



# S4PST: Stewardship of Programming Systems and Tools

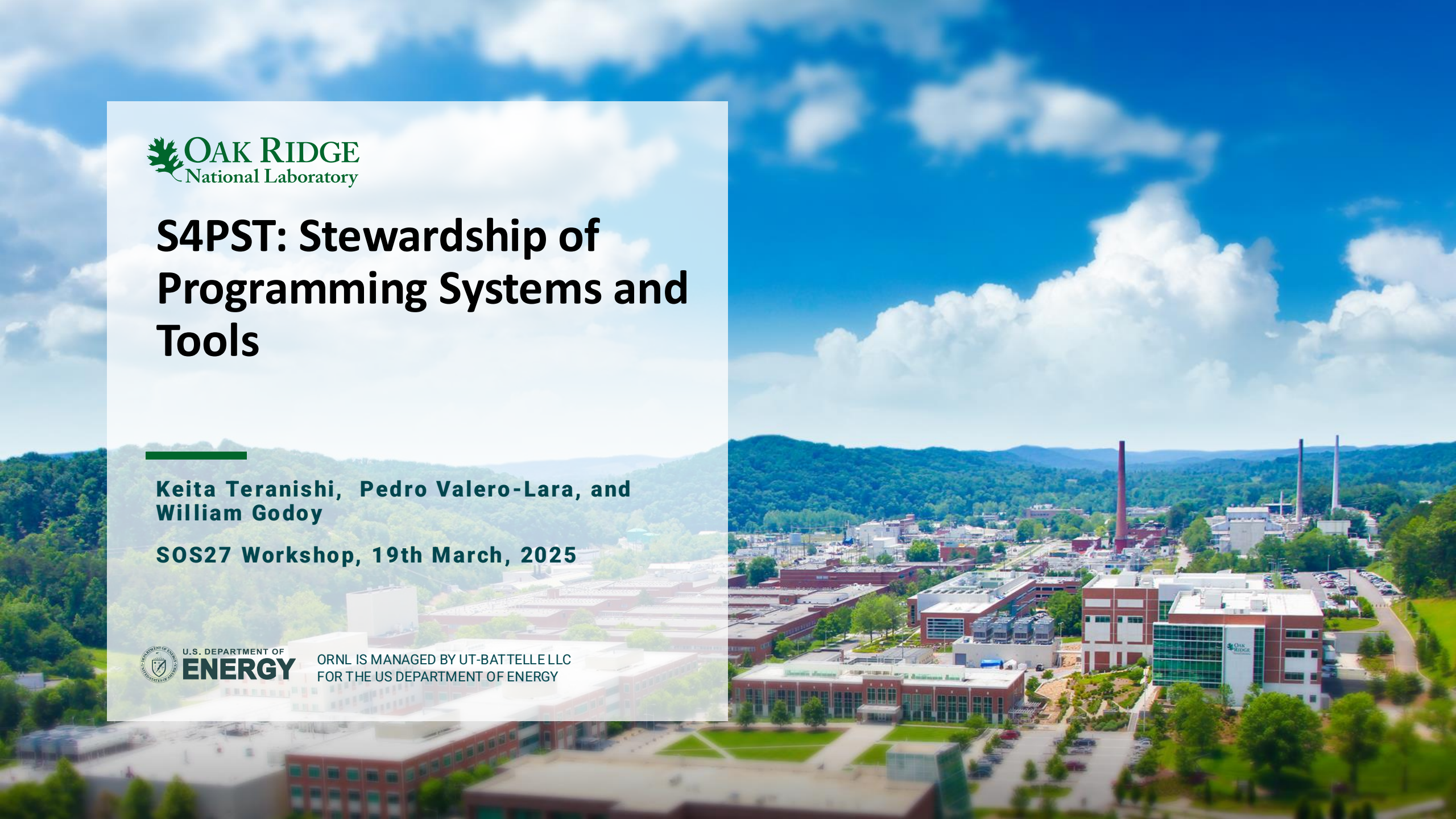
**Keita Teranishi, Pedro Valero-Lara, and  
William Godoy**

**SOS27 Workshop, 19th March, 2025**



U.S. DEPARTMENT OF  
**ENERGY**

ORNL IS MANAGED BY UT-BATTELLE LLC  
FOR THE US DEPARTMENT OF ENERGY



# Acknowledgement

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Next-Generation Scientific Software Technologies program, under contract number DE-AC05-00OR22725.



# Scientific Computing Software Development During Exascale Computing Project (ECP) and Post-ECP Era

## US DOE Exascale Computing Project (2016-2023)

- **Software Technology Project:**
  - Development of 70 software products
  - Annual budget of \$70M.
- **Programming Systems:**
  - Developed over 10 programming system products.
- **Supercomputing Infrastructure**
  - All DOE supercomputing systems employ accelerator technology.

## Post-ECP Initiative

- **End of 2022: Software Sustainability Proposal Call**
  - Issued by the US DOE Advanced Scientific Computing Research (ASCR)
- **April 2023: Phase 1 – Incubation**
  - 6 projects initiated
- **Fall 2023: the Next Generation Scientific Software Technology (NGSST)**
  - Four existing projects and one new project were recommended for funding
- **January 2024: Phase 2 Commencement**
  - Annual funding of \$11.5M allocated through 2028.



# 7 (6+1) Software Stewardship Organizations

**NGSST** is the next step after the successful DOE Exascale Computing Project (2016-2023) as we evolved into “Heterogeneous and AI dominated landscape”

Toward a Post-ECP  
Software  
Sustainability  
Organization (PESO)

Michael  
Heroux  
(SNL)

Collaboration for  
Better Software  
(COLABS)

Anshu  
Dubey  
(ANL)

Open Scientific  
Software  
Foundation (CORSA)

Greg  
Watson  
(ORNL)

Center for Sustaining  
Workflows and  
Application Services

Rafael  
Ferreira  
Da Silva  
(ORNL)

Stewardship for  
Programming  
Systems and  
Tools (S4PST)

Keita  
Teranishi  
(ORNL)

Sustainable Tools  
Ecosystem Project  
(STEP)

Terry  
Jones  
(ORNL)

OASIS (SciDAC  
FastMath and  
RAPIDS-2)

Rob  
Ross  
(ANL)

# Consortium for the Advancement of Scientific Software (CASS)

CASS and its member organizations work with our software product teams to improve the quality, sustainability, and interoperability of the software products in our ecosystem

The CASS software catalog covers a range of freely available libraries supporting leading-edge computational science and engineering research on high-performance computers. Most products available via Spack in the E4S distribution.

## Engage with us:

- Learn about the impacts of CASS software
- Join our announcement mailing list
- Participate in our working groups
- Reach out to a member organization responsible for specific areas of the software ecosystem of interest
- Become a CASS member. We welcome projects and organizations with similar scientific software stewardship missions

## Software Catalog

### Areas

*Number of products in parentheses*

#### Data and visualization (10)

*Provide capabilities for analyzing, visualizing, compressing, moving, and managing data*

#### Development tools (6)

*Help address performance challenges and facilitate efficient support of the latest HPC node architectures*

#### Mathematical libraries (14)

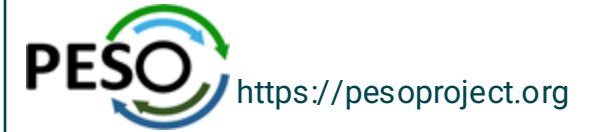
*Scalable libraries supporting the latest HPC architectures for coupled systems, ensemble calculations, and more*

#### Programming models and runtimes (11)

*Support intra-node and inter-node concurrency on current and next-generation HPC node architectures*

#### Software ecosystem and delivery (2)

*Provide software build, test, and integration tools, and containers environments*



