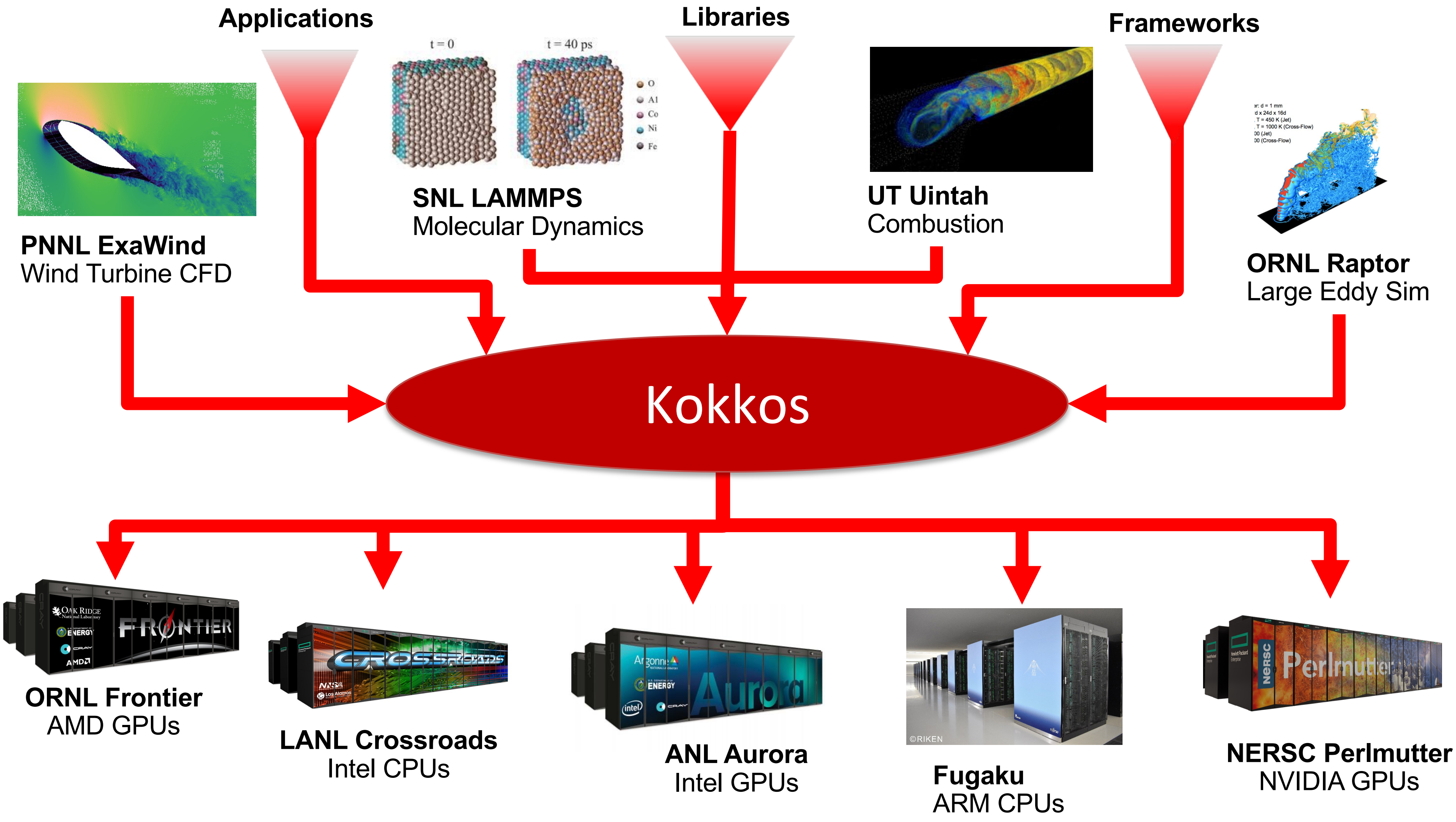
The Kokkos ecosystem - Sustaining performance portability at the exascale era

