December 2017 ECP ST Project Review

SAND2017-13792PE

ECP Project WBS 2.3.1.04 : SNL ATDM PMR

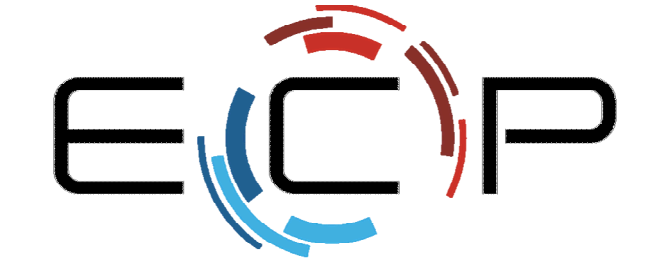
Kokkos Research & Development
DARMA Research & Development

SNL ATDM PMR PM: Robert Clay

Kokkos PI: H. Carter Edwards; *acting* PI: Christian Trott

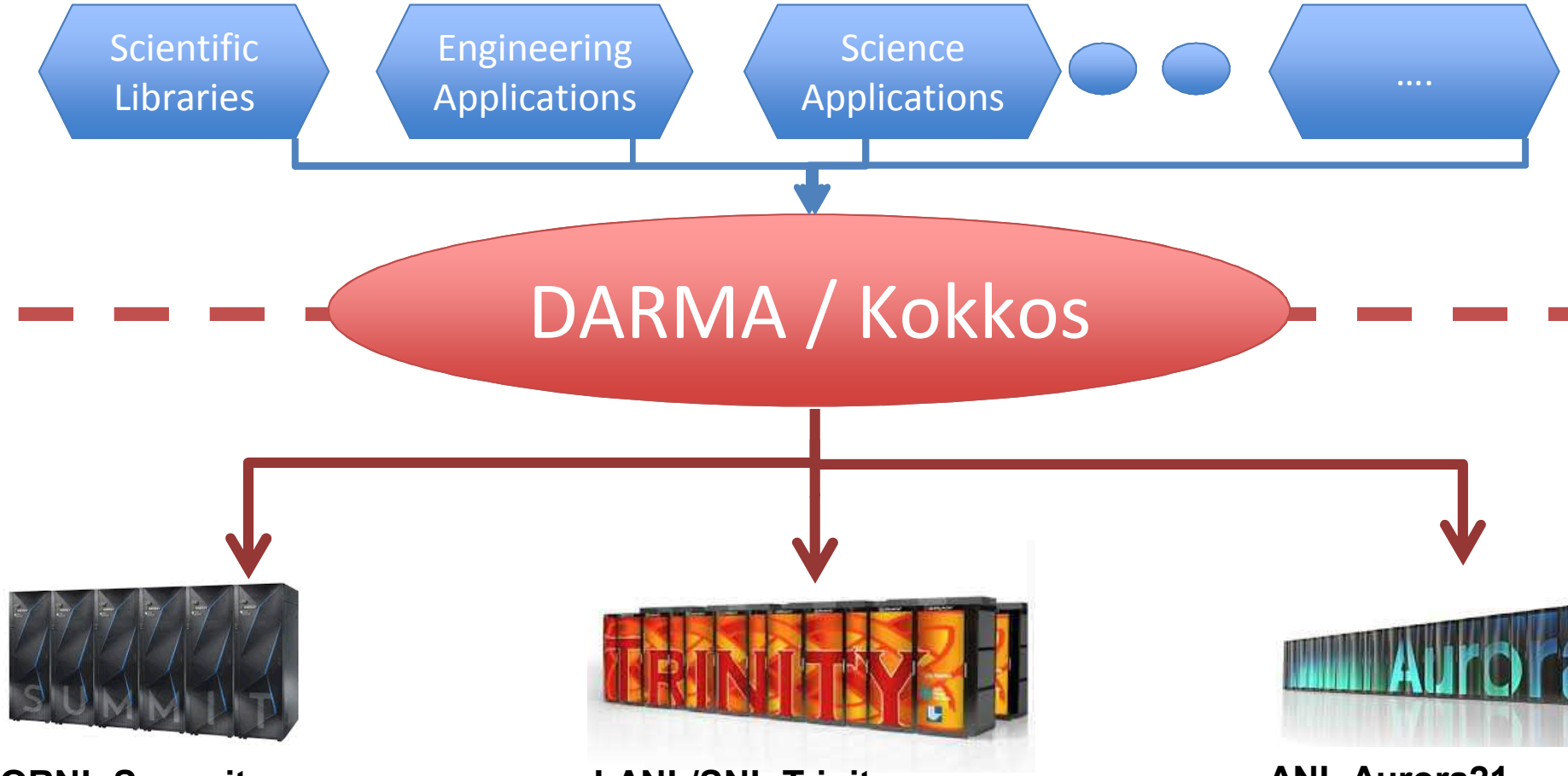
DARMA PI: Jeremiah Wilke

Date: December 20, 2017



SNL ATDM PMR Overview

C++ Abstractions to isolate applications from HPC HW and RT diversity



ORNL Summit
IBM Power9 / NVIDIA Volta

LANL/SNL Trinity
Intel Haswell / Intel KNL

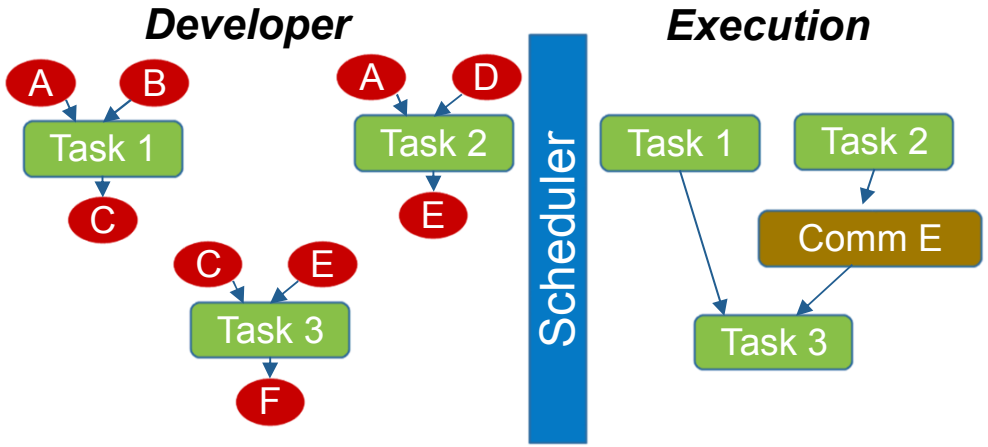
ANL Aurora21
Intel unannounced Novel Architecture