## Scientific software ecosystem benefits (technical and community)

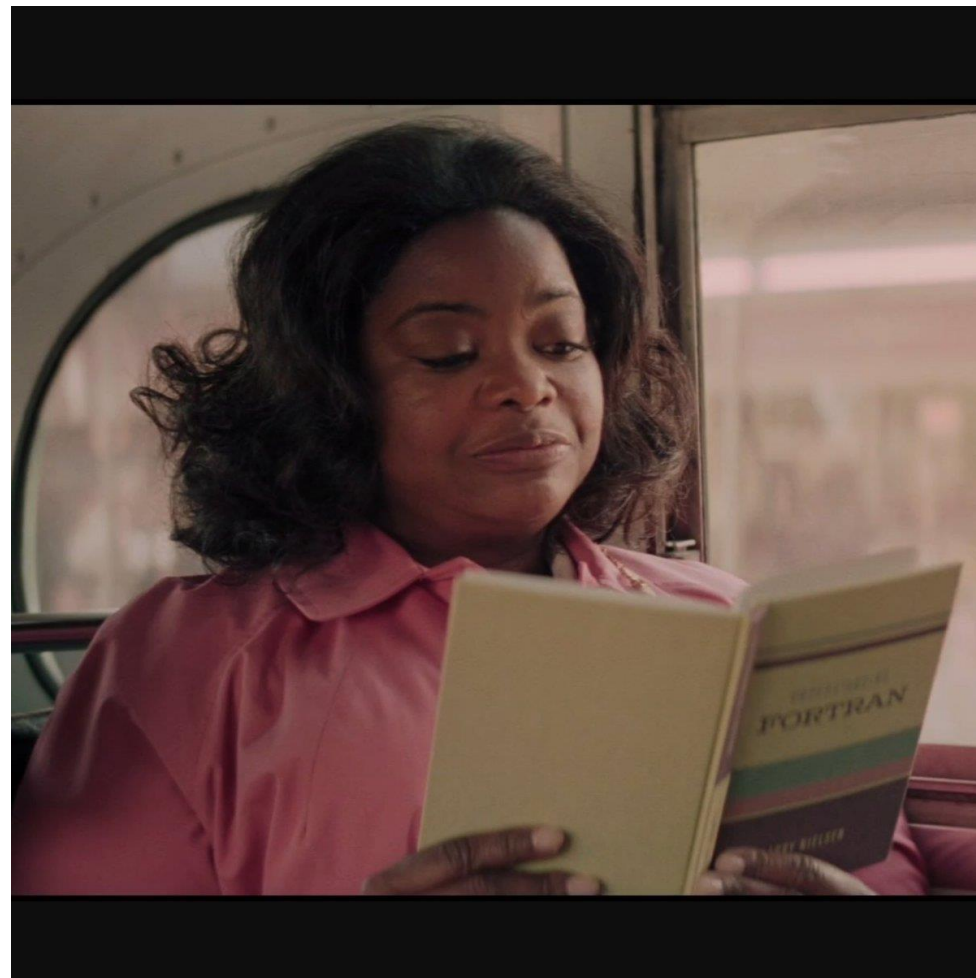


# Why do we care about programming systems?

From *Hidden Figures*, 2016.

“FORTRAN is a new and exciting language used by programmers to **communicate with computers**. It is exciting as **it is the wave of the future**.” -- Octavia Spencer as Dorothy Vaughan, Langley Research Center, NASA in 1961.

Today, **we continue to use programming systems as a critical tool to communicate with computers and shape the future.**

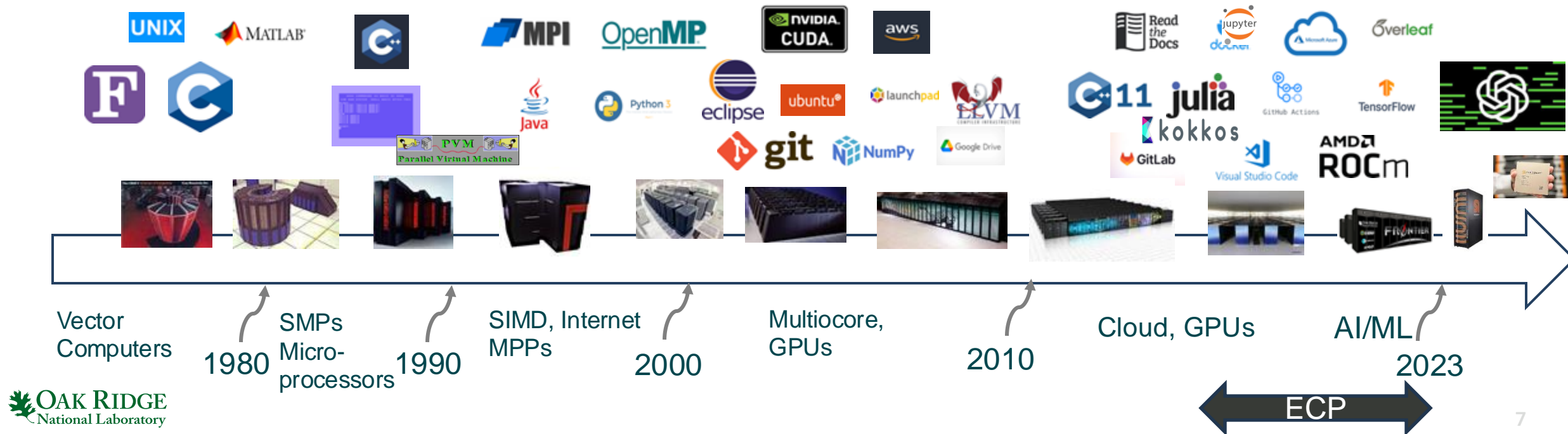


Courtesy: Damian Rouson at LBL and Kengo Nakajima at RIKEN/U of Tokyo.



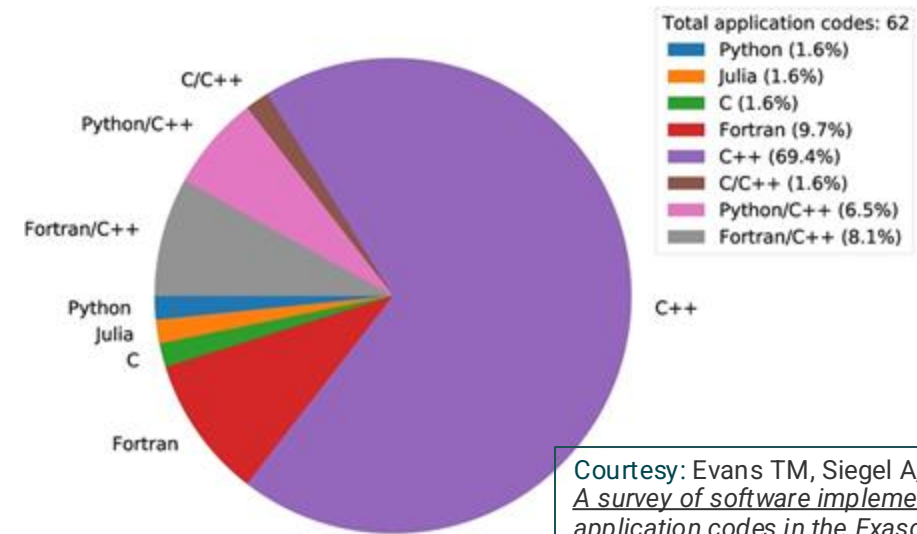
# Why do we care about programming systems?

- Programming Systems are the first point of contact in human-computer interactions
- The DOE HPC programming environment is notoriously challenging:
  - **Key Facilities: OLCF, ALCF, NERSC**
  - **Historical Challenges in Programming:**
    - 80s: Vectorization is complex (Cray), SMP programming is significantly challenging (Cray YMP)
    - 90s: SIMD programming is intricate (CM-2/CM-5), SPMD is hard to debug (PVM and MPI)
    - 00s: Multicore Programming (hybrid of MPI + X) remains difficult
    - 10s: GPU Programming adds more complexities (discrete memory spaces, massive SIMD, etc.)
    - 20s: Domain specific accelerators, Dataflow Architecture? (Sambanova, Cerebras, etc.)
  - Scientific discovery, software development, and hardware technology never cease to **evolve**

