ECP Software Technology Overview

Michael A. Heroux, Sandia National Laboratories
Director of Software Technology

E4S at DOE Facilities with Deep Dive at NERSC, October 4, 2021
ECP Organizational Sketch
ECP Software Technology (ST) is one of three focus areas

**Performant mission and science applications @ scale**

- **Aggressive RD&D Project**
- **Mission apps & integrated S/W stack**
- **Deployment to DOE HPC Facilities**
- **Hardware tech advances**

**Application Development (AD)**
- Develop and enhance the predictive capability of applications critical to the DOE
  - **24 applications** including national security, to energy, earth systems, economic security, materials, and data

**Software Technology (ST)**
- Deliver expanded and vertically integrated software stack to achieve full potential of exascale computing
  - **70 unique software products** spanning programming models and run times, math libraries, data and visualization

**Hardware and Integration (HI)**
- Integrated delivery of ECP products on targeted systems at leading DOE HPC facilities
  - 6 US HPC vendors focused on exascale node and system design; application integration and software deployment to facilities
ECP ST has six technical areas

Programming Models & Runtimes
- Enhance and get ready for exascale the widely used MPI and OpenMP programming models (hybrid programming models, deep memory copies)
- Development of performance portability tools (e.g., Kokkos and Raja)
- Support alternate models for potential benefits and risk mitigation: PGAS (UPC++/GASNet), task-based models (Legion, PaRSEC)
- Libraries for deep memory hierarchy and power management

Development Tools
- Continued, multifaceted capabilities in portable, open-source LLVM compiler ecosystem to support expected ECP architectures, including support for F18
- Performance analysis tools that accommodate new architectures, programming models, e.g., PAPI, Tau

Math Libraries
- Linear algebra, iterative linear solvers, direct linear solvers, integrators and nonlinear solvers, optimization, FFTs, etc
- Performance on new node architectures; extreme strong scalability
- Advanced algorithms for multi-physics, multiscale simulation and outer-loop analysis
- Increasing quality, interoperability, complementarity of math libraries

Data and Visualization
- I/O via the HDF5 API
- Insightful, memory-efficient in-situ visualization and analysis – Data reduction via scientific data compression
- Checkpoint restart

Software Ecosystem
- Develop features in Spack necessary to support all ST products in E4S, and the AD projects that adopt it
- Development of Spack stacks for reproducible turnkey deployment of large collections of software
- Optimization and interoperability of containers on HPC systems
- Regular E4S releases of the ST software stack and SDKs with regular integration of new ST products

NNSA ST
- Open source NNSA Software projects
- Projects that have both mission role and open science role
- Major technical areas: New programming abstractions, math libraries, data and viz libraries
- Cover most ST technology areas
- Subject to the same planning, reporting and review processes
ECP Software Technology Leadership Team

**Mike Heroux, Software Technology Director**
Mike has been involved in scientific software R&D for 30 years. His first 10 were at Cray in the LIBSCI and scalable apps groups. At Sandia he started the Trilinos and Mantevo projects, is author of the HPCG benchmark for TOP500, and leads productivity and sustainability efforts for DOE.

**Lois Curfman McInnes, Software Technology Deputy Director**
Lois is a senior computational scientist in the Mathematics and Computer Science Division of ANL. She has over 20 years of experience in HPC numerical software, including development of PETSc and leadership of multi-institutional work toward sustainable scientific software ecosystems.

**Rajeev Thakur, Programming Models and Runtimes**
Rajeev is a senior computer scientist at ANL and most recently led the ECP Software Technology focus area. His research interests are in parallel programming models, runtime systems, communication libraries, and scalable parallel I/O. He has been involved in the development of open source software for large-scale HPC systems for over 20 years.

**Jeff Vetter, Development Tools**
Jeff is a computer scientist at ORNL, where he leads the Future Technologies Group. He has been involved in research and development of architectures and software for emerging technologies, such as heterogeneous computing and nonvolatile memory, for HPC for over 15 years.

**Xaioye (Sherry) Li, Math Libraries**
Sherry is a senior scientist at Berkeley Lab. She has over 20 years of experience in high-performance numerical software, including development of SuperLU and related linear algebra algorithms and software.

