OAK RIDGE National Laboratory

S4PST: Stewardship for Programming Systems and Tools

Keita Teranishi, Group Leader, Programming Systems, Computer Science and Mathematics Division
S4PST Project PI

PCCC24, December 6th 2024

U.S. DEPARTMENT OF ENERGY

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

S4PSTプロジェクト

- 目標：次世代HPCシステムへ向けのプログラミングシステムのメンテナンス、開発、利用促進

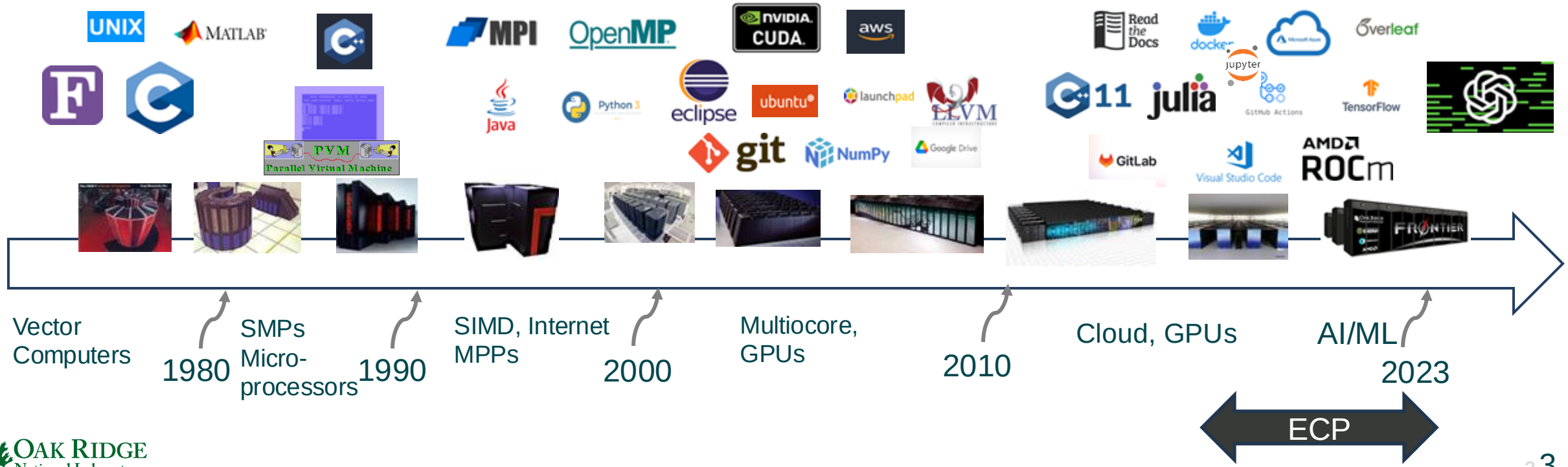
- 対象：米国エネルギー省エクサスケールコンピューティングプロジェクト(DOE ECP, 2016-2023)で研究開発されたプログラミングシステム、コンパイラ
  - ECP下では70近くのソフトウェアプロダクトが開発
  - S4PSTではその内10をカバー

- 課題点:
  - 技術面: フルスタック開発による困難（HPC向けアプリーケーション、アルゴリズム、コンパイラ、システムソフトウェアへの幅広い理解が必要）
  - 開発方針: 既存のアプリケーションとユーザを優先するが、新技術や新しいアプリケーション向けの機能を導入。また新規ユーザへの啓発、教育も重要。
  - 人材面: 開発者へのコミットメント、開発者の確保、人材育成