SNL ATDM PMR Overview

DARMA

Research and Prototyping

- Implicit data-effects task-parallel programming

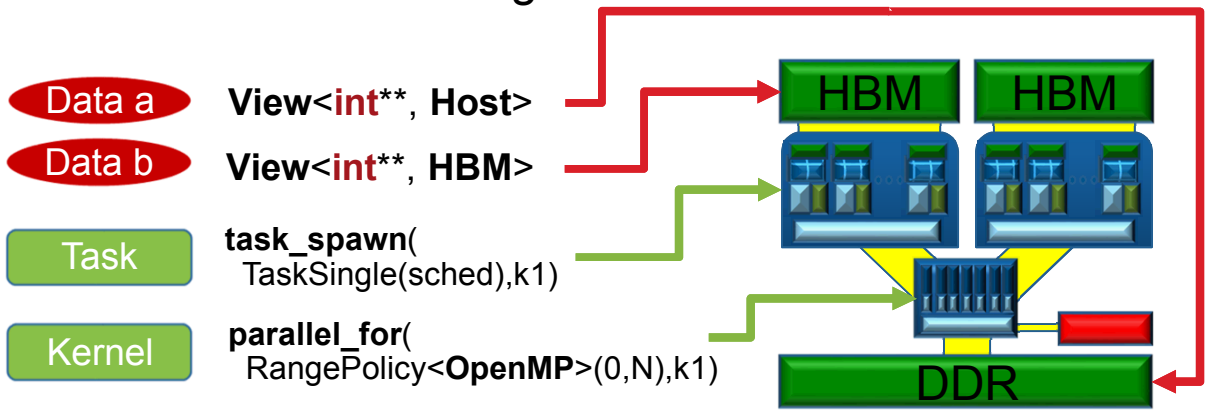


- Goals:
  - More productive app development
  - Implicit communication, load balancing
  - Exploit data utilization knowledge for more effective work scheduling
  - Efficient use of both shared and distributed memory

Kokkos

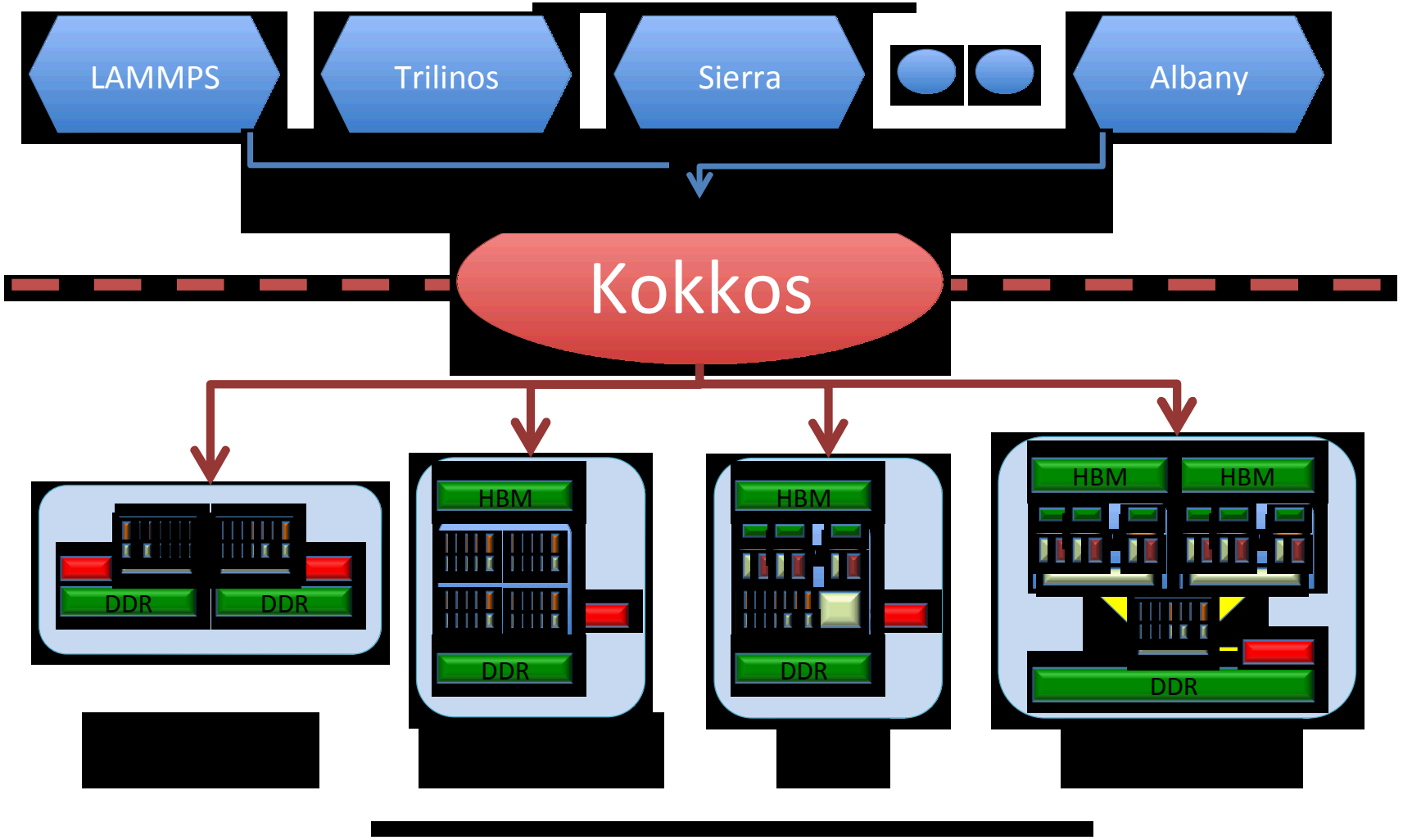
Production Growth

- Explicit data and task parallel programming for hierarchical and heterogeneous architectures