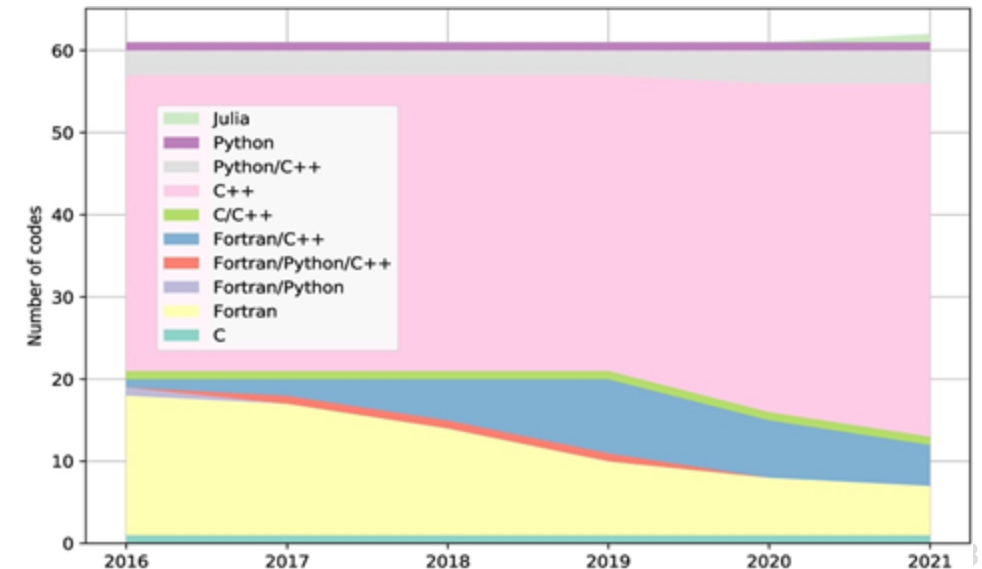
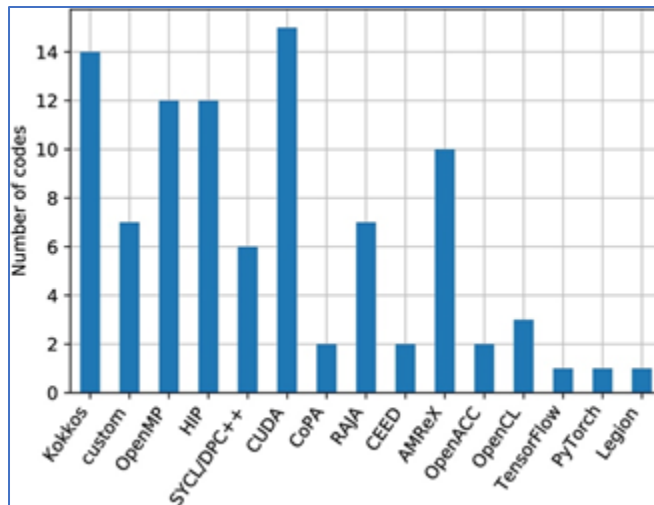


# Observations during ECP

- Adapting to Heterogeneous Computing Norms
  - NVIDIA, AMD and Intel GPUs
- Enhancing Productivity Through Performance Portability
  - Same source code across different platforms
- Evolving Programming Systems
  - **C++ has become dominant language for ECP apps**
  - **High productivity Languages, AI/ML**
- Legacy Applications
  - Many Fortran applications are still actively used in other DOE programs.



Courtesy: Evans TM, Siegel A, Draeger EW, et al.  
*A survey of software implementations used by application codes in the Exascale Computing Project*. IJHPCA. 2022;36(1):5-12





# Observations during ECP (Continued)

## HPC Vendor Shift

- Transitioning from traditional HPC to AI/ML-focused systems.

## Cloud HPC Growth

- Cloud-based solutions are overtaking on-premises HPC.

## DOE's Custom Solutions

- Features compilers, runtime, APIs, and testing suites with fast support.
- Committed to standardizing C++, Fortran, OpenMP/ACC, MPI to meet DOE's specific needs.
- Aims to establish a unique position within the broader open-source software community.

## Strategic Solution

- Embrace a community-wide, proactive approach to new technology integration.

# What is S4PST?

- **Objective:** Enhance Programming Systems for next-generation high-performance computing (HPC) systems and ensure their seamless integration with emerging AI technologies for scientific advancement.
- **Scope:** Focus on Performance Portable Parallel Programming Frameworks, Compilers, Distributed Computing Framework, and High Productivity Languages.
- **Challenges:**
  - **Technical:** Requires unique skill sets, including comprehensive knowledge of full-stack technologies, system architecture, and application needs.
  - **Diversified Portfolio:** Commitment to serving existing users and computing systems while embracing new technologies, meeting evolving application demands, and fostering the next generation of HPC experts.
  - **People:** Dedicated engagement is essential for the development of individual products, support of users, and mentoring of emerging talents (future HPC experts and leaders).

# S4PST Community

- PI: Keita Teranishi (ORNL)
- CoPI: William Godoy and Pedro Valero Lara
- 7 National Laboratories:
  - Oak Ridge National Laboratory
  - Argonne National Laboratory
  - Lawrence Livermore National Laboratory
  - Lawrence Berkeley National Laboratory
  - Sandia National Laboratories
  - Los Alamos National Laboratory
  - SLAC National Accelerator Laboratory
- University Partners:
  - University of Delaware
  - Massachusetts Institute of Technology
- Collaborations:
  - Louisiana State University
  - Pacific Northwest National Laboratory
  - Carnegie Mellon University
  - University of Tennessee, Knoxville
  - Stanford University
  - Other 6 NSSGT projects



Keita Teranishi  
(ORNL)



Pedro Valero-  
Lara (ORNL)



William Godoy  
(ORNL)



Christian Trott  
(SNL)



Damien Lebrun-  
Grandie (ORNL)



Brice Videau  
(ANL, ALCF)



Damian Rouson  
(LBL)



Johannes Doerfert  
(LLNL)



Sunita  
Chandrasekaran  
(UD,BNL)



Johannes Blaschke  
(LBL,NERSC)



Pat McCormick  
(LANL)



Alex Aiken (SLAC)



Paul Hargrove  
(LBL)



Katherine  
Rasmussen (LBL)



Joel Denny  
(ORNL)



Thomas Naughton  
(ORNL)



Suzanne Parete-  
Koon (ORNL)



Swaroop  
Pophale (ORNL)



Michel Schanen  
(ANL)



Hui Zhou (ANL)



Rajeev Thakur  
(ANL)



Ignacio  
Laguna (LLNL)



Seyong Lee  
(ORNL)



Thomas  
Applencourt  
(ANL,ALCF)



Ken Raffanetti  
(ANL)



Yanfei Guo  
(ANL)



Siva  
Rajamanickam  
(SNL)



Rabab  
Alomairy (MIT)



Alan Edelman  
(MIT)



Jan Hückelheim  
(ANL)



Georgios  
Georgakoudis  
(LLNL)



# S4PST: Our portfolio for DOE's Scientific Mission

## Product Stewardship

- OpenMP/OpenACC
- LLVM
- Fortran
- Kokkos
- MPICH
- Open MPI
- Legion
- GASNet-EX/UPC++
- HIP
- SYCL

## Emerging Technologies

- Automatic Differentiation (AD)
- HPC+AI ecosystems: Julia, Mojo, Python
- Large language models for HPC
- Kokkos ecosystems adapting AI-hardware/special arithmetic units



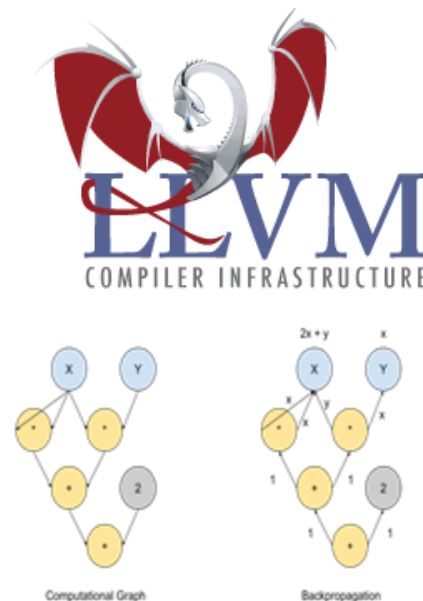
# S4PST Activities

- Advancement of Programming Systems
  - Development for new node architectures and interconnects
  - Adapting new architecture features
  - Incorporating AI technology and emerging programming systems
- Software Maintenance
  - Addressing bugs
  - Providing version updates
  - Integrating multiple programming systems as PESO and E4S
- Standardization
  - Fortran: Emphasizing DO CONCURRENT, Fortran 2023
  - MPI: Including MPICH and OpenMPI with version of MPI-4.2 and 5.0
  - OpenMP/OpenACC: Advancing OpenMP 6.0



# S4PST Activities (Continued)

- Vendor Independent Test Suites
  - OpenMP and OpenACC
- Incubation
  - Automatic Differentiation (Enzyme)
  - High Productivity Language (Julia)
  - Adapting AI/ML techniques
- Community Engagement
  - Documentation
  - Tutorials, Hackathons, Webinars
  - Example Programs
  - HPC Representative (LLVM)
  - Tutorials (SC, ISC, HPC-ASIA, DOE-MEXT)
- Synergistic Efforts with other SSOs and HPSF
- Website
  - <https://s4pst.org>



Mojo 



The Julia logo, featuring the word "julia" in a lowercase, black, sans-serif font. Above the letters "i", "l", and "a" are small, colored circles in blue, green, and purple respectively.