Damien Lebrun-Grandié
Christian Trott

ORNL is managed by UT-Battelle LLC
for the US Department of Energy



Applications

Libraries

Frameworks

PNNL ExaWind
Wind Turbine CFD

SNL LAMMPS
Molecular Dynamics

UT Uintah
Combustion

ORNL Raptor
Large Eddy Sim

Kokkos

ORNL Frontier
AMD GPUs

LANL Crossroads
Intel CPUs

ANL Aurora
Intel GPUs

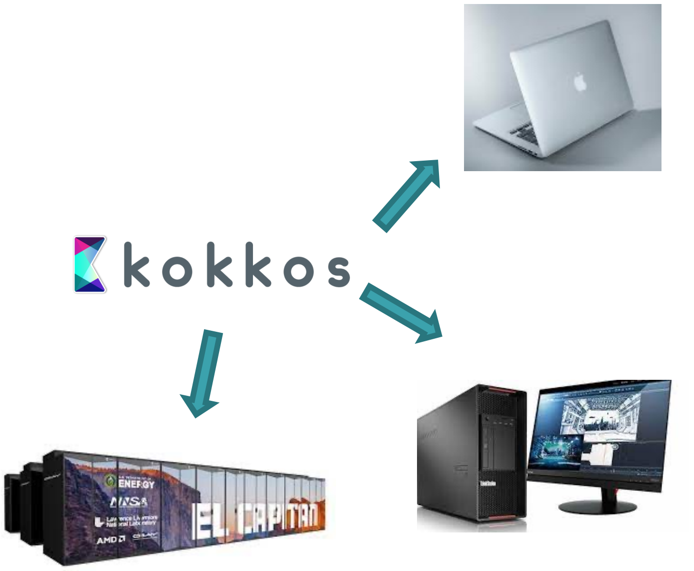
Fugaku
ARM CPUs

NERSC Perlmutter
NVIDIA GPUs

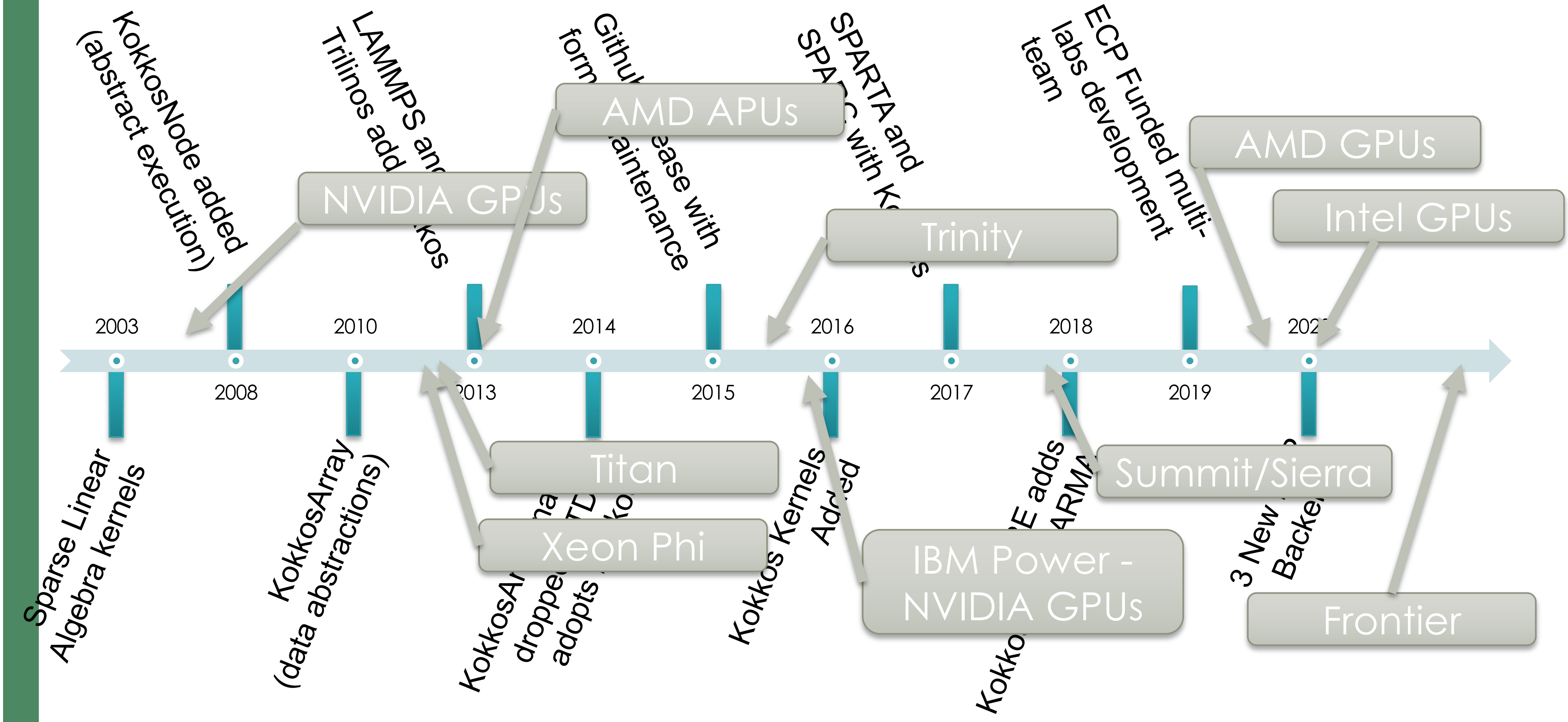
Open slide master to edit

What is Kokkos?

