MPI)
  - Load balancing
  - Data type for common mesh centerings (cell, node, face, . . .)
  - Data transfer operations (copy, coarsen, refine, time interpolation)

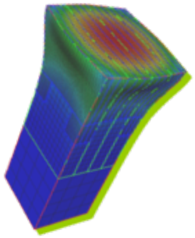


- **Flexibility provided to applications**
  - Applications provide numerical kernels to operate on distributed patches
  - Users may define and own their own data structures
  - Works on different geometries (Cartesian, staggered, multiblock, etc.)
  - Applications choose when and where to use SAMRAI data structures
  - Interfaces to solver libraries included (hypre, SUNDIALS, PETSc)
  - VisIt visualization and HDF5 checkpoint/restart supported
- **Open source**
  - LGPL 2.1 license, available on GitHub



Fixed geometry Eulerian methods

Lagrangian moving grids



Multi-physics applications

- RAJA threading interfaces and Umpire memory management for GPUs are being developed
- CMake-based build system coming soon

Performance and Workflow Tools

Performance and Workflow Tools

Technical Contact

Matthew LeGendre

Project	Description	License	Maturity (years)	Website	Repository	Contact
Caliper	Always-on performance measurement library	BSD	~5	llnl.github.io/Caliper/	github.com/LLNL/Caliper	David Boehme
SPOT	Performance history tracking	BSD	In development	computing.llnl.gov/projects/caliper	github.com/LLNL/Caliper	Matthew LeGendre
Flux	Resource management and scheduling	LGPL-3.0	~6	flux-framework.org	github.com/flux-framework	Dong Ahn
Maestro WF	A tool and library for specifying and conducting general workflows	MIT	~2.5	maestrowf.readthedocs.io	github.com/LLNL/maestrowf	Frank Di Natale
Spindle	Library loading and program start-up at scale	LGPL-2.1	~6	computing.llnl.gov/projects/spindle	github.com/hpc/spindle	Matthew LeGendre
LBANN	Machine learning training and inference at extreme scale	Apache-2	~5.5	lbann.readthedocs.io	github.com/LLNL/lbann	Brian Van Essen
Hatchet	Performance analysis for hierarchical data	MIT	~2	hatchet.readthedocs.io	github.com/LLNL/hatchet	Stephanie Brink

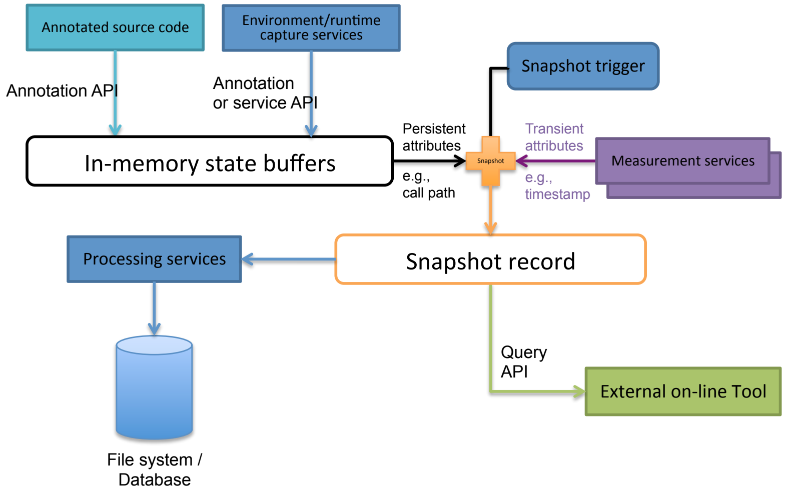
Caliper

A library for always-on performance monitoring

- **Add simple annotations to source code**
  - Physics regions, Key loops, other semantics

```
// Mark the "intialization" phase
CALI_MARK_BEGIN("initialization");
int count = 4;
double t = 0.0, delta_t = 1e-6;
CALI_MARK_END("initialization");

// Mark the loop
CALI_CXX_MARK_LOOP_BEGIN(mainloop, "main loop");

for (int i = 0; i < count; ++i) {
    // Mark each loop iteration
    CALI_CXX_MARK_LOOP_ITERATION(mainloop, i);

    // A Caliper snapshot taken at this point will contain
    // { "function"="main", "loop"="main loop", "iteration#main loop"=<i> }

    // ...
}

CALI_CXX_MARK_LOOP_END(mainloop);
```

- **Link code with Caliper library from C++, C, or Fortran**
- **Attach arbitrary performance measurement tools to your regions**
- **Leave Caliper in and *always* have performance data available**