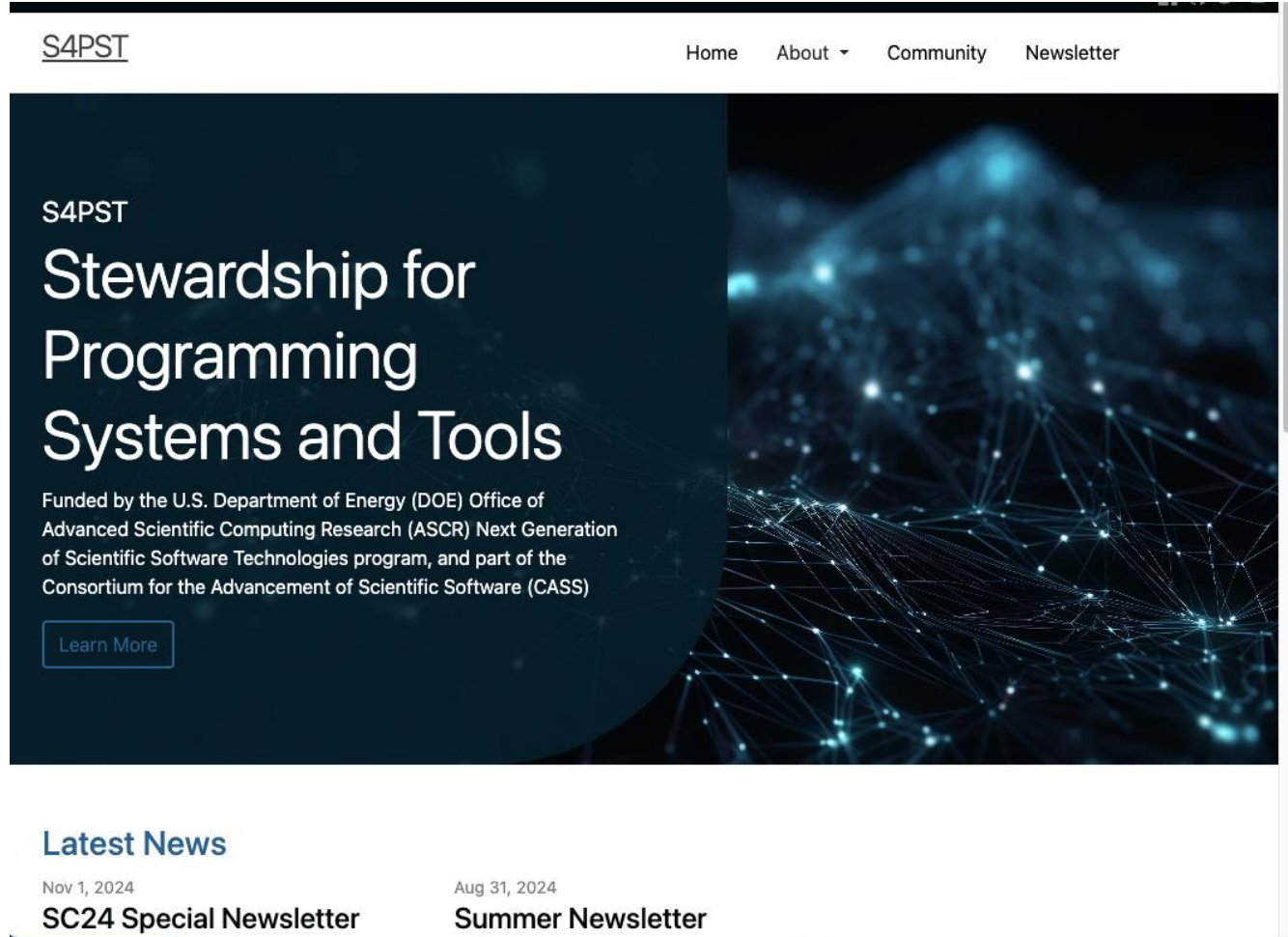
# S4PST Highlight



## S4PST.org

S4PST was involved in:

- 4 Tutorials
- 4 Publications
- 1 Invited Presentation
- 1 Research Poster
- 9 BoF Events
- Several Booth Talks

A screenshot of the S4PST website. The header includes the "S4PST" logo and navigation links: "Home", "About", "Community", and "Newsletter". The main content area features a large dark blue banner with the text "S4PST Stewardship for Programming Systems and Tools". Below this, it states: "Funded by the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) Next Generation of Scientific Software Technologies program, and part of the Consortium for the Advancement of Scientific Software (CASS)". A "Learn More" button is present. The background of the banner shows a network of glowing blue nodes and lines. Below the banner, there is a "Latest News" section with two items: "SC24 Special Newsletter" dated "Nov 1, 2024" and "Summer Newsletter" dated "Aug 31, 2024".

S4PST

Home About Community Newsletter

S4PST

## Stewardship for Programming Systems and Tools

Funded by the U.S. Department of Energy (DOE) Office of Advanced Scientific Computing Research (ASCR) Next Generation of Scientific Software Technologies program, and part of the Consortium for the Advancement of Scientific Software (CASS)

[Learn More](#)

### Latest News

Nov 1, 2024 SC24 Special Newsletter

Aug 31, 2024 Summer Newsletter

# S4PST – Highlights

## Software Release:

- Kokkos 4.5,
- OpenMPI 5.0.2, MPICH 4.3
- OpenMP 5.2 and 6.0 Verification Suites
- OpenACC 3.3 Verification Suites
- ChipStar v1.1 (HIP on Aurora)
- Julia 1.1
- JACC (Julia for Accelerators)



## Outreach:

- S4PST Presentation in Japan (HPC-AI Council)
- Kokkos User Group Meeting at SNL, NM.
- Kokkos Developers Meeting at ORNL
- Kokkos Presentation and Tutorial in Japan
- LBL hosted Fortran Standards Committee Meeting in Berkeley
- ANL hosted IWOCL24 in Chicago
- MPICH and OpenMPI teams collaborate Forum for **Abstract Binary Interface** standardization
- SIAM PP24: OpenMP, LLVM, Kokkos, Fortran, Julia
- SC24 BoFs, Tutorials,
- SIAM CSE25: Julia, CASS, etc.

## Technical Accomplishments:

- **Kokkos-3** won IEEE TPDC Best Paper Award
- **Legion** runs on 8,000 nodes of Frontier
- ComPILE **LLVM-IR AI model** released
- 5+ IJHPCA Journal paper submissions

## Collaborations

- OpenACC Specification Committee
- OpenMP Specification Committee
- Enzyme team and several universities under NSF project
- SciDAC Next Generation Power Grid Analysis

## International Collaboration:

- DOE-MEXT (Japan) on Fortran and Kokkos
- Kokkos (DOE team) - CEA Collaboration
- ADAC: The Accelerated Data Analytics and Computing Institute



# Thank you!



ORNL IS MANAGED BY UT-BATTELLE LLC  
FOR THE US DEPARTMENT OF ENERGY

# S4PST Product Introduction (Extra Materials)

---

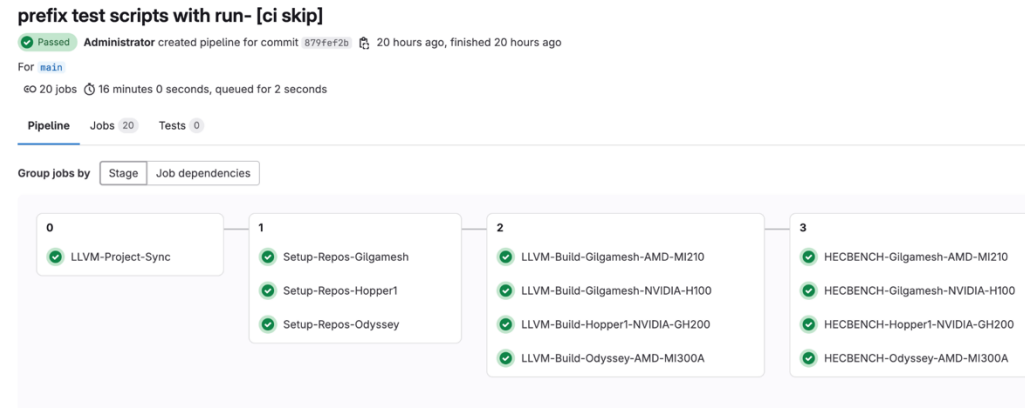
# OpenMP and OpenACC (UD, ORNL and ANL)