**Jim Ahrens, Data and Visualization**
Jim is a senior research scientist at the Los Alamos National Laboratory (LANL) and an expert in data science at scale. He started and actively contributes to many open-source data science packages including ParaView and Cinema.

**Todd Munson, Software Ecosystem and Delivery**
Todd is a computational scientist in the Math and Computer Science Division of ANL. He has nearly 20 years of experience in high-performance numerical software, including development of PETSc/TAO and project management leadership in the ECP CODAR project.

**Kathryn Mohror, NNSA ST**
Kathryn is Group Leader for the CASC Data Analysis Group at LLNL. Her work focuses on I/O for extreme scale systems, scalable performance analysis and tuning, fault tolerance, and parallel programming paradigms. She is a 2019 recipient of the DOE Early Career Award.
ST L4 Teams

- WBS
- Name
- PIs
- PCs - Project Coordinators

ECP ST Stats

- 35 L4 subprojects
- 11 PI/PC same
- 24 PI/PC different
- ~27% ECP budget

<table>
<thead>
<tr>
<th>WBS</th>
<th>WBS Name</th>
<th>CAM/PI</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Software Technology</td>
<td>Thakur, Rajeev</td>
<td></td>
</tr>
<tr>
<td>2.3.1</td>
<td>Programming Models &amp; Runtimes</td>
<td>Shende, Sameer</td>
<td>Shende, Sameer</td>
</tr>
<tr>
<td>2.3.1.01</td>
<td>PMR SDK</td>
<td>Balajji, Pavan</td>
<td>Guo, Yanfei</td>
</tr>
<tr>
<td>2.3.1.07</td>
<td>Exascale MPI (MPICH)</td>
<td>McCormick, Pat</td>
<td>McCormick, Pat</td>
</tr>
<tr>
<td>2.3.1.08</td>
<td>Legion</td>
<td>Bosilica, George</td>
<td>Carr, Earl</td>
</tr>
<tr>
<td>2.3.1.09</td>
<td>PaRSEC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.1.14</td>
<td>Pagoda: UPC++/GASNet for Lightweight Communication and Global Address Space Support</td>
<td>Hargrove, Paul</td>
<td>Hargrove, Paul</td>
</tr>
<tr>
<td>2.3.1.16</td>
<td>SICM</td>
<td>Lang, Michael</td>
<td>Vigil, Brittney</td>
</tr>
<tr>
<td>2.3.1.17</td>
<td>OMPI-X</td>
<td>Bernholdt, David</td>
<td>Grundhoffer, Alicia</td>
</tr>
<tr>
<td>2.3.1.18</td>
<td>RAJA/Kokkos</td>
<td>Trott, Christian Robert</td>
<td>Trujillo, Gabrielle</td>
</tr>
<tr>
<td>2.3.1.19</td>
<td>Argo: Low-level resource management for the OS and runtime</td>
<td>Beckman, Pete</td>
<td>Gupta, Rinku</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Development Tools</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2.01</td>
<td>Development Tools Software Development Kit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2.06</td>
<td>Exa-PAPI++: The Exascale Performance Application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2.08</td>
<td>Extending HPF/C++ to Measure and Analyze Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2.10</td>
<td>PROTEAS-TUNE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2.11</td>
<td>SOLLVE: Scaling OpenMP with LLVM for Exascale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.2.12</td>
<td>FLANG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3</td>
<td>Mathematical Libraries</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.01</td>
<td>Extreme-scale Scientific xSDK for ECP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.06</td>
<td>Preparing PETSc/TAO for Exascale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.07</td>
<td>STRUMPACK/SuperLU/FFTX: sparse direct solvers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.12</td>
<td>Enabling Time Integrators for Exascale Through SUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.13</td>
<td>CLOVER: Computational Libraries Optimized Via Exascale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.14</td>
<td>ALEXa: Accelerated Libraries for Exascale/ForTrilinos</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.3.15</td>
<td>Sake: Scalable Algorithms and Kernels for Exascale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4</td>
<td>Data and Visualization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.01</td>
<td>Data and Visualization Software Development Kit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.09</td>
<td>ADIOS Framework for Scientific Data on Exascale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.10</td>
<td>DataLib: Data Libraries and Services Enabling Exascale</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.13</td>
<td>ECP/VTK-m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.15</td>
<td>Veloci: Very Low Overhead Transparent Multilevel Co</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.16</td>
<td>ALPINE: Algorithms and Infrastructure for In Situ Visualization and Analysis/ZFP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.5</td>
<td>Software Ecosystem and Delivery</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.5.01</td>
<td>Software Ecosystem and Delivery Software Development Kit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.5.09</td>
<td>SW Packaging Technologies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.5.10</td>
<td>ExaWorks</td>
<td>Laney, Dan</td>
<td>Laney, Dan</td>
</tr>
<tr>
<td>2.3.6</td>
<td>NNSA ST</td>
<td>Mohror, Kathryn</td>
<td></td>
</tr>
<tr>
<td>2.3.6.01</td>
<td>LANL ATDM</td>
<td>Mike Lang</td>
<td>Vandenbusch, Tanya Marie</td>
</tr>
<tr>
<td>2.3.6.02</td>
<td>LLNL ATDM</td>
<td>Beckey Springmeyer</td>
<td>Gamblin, Todd</td>
</tr>
<tr>
<td>2.3.6.03</td>
<td>SNL ATDM</td>
<td>Jim Stewart</td>
<td>Trujillo, Gabrielle</td>
</tr>
</tbody>
</table>