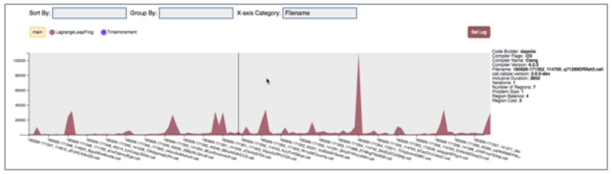


SPOT

Performance analysis and history tracking

- **Collect performance results from arbitrary application runs, track performance across users and history**
- **Integrate performance analysis tools into applications**
  - Annotate code regions with Caliper
  - Control performance collection through command line or input deck
  - Store history of performance data and visualize through web interfaces
- **Caliper interfaces with applications**
  - Annotation interface puts labels on code and data regions
  - Varity of metrics (time, memory bandwidth, MPI usage, etc.) are collected and reported against annotation labels.
  - More reliable that traditional performance tools.
- **SPOT visualizes history of Caliper-collected runs**
  - Any application run can report performance data to SPOT.
  - Track how performance changes with code releases and across systems
  - Explore performance data to identify issues
- **Under active development & integrated into several large codes**

Performance Dashboards



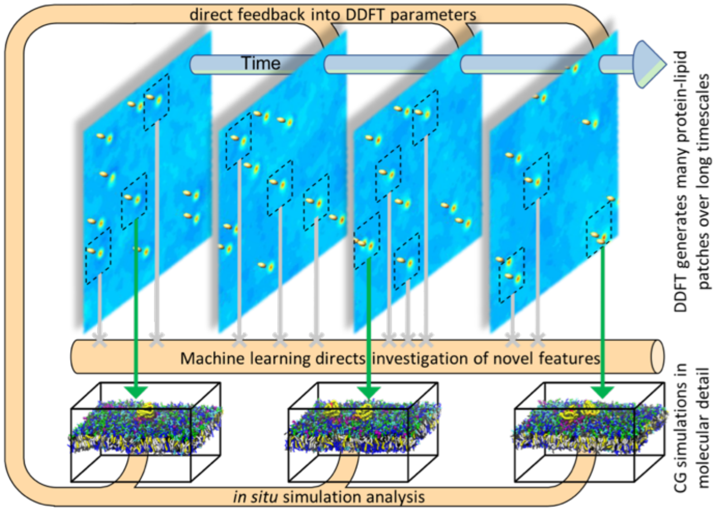
History Tracking

Drill-Down on Performance with Specialized Visualizations

Flux

Next-generation resource management and scheduling framework to address emerging challenges

- **Workflow challenges**
  - Modern Workflows are increasingly difficult to schedule
  - Cancer Moonshot Pilot2, Machine Learning LDRD Strategic Initiative, …
- **Resource challenges**
  - Changes in resource types are equally challenging
  - GPGPUs, Burst buffers, under-provisioned PFS BW, …
- **Fully hierarchical approach for job throughput/co-scheduling**
- **Graph-based resource model for resource challenges**
- **Rich APIs for workflow communication and coordination**
- **Consistent APIs for workflow portability and reproducibility**

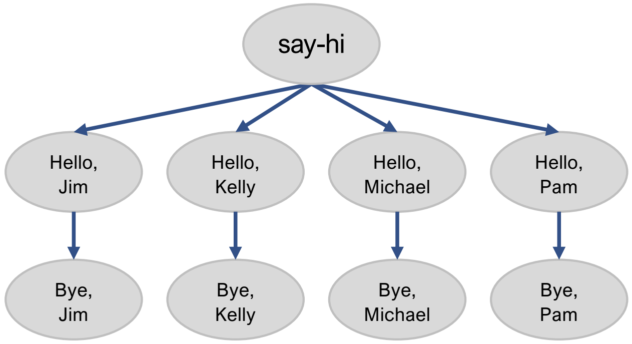

direct feedback into DDFT parameters
Time
Machine learning directs investigation of novel features
in situ simulation analysis
DDFT generates many protein-lipid patches over long timescales
CG simulations in molecular detail

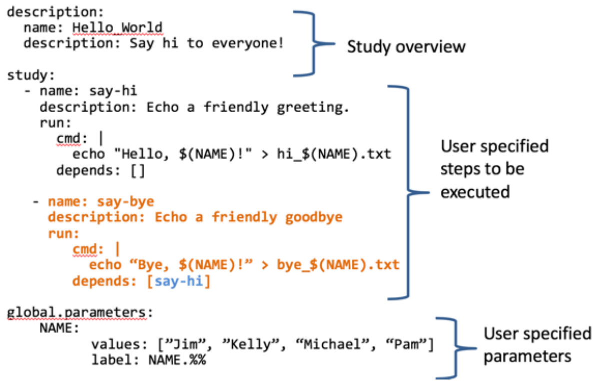
@FluxFramework

MaestroWF

A standard framework to make simulation studies easier to manage, run, and expand