- A C++ Programming Model for Performance Portability
  - Implemented as a template library on top of CUDA, OpenMP, …
  - Aims to be descriptive not prescriptive
  - Aligns with developments in the C++ standard
- Expanding solution for common needs of modern science/engineering codes
  - Math libraries based on Kokkos
  - Tools which enable insight into Kokkos
- It is Open Source
  - Maintained and developed at https://github.com/kokkos
- It has many users at wide range of institutions

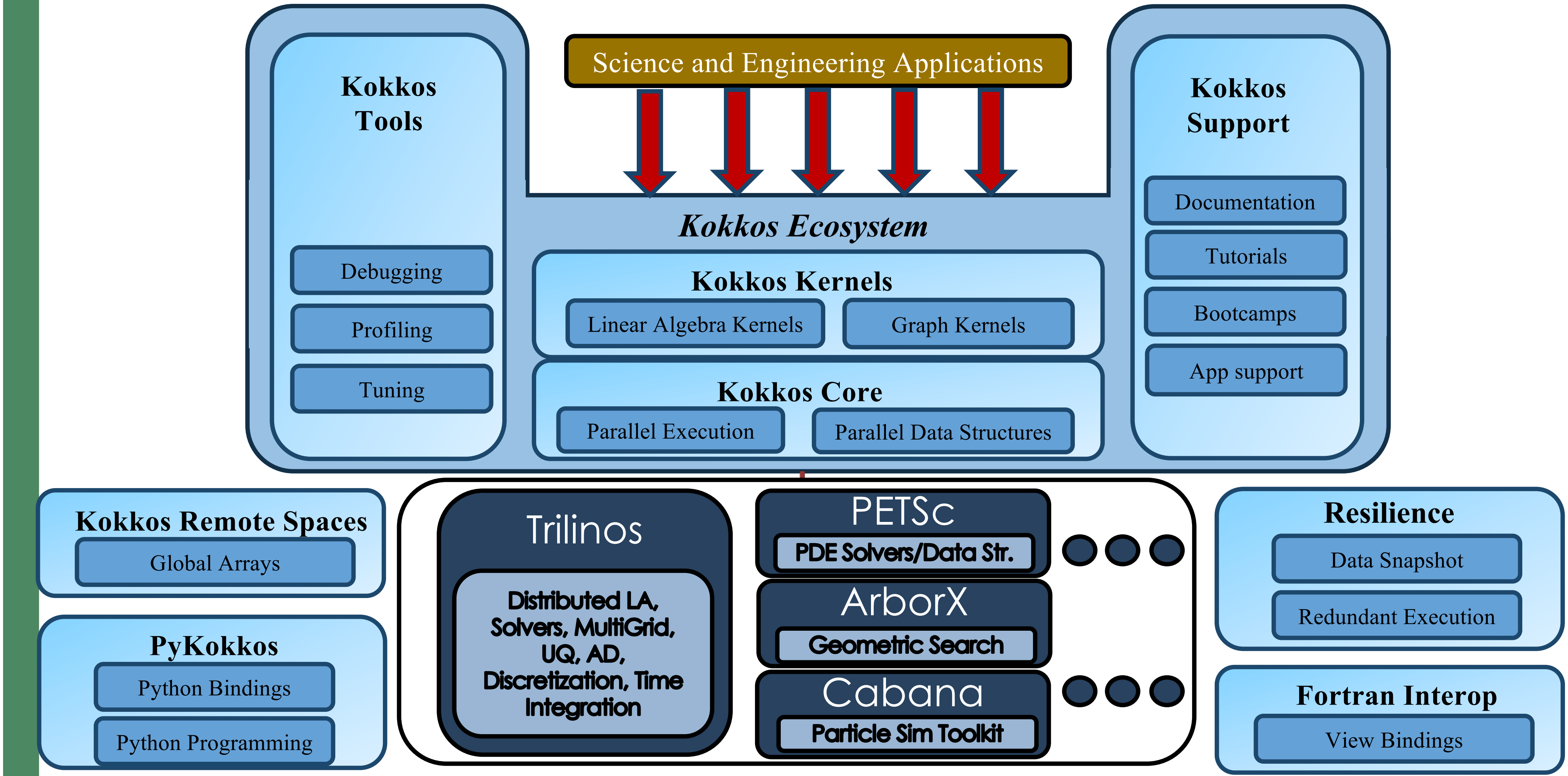


Kokkos Timeline


KokkosNode added (abstract execution)
LAMMPS an Trilinos add kos
Githu form ease with aintenance
SPARTA and SPA C with Ko s
ECP Funded multi-labs development team
AMD APUs
NVIDIA GPUs
Trinity
AMD GPUs
Intel GPUs
2003
2008
2010
2013
2014
2015
2016
2017
2018
2019
202
Sparse Linear Algebra kernels
KokkosArray (data abstractions)
Titan
Xeon Phi
KokkosAr droppe adopts na TD ko
Kokkos Kernels Added
IBM Power - NVIDIA GPUs
Kokko E adds ARM
Summit/Sierra
3 New Backe
Frontier

Open slide master to edit

The Kokkos Ecosystem - Today