ECP Staff

- ~250 staff
- ~70 products
- ~30 teams
- ~9 universities
- ~6 DOE labs
- ~3 technical areas
- focus area of 3 in ECP
We work on products applications need now and into the future

Key themes:
- Focus: GPU node architectures and advanced memory & storage technologies
- Create: New high-concurrency, latency tolerant algorithms
- Develop: New portable (Nvidia, Intel, AMD GPUs) software product
- Enable: Access and use via standard APIs

Software categories:
- **Next generation established products:** Widely used HPC products (e.g., MPICH, OpenMPI, PETSc)
- **Robust emerging products:** Address key new requirements (e.g., Kokkos, RAJA, Spack)
- **New products:** Enable exploration of emerging HPC requirements (e.g., SICM, zfp, UnifyCR)

<table>
<thead>
<tr>
<th>Example Products</th>
<th>Engagement</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPI – Backbone of HPC apps</td>
<td>Explore/develop MPICH and OpenMPI new features &amp; standards</td>
</tr>
<tr>
<td>OpenMP/OpenACC – On-node parallelism</td>
<td>Explore/develop new features and standards</td>
</tr>
<tr>
<td>Performance Portability Libraries</td>
<td>Lightweight APIs for compile-time polymorphisms</td>
</tr>
<tr>
<td>LLVM/Vendor compilers</td>
<td>Injecting HPC features, testing/feedback to vendors</td>
</tr>
<tr>
<td>Perf Tools - PAPI, TAU, HPCToolkit</td>
<td>Explore/develop new features</td>
</tr>
<tr>
<td>Math Libraries: BLAS, sparse solvers, etc.</td>
<td>Scalable algorithms and software, critical enabling technologies</td>
</tr>
<tr>
<td>IO: HDF5, MPI-IO, ADIOS</td>
<td>Standard and next-gen IO, leveraging non-volatile storage</td>
</tr>
<tr>
<td>Viz/Data Analysis</td>
<td>ParaView-related product development, node concurrency</td>
</tr>
</tbody>
</table>
**Scope and objectives**

- SLATE is a distributed, GPU-accelerated, dense linear algebra library, intended to replace ScaLAPACK
- SLATE covers parallel BLAS, linear system solvers, least squares, eigensolvers, and the SVD

**Port to AMD and Intel**

- SLATE and BLAS++ now support all three major GPU platforms

**Impact**

- Initially supported NVIDIA’s cuBLAS for use on current machines like Summit
- Can now use AMD’s rocBLAS in preparation for Frontier, and Intel’s oneMKL in preparation for Aurora
- Other projects can also leverage BLAS++ for portability