- **Consistent study specification definition**
  - Specify multi-step workflows in a human-readable and self-documenting YAML specification.
  - Studies can be linear or parameterized, are easily shareable between users, and can be software generated.
  - Easily repeat studies simply by launching an existing specification.
- **Lightweight workflow automation and monitoring**
  - Studies are parsed, expanded based on parameters, and monitored automatically.
  - Workflows are expanded into DAGs, with workflow steps being launched as their dependencies allow them.
- **Easy for users to specify and launch workflows**
  - Specifications being shareable allows existing studies to serve as templates for new ones (making both set up and knowledge sharing easier).
  - A study specification allows users to build standard infrastructure to generate the necessary YAML to run larger collections of studies.





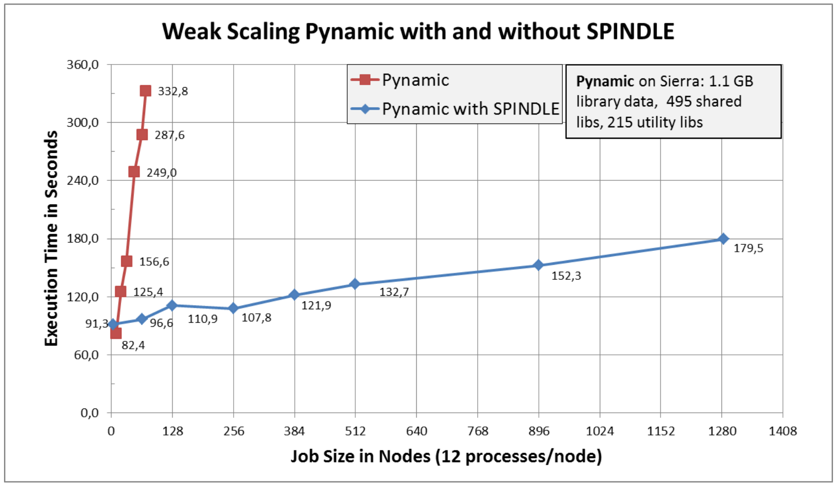
```
description:
  name: Hello_World
  description: Say hi to everyone!

study:
  - name: say-hi
    description: Echo a friendly greeting.
    run:
      cmd: |
        echo "Hello, $(NAME)!" > hi_$(NAME).txt
      depends: []

   - name: say-bye
     description: Echo a friendly goodbye
     run:
        cmd: |
          echo “Bye, $(NAME)!” > bye_$(NAME).txt
        depends: [say-hi]

global.parameters:
    NAME:
          values: [”Jim”, ”Kelly”, “Michael”, “Pam”]
          label: NAME.%%
```

Study overview
User specified steps to be executed
User specified parameters

Spindle

Scalable application start-up

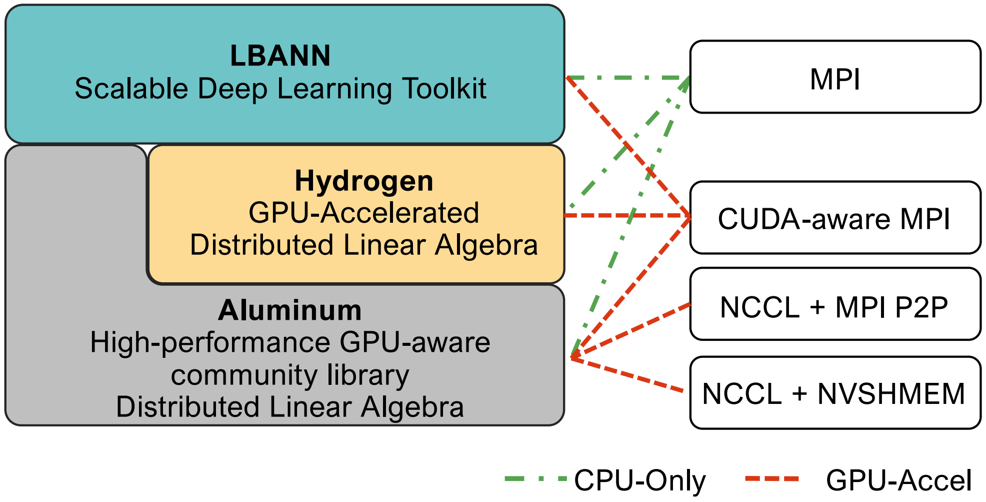
- **Job launch not scalable with many libraries or Python**
  - Solves start-up issues from loading libraries and Python modules at scale
  - Nodes hammer shared file systems when searching and loading libraries
  - Impacts users across whole center
- **Spindle makes job launch scalable**
  - Single node loads libraries/python-modules.
  - Broadcasts libraries to other nodes over high-bandwidth communication network.
  - Run by: `% spindle srun -n 512 ./myapp`
- **Open source**
  - LGPL-2.1 with thousands of downloads/year worldwide
  - Available on GitHub