Lead: Sunita Chandrasekaran  
U of Delaware & BNL

## Project Overview

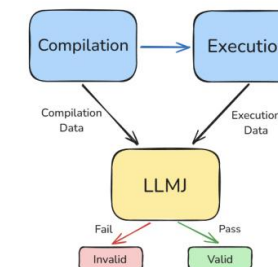
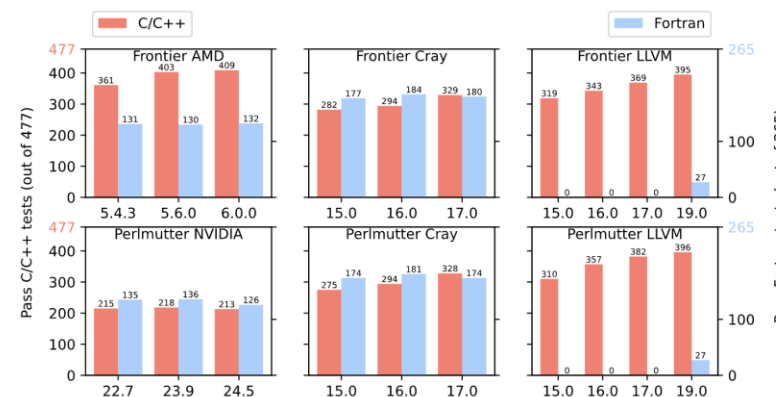
- Standardization, implementation, validation, benchmarking, code migration
- Automate test generation and explore its applicability for other models such as SYCL, CUDA and Kokkos



```
#pragma acc data copyin( a[0:n], b[0:n] ), copyout( c[0:n] )
{
    #pragma acc kernels
    {
        #pragma acc loop independent
        for ( i = 0; i < n; i++ ) {
            c[i] = a[i] + b[i];
        }
    }
}
```

## Synergistic Activities

- Close coordination with the LLVM and MLIR community along with E4S and SPACK



```
1 /* Host function that performs SAXPY on the device */
2 void saxpy_device(float a, float x[], float y[], int size)
3 {
4     #pragma omp target map(to: a,size,x[0:size]) \
5         map(tofrom: y[0:size])
6 {
7     int i;
8     #pragma omp parallel for
9     for (i = 0; i < size; i++)
10         y[i] = a * x[i] + y[i];
11 }
12 }
```

## Evolution of OpenMP Compilers

## An Overview of Agent-based Approach for LLMJ

# LLVM in S4PST (LLNL, ANL, LBNL, ORNL)

Lead:  
Johannes Doerfert  
LLNL

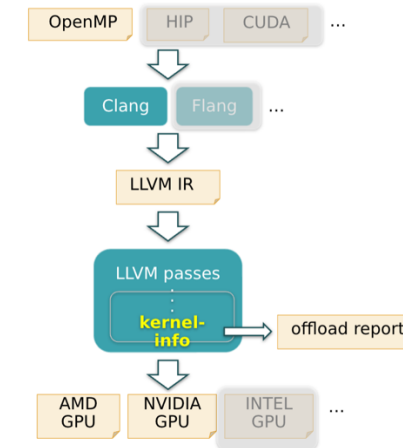


## Project Overview

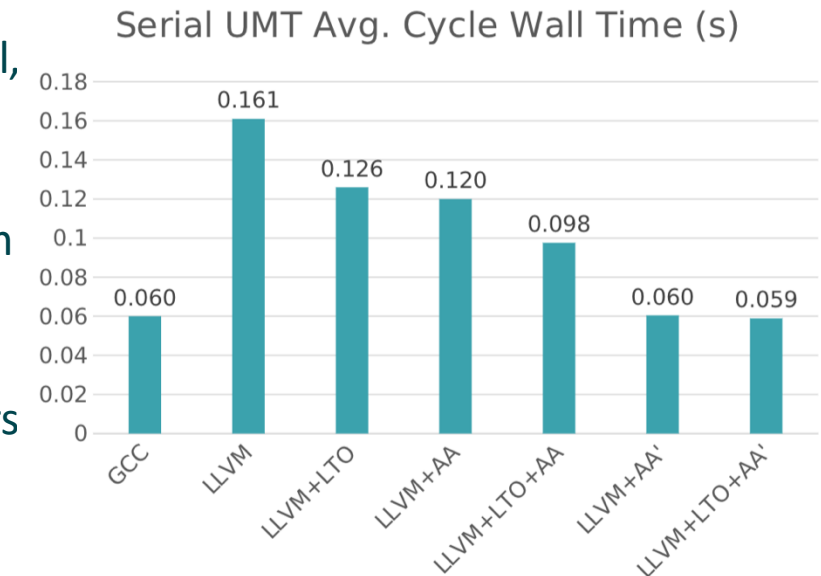
- Base compiler for all HPC vendors; vehicle for DOE to collaborate with academia and industry, innovate and productize compiler and toolchain technology
- Focus on DOE application issues, esp. wrt. Perlmutter, Frontier, Aurora, and El Capitan
- Offload for LLVM/Flang and native C++ parallelism; collaborations with AMD and others

## Accomplishments

- Creating of LLVM/Offload; portability across devices and interoperability between programming languages; backed by AMD, Apple, Codeplay, Intel, HPE, and others
- Improvement of Alias analysis of LLVM/Flang (+ performance )
- Analysis tool of LLVM-IR to detect potential performance bugs in program source
- ML in LLVM; ComPile dataset created; foundation models and GPU specific tuning strategies will follow; collaboration with Google and others



*The LLVM Kernel Info pass that produces a report for offload/GPU code. Known problematic code patterns are explicitly listed with their source location. Profile information (PGO) is integrated for FLOP calculations.*





# Kokkos Core – Maintaining Performance Portability

## Kokkos AI – MLIR and future AI application support



Lead: Christian Trott (SNL)

### Product Description:

- Pure C++ library targeting CUDA, HIP, SYCL, OpenMP and others
- Used by applications and libraries to write single source parallel code for heterogeneous architectures
- Used at more than 150 institutions by hundreds of projects



Kokkos-AI Lead:  
Siva Rajamanickam  
SNL

### Accomplishments

- Development of Kokkos Dialect in LLVM MLIR
- Kokkos 4.5 Release. (Full SYCL Support)

### Synergistic Efforts

- HPSF CI Working group and support efforts could be contributed to, and leveraged for Kokkos Core
- Collaboration with CEA, France
- Collaboration with RIKEN, Japan



# Modern Fortran (LBNL)

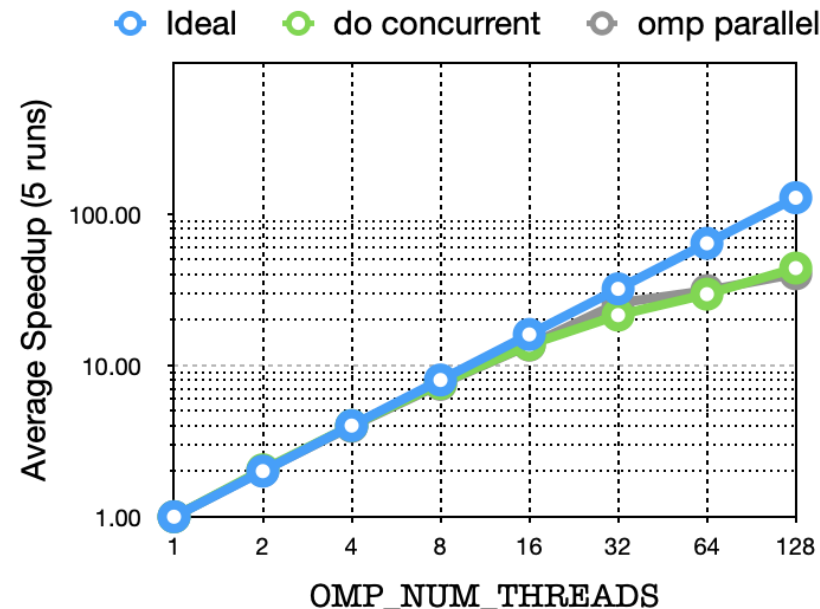


Lead: Damian Rouson, LBNL

## Project Overview

- There are ~15 actively developed or maintained compilers:
  - Open-source: GNU, LFortran, LLVM, ...
  - Vendors: Arm, AMD, HPE (Cray), IBM, Intel, NAG, NVIDIA, ...
- Fortran 2023 was published in November:
  - Runtime-agnostic parallelism for shared or distributed memory
  - Concurrent iterations supporting automatic GPU-offloading

- The next standard will include
  - Asynchronous task-based parallelism
  - Type-safe generic programming for better error messages
- DOE funds support LLVM Flang and launched LFortran



```
! Outline the objects in the binary image
!$omp target data map(to:image) map(from:edge_mask)
!$omp target
!$omp parallel do
do j = 1, n
  do i = 1, n
    edge_mask(i, j) = .false.
    if (image(i, j) /= 0) then
      if (i == 1 .or. i == n .or. &
         j == 1 .or. j == n) then
        edge_mask(i, j) = .true.
      else
        if (any(image(i-1:i+1, j-1:j+1) == 0)) edge_mask(i, j) = .true.
      endif
    endif
  enddo
enddo
!$omp end target
!$omp end target data
```

do concurrent facilitates optimizations, including automatic GPU-offloading

```
integer, allocatable :: image(:, :)
logical, allocatable :: edge_mask(:, :)

! Allocate image and edge mask
allocate (image(n, n), source = 0, stat = allocstat, errmsg = allocmsg)
allocate (edge_mask(n, n), source = .false., stat = allocstat, errmsg = allocmsg)

! Initialize image

! Outline the objects in the binary image
do concurrent (j = 1:n, i = 1:n, image(i, j) /= 0)
  if (i == 1 .or. i == n .or. &
      j == 1 .or. j == n) then
    edge_mask(i, j) = .true.
  else
    if (any(image(i-1:i+1, j-1:j+1) == 0)) edge_mask(i, j) = .true.
  endif
enddo
```

# Open MPI (ORNL) and MPICH (ANL)

## Project Overview

- Full support of DOE's Supercomputing Systems
  - GPU, Resource Management, Batch Systems
- GPU support and performance tuning
- MPI ABI standardization
- MPI Standard (Forum) activities

## Synergistic efforts

- PETSc (external work to optimize support for GPU communication)
- OpenPMIx: Implements PMIx interface for resource managers/process launch
  - Support/Engagements/Use: SLURM, OpenPBS, Flux, OpenSHMEM, DEEP-SEA project, etc.