**Accomplishment**

- Refactored SLATE to use BLAS++ as portability layer
- Ported BLAS++ to AMD rocBLAS and Intel oneMKL

**Deliverables**

- Report: [https://www.icl.utk.edu/publications/swan-016](https://www.icl.utk.edu/publications/swan-016)
- Code in git repos: [bitbucket.org/icl/slate/](https://bitbucket.org/icl/slate/) and [bitbucket.org/icl/blaspp/](https://bitbucket.org/icl/blaspp/)

**Key ECP Software Stack Legacy:**

- Portable execution on:
  - CPUs
  - 3 different GPUs
- A bridge from CPUs to GPUs
E4S Planning, Executing, Delivering
ECP ST Planning Process: Hierarchical, three-phase, cyclical

### Baseline
- FY20–23 Baseline Plan
- High level Definitions
  - Q2 FY19 start
  - FY20 Base plan
  - FY21–23 planning packages

### Annual Refinement
- FY Refine Baseline Plan
- As Needed
- Basic activity definitions
  - 6 months prior to FY
  - 4–6 P6 Activities/year
  - Each activity:
    - % annual budget
    - Baseline start/end
    - High level description

### Per Activity
- Detailed Plan
- Complete activity definitions
  - 8 weeks prior to start
  - High-fidelity description
  - Execution strategy
  - Completion criteria
  - Personnel details

---

### Two-level Review Process

<table>
<thead>
<tr>
<th>Changes to Cost, Scope, and Schedule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor</td>
</tr>
<tr>
<td>Lightweight Review in Jira, L3 and L2 leads</td>
</tr>
<tr>
<td>Major</td>
</tr>
<tr>
<td>Change Control Board Review, ECP leadership</td>
</tr>
</tbody>
</table>

Variance Recorded in Jira

Proceed with Execution
KPP-3: Focus on capability integration

- **Capability**: Any significant product functionality, including existing features adapted to the pre-exascale and exascale environments, that can be integrated into a client environment.

- **Capability Integration**: Complete, sustainable integration of a significant product capability into a client environment in a pre-exascale environment (tentative score) and in an exascale environment (confirmed score).
ECP ST Lifecycle summary

Create Annual Planning Package
• Each product has its own planning packages
• Defined for all FYs

Refine upcoming FY plan
• Complete 6 months prior to FY
• 4 or more P6 activities per product

Refine upcoming P6 activity
• Complete 8 weeks prior to activity start
• Include all details

Develop capabilities and track progress via tailored EVM

Integrate into product
• Full testing, documentation, etc.
• Direct access for some users

Integrate into SDK
• Satisfy SDK community policies
• Direct access for some users

Integrate into E4S
• Satisfy E4S community policies
• Full ecosystem with high value

Deliver to users
• From source (spack)
• Containers, cloud

Managed by P6 Activity Process

Measured by KPP-3 Process
The Extreme-Scale Scientific Software Stack (E4S) and Software Development Kits (SDKs)
Extreme-scale Scientific Software Stack (E4S)

- **E4S**: HPC Software Ecosystem – a curated software portfolio
- A **Spack-based** distribution of software tested for interoperability and portability to multiple architectures
- Available from **source, containers, cloud, binary caches**
- Leverages and enhances SDK interoperability thrust
- Not a commercial product – an open resource for all
- Oct 2018: E4S 0.1 - 24 full, 24 partial release products
- Jan 2019: E4S 0.2 - 37 full, 10 partial release products
- Nov 2019: E4S 1.0 - 50 full, 5 partial release products
- Feb 2020: E4S 1.1 - 61 full release products
- Nov 2020: E4S 1.2 (aka, 20.10) - 67 full release products
- Feb 2021: E4S 21.02 - 67 full release, 4 partial release
- May 2021: E4S 21.05 - 76 full release products
- August 2021: E4S 21.08 - 88 full release products

Also include other products e.g.,
AI: PyTorch, TensorFlow, Horovod
Co-Design: AMReX, Cabana, MFEM
xSDK: Primary delivery mechanism for ECP math libraries’ continual advancements toward predictive science