LBANN

Livermore Big Artificial Neural Network Toolkit

- **Distributed deep learning training and inference**
  - Optimize for strong and weak scaling network training
  - Train large networks quickly
  - Enable training on data samples or data sets too large for other frameworks (e.g., 3D data cubes, billion sample data sets)
  - Optimized distributed memory algorithm
  - Including spatially decomposed convolutions
  - Multi-level parallelism (model / data / ensemble)
  - Hydrogen GPU-accelerated distributed linear algebra library
  - Optimized asynchronous GPU-aware communication library
- **Utilize unique HPC resources at scale**
  - InfiniBand and next-generation interconnect
    - Low latency / high cross-section bandwidth
  - Tightly-coupled GPU accelerators
  - Node-local NVRAM
  - High bandwidth parallel file system
- **C++ / MPI + OpenMP / CUDA / ROCm / NCCL / cuDNN**



LBANN
Scalable Deep Learning Toolkit

Hydrogen
GPU-Accelerated
Distributed Linear Algebra

Aluminum
High-performance GPU-aware
community library
Distributed Linear Algebra

MPI

CUDA-aware MPI

NCCL + MPI P2P

NCCL + NVSHMEM

CPU-Only GPU-Accel

- **Open source under Apache license**
  - github.com/LLNL/lbann
  - github.com/LLNL/Elemental
  - github.com/LLNL/Aluminum

Hatchet

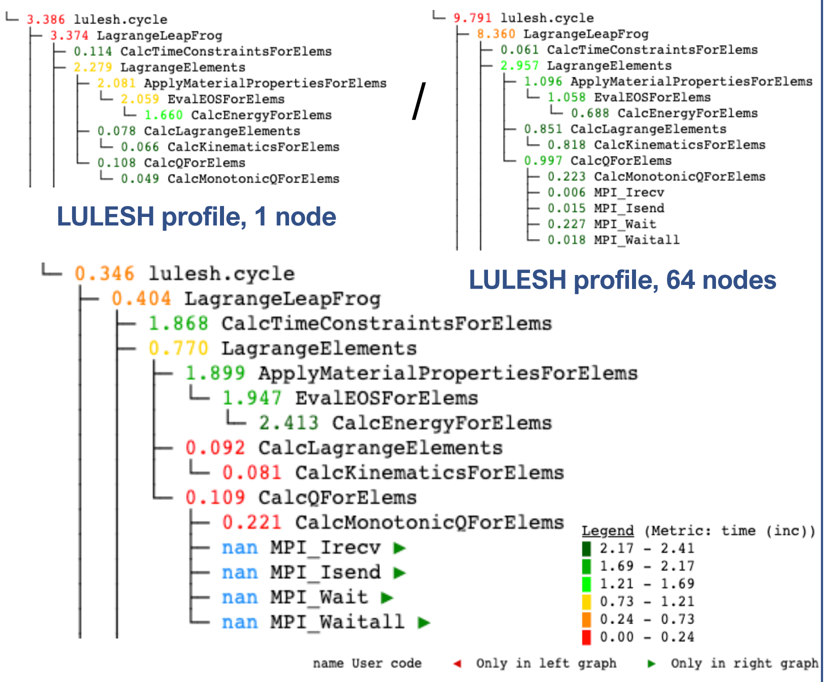
Performance analysis for hierarchical data

- **Hatchet enables programmatic analysis of hierarchical data**
  - Hatchet reads profiles from common HPC performance and tracing tools (*e.g.*, Caliper, HPCToolkit, gprof, Callgrind)
  - Hatchet users can write their performance analyses in Python scripts, leveraging the full Pandas API to perform operations