Science and Engineering Applications

Kokkos Ecosystem

Kokkos Tools
- Debugging
- Profiling
- Tuning

Kokkos Kernels
- Linear Algebra Kernels
- Graph Kernels

Kokkos Core
- Parallel Execution
- Parallel Data Structures

Kokkos Support
- Documentation
- Tutorials
- Bootcamps
- App support

Kokkos Remote Spaces
- Global Arrays

PyKokkos
- Python Bindings
- Python Programming

Trilinos
- Distributed LA, Solvers, MultiGrid, UQ, AD, Discretization, Time Integration

PETSc
- PDE Solvers/Data Str.

ArborX
- Geometric Search

Cabana
- Particle Sim Toolkit

Resilience
- Data Snapshot
- Redundant Execution

Fortran Interop
- View Bindings

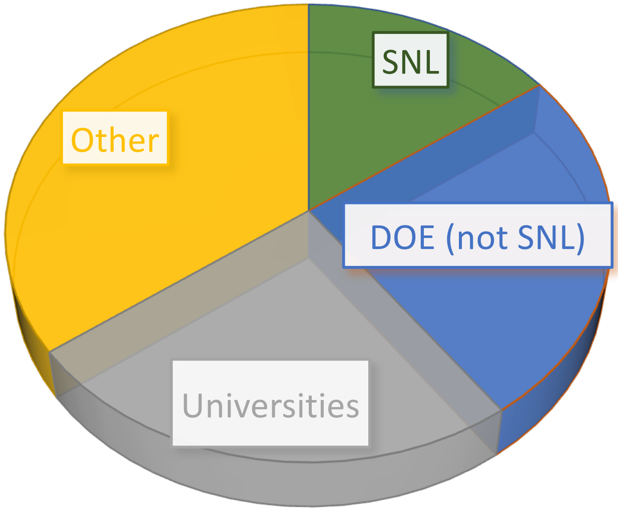
Open slide master to edit

Kokkos Community

Kokkos Slack

https://kokkosteam.slack.com

- >1200 registered users
- >150 institutions
  - Including 34 European



Kokkos Developers

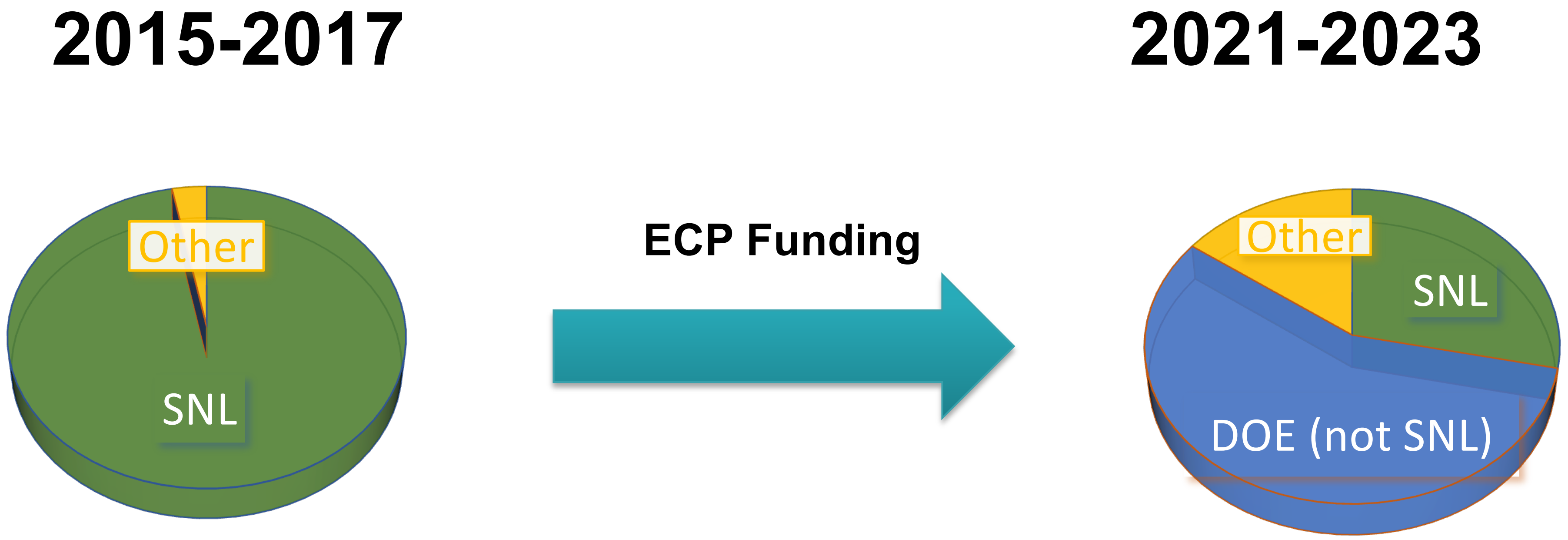
Sandia National Laboratories, Oak Ridge National Laboratory, Argonne National Laboratory, cea, Berkeley Lab, Los Alamos National Laboratory, CSCS, NexGen Analytics