As motivated and validated by the needs of ECP applications:

- Performance on new node architectures
- Extreme strong scalability
- Interoperability, complementarity: xSDK
- Optimization, UQ, solvers, discretizations
- Advanced, coupled multiphysics, multiscale

Next-generation algorithms

Toward predictive scientific simulations

Advances in data structures for new node architectures

Improving library quality, sustainability, interoperability

Increasing performance, portability, productivity

Timeline:
- xSDK release 1
- xSDK release 2
- ... ...
- xSDK release n

Ref: xSDK: Building an Ecosystem of Highly Efficient Math Libraries for Exascale, SIAM News, Jan 2021
Delivering an open, hierarchical software ecosystem
More than a collection of individual products

<table>
<thead>
<tr>
<th>Levels of Integration</th>
<th>Product</th>
<th>Source and Delivery</th>
</tr>
</thead>
</table>
|                       | E4S     | Source: ECP E4S team; Non-ECP Products (all dependencies)  
|                       |         | Delivery: spack install e4s; containers; CI Testing |
| Group similar products| SDKs    | Source: SDK teams; Non-ECP teams (policy compliant, spackified)  
|                       |         | Delivery: Apps directly; spack install sdk; future: vendor/facility |
| Standard workflow     | ST Products | Source: ECP L4 teams; Non-ECP Developers; Standards Groups  
|                       |         | Delivery: Apps directly; spack; vendor stack; facility stack |
E4S Community Policies
E4S Community Policies V1.0 Released

What is E4S?

The Extreme-scale Scientific Software Stack (E4S) is a community effort to provide open source software packages for developing, deploying and running scientific applications on high-performance computing (HPC) platforms. E4S provides from-source builds and containers of a broad collection of HPC software packages.

Purpose

E4S exists to accelerate the development, deployment and use of HPC software, lowering the barriers for HPC users. E4S provides containers and turn-key, from-source builds of more than 80 popular HPC products in programming models, such as MPI; development tools such as HPCToolkit, TAU and PAPI; math libraries such as PETSc and Trilinos; and Data and Viz tools such as HDF5 and Paraview.

Approach

By using Spack as the meta-build tool and providing containers of pre-built binaries for Docker, Singularity, Shifter and CharlieCloud, E4S enables the flexible use and testing of a large collection of reusable HPC software packages.
E4S Community Policies Version 1
A Commitment to Quality Improvement

• Will serve as membership criteria for E4S
  – Membership is not required for inclusion in E4S
  – Also includes forward-looking draft policies

• Purpose: enhance sustainability and interoperability

• Topics cover building, testing, documentation, accessibility, error handling and more

• Multi-year effort led by SDK team
  – Included representation from across ST
  – Multiple rounds of feedback incorporated from ST leadership and membership

• Modeled after xSDK Community Policies

• https://e4s-project.github.io/policies.html

---

P1: Spack-based Build and Installation Each E4S member package supports a scriptable Spack build and production-quality installation in a way that is compatible with other E4S member packages in the same environment. When E4S build, test, or installation issues arise, there is an expectation that teams will collaboratively resolve those issues.

P2: Minimal Validation Testing Each E4S member package has at least one test that is executable through the E4S validation test suite (https://github.com/E4S-Project/testsuite). This will be a post-installation test that validates the usability of the package. The E4S validation test suite provides basic confidence that a user can compile, install and run every E4S member package. The E4S team can actively participate in the addition of new packages to the suite upon request.

P3: Sustainability All E4S compatibility changes will be sustainable in that the changes go into the regular development and release versions of the package and should not be in a private release/branch that is provided only for E4S releases.

P4: Documentation Each E4S member package should have sufficient documentation to support installation and use.

P5: Product Metadata Each E4S member package team will provide key product information via metadata that is organized in the E4S DocPortal format. Depending on the filenames where the metadata is located, this may require minimal setup.

P6: Public Repository Each E4S member package will have a public repository, for example at GitHub or Bitbucket, where the development version of the package is available and pull requests can be submitted.