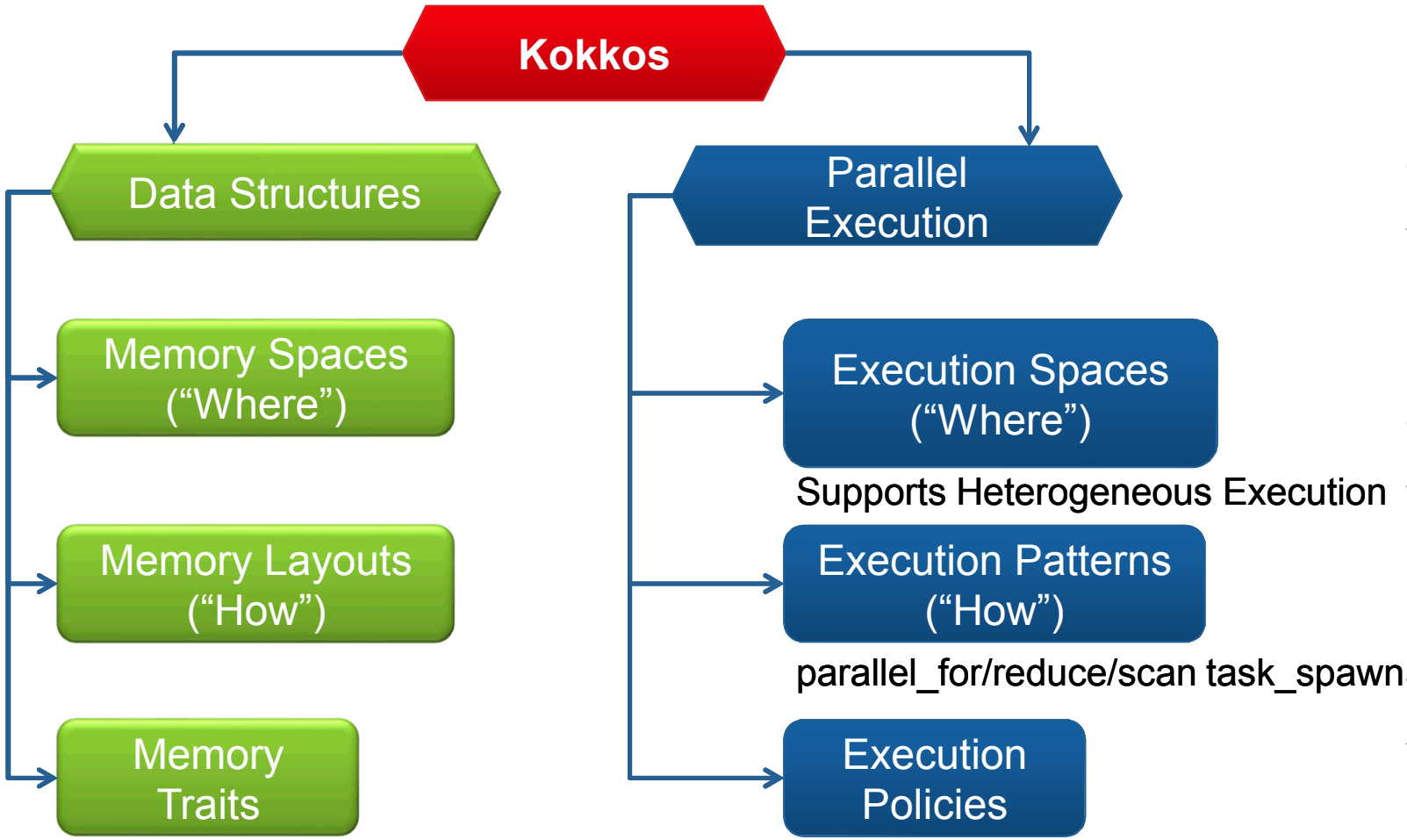
- Goals:
  - Performance portability over hardware architectures
  - Delivering well established, productive abstractions for explicit parallelism
  - Provide high quality implementation for production applications

Kokkos: Mapping Applications to Diverse Node-Architectures



https://github.com/kokkos

Kokkos: On-Node Data and Execution Abstractions



Supports Heterogeneous Execution

parallel_for/reduce/scan task_spawn

Access Intent and Behavior
Stream, Random, Atomic

- Range, Team, Task-Dag
- Dynamic / Static Scheduling
- Support non-persistent scratch-pads

Example Memory Layouts:

AOSOA SIMD transform

Typedefs to change data Layout

Traditional C-Layout:

View<float*[3],LayoutRight>

Traditional C-Layout but with SIMDType:

View<SIMDfloat*[3],LayoutRight>

View the SIMD data as simple scalar:

View<float*[3],LayoutSIMDRight>

EXASCALE COMPUTING PROJECT

DARMA: Data effects programming embedded in C++ for handling multi-level, dynamic, or irregular parallelism

C++ abstraction to capture data effects in tasks

Library calls for defining tasks/granularity

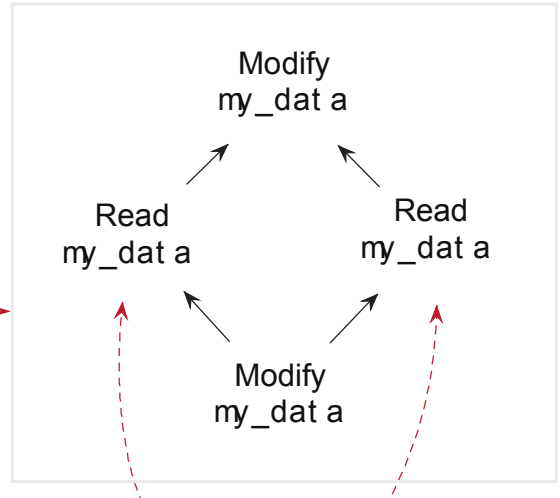
Programming Model

Example Program

```
AccessHandle<int> my_data;

darma::create_work([=]{
  my_data.set_value(29);
});

darma::create_work(
  reads(my_data), [=]{
    cout << my_data.get_value();
  }
);

darma::create_work(
  reads(my_data), [=]{
    cout << my_data.get_value();
  }
);

darma::create_work([=]{
  my_data.set_value(31);
});
```

Semantics

DAG (Directed Acyclic Graph)