S4PST: Stewardship for Programming Systems and Tools

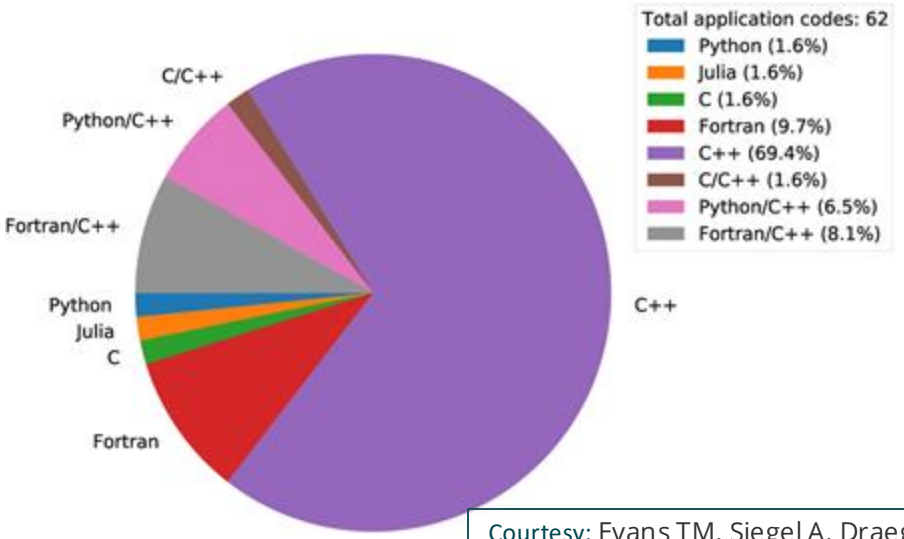
S4PSTのビジョン:HPCのデモクラタイゼーション

- ハードウェア、システム、ソフトゥエアは日進月歩で発展するが・・・。
- 「HPCプログラミングは難しい」は未だ解決せず！

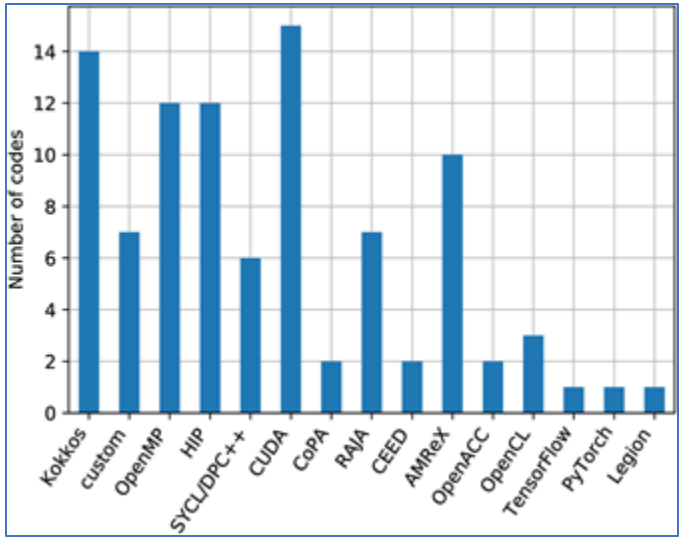


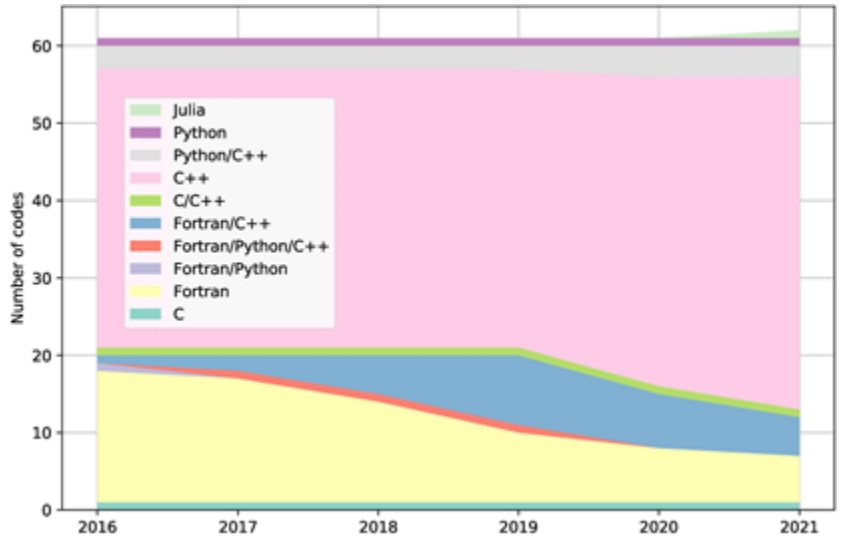
ECPアプリのプログラミング言語・モデルの利用状況

- GPUノードへの対応の完了
  - NVIDIA, AMD and Intel GPUs
- **KokkosとRaja**というパフォーマンスポータブルフレームワークによる生産性の大幅な改善
- プログラム言語、ライブラリの利用の推移
  - C++が主流へ
  - PythonやJuliaの利用の増加
  - Fortranは減少傾向に
  - ECP外だとまだFortranは健在



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



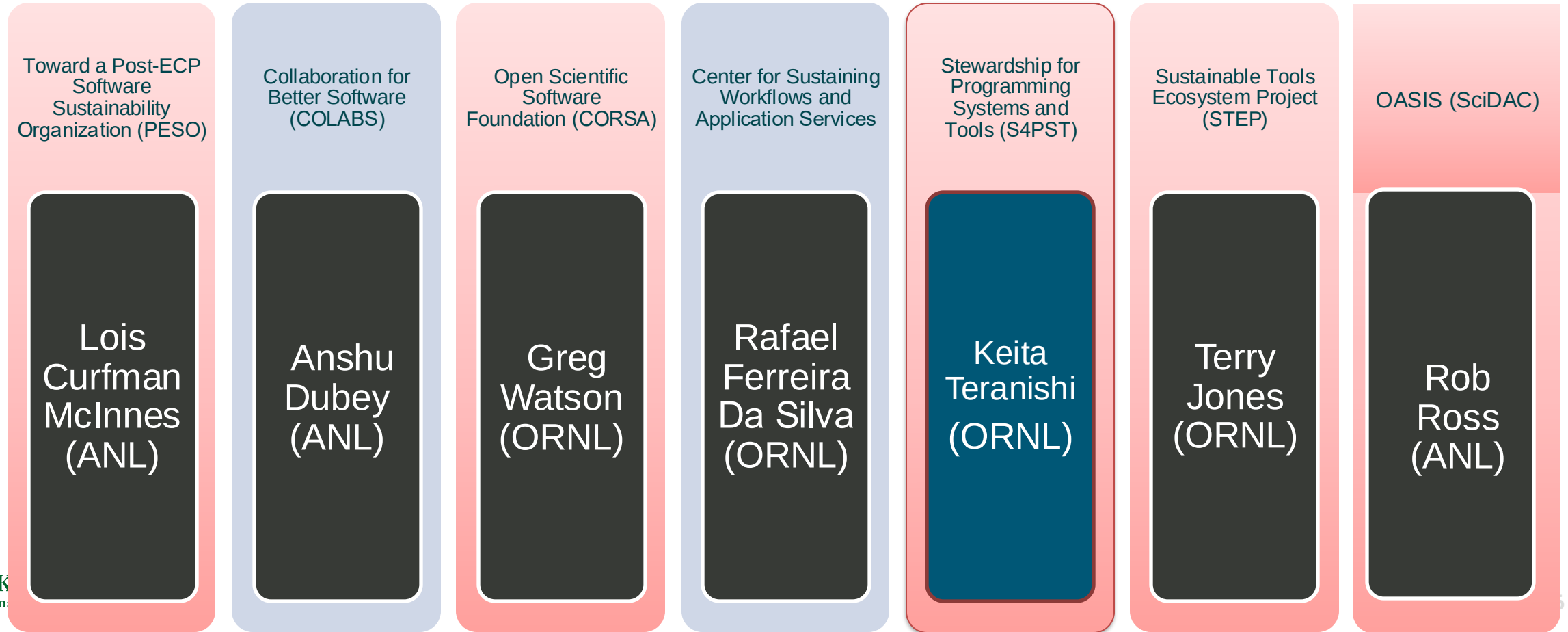


エクサスケール時代のHPC向けソフトウェアの動向の変化

- **HPCシステムベンダはAI/MLシステムベンダへと変化**
  - AI/ML向けシステム、ソフトウェアの開発は急速に進むも・・・
  - ベンダによる従来のHPC、科学計算向けソフトウェア開発は
- 一般ユーザの需要はオンプレミスでの HPC ソリューションから、HPCクラウドへ
- その結果DOEは自前でのプログラミング環境の開発が必須に
  - コンパイラ、ランタイム、デレクティブ言語
  - ソフトウェアのテスト環境 、テストプログラムスイーツ
  - 科学計算のニーズを満たすため、プログラミング言語、APIの標準化に向けての積極的な活動　(C++, Fortran, OpenMP/ACC, MPI, etc.)
  - テクニカルサポート、バグフィックスへの迅速な対応
- DOEのソリューション：
  - プログラミングシステムのコミュニティ
  - 積極的な新技術の導入 – (Early Adopter)