- **Identify performance bottlenecks to enhance application development**
  - *Compare* multiple execution profiles (*e.g.*, speedup of GPU to CPU performance, performance changes over time)
  - *Sub-select and focus* on a specific region of the data (*e.g.*, all MPI nodes, nodes with exclusive time > 10)

- **Open-source software**
  - Released with SPOT web-based visualization tool

Tree Visualizations in Hatchet



```
└─ 3.386 lulesh.cycle
   ├─ 3.374 LagrangeLeapFrog
   │  ├─ 0.114 CalcTimeConstraintsForElems
   │  ├─ 2.279 LagrangeElements
   │  │  ├─ 2.081 ApplyMaterialPropertiesForElems
   │  │  │  └─ 2.059 EvalEOSForElems
   │  │  │     └─ 1.660 CalcEnergyForElems
   │  │  ├─ 0.078 CalcLagrangeElements
   │  │  │  └─ 0.066 CalcKinematicsForElems
   │  │  └─ 0.108 CalcQForElems
   │  │     └─ 0.049 CalcMonotonicQForElems
```

LULESH profile, 1 node

/

```
└─ 9.791 lulesh.cycle
   ├─ 8.360 LagrangeLeapFrog
   │  ├─ 0.061 CalcTimeConstraintsForElems
   │  ├─ 2.957 LagrangeElements
   │  │  ├─ 1.096 ApplyMaterialPropertiesForElems
   │  │  │  └─ 1.058 EvalEOSForElems
   │  │  │     └─ 0.688 CalcEnergyForElems
   │  │  ├─ 0.851 CalcLagrangeElements
   │  │  │  └─ 0.818 CalcKinematicsForElems
   │  │  └─ 0.997 CalcQForElems
   │  │     ├─ 0.223 CalcMonotonicQForElems
   │  │     ├─ 0.006 MPI_Irecv
   │  │     ├─ 0.015 MPI_Isend
   │  │     ├─ 0.227 MPI_Wait
   │  │     └─ 0.018 MPI_Waitall
```

LULESH profile, 64 nodes

```
└─ 0.346 lulesh.cycle
   ├─ 0.404 LagrangeLeapFrog
   │  ├─ 1.868 CalcTimeConstraintsForElems
   │  ├─ 0.770 LagrangeElements
   │  │  ├─ 1.899 ApplyMaterialPropertiesForElems
   │  │  │  └─ 1.947 EvalEOSForElems
   │  │  │     └─ 2.413 CalcEnergyForElems
   │  │  ├─ 0.092 CalcLagrangeElements
   │  │  │  └─ 0.081 CalcKinematicsForElems
   │  │  └─ 0.109 CalcQForElems
   │  │     ├─ 0.221 CalcMonotonicQForElems
   │  │     ├─ nan MPI_Irecv ▶
   │  │     ├─ nan MPI_Isend ▶
   │  │     ├─ nan MPI_Wait ▶
   │  │     └─ nan MPI_Waitall ▶

Legend (Metric: time (inc))
█ 2.17 - 2.41
█ 1.69 - 2.17
█ 1.21 - 1.69
█ 0.73 - 1.21
█ 0.24 - 0.73
█ 0.00 - 0.24

name User code    ◀ Only in left graph    ▶ Only in right graph
```

Resulting speedup of 1 to 64 nodes

- **Open source under MIT license**
  - github.com/LLNL/hatchet

Data Management and Visualization

Data Management and Visualization

Technical Contact

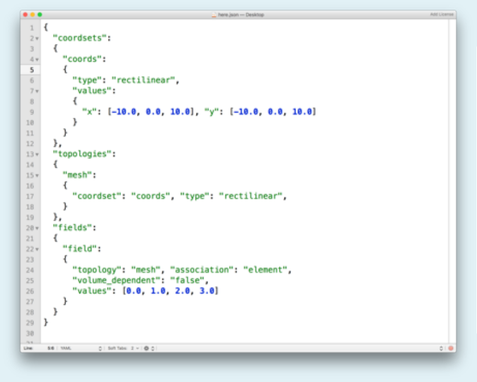
Cyrus Harrison

Project	Description	License	Maturity (years)	Website	Repository	Contact
Conduit	Simplified data exchange for HPC simulations	BSD	~6	software.llnl.gov/conduit	github.com/llnl/conduit	Cyrus Harrison
Ascent	Flyweight in situ visualization and analysis for HPC simulations	BSD	~4	ascent-dav.org	github.com/alpine-dav/ascent	Matt Larsen
zfp	In-memory compression of floating-point arrays	BSD	~6	zfp.readthedocs.io	github.com/LLNL/zfp	Peter Lindstrom
SCR	Multilevel checkpointing support and burst buffer interface	BSD	~13	scr.readthedocs.io	github.com/LLNL/scr/	Kathryn Mohror
VisIt	Feature-rich mesh-based visualization and analysis platform	BSD	~20	visit.llnl.gov	github.com/visit-dav/visit	Cyrus Harrison
GLVis	Lightweight high order visualization for MFEM	LGPL-2.1	~11	glvis.org	github.com/GLVis/glvis	Tzanio Kolev