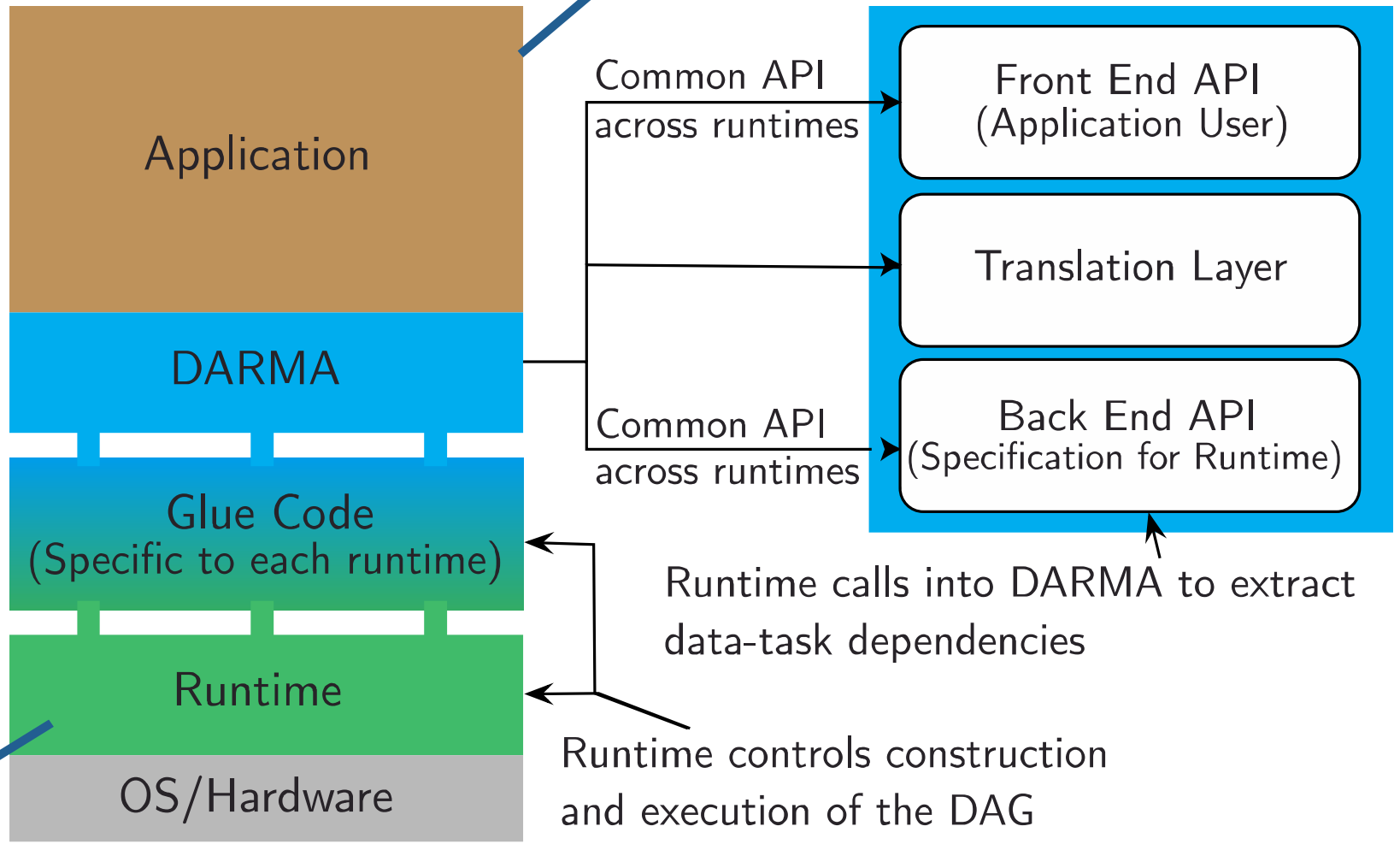


These two tasks are concurrent and can be run in parallel by a DARMA backend!

Distributed, ***A***synchronous, ***R***esilient ***M***odels for ***A***pplications

The DARMA software stack model

- App-facing frontend provides programming abstractions
  App-driven co-design
- Translation layer provides semantics and captures data effects
- Runtime-facing backend expresses granularity, concurrency, locality, etc
  Runtime-driven co-design



Current focus: MPI + Kokkos
Previous (and future?) focus:
Charm++, std::threads, HPX

Development Timeline

DARMA

- Load balancing and tasking abstractions
- Using MPI + Kokkos/OpenMP
- DARMA tasks can dispatch Kokkos kernels
- ProxyApp: PIC code

2018

- ProxyApp: MiniMultiScale including Contact
- Dispatch Kokkos GPU kernels
- Data Tiling Specification

2019

- Performance for MiniMultiScale
- Demonstrate capability in real application

2020

Kokkos

- Support Node Partitioning for GPUs
- Remote Memory Spaces
  - Initial API + MMap and PGAS implementation
  - Potentially MPI OneSided support: Resource Constraint
- Pay down of Technical debt
- ROCm Backend (AMD) maturation

- Tiling data structures and execution
- OpenMP5 Target Backend Functional
- Backend for A21 Intel Architecture

- Mature Backend for A21

DARMA / Kokkos Integration

- Kokkos provides tools to execute parallel work and manage memory
- DARMA analyzes data flow, determines scheduling order, and facilitates communication
- Goal: Maximize leveraging / minimize duplication

DARMA		Kokkos / Communication Layer
Data Allocation	→	Kokkos::View / Kokkos::alloc
Data Movement Intra Node	→	Kokkos::deep_copy
Data Movement Inter Node	→	MPI, PGAS, …
Execute Data Parallel Task	→	Kokkos::parallel_for/reduce/scan
Execute Portion of Task Graph	→	Kokkos::task_spawn

Kokkos Project Overview: Goals

- Goal #1: Performance Portability and Productivity Across NGP
  - HPC applications and libraries using C++ and intra-node parallelism
  - Diverse architectures: multicore CPU, Xeon Phi, NVIDIA GPU, AMD GPU, ...
  - R&D for programming model abstractions
  - R&D for back-end implementations
  - R&D collaborations with labs, vendors, universities, ...
  - Supporting ASC/ ATDM and IC applications and libraries

- Goal #2: Influencing ISO/C++ Standards
  - Adopt abstractions enabling productivity and performance portability
  - Emphasizing broadly applicability and high productivity