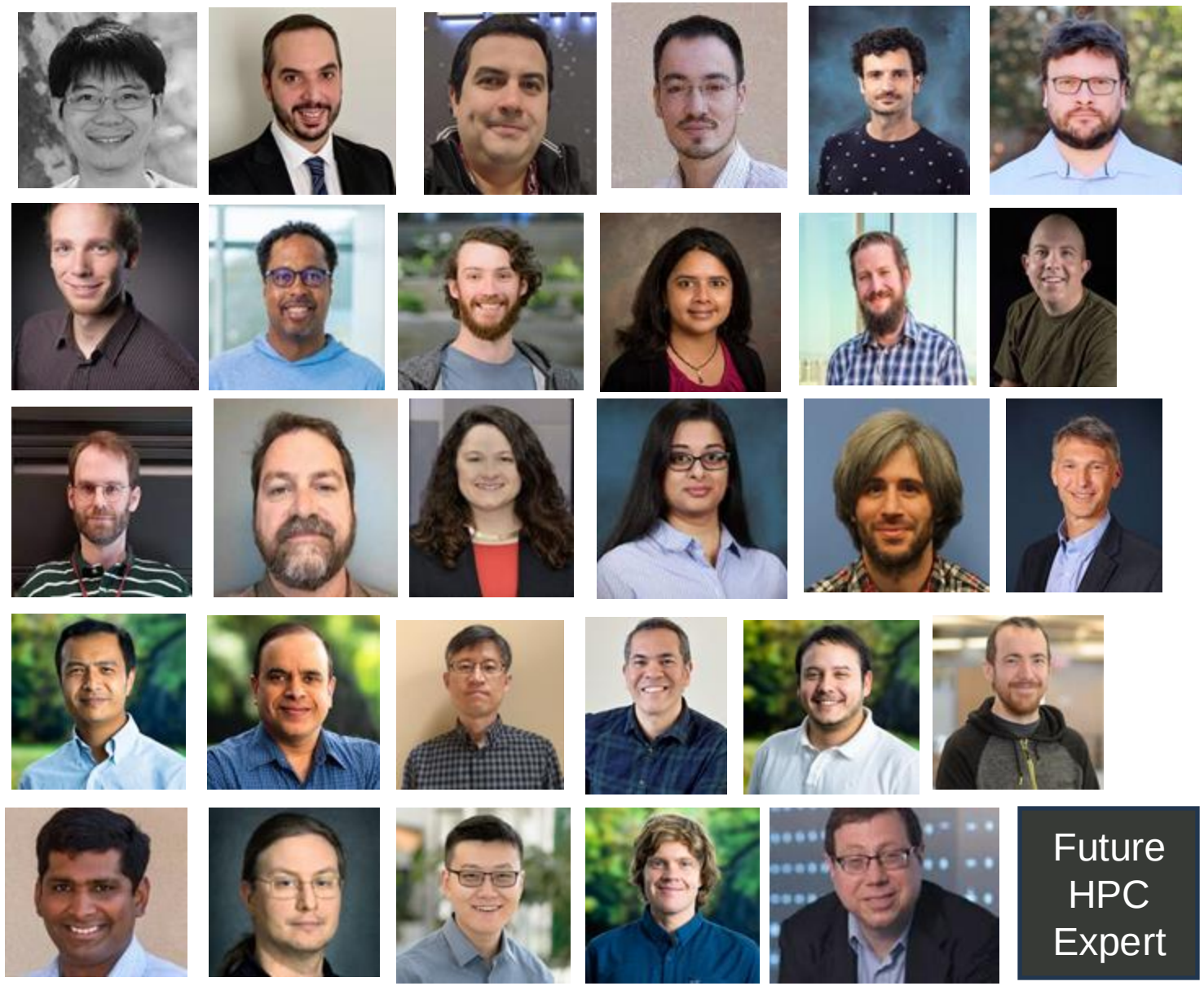
S4PST: Stewardship for Programming Systems and Tools

- Sustaiability（維持可能）からStewardship（幹事）へ。
- ２０２３年：Phase-1。７団体への萌芽的活動開始
- ２０２４年：Phase-2。５団体の活動が中心に。
- ソフトウェアプロダクトの幹事団体は以下の通り
  - STEP：性能解析ツール（５）
  - S4PST：プログラミングシステム（１０）
  - OASIS：科学計算ライブラリ、I/O,可視化（２０＋）
  - PESO：垂直統合



S4PST コミュニティ

- PI: Keita Teranishi (ORNL)
- CoPI: William Godoy and Pedro Valero Lara
- 国立研究所:
  - Oak Ridge National Laboratory
  - Argonne National Laboratory
  - Lawrence Livermore National Laboratory
  - Lawrence Berkeley National Laboratory
  - Sandia National Laboratories
  - Brookhaven National Laboratory
  - Los Alamos National Laboratory
  - SLAC National Accelerator Laboratory
- 大学:
  - University of Delaware
  - Massachusetts Institute of Technology
- コラボレーター:
  - Louisiana State University
  - Pacific Northwest National Laboratory
  - Carnegie Mellon University
  - University of Tennessee, Knoxville
  - Stanford University
  - Other 6 NSSGT projects



Future HPC Expert

S4PST: ポートフォリオ

プロダクトスチュワードシップ

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

エマージングテクノロジー

- Enzyme: Automatic Differentiation (AD) Package
- High Productivity Language: **Julia,** Python
- Large language models for HPC
- Kokkos ecosystems adapting AI-hardware/special arithmetic units

S4PST の活動

- 開発
  - 新しいノード、ネットワークへの対応
  - AIハードウェア、AIソフトウェアスタックとの融合
- メンテナンス
  - バグフィックス
  - バージョン更新
  - E4S(ECPで開発されたソフトウェアの統合パッケージ　https://e4s.io ） ソフトウェアスタックとのインテグレーション(Spack)
- 標準化

C++ (Kokkos) MDSpan, std::Linear Algebra, MDSubSpan, C++26

Fortran: DO CONCURRENT, Fortran 2023

MPI (MPICH and OpenMPI): MPI-4.2 and 5.0

OpenMP/OpenACC: OpenMP 6.0

S4PST の活動

- ベンダ非依存のテストプログラム群
  - •OpenMP
  - •OpenACC
- インキューベーション、実用化
  - Enzyme Automatic Differentiation Framework (LLVM)
  - High Productivity Language (Julia, Python)
  - 生成AIによるHPCプログラミングサポート（ChatHPC)
- コミュニティサポート
  - マニュアル等のドキュメントの整備
  - チュートリアル、ハッカソン、ウェビナー
  - プログラム例の整備
  - s4pst.org
- 他のSoftware Stewardship Organizationとの連携

OAK RIDGE National Laboratory

S4PSTプロジェクトの教訓

- 多様性
  - 開発者のポジション、所属等によって、目標設定が異なる
    - 研究部門
    - 開発部門
    - 運用部門
    - 大学vs研究所

  各開発者、チームによって優先順位が大きく異なる
  論文優先？ソフトウェアリリースが優先？ステークホルダー優先？

  開発資金の出資元、プログラムの違い(NSF, ASCR, SciDAC, Other DOE, NNSA ASC, Facility, DOD, etc.)
- 相互依存、補完関係
- 開発のための時間確保
- HPSFの役割
  - 国際的協業のハブに
    - Kokkosは立ち上げメンバーとして参加。現在はCEA（フランス）とDOEによる開発が進む。
  - LLVM and OpenMPIは既にFoundationを設立
  - プロダクトの所有権はHPSFに側に移管。それを嫌う開発者も。

各プロダクトの概要

OpenMP and OpenACC (UD, BNL, ORNL and ANL)

Lead: Sunita Chandrasekaran
U of Delaware & BNL

プロジェクトの目的

- 標準化、プロトタイプ実装、各コンパイラの機能の確認
- Intel,AMD,NVIDIA,Arm 等のCPUとGPUでの動作、機能のテスト
- テストプログラムの自動作成
- MLIRのDialectの整備（GPU Offloading機能の共通化）

プロジェクト成果

- OpenMP6.0標準化（SC24直前で発表）の取り組み
- OpenMP、OpenACCの各機能のテストプログラム集のE4Sへの組み込み
- LLMを使ったOpenMP向けテストプログラム自動作成