Applications and Libraries

- Estimated 150-300 HPC projects using Kokkos
- On the order of three-dozen apps run science and engineering production runs with Kokkos
  - Many apps use multiple Kokkos based libraries
- Similar distribution as the Slack User

50% of C++ based DOE ECP codes use Kokkos

Kokkos Core - Contributions



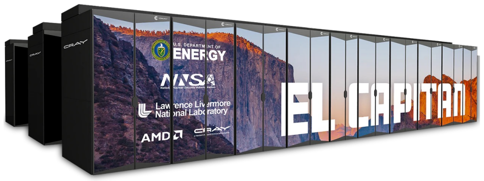
2015-2017

ECP Funding

2021-2023

- Most of Kokkos-Tools and Kokkos-Kernels development still at Sandia
- ISO C++ Contribution well distributed over labs

Frontier/Aurora support status



AMD
Frontier/El Capitan: HIP

Production-ready since Kokkos 4.0

- Fine grained tasking is missing

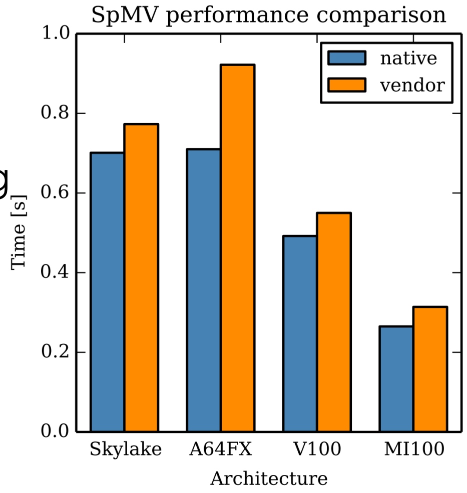
PR and nightly testing on AMD GPUs

Generally, performance is good

AMD GPUs struggle a bit compared to NVIDIA GPUs with cache intensive workloads

Performance Portability of Kokkos code is excellent however

See for example native Kokkos SPMV implementation beating vendor libraries for a range of use cases



Intel
Aurora: DPC++/SYCL

Still experimental

- DPC++/SYCL is still evolving (not fully stabilized)
- Tracking latest toolchain developments
- Regressions in functionality still common