DARMA Project Overview: Goals

- Goal #1: Data-centric higher-leveling programming abstractions with focus on community involvement, best practices, and standards
  - Make it *possible* but not *required* for runtime to tune granularity, concurrency, operation order, data movement, and load balancing
  - Leverage existing runtime infrastructure wherever possible/feasible
    - Kokkos provides knobs for controlling execution
  - Emphasis on standards-compliant C++
- Goal #2: Application and results-driven development process
  - Design driven by apps and existing/expected productivity/performance bottlenecks
  - Identify abstractions and high-level semantics that help compiler or runtime to solve performance and portability challenges
- Goal #3: Efficient use of both shared and distributed memory
  - Hooks for converting shared-memory to distributed-memory tasks created at *compile-time*
  - Load balancing or distributed overheads only incurred at runtime when required

SNL ATDM PMR: Team and Budget

- Kokkos
  - Team staffing: Edwards (PI), Trott (*acting* PI), Sunderland, Ibanez, Ellingwood, Labreche
  - Budget FY18: $1590k

- DARMA
  - Team staffing: Wilke (PI), Hollman (C++ lead,Frontend), Borghesi (Frontend), Lifflander (Backend), Markosyan (Backend, PIC app), Morales (arrives Jan; Kokkos integration, solid mechanics app)
  - Budget FY18 = $1600k

Project Overview: Impact Goals and Metrics

- ECP, ASC ATDM and IC applications using DARMA/Kokkos demonstrate performance portability and productivity
  - Measured through applications' successes
  - Kokkos: Production apps effectively using all HPC node architectures
  - DARMA: Demonstrate applicability and potential productivity/performance gains
- Refinement of abstractions to meet applications' needs
  - Enhancements requested by and developed for ASC and ECP applications
- Leverage best programming mechanisms for target architectures
  - Kokkos: On-Node e.g., OpenMP4, CUDA9 and ROCm
  - DARMA: On-Node (Kokkos or native OpenMP) and Inter-Node (MPI, Charm++, HPX)
- Advocacy for performance portability and productivity abstractions in future ISO/C++ Standard for HPC
  - Enabling proposals identified/developed and shepherded through ISO/C++ committee

Kokkos Project Plan : ECP JIRA Deliverables

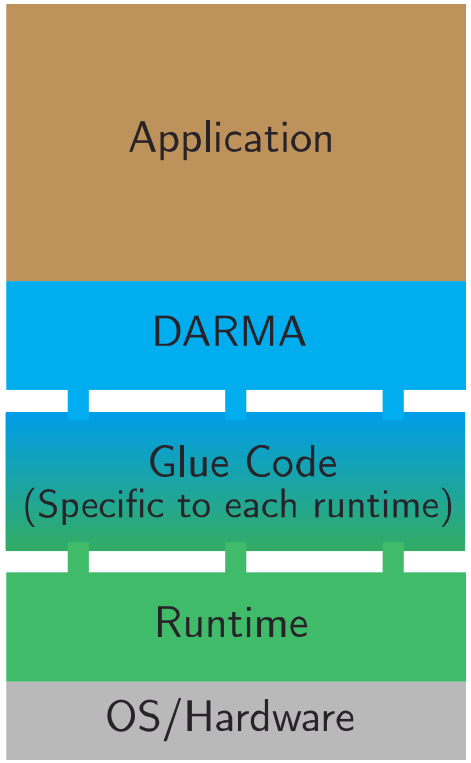
- Back-end R&D and external collaborations (STPR04-4)
  - Focusing on ATS1/ATS2/ARM; OpenMP4 and CUDA9
  - Collaborating with labs, vendors, universities, ISO/C++ standard
- Abstraction R&D for SIMD types and remote memory spaces (STPR04-5,7)
  - Portable, explicit SIMD vectorization because compilers cannot
    - Collaboration with 2.3.3.04 SNL ATDM Math Libraries
  - Remote memory spaces for one-sided halo-exchange deep copy
- Support for ASC application and project management (STPR04-06)
  - Training, consulting, enhancements for ASC ATDM and IC
  - Substantial project management load

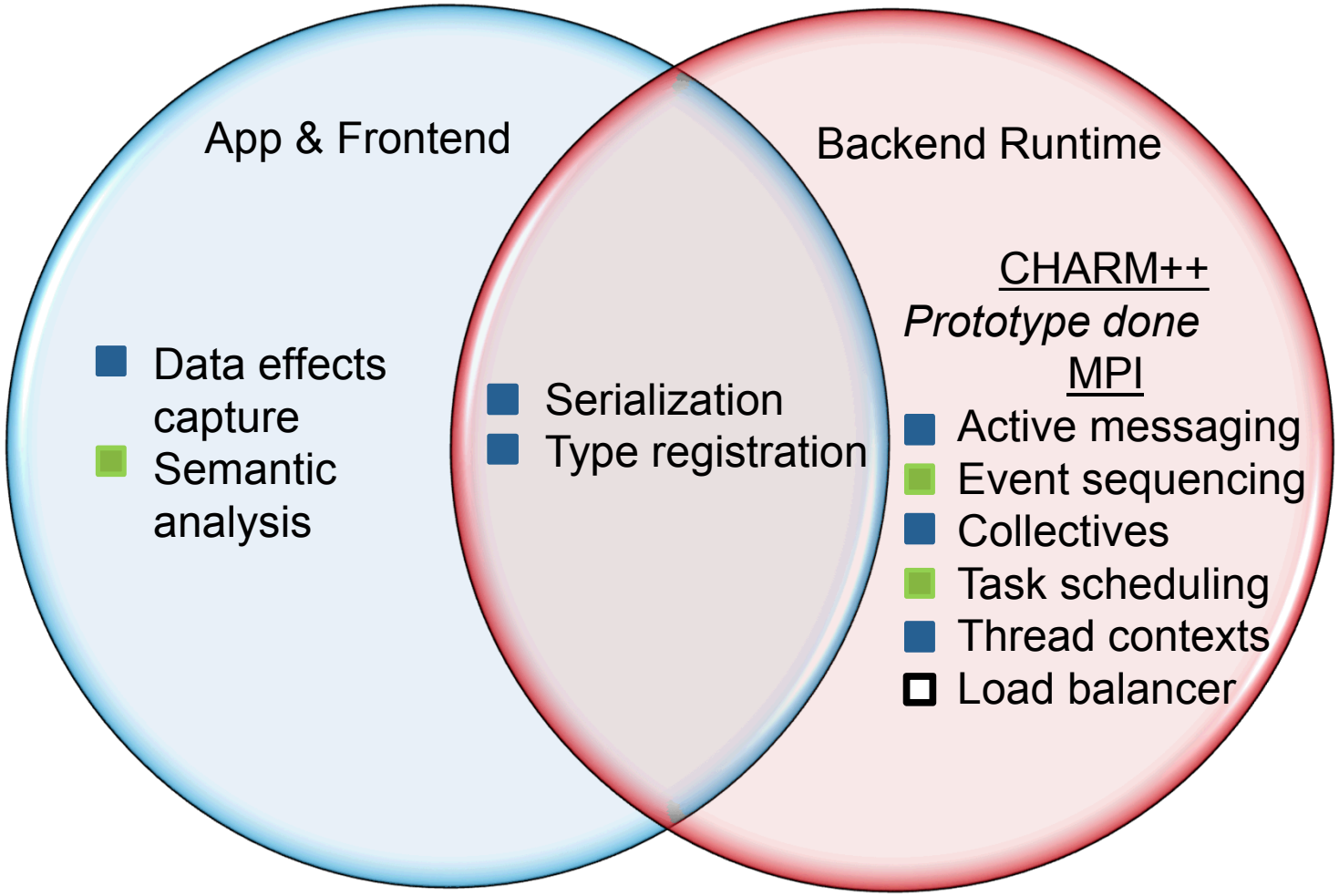
DARMA Project Plan : ECP JIRA Deliverables