- OpenMPのテストプログラム集と各コンパイラ、アーキテクチュアでの動作確認状況. https://crpl.cis.udel.edu/ompvv/
- OpenACCのテストプログラム集と各コンパイラ、アーキテクチュアでの動作確認状況. https://crpl.cis.udel.edu/oaccvv/

```
#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];
    }
  }
}
```

```
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 }
```

他プロジェクトとの協業

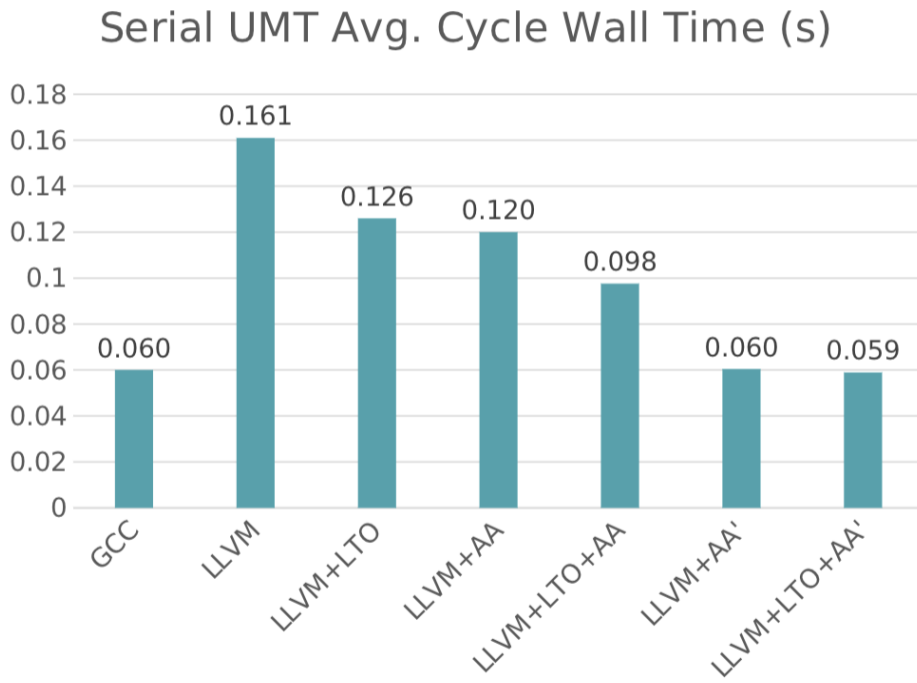
- E4S
- LLVMチーム

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

Lead:
Johannes Doerfert
LLNL

プロジェクトの目的

- 企業、大学、他の研究機関との連携によるLLVMコンパイラのHPC向けサポート、ツールの整備
- DOEのHPCシステム、アプリケーション開発のロードマップをLLVMコミュニティに反映させる
- ワークショップ、チュートリアルの開催
- DOEのスパコンを対象にした機能の開発(Perlmutter, Frontier, Aurora, and El Capitan)
  - FlangコンパイラのOpenMPのGPUサポートの改善
  - DOEスパコンを対象にしたCI環境の充実



プロジェクト成果

- Flangコンパイラの性能向上（メモリのエイリアス解析の改善）Gfortranに近い性能に
- LLVM-IR解析ツールの開発（GPU向けアプリのスタック割り当ての解析）

Kokkos Core – Maintaining Performance Portability

Lead: Christian Trott (SNL)

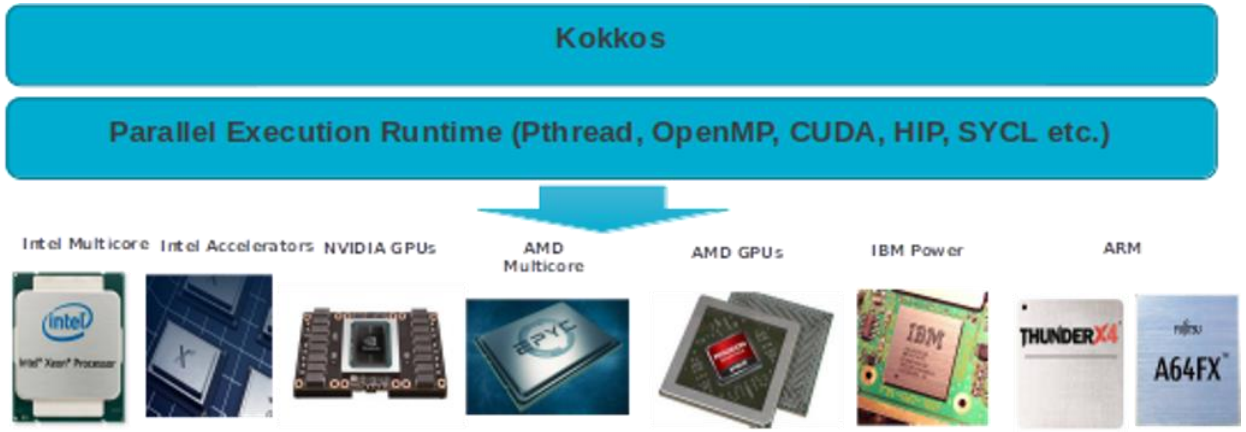
プロジェクトの目的

- C++言語によるパフォーマンスポータビリティ環境
- CUDA, HIP, SYCL, OpenMP, OpenACCをバックエンドにすることで互換性、柔軟性を担保
- AI/MLアプリケーションのサポート
- 150 以上の大学、研究機関で利用

Kokkos-AI Lead:
Siva Rajamanickam
SNL

プロジェクト成果

- High Performance Software Foundationへの移管
- CEA（フランス）が開発者として積極的に参加
- KokkosをMLIRへの組み込み
- Kokkos4.5リリース(12月10日)



Modern Fortran (LBNL)