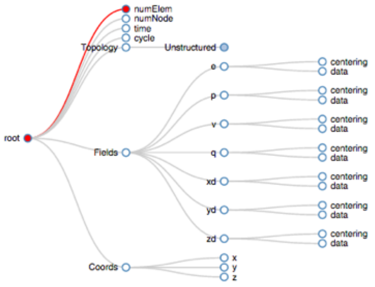
Conduit

Simplified data exchange for HPC simulations

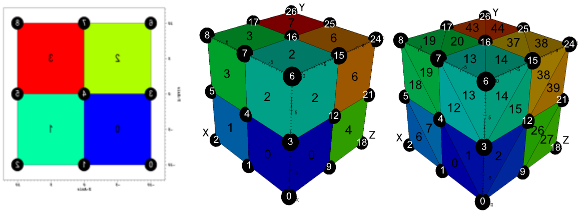
- **Provides an intuitive API for in-memory data description**
  - Enables *human-friendly* hierarchical data organization
  - Can describe in-memory arrays without copying
  - Provides C++, C, Python, and Fortran APIs
- **Provides common conventions for exchanging complex data**
  - Shared conventions for passing complex data (e.g., simulation meshes) enable modular interfaces across software libraries and simulation applications
- **Provides easy to use I/O interfaces for moving and storing data**
  - Enables use cases like binary checkpoint restart
  - Supports moving complex data with MPI (serialization)
- **Open source**
  - Leveraged by Ascent, VisIt, and Axom



```
{
  "coordsets":
  {
    "coords":
    {
      "type": "rectilinear",
      "values":
      {
        "x": [-10.0, 0.0, 10.0], "y": [-10.0, 0.0, 10.0]
      }
    }
  },
  "topologies":
  {
    "mesh":
    {
      "coordset": "coords", "type": "rectilinear",
    }
  },
  "fields":
  {
    "field":
    {
      "topology": "mesh", "association": "element",
      "volume_dependent": "false",
      "values": [0.0, 1.0, 2.0, 3.0]
    }
  }
}
```



Hierarchical in-memory data description



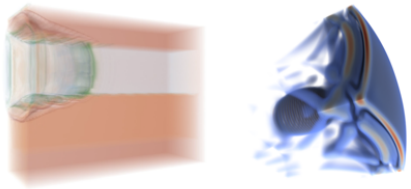
Conventions for sharing in-memory mesh data

Ascent

Flyweight in-situ visualization and analysis for HPC simulations

- **Ascent is an easy to use in-memory visualization and analysis library**
  - Use cases: **making pictures, transforming data,** and **capturing data**
  - Young effort, yet already supports most common visualization operations
  - Provides a simple infrastructure to integrate custom analysis
  - Provides C++, C, Python, and Fortran APIs
- **Uses a flyweight design targeted at next-generation HPC platforms**
  - Efficient distributed-memory (MPI) and many-core (CUDA or OpenMP) execution
  - Has lower memory requirements then current tools
    - Demonstrated scaling: In situ filtering and ray tracing across **16,384 GPUs** on LLNL's Sierra Cluster
  - Requires less dependencies than current tools (e.g., no OpenGL)
- **Open source**
  - Leverages Conduit, will also be released with Visit