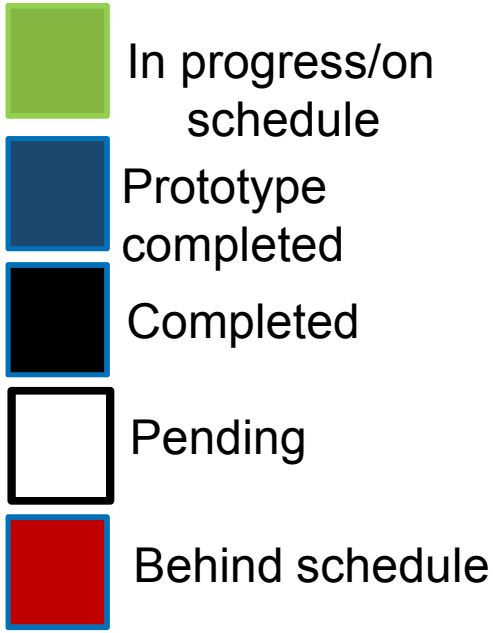
- Milestone STPR04-13: Reliable and stable code base
  - Open-source release plan completed with Sandia copyright
  - Sandia Jenkins infrastructure configured and linked to GitHub with unit tests for each individual frontend/backend feature implemented
  - To-do: Implement auto-tester/auto-merger, finalize 1.0 release
- Milestone STPR04-15: DARMA-MPI interoperability
  - MPI-Compatible components in development (See 04-14)
  - To-do: Frontend programming abstractions for DARMA-MPI handoff in development (Q2), full demonstration (late Q3/early Q4)
- Milestone STPRO4-16: Kokkos interoperability
  - Use Kokkos for executing thread-parallel tasks instead of just serial tasks
  - Backend prototype complete using Kokkos thread pools/partitioning
  - To-do: Finalize abstractions for expressing intra-task parallelism and elasticity

DARMA Project Plan : ECP JIRA Deliverables STPR04-14: Component-based development

DARMA software stack model







Kokkos is essential for sustainable, production-quality ASC applications and libraries on NGPs

- Exascale compute node challenges
  - Heterogeneous, diverse, and evolving execution and memory architectures
  - Concurrency increasing must faster than memory capacity
  - Diversity of “best” programming mechanism (e.g., OpenMP, CUDA) for each architecture
- Performance portable and productive programming model, for C++
  - On-node shared memory parallelism with heterogeneous execution and memory
  - Productivity via explicit data and task parallel patterns and abstractions for heterogeneous arch.
  - Applications can develop and maintain a single source code base for diverse NGPs
  - Unique (vs. OpenMP, OpenACC, ...) capability to trivially transform data layout
  - Agile research, prototyping, evaluation, development, and deployment of capabilities
- Future-proofing
  - Engaged with ISO/C++ to semantically align with future standard
  - Collaborating with vendors to anticipate and influence architectures and programming mechanisms

DARMA provides data-centric programming abstractions essential for productivity/performance in exascale era

- Data-effects programming makes app developers more productive
  - *Now*: Lower threshold to experiments with tasking, load balancing, code coupling, resilience
  - *Future*: Mapping applications to fundamentally new architectures, resilience methods, or even problem domains requires significant code changes for execution-centric models
- Lack of data-centric models may delay new dynamic or multi-physics codes
  - Instead of desired algorithms, developers settle for algorithms that are ”good enough”
  - Slows pace of algorithmic exploration or adding new physics
- We need robust programming models for the entire exascale era, not just models good enough to cross the initial exascale threshold

Delivery of Kokkos to Users, since 2015

- Regular releases at github.com/kokkos/kokkos
  - First released stand-alone March 2015; BSD 3-clause
    - Prior to March 2015 development and release was through Trilinos
  - development -> master promotion model with extensive up-stream integration testing
  - Nightly multi-architecture and multi-configuration testing of develop branch (using Jenkins)
    - Over 200 configurations (Architecture/Compiler/Backend/Options combinations)
  - Fully utilizing github pull requests, issue management, and project kanban boards
- User support – ramping up
  - Augmented by ECP ST 2.3.1.10 Kokkos Support (starting Jan'2017)
  - User guide wiki at github.com/kokkos/kokkos/wiki (released Nov'2017, previously as PDF)
  - Slack channel at kokkosteam.slack.com (currently underutilized)
  - Many tutorials, hackathons, and bootcamps for ASC, ECP, at conferences, with collaborators
  - More than 250 developers attended some form of Kokkos Tutorial

Delivery of DARMA to Users (Feb 2018)