Lead: Damian Rouson, LBNL

- プロジェクトの目的
  - Fortran標準化、Fortranコンパイラの各機能のテスト、新機能の導入と普及、啓発
  - Fortranは８位のプログラミング言語
- Fortranコンパイラ
  - Open-source: GNU, LFortran, LLVM, ...
  - Vendors: Arm, AMD, HPE (Cray), IBM, Intel, NAG, NVIDIA, Fujitsu, .
- 成果：Fortran 2023の 標準化
  - ノード間、ノード内（GPUも含めて）の並列化文法の標準化
  - GPU-offloadingの自動化
- 今後の課題
  - LLVM,OpenMP/ACCチームとの連携
  - 非同期タスク並列のサポート
  - タイプセーフティ

```
! 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

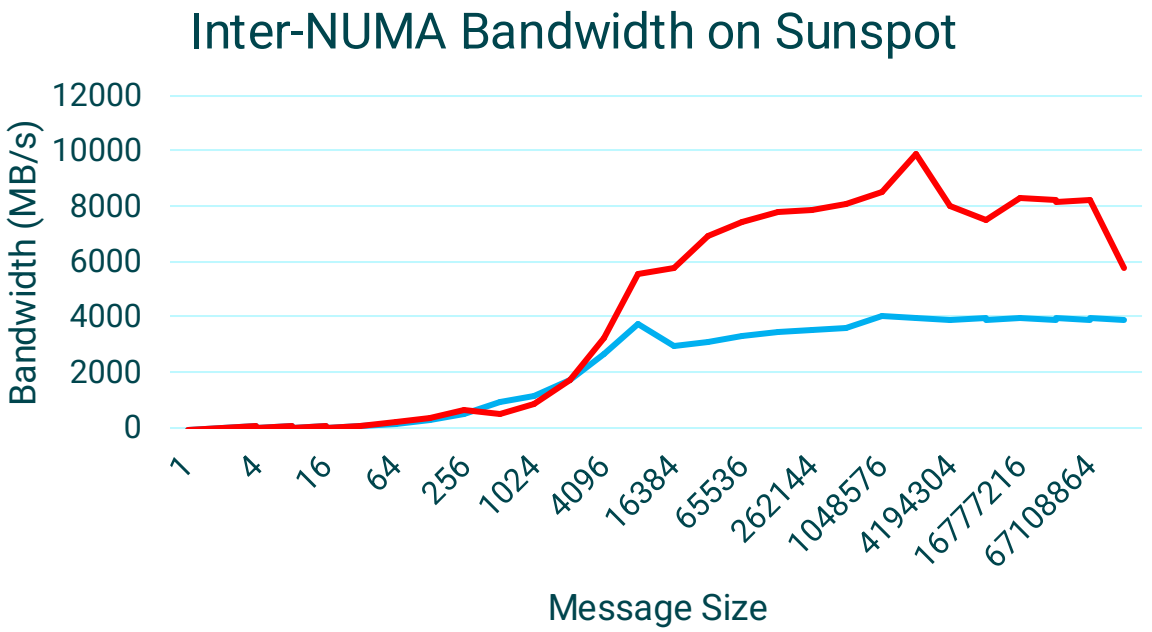
```
integer, allocatable :: image(:,:)
logical, allocatable :: edge_mask(:,:)

! 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)

OpenMPI Lead: Thomas Naughton III
ORNL

MPICH Lead: Yanfei Guo
ANL

- プロジェクトの目的
  - OpenMPI とMPICHのメンテナンスとエクサスケールシステムでの性能向上
  - ベンダーとの連携
- 成果
  - Aurora,Frontierシステムでの性能向上
    - Libfabricのサポート
    - GPU Directの
    - PMIx
  - CI/CDシステムの改善
  - MPI Forumでの参加



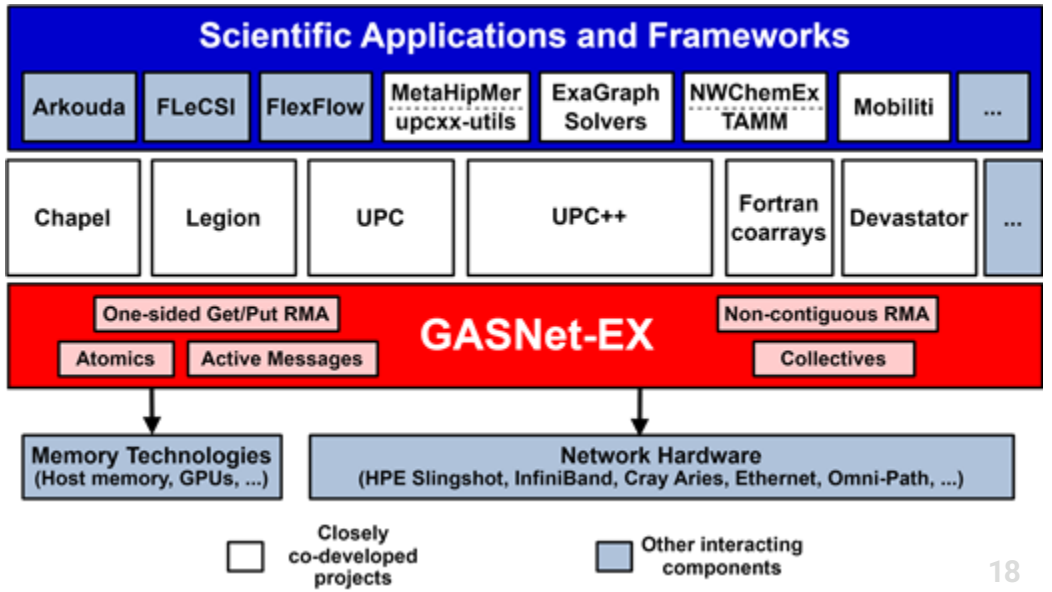
UPC++ & GASNet-EX (LBNL)

GASNet-EXはPartitioned Global Address Space (PGAS) とdistributed asynchronous many-task (AMT) programming systems実装するためのポータブルなトランスポートレイヤ

UPC++ はGASNet-EX 上に実装されたPGAS及び Remote Procedure Call (RPC)の機能を持つC++ライブラリ

他のプロジェクトの連携
- Legion （トランスポートレイヤ）
- Flang （Fortran23の分散並列機能のサポート）

Lead: Paul H. Hargrove (LBL)



Legion (LANL,SLAC)

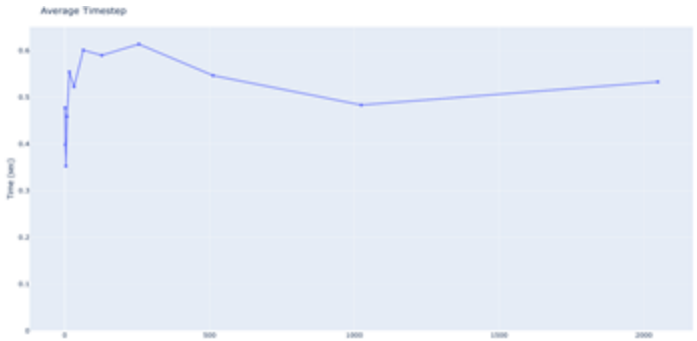
特徴
分散非同期タスク並列ランタイム
CPU,GPUの両方をサポート
Frontier, Perlmutter, and El Capitan上で動作

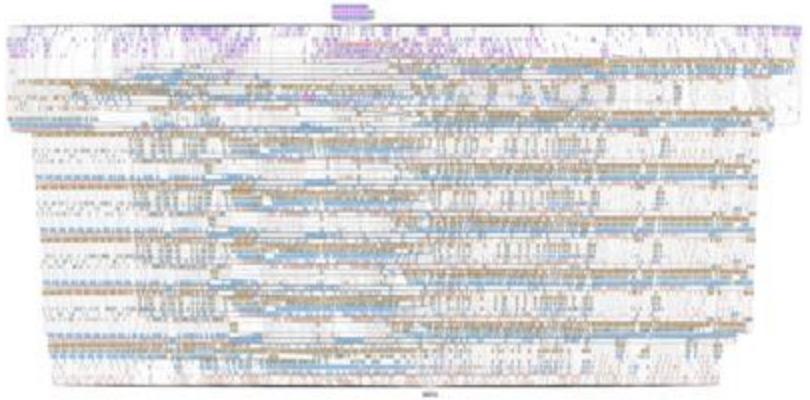
Legionの利用例
- S3D (SNL)
- FlexFlow (CMU/LANL/Stanford/UCSD)
- SpiniFEL (SLAC)
- cuPyNumeric