Ascent



Visualizations created using Ascent

python™ HDF CINEMA

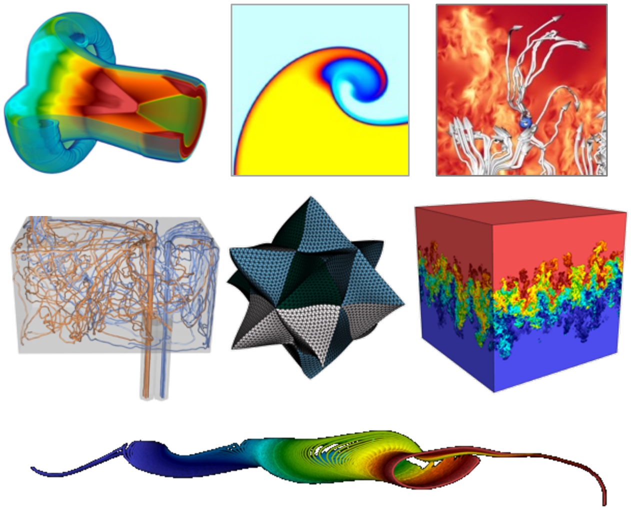
Extracts supported by Ascent

ASC ECP

VisIt

Full-featured visualization and analysis for HPC simulations

- **Production end-user tool supporting scientific and engineering applications**
  - Use cases: ***data exploration***, ***quantitative analysis***, ***visual debugging***, ***comparative analysis*** and generation of ***presentation graphics***
  - Provides a rich feature set and a flexible data model suitable for many scientific domains
  - Includes more than 100 file format readers
  - Provides GUI and Python interfaces, extendable via C++ and Python
- **Provides parallel post-processing infrastructure that scales from desktops to massive HPC clusters**
  - Uses MPI for distributed-memory parallelism on HPC clusters
  - Development underway to leverage on-node many-core (CUDA or OpenMP) parallelism
- **Open source**
  - Used as a platform to deploy research from the DOE visualization community
  - Initially developed by LLNL to support ASC, now co-developed by several organizations

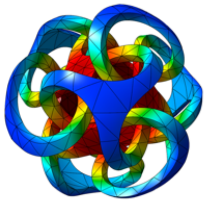


Visualizations created using VisIt

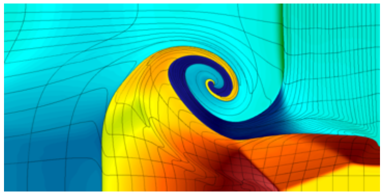
GLVis

Lightweight OpenGL tool for accurate and flexible interactive finite element visualization

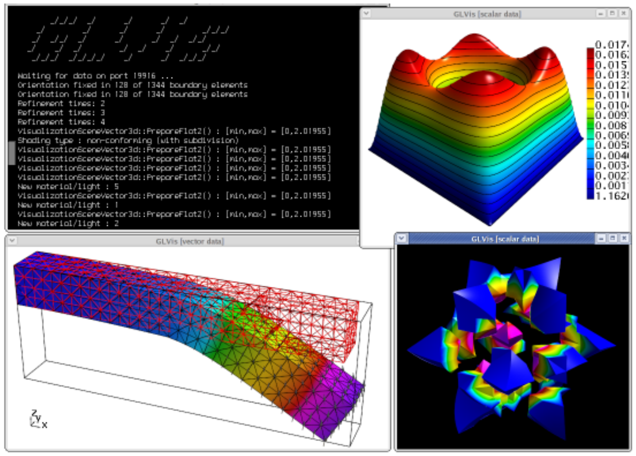
- **Accurate visualization**
  - 1D/2D/3D, volume/surface, triangular/quad/tet/hex, low/high-order meshes
  - Arbitrary high-order, scalar and vector finite element and NURBS solutions
  - Visualization of parallel meshes and solutions
- **Lightweight and interactive**
  - Unlimited number of refinement and de-refinement levels
  - Support for antialiasing, accurate cutting planes, materials, lighting, and transparency
  - Processor and element shrinking for better visualization of 3D mesh interiors
- **Flexible server support**
  - Simultaneous visualization of multiple fields/meshes in separate GLVis windows
  - Local visualization for remote parallel runs with secure socket connections
  - Persistent visualization of time-evolving fields
- **Open source**
  - LGPL-2.1. Available on GitHub
  - Based on the MFEM finite element library
  - Used in MFEM, MARBL/BLAST, LiDO, and more



Supports general meshes and fields



Visualization of a time-dependent high-order BLAST simulation



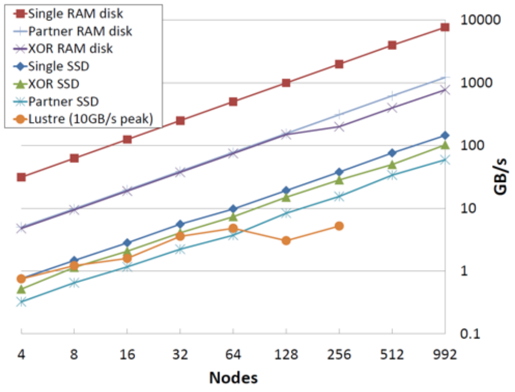
```
Waiting for data on port 19916 ...
Orientation fixed in 128 of 1344 boundary elements
Orientation fixed in 128 of 1344 boundary elements
Refinement times: 2
Refinement times: 3
Refinement times: 4
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
Shading type : non-conforming (with subdivision)
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
New material/light : 5
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
New material/light : 1
VisualizationSceneVector3d::PreparePlot2() : [min,max] = [0,2.01955]
New material/light : 2
```