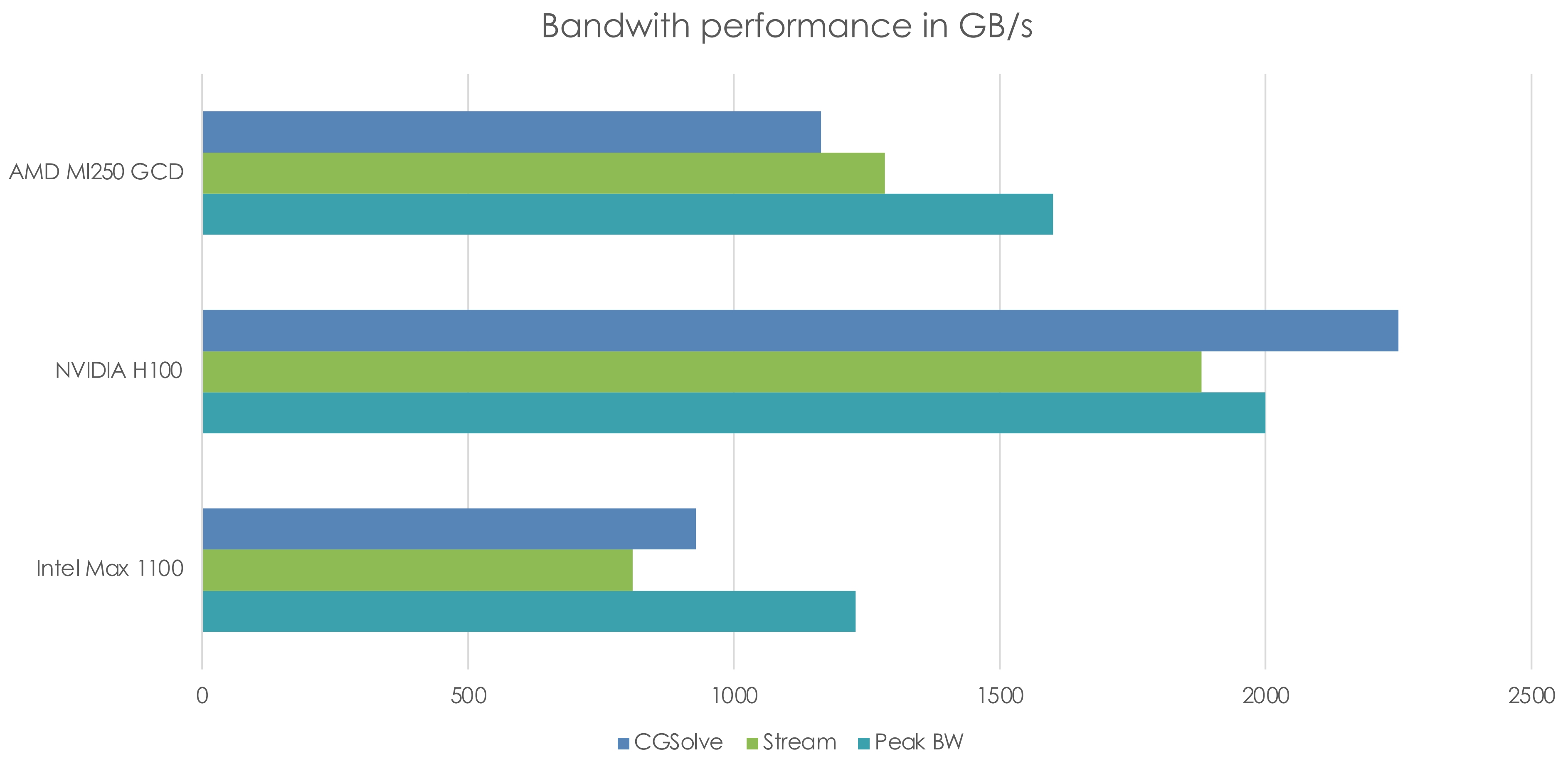
PR testing on NVIDIA GPUs, nightly testing on actual Intel PVC hardware