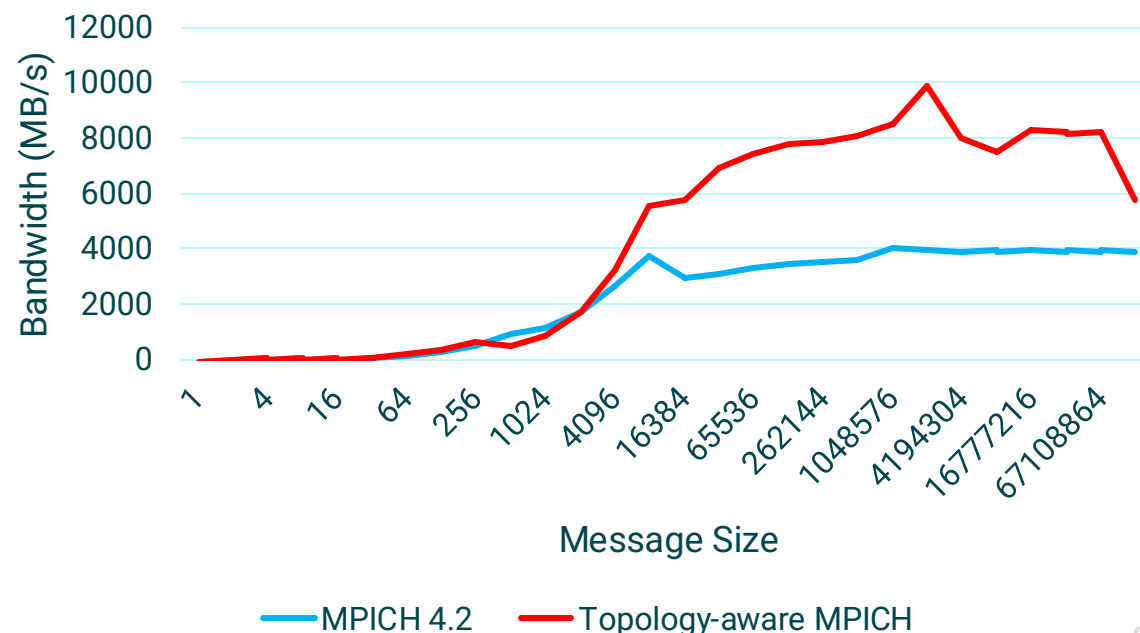
OpenMPI Lead: Thomas Naughton III  
ORNL



MPICH Lead: Yanfei Guo  
ANL



Inter-NUMA Bandwidth on Sunspot



# UPC++ & GASNet-EX (LBNL)



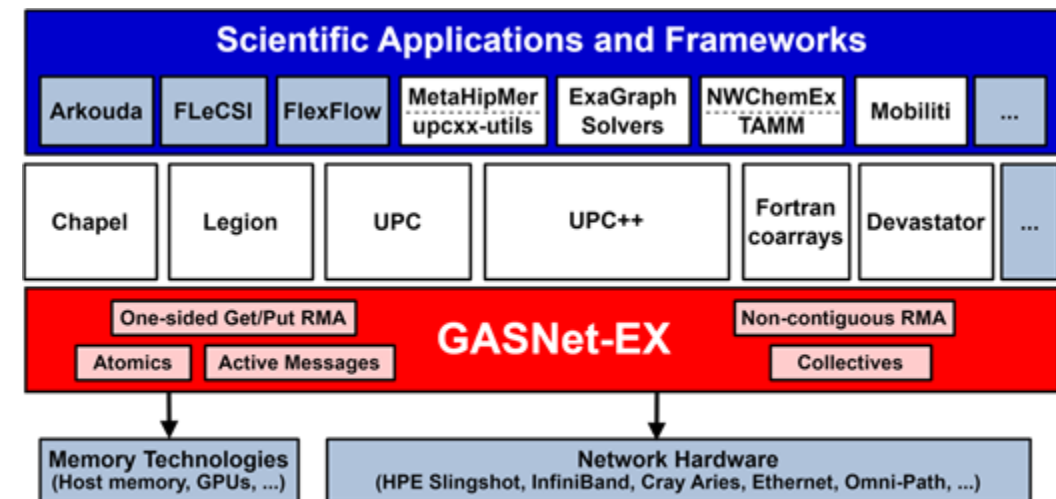
Lead: Paul H. Hargrove (LBL)

## Description and Capabilities

- GASNet-EX provides communication middleware for scalable Partitioned Global Address Space (PGAS) and distributed asynchronous many-task (AMT) programming systems
  - Remote Memory Access (RMA) and Active Messages (AM)
  - Direct RMA to/from accelerator memory (e.g. GPUs)
- UPC++ provides high-level productivity abstractions over GASNet-EX through modern C++ features to enable PGAS programming and Remote Procedure Call (RPC)

## Synergistic Efforts

- Legion (including within S4PST)
- Flang (external work to implement parallel Fortran features)





# Legion (LANL,SLAC)



Distributed, asynchronous task parallel programming framework

- Support for CPUs & GPUs
- Used in production on Frontier, Perlmutter, and Summit

Applications and libraries developed under ECP, supported under S4PST:

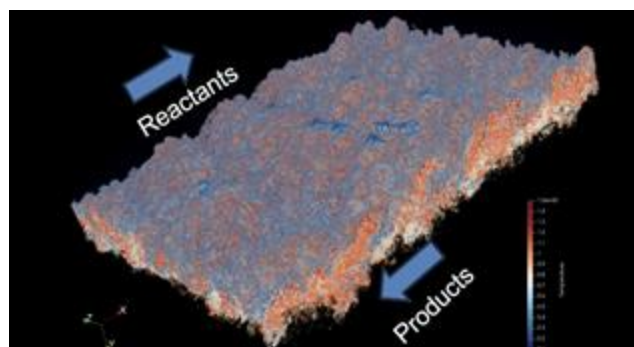
- S3D (SNL)
- FlexFlow (CMU/LANL/Stanford/UCSD)
- SpiniFEL (SLAC)
- New work: support cuNumeric on non-NVIDIA machines

Synergistic Efforts

- Legion is core technology for NVIDIA's Legate and cuNumeric products
- GASNet-EX for low latency communication (including within S4PST)



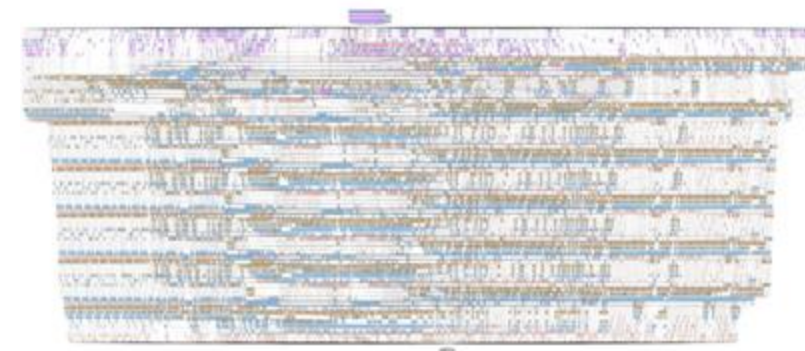
S3D Scalability on Frontier



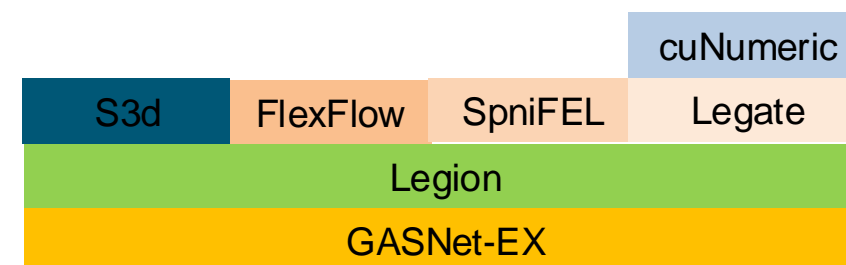
Visualized output from S3D



Co-Leads: Pat McCormick (LANL) and Alex Aiken (SLAC)