GLVis server sessions with multiple windows

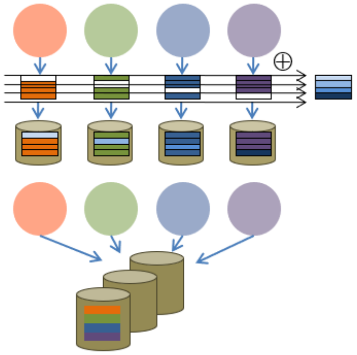
Scalable Checkpoint/Restart (SCR) Library

Enables fast, portable I/O to burst buffers across HPC systems

- **SCR provides fast, scalable I/O performance for LLNL applications**
  - SCR caches output data in node local storage like RAM disk or burst buffer, which can be as much as 1000x faster than the parallel file system
  - SCR hides the complexity of different burst buffer systems and storage architectures
- **Easy integration into application codes**
  - Simple wrapper API around existing checkpoint/restart code
  - Full featured scripting tools wrap existing job launch commands, e.g. srun → scr_srun
- **SCR now enables fast I/O for general output from applications**
  - SCR can now cache visualization dumps or other output to node local storage and drain data to the parallel file system in the background
  - Applications can output data more frequently without the overhead
- **Open source**
  - Available on GitHub with BSD license



SCR's I/O strategies scale with the number of nodes in an HPC job

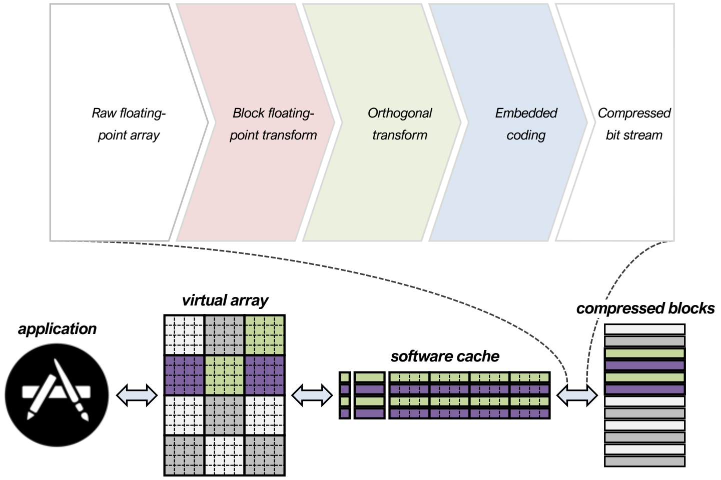


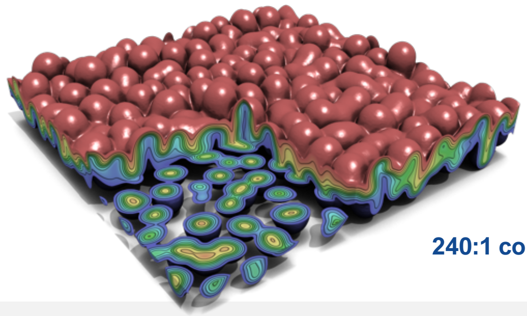
SCR protects your data and manages the complexity of HPC storage hierarchies for performance portable I/O

ZFP

In-memory compression of floating-point and integer arrays

- **Provides a conventional array interface for multidimensional scalar fields**
  - Supports constant-time read & write random access to any array element
  - Hides complexity of (de)compression via C++ operator overloading
  - Provides efficient data access via iterators, views, proxy references and pointers
  - Supports thread safe access and STL algorithms
- **Provides a simple API for (de)compression of whole arrays**
  - Supports prescribed error tolerance or precision, exact storage, lossless compression
  - Supports OpenMP and CUDA parallel (de)compression at up to 150 GB/s throughput
  - Provides C++, C, Python, and Fortran APIs
  - Suitable for compressing checkpoints, viz dumps, MPI messages, CPU-GPU transfers
- **Open source**
  - BSD licensed and available via GitHub, Spack, and Fedora RPM
  - Supported by Intel IPP, HDF5, Silo, ADIOS, VTK-m, LEOS, E4S, …





240:1 compression

radiuss.llnl.gov

lc.llnl.gov/confluence/display/RAD/RADIUSS

Lawrence Livermore National Laboratory

Disclaimer
This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.