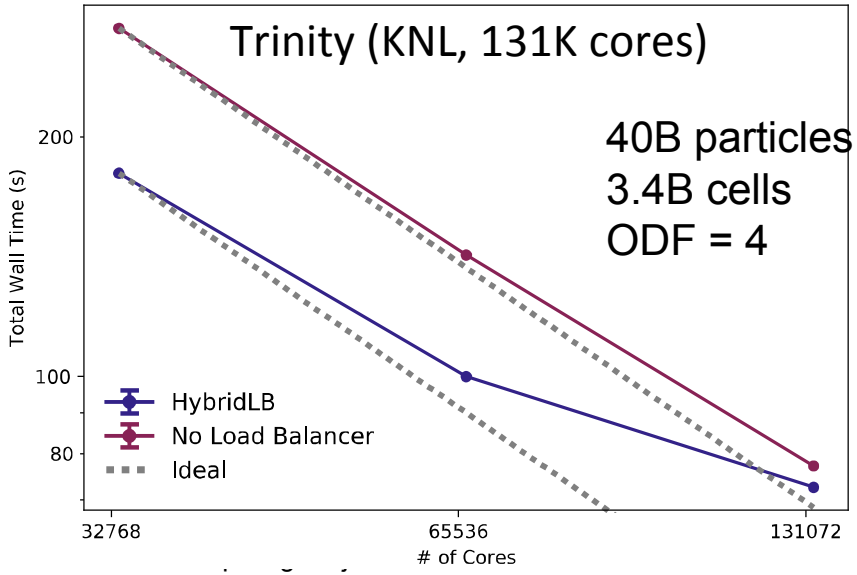
- First public release at ECP all-hands meeting in Feb 2018
  - Open-source copyright process completed with Sandia-modified BSD-3 license
  - Spack packages to follow
  - API preservation across minor release numbers
- Documentation and user-support through GitHub repository
  - Wiki with documentation (PDF manuals in releases)
  - Issue tracker and mailing list for user questions
  - Regular DARMA Skype/BlueJeans help sessions to discuss open issues following 1.0 release
- Feature/Devel/Master branch model for repository
  - Pull requests accepted into Devel after Jenkins tests
  - Devel tested nightly on Jenkins
  - Master/Devel merge at regular intervals after consistent nightlies

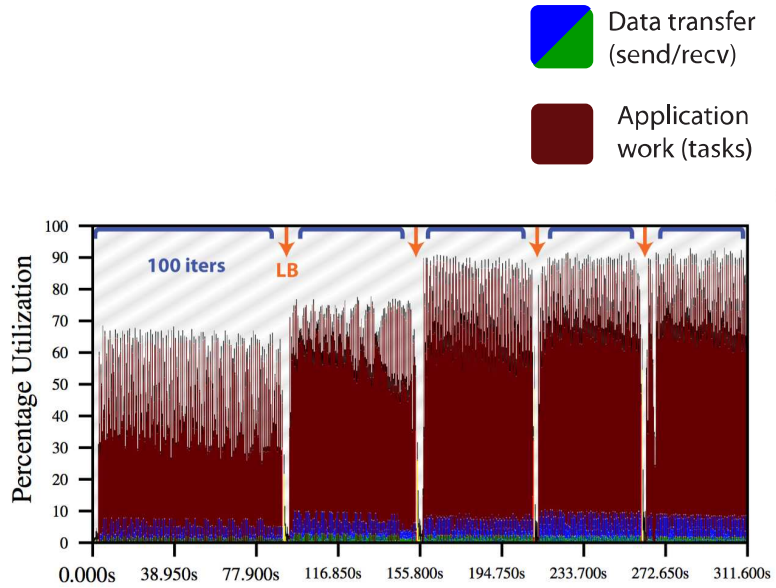
Kokkos progress toward goals; FY17 highlights

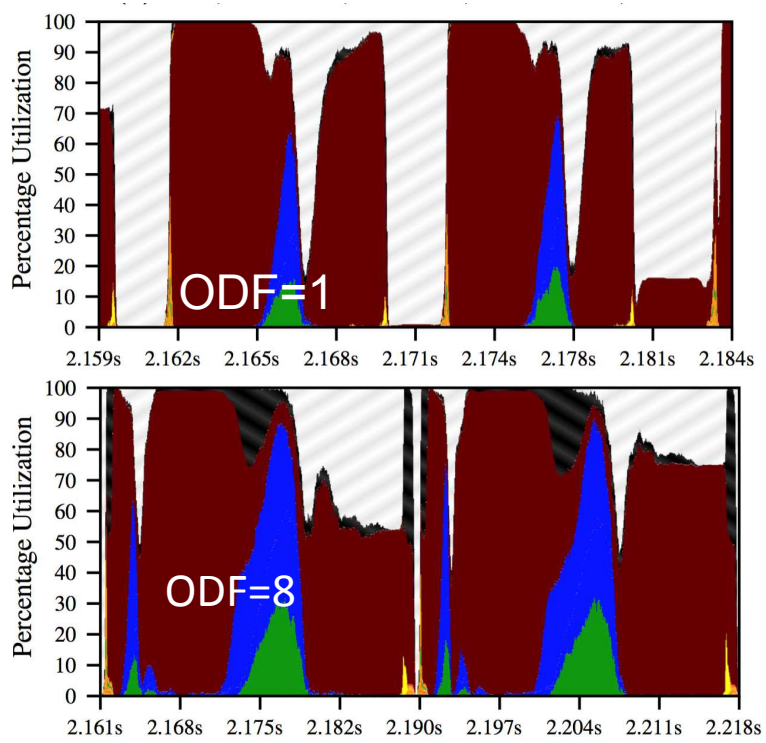
- Deployed production portable task-dag capabilities (micro-tasking)
  - Dynamic heterogeneous task-dag; used in Tacho sparse matrix factorization
  - Static homogeneous work-dag; used in LANL Tycho2 neutral particle transport via sweeps mini-app
  - Thread scalable memory pool
- Deployed production support for “macro tasking” / resource management
  - Coarse grain, long duration splitting and management of CPU threads
  - Fully interoperable with OpenMP
  - Integrated into Uintah @ U-Utah, publication in progress
- Application-driven enhancements
  - Dynamic rank multidimensional array view
  - Multidimensional parallel range policy (~OpenMP loop collapse)
- ISO/C++ Standard engagement
  - Championed “lambda-capture-*this” in C++17 and CUDA host-device offload
  - Hosted November ISO/C++ Standard committee meeting
- Presentations and Tutorials
  - PADAL, SC, JOWOG, PSAPP WEST, SIAM CS&E, GPU-Tech