S3D Task Graph



Legion Software in S4PST's scope



# SYCL (ALCF, OLCF, NERSC)

## Performance Portable Programming Systems based on Modern C++

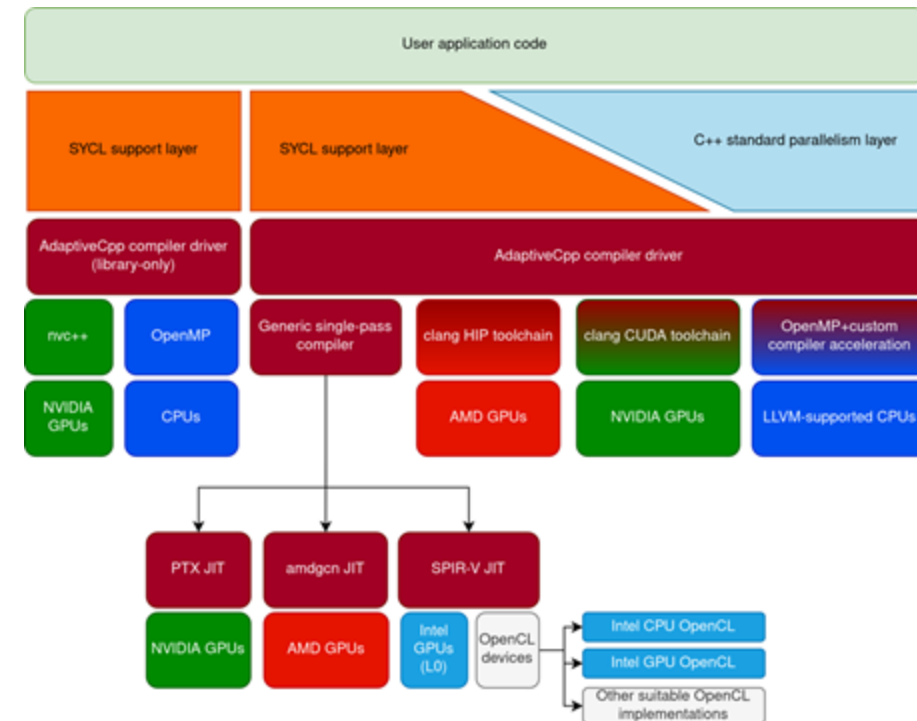
- Successor of OpenCL (Open Standard) and Support CPUs, GPUs and FPGAs
- Leverage LLVM compiler toolchain
- Two major Open-Source implementations: Intel's DPC++ and AdaptiveCpp

## Synergistic efforts

- HIP (HIP is a backend of SYCL)
- Kokkos (Kokkos have a SYCL backend)
- LLVM (LLVM is the infrastructure used by all SYCL implementations)



Lead: Thomas Applencourt, ALCF



# HIP (ALCF, NERSC, OLCF)

## Heterogeneous-Computing Interface for Portability

Project Goals: HIP is AMD's Portable GPU programming model

- Very similar to CUDA
- Single source C++ language, with C runtime library
- Large ecosystem of math/support libraries
- DOE implementation in collaboration with Intel: chipStar
  - LLVM + SPIR-V + Level-Zero/OpenCL
  - Intel math libraries

### Accomplishments:

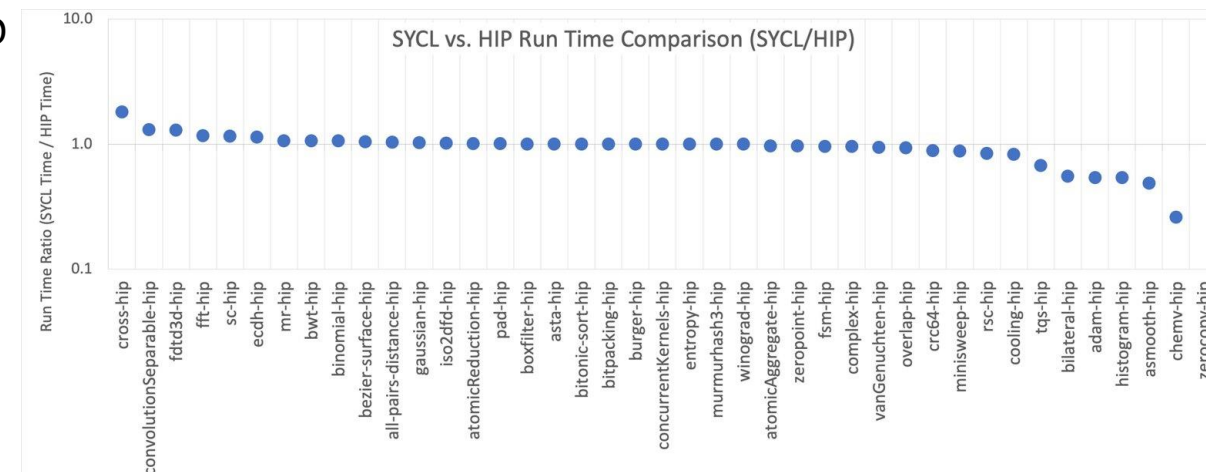
- Provide HIP support on LLVM 16 to 19: chipStar v1.2
- Supported Applications CP2K, GAMESS, libCEED, OpenMM
- Supported libraries: hipBLAS, hipSOLVER, hipRTC, hipFFT, hipRAND
- Upcoming support for: rocPRIM, rocThrust, CuPy, and more
- Pre-release CUDA support

### Synergistic Efforts

- LLVM (base tool-chain of AMD HIP and chipStar)
- SYCL (also a Khronos standard like SPIR-V and OpenCL)



Lead: Brice Videau, ALCF



# Incubation

---

# Automatic Differentiation (Enzyme)

**Team:**  
A. Edelman, MIT  
J. Hueckelheim, ANL  
W. Moses, UIUC  
M. Schanen, ANL

## Goals

- Facilitate integration of scientific simulations into AI models

## Strategy

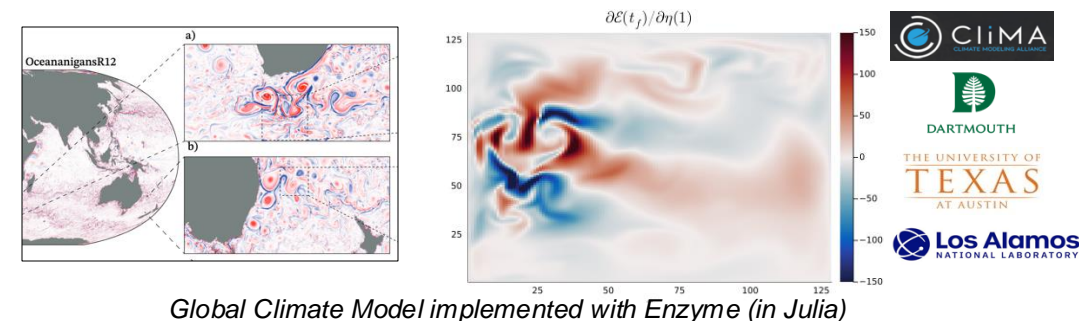
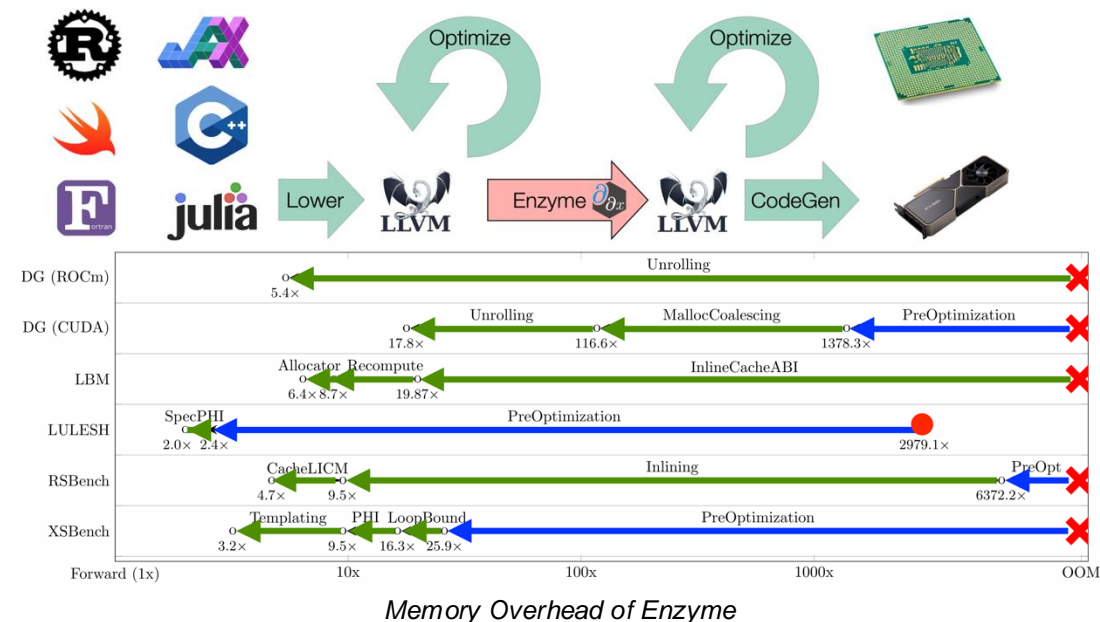
- This requires a workflow for generating the gradient of scientific simulations using techniques such as automatic differentiation and differentiable programming
- Focus on LLVM-based AD tool Enzyme as LLVM is the backend for most vendor architectures
- “Scalable automatic differentiation of multiple parallel paradigms through compiler augmentation”, SC22, Best Student Paper
  - C/C++, Julia, Rust, OpenMP, Fortran, RAJA, CUDA, ROCm, MPI