連携
NVIDIAのcuPyNumericの開発
GASNet-EX の利用

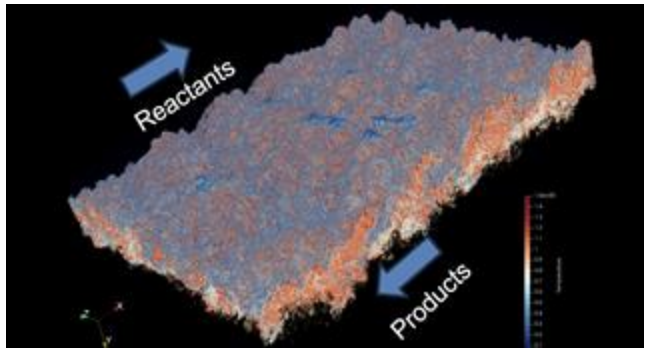
成果
Frontier上S3D(燃焼シミュレーション）が
8192ノードまでスケール

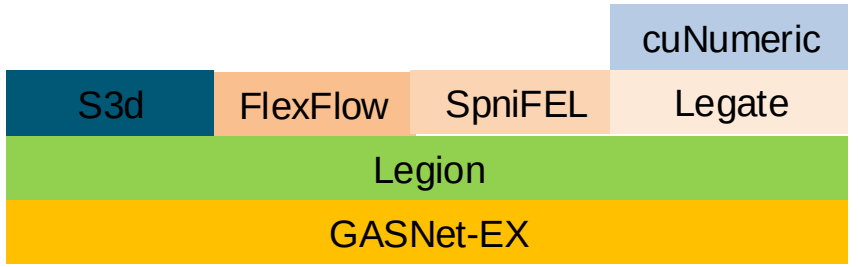
Co-Leads: Pat McCormick (LANL)
and Alex Aiken (SLAC)





S3D Task Graph





Legion Software in S4PST's scope

SYCL (ALCF, OLCF, NERSC)

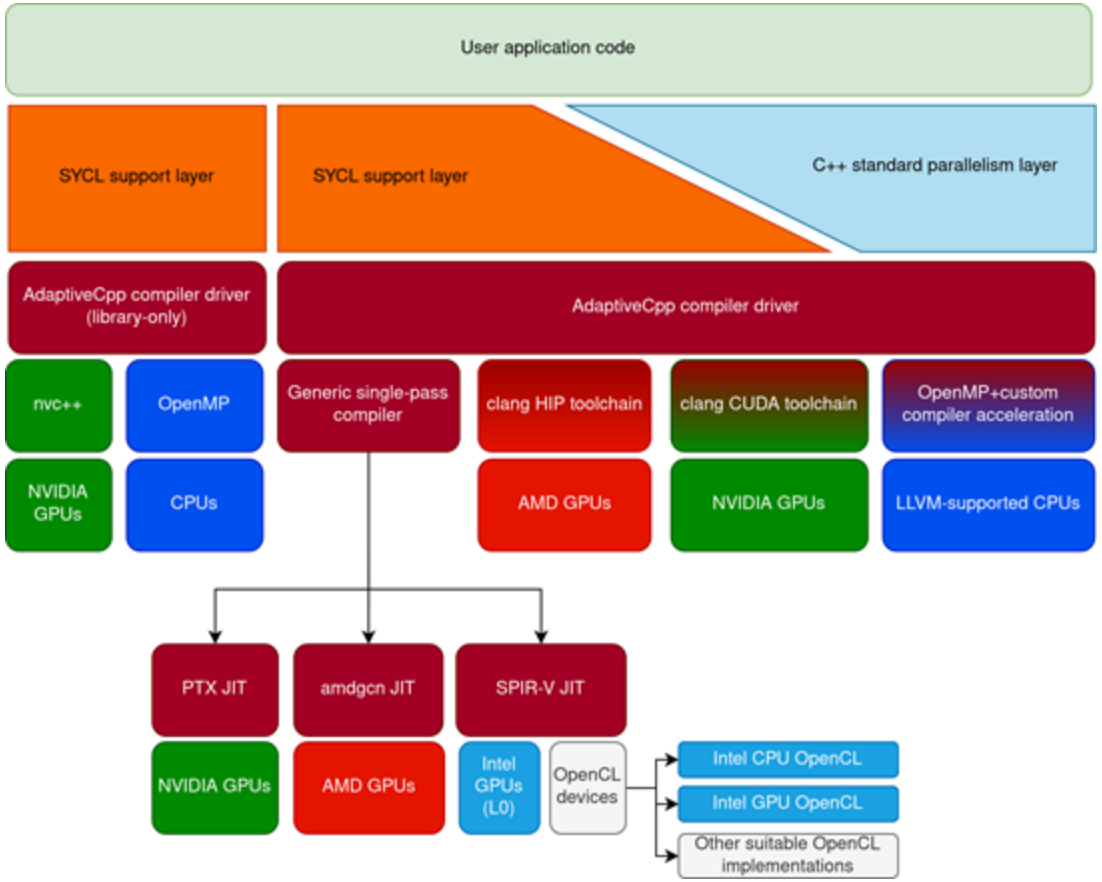
Lead: Thomas Applencourt, ALCF

C++ベースのパフォーマンスポータブルプログラミングシステム

- OpenCLの後継 CPU、GPU、 FPGA等を広くサポート
- LLVM コンパイラを利用
- 利用可能の実装はIntel製 DPC++ と オープンソースのAdaptiveCpp
- SYCLの普及、性能解析、標準化委員会での活動が中心

協業、連携

- HIP (HIP をSYCLのバックエンドとして利用)
- Kokkos (SYCLバックエンド)
- LLVM

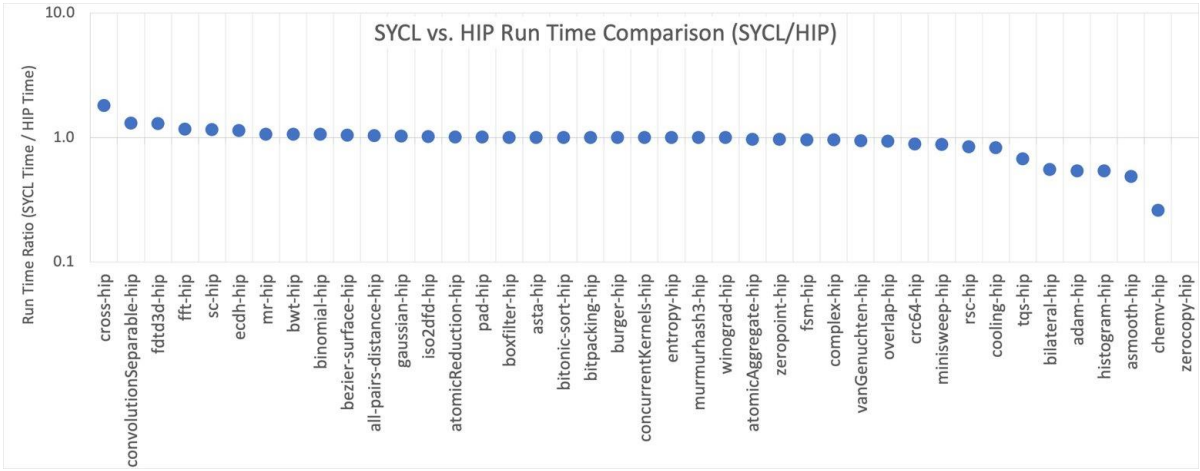


HIP (ALCF, NERSC, OLCF)