Most Kokkos-based ECP applications pass testing with the SYCL backend

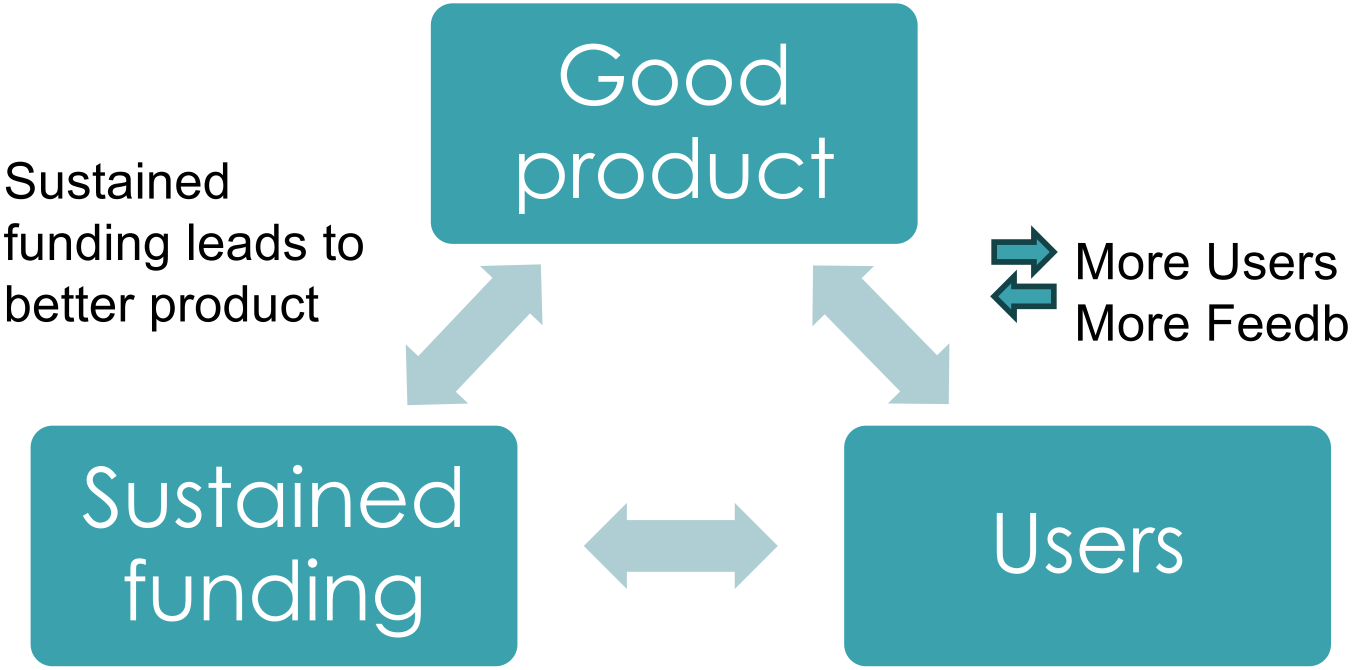
Performance similar to AMD

- Some issues around bandwidth – only getting about 65% of peak

Performance benchmarks



Sustainment: a self reinforcing cycle?


Good product

Sustained funding

Users

⇨ Sustained funding leads to better product

⇄ More Users
More Feedback

⇄ Sustained funding == trust
More usage leads to more funding

There is strength in numbers: collaboration on core product good for everyone!

Pillars for Long Term Sustainment

Open Source

- Enable wider set of contributor
- Risk mitigation for partner institutions – no one can just take the project away; worst case scenario is institutional fork with internal continued development
- Permissive license critical for industry participation

Core Funding

- Need a group of institutions to sustain core development team
  - NNSA – Sandia National Laboratories *(+ Los Alamos National Laboratory?)*
  - DOE – ASCR Facilities – Oak Ridge Leadership Computing Facility, NERSC, … ?
  - CEA starting now