## Tasks and Objectives

- Differentiable Linear Solvers
- Differentiable GPU Kernels
- Demonstrate integration into an AI framework
- Written guidance and challenges for applying Enzyme
  - Language support (e.g., C/C++, Julia, ...)
  - Numerical challenges (e.g., iterative algorithms)
  - Architecture support

## Synergies

- LLVM project
- Julia programming language
- SciDAC Next Generation Grid Simulation





- High-level, high-performance, high-productivity programming language
- Built on top of LLVM compiler toolchain

### Tasks in CY24:

- Support AMD MI300 and NVIDIA H200 GPUs
- Address startup-latency issues for accelerated computing with Julia
- Continuous development of Julia for HPC (LLVM 17+, new architectures)

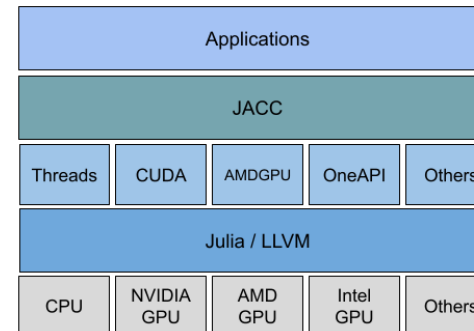
### Synergistic Efforts:

- Enable Automatic Differentiation (AD) for HPC codes
- Support MPI 4.x in MPI.jl / MPI-ABI standardization

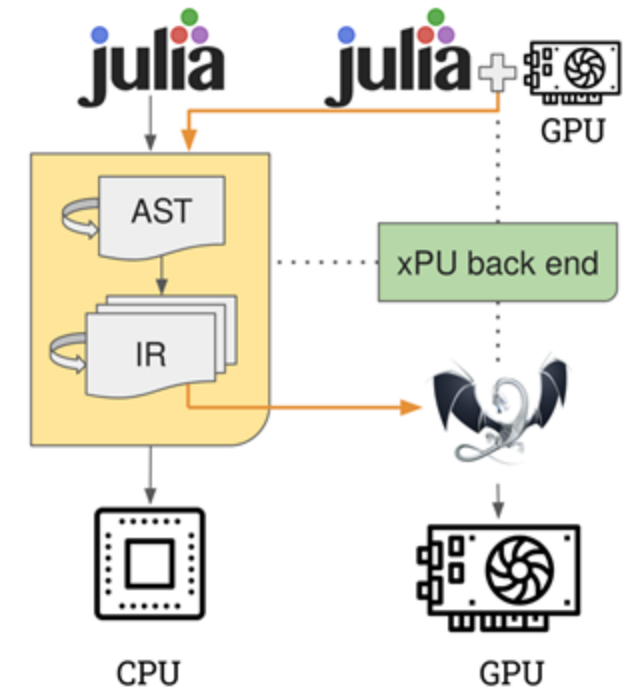


```
# Unidimensional arrays
function axpy(i, alpha, x, y)
    x[i] += alpha * y[i]
end
function dot(i, x, y)
    return x[i] * y[i]
end
SIZE = 1_000_000
x = round.(rand(Float64, SIZE) * 100)
y = round.(rand(Float64, SIZE) * 100)
alpha = 2.5
dx = JACC.Array(x)
dy = JACC.Array(y)
JACC.parallel_for(SIZE, axpy, alpha, dx, dy)
res = JACC.parallel_reduce(SIZE, dot, dx, dy)

# Multidimensional arrays
function axpy(i, j, alpha, x, y)
    x[i,j] = x[i,j] + alpha * y[i,j]
end
function dot(i, j, x, y)
    return x[i,j] * y[i,j]
end
SIZE = 1_000
x = round.(rand(Float64, SIZE, SIZE) * 100)
y = round.(rand(Float64, SIZE, SIZE) * 100)
alpha = 2.5
dx = JACC.Array(x)
dy = JACC.Array(y)
JACC.parallel_for((SIZE, SIZE), axpy, alpha, dx, dy)
res = JACC.parallel_reduce((SIZE, SIZE), dot, dx, dy)
```



Lead: Alan Edelman, MIT

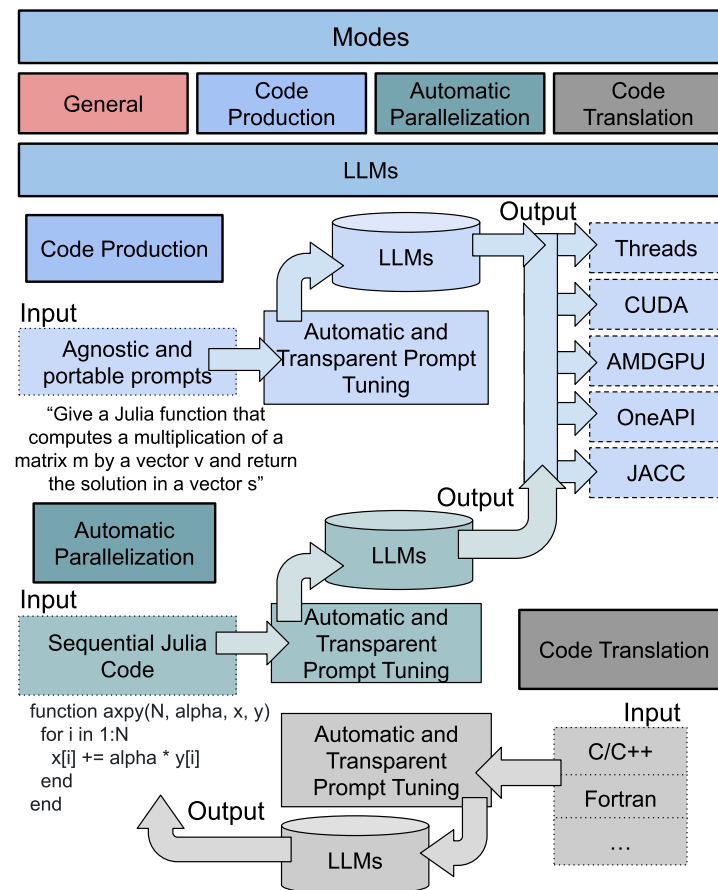




# LLMs for HPC Programming (ORNL)

- Development of LLMs fine-tuned for HPC Software Development
- Framework to offer users to develop their own LLM
- Pre-Fine-Tuned Models
  - Julia, Kokkos, OpenMP, BLAS, High Order Explicit PDE

```
root@lap125354: ~  
julia> ChatHPC.prompt("general", "What is Julia?")  
Julia is a high-level, high-performance, dynamic programming language for numerical computing. It provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library. Julia is often used for data analysis and computational science.  
julia> ChatHPC.prompt("threads", "Give a code that computes the multiplication of a Matrix A with a vector X and stores the result in a vector Y")  
Here is the Julia code that performs the multiplication of a matrix A with a vector X and stores the result in a vector Y.  
  
```julia  
Threads.@threads for j = 1:size(A, 2)  
    for i = 1:size(A, 1)  
        Y[i] = Y[i] + A[i,j] * X[j]  
    end  
end  
```  
julia> |
```



Lead: Pedro Valero-Lar, ORNL