DARMA progress toward goals; FY17 highlights

- Deployed prototype DARMA-compliant stack (Charm++ backend) as part of successful ASC ATDM L2 milestone
- Initial scalability on Trinity to test load balancing, task pipelining via problem over-decomposition factor (ODF)
  - Identified most critical areas for improvement for MPI + OpenMP/Kokkos backend







EXASCALE COMPUTING PROJECT

Kokkos progress toward impact goals

- Capabilities are being enhanced in response to application needs
  - Documented through many enhancement requests closed / release notes
- SNL ASC ATDM and IC C++ applications are “all in”
  - Milestones with performance goals using Kokkos for Aria, Empire, Sparta, SPARC, Sierra SM Contact among others
- Numerous ECP applications using/integrated/exploring/considering (FY17 WBS)

  AD-1.2.1.03-LatticeQCD, AD-1.2.1.04 EXAALT, AD-1.2.1.07-ExaWind, AD-1.2.1.08-EXASMR, AD-1.2.1.09-QMCPACK, AD-1.2.1.10-ExaAM, AD-1.2.1.11-NWChemEx, AD-1.2.1.14-CombustionPele, AD-1.2.1.20 ExaBiome, AD-1.2.2.01-LANLapp, AD-1.2.2.03a-SNLapp, AD-1.2.2.03b-SNLapp, AD-1.2.5.3.3-CoPA, AD-1.2.5.3.5-CEED, ST-1.3.1.06-DARMA, ST-1.3.1.10-Legion, ST-1.3.1.11-Parsec, ST-1.3.2.09-ExaPAPI, ST-1.3.3.01-KokkosKernels, ST-1.3.3.02-TrilinosSolvers, ST-1.3.3.03a-AgileComponents, ST-1.3.3.03b-DataProp, ST-1.3.3.05-xSDK4ECP, ST-1.3.3.12-FleCSI, ST-1.3.3.15-ALExa, ST-1.3.4.05-DataWarehouse, ST-1.3.5.05-VTKm, ST-1.3.8.04-AAPS
- ISO/C++: lambda-capture-*this, floating point atomic, atomic reference, multidimensional array, SIMD
- Utilizing SNL Testbeds for early porting and optimization of back-ends
- FY17-18 back-ends: std::threads, OpenMP4, OpenMP4.5, NVIDIA CUDA9, AMD ROCm

DARMA progress toward impact goals

- Applications using/considering DARMA:
  - 2.2.5.03 ADNN03-ASC ATDM SNL Application
  - 2.2.1.02 ADSE11-NWChemEx
- FY17-18 leveraging infrastructure/vendor-supported libraries:
  - Charm++ backend tested at scale
  - MPI+OpenMP, std::threads, HPX backends in development
  - Engagement with HiHat
- Interoperability being explored with:
  - ST-1.3.3.01-KokkosKernels
  - ST-1.3.3.02-TrilinosSolvers
  - ST-1.3.4.05-DataWarehouse
- Team members active on C++ standards committee

Kokkos relationship with other software

- Comparison to RAJA @ LLNL
  - This Kokkos Support project is actively engaging ECP projects with training and support
    - Intentionally diversifying usage domains to insure broad applicability of programming model
  - Relative to our outward focus, RAJA is inwardly focused on LLNL applications
  - RAJA, aided by other LLNL projects, are working to catch up with Kokkos capabilities...
    - E.g., LLNL CHAI for heterogeneous memory data management
    - Details are in flux / limited information about current development state
  - Kokkos has single, holistic programing model strategy
- Comparison to OpenMP, OpenACC, OpenCL, ...
  - Language / compiler extensions versus pure C++ library interface
    - Kokkos is more agile for prototyping / developing new abstractions
  - No multidimensional array (with polymorphic layout); key for performance portability
  - Explicit multi-dependence directed acyclic graph of tasks (task-DAG)
  - Low overhead work-DAG via CRS graph of dependences

DARMA relationship with other software

- Data-effects programming through standards-compliant C++
  - Similar to Legion, but focus there on new Regent language
  - HPX also C++-centric, but lacks data-centric abstractions
  - RUST language defines similar asynchronous semantics, but new language
- High-level programming abstractions with performance-portable mapping to architectures
  - Similar to FleCSI, but DARMA focus on expressing tasks + data effects while FleCSI is one-step above as almost a DSL for expressing mesh operations
- Community best practices, standards for (AMT) tasking runtimes
  - Beginning engagement with HiHat
  - Worked with Open Community Runtime (OCR)
  - Participated in various conference groups (Workshops, Panels, BOFs, etc)

Kokkos Risks / Concerns / Issues

- Understaffed / underfunded
  - Scaling up user support with growing user base
  - Regrets given for R&D of ATDM application requested execution and data tiling capability
  - Publication-worthy R&D going unpublished
- Risk: accumulating technical debt
  - Understaffing and high support demands preventing “paying down” technical debt
- Risk: staffing / structure changes
  - Project PI is *acting* line manager, further reducing available technical resources
    - Temporary? To-be-determined March’18
  - We’re working through the new PMR alignment – so far the results are positive but has introduced some uncertainty

DARMA Risks / Concerns / Issues

- Application-driven design requires buy-in from app and performance analysis teams
  - Many app teams driven by short-term priorities for 2019 and 2020
  - Lacking a list of “diagnosed performance bottlenecks” from performance analysis teams
  - Application space is moving target
- Staffing
  - Highly specialized compiler/runtime/C++ knowledge required for implementing DARMA libraries
- Compiler support for C++11/14
  - C++11 required (and a few C++14 features)
  - Nominally supported by older compiler versions, but DARMA has exposed internal compiler bugs
  - There *exist* Clang/GCC/ICC versions that work, but may not be versions apps teams want
- Demonstrating incremental value when full payoff potentially not realized until after first exascale machines delivered
  - DARMA likely not critical to cross *exascale threshold;* needed for *exascale era*
  - Management changes (within Sandia and ECP) can change risk tolerance and focus
  - On-node performance optimization may shift more performance bottlenecks into DARMA domain – application space is moving target