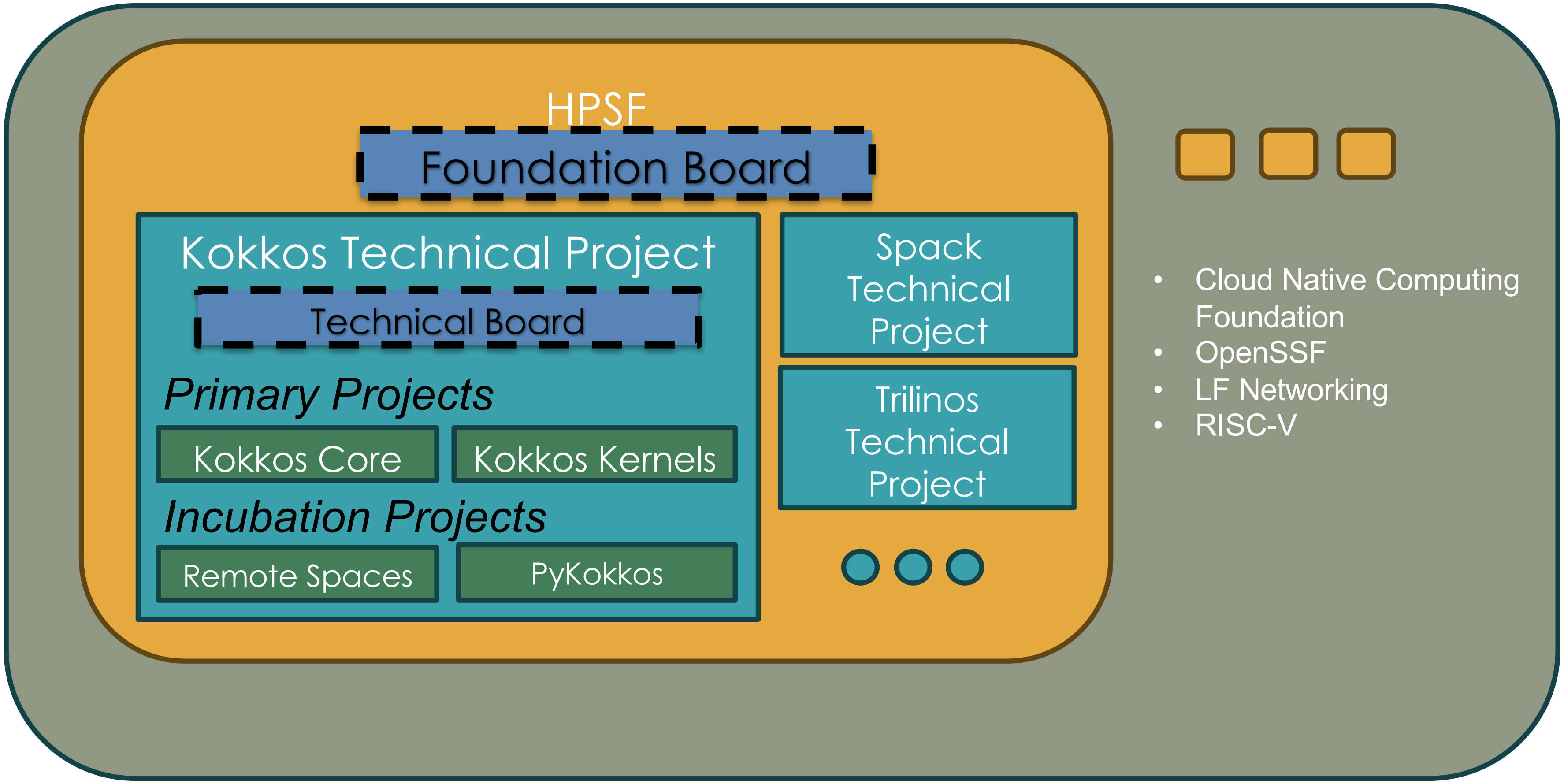
Open Governance

- Encourage participation of institutions by enabling say in direction
  - Enable path for new core funding teams to enter

High Performance Software Foundation

https://hpsf.io

Primary Goal: *Enable true partnerships on Kokkos via open governance.*



- Cloud Native Computing Foundation
- OpenSSF
- LF Networking
- RISC-V

Open slide master to edit

Kokkos and ISO C++

Long term sustainment via integration of Kokkos features into ISO C++ standard

Getting something into ISO C++

- Requires a lot of effort
  - mdspan was 9 years, but we didn’t know what we were doing
  - linalg took 5 years to get into draft
- Requires prototype and usage experience
  - Need to be able to show successful use in field by sizeable community

Kokkos as the HPCs proving ground

- Large enough community
- More
- Kokk nters

In the standard

- ”this” capture C++17
- atomic_ref C++20
- mdspan C++23

In flight for 26

- linalg – BLAS with extensions – *in draft*
- Batched linalg
- mdarray
- submdspan – *in draft*
- More accessors and layouts
- simd
- senders/receivers

We need long term engagement with ISO C++ as integral part of Kokkos effort.

Sustainment through standardization
Multi-dimensional arrays

```
Kokkos::View<double**> A("A", M, N);
Kokkos::View<double[4][4], Kokkos::LayoutLeft> B("B");
```

```
std::mdspan A(ptr, M, N);
std::mdspan<double, std::extents<int, 4, 4>, std::layout_left> B(ptr);
```

```
template <
   class DataType
   [, class LayoutType]
   [, class MemorySpace]
   [, class MemoryTraits]>
class Kokkos::View;
```

```
template<
   class T,
   class Extents,
   class LayoutPolicy = std::layout_right,
   class AccessorPolicy = std::default_accessor<T>
> class mdspan; (since C++23)
```

Sustainment through standardization
Linear algebra

```
dgemv('N', M, N, 1., A, 1, x, 1, 0., y, 1);  // 11 parameters     BLAS

KokkosKernels::gemv('N', 1., A, x, 0., y);                         KokkosKernels

std::matrix_vector_product(A, x, y);                               Standard C++
```

```
template<
   [class ExecutionHandle,]
   class InMat,
   class InVec,
   class OutVec>
void KokkosKernels::gemv (
    [const ExecutionHandle& exec,]
    const char trans[],
    typename InMat::const_value_type& alpha,
    InMat A,
    OutMat x,
    typename OutVec::const_value_type& beta,
    OutVec y);
```

```
template<
   [class ExecutionPolicy,]
   InMatrix InMat,
   InVector InVec,
   OutVector OutVec
> void matrix_vector_product( [ExecutionPolicy&& exec,]
                              InMat A,
                              InVec x,
                              OutVec y );  (since C++26)
```

Our ideas for future directions of Kokkos

Edge computing / Embedded Support

- Many of the same concerns as HPC – resource constraint, performance critical
- Many different devices including FPGAs

Programming Language Safety

- More concern about cyber security – how do we write safer code?
- Kokkos data abstractions (View/mdspan/mdarray) enable safer encapsulation – could make it almost impossible to have out-of-bounds memory access
- Combined with static analysis could be significant step to enable C++ codes which are memory safe by design

Better integration with distributed computing

- Remote spaces
- MPI interface taking Kokkos data structures

kokkos