Heterogeneous-Computing Interface for Portability

Lead: Brice Videau, ALCF

- プロジェクト概要
  - AMD GPU向けのプログラミング環境
  - NVIDIA CUDAに非常に近いAPI
  - DOE とIntelの協業でIntelアクセラレータ向けのHIP （**chipStar**）の開発
    - LLVM + SPIR-V + Level-Zero/OpenCL
    - Intel math librariesのサポート
- 成果
  - chipStarの互換性の強化　（AMD,NVIDIA)
- 今後の課題
  - Intelアクセラレータ以外のアーキテクチャのサポート
- 協業、連携
  - LLVM (base tool-chain of AMD HIP and chipStar)
  - SYCL



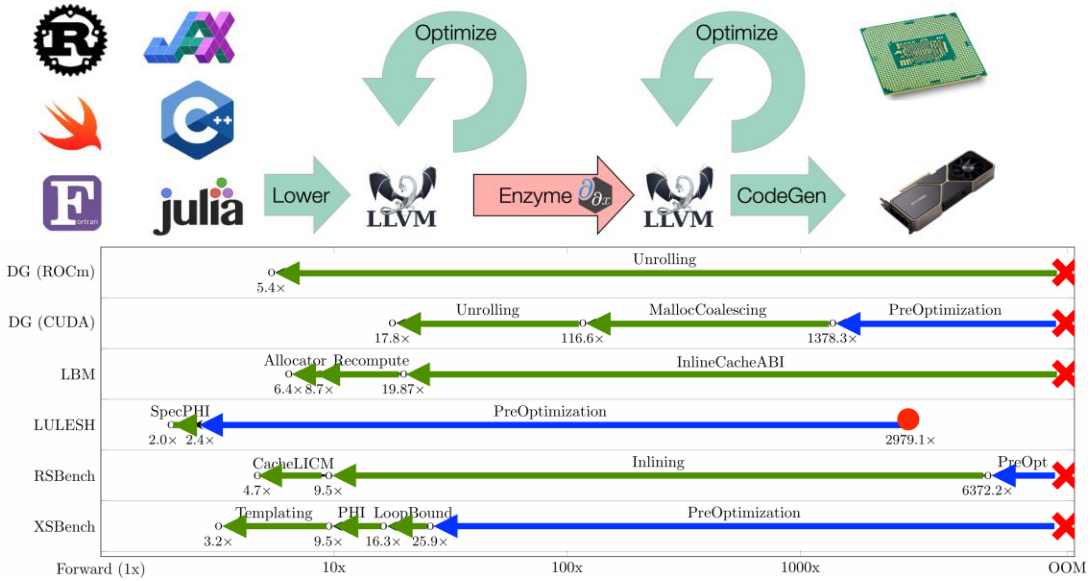
Incubation

Automatic Differentiation (Enzyme)

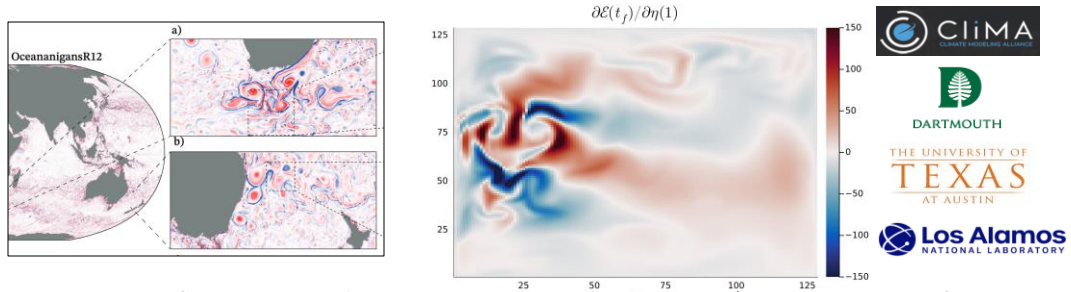
Team:
A. Edelman, MIT
J. Hueckelheim, ANL
W. Moses, UIUC
M. Schanen, ANL

Enzyme: Automatic Differentiation （自動微分）ツール

- プロジェクト目標：
  - 実用化とHPCとAIの融合
  - 幅広いHPCアーキテクチャ、プログラミング言語、ランタイムのサポート
  - C/C++, Julia, Rust, OpenMP, Fortran, RAJA, CUDA, ROCm, MPI
  - 線形計算ライブラリの有効活用
  - 生成コードの最適化
- 連携、協業
  - LLVM project
  - Julia programming language



Enzymeで生成されたコードのメモリ消費。これらのベンチマークはEnzymeなしではGPU上では実行不可能



Enzymeで生成されたコードで開発されたClima海洋気候モデル　（Juliaで実装）

OAK RIDGE National Laboratory
Argonne NATIONAL LABORATORY
MIT
ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

https://github.com/EnzymeAD
https://enzyme.mit.edu/conference

MIT, ORNL

- MATLABの近いシンタックスを持つ科学計算向け言語
- LLVMの利用とJust-in-Timeでのコンパイルが特徴

技術的課題:

- アクセラレータのコールによる遅延の低下
- LLVMの最新版へのキャッチアップ

成果：

- JACC (Julia for Accelerator)

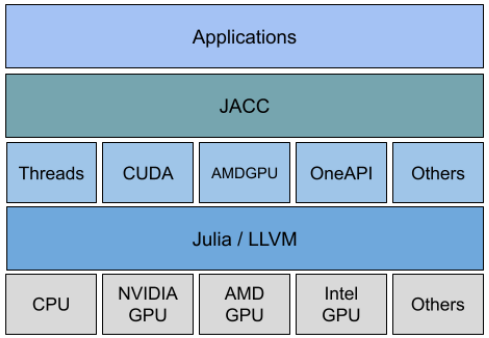
連携、協業:

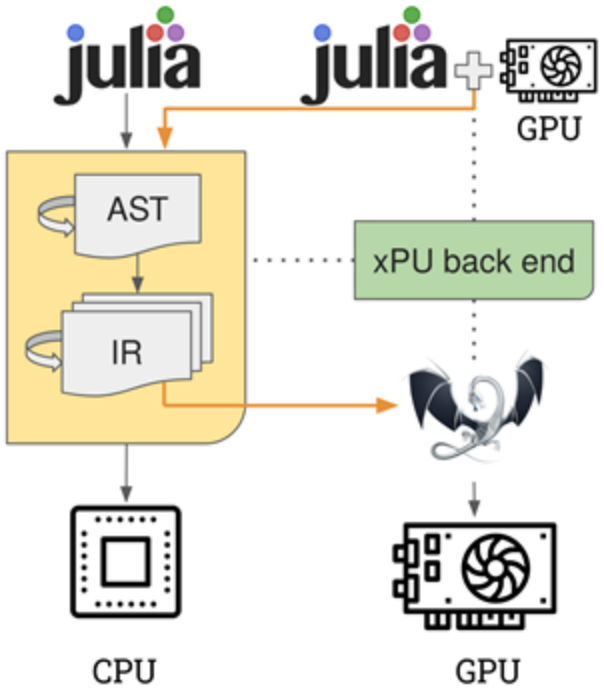
- Enzyme(AD)フレームワーク
- MPIインターフェース(ABI)の標準化

JACC
JULIA FOR ACCELERATORS

Lead: Alan Edelman, MIT

```
# 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)
```

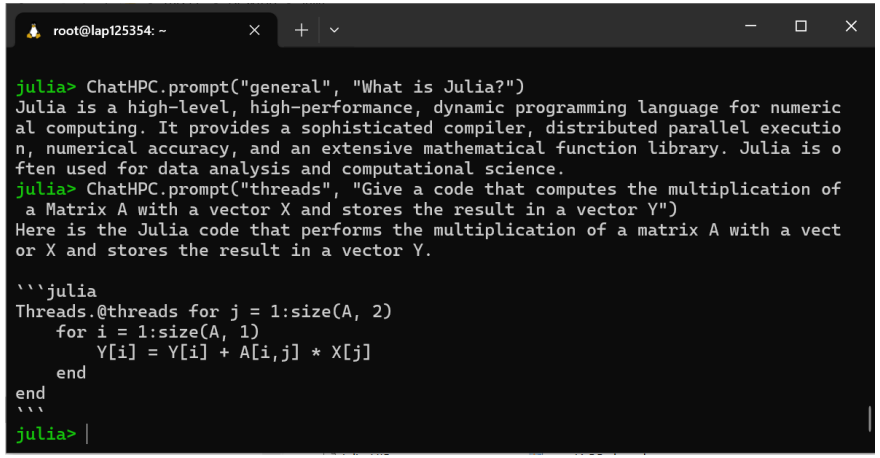


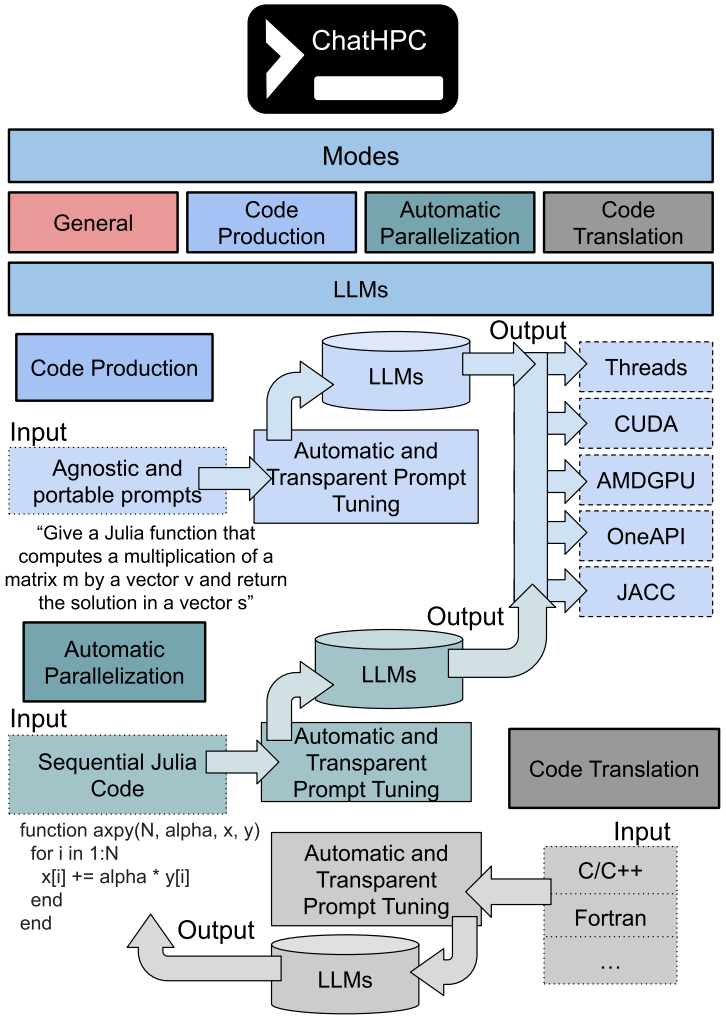


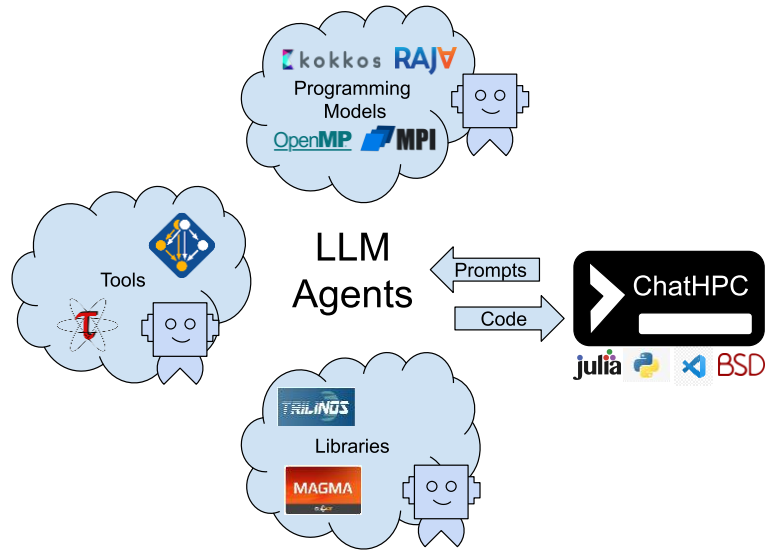
LLMs for HPC Programming (ORNL)

Lead: Pedro Valero-Lar, ORNL

- HPCプログラム向けのLLMモデルの改良、利用方法の研究開発
- CodeLlama等の大規模コーディングモデルを特定のHPCプログラミングに特化させた複数のモデルを提供
  - Julia,Kokkos,OpenMP,BLAS,High Order Explicit PDE
  - ライブラリ使用に特化したものも作成予定
- 来年春までには公開予定







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. All public copies of research results, including pre-prints, project webpages, blog postings, etc. should include a funding acknowledgement that specifically mentions the Department of Energy and, unless impractical, includes the string “Advanced Scientific Computing Research