P7: Imported Software If an E4S member package imports software that is externally developed and maintained, then it must allow installing, building, and linking against a functionally equivalent outside copy of that software. Acceptable ways to accomplish this include (1) forsaking the internal copied version and using an externally-provided implementation or (2) changing the file names and namespaces of all global symbols to allow the internal copy and the external copy to coexist in the same downstream libraries and programs. This pertains primarily to third party support libraries and does not apply to key components of the package that may be independent packages but are also integral components to the package itself.

P8: Error Handling Each E4S member package will adopt and document a consistent system for signifying error conditions as appropriate for the language and application. For e.g., returning an error condition or throwing an exception. In the case of a command line tool, it should return a sensible exit status on success/failure, so the package can be safely run from within a script.

P9: Test Suite Each E4S member package will provide a test suite that does not require special system privileges or the purchase of commercial software. This test suite should grow in its comprehensiveness over time. That is, new and modified features should be included in the suite.
E4S DocPortal

- Single point of access
- All E4S products
- Summary Info
  - Name
  - Functional Area
  - Description
  - License
- Searchable
- Sortable
- Rendered daily from repos

https://e4s-project.github.io/DocPortal.html

All we need from the software team is a repo URL + up-to-date meta-data files
Goal: All E4S product documentation accessible from single portal on E4S.io

(working mock webpage below)

https://e4s-project.github.io/DocPortal.html
E4S: Better quality, documentation, testing, integration, delivery, building & use

Delivering HPC software to facilities, vendors, agencies, industry, international partners in a brand-new way

Community Policies
Commitment to software quality

DocPortal
Single portal to all E4S product info

Portfolio testing
Especially leadership platforms

Curated collection
The end of dependency hell

Quarterly releases
Release 1.2 – November

Build caches
10X build time improvement

Turnkey stack
A new user experience

https://e4s.io

Community & LSSw
US agencies, industry, international

https://e4s.io
Growing and Sustaining the Software Community
IDEAS-ECP team works with the ECP community to improve developer productivity and software sustainability as key aspects of increasing overall scientific productivity.

1. **Customize and curate methodologies**
   - Target scientific software productivity and sustainability
   - Use workflow for best practices content development

2. **Incrementally and iteratively improve software practices**
   - Determine high-priority topics for improvement and track progress
   - *Productivity and Sustainability Improvement Planning (PSIP)*

3. **Establish software communities**
   - Determine community policies to improve software quality and compatibility
   - Create Software Development Kits (SDKs) to facilitate the combined use of complementary libraries and tools

4. **Engage in community outreach**
   - Broad community partnerships
   - Collaboration with computing facilities
   - Webinars, tutorials, events
   - WhatIs and HowTo docs
   - Better Scientific Software site ([https://bssw.io](https://bssw.io))
Advancing Scientific Productivity through Better Scientific Software: Developer Productivity & Software Sustainability Report

Disruptive changes in computer architectures and the complexities of tackling new frontiers in extreme-scale modeling, simulation, and analysis present daunting challenges to software productivity and sustainability.

This report explains the IDEAS approach, outcomes, and impact of work (in partnership with the ECP and broader computational science community).

Target readers are all those who care about the quality and integrity of scientific discoveries based on simulation and analysis. While the difficulties of extreme-scale computing intensify software challenges, issues are relevant across all computing scales, given universal increases in complexity and the need to ensure the trustworthiness of computational results.

https://exascaleproject.org/better-scientific-productivity-through-better-scientific-software-the-ideas-report
Preparing for Sustainable Software Efforts after ECP: Leadership Scientific Software (LSSw.io)
Background

• US Department of Energy (DOE) Exascale Computing Project (ECP)
  – Developing enabling technologies for upcoming exascale computers
    • ECP Software Technology (ST) focus area:
      – Uses a macro-engineering software lifecycle to
        • Plan, execute, track, and assess product development toward the
          • Delivery of a curated portfolio of reusable, open-source software products
t            • The Extreme-scale Scientific Software Stack or E4S (https://e4s.io)

• During the final years of ECP, one key objective is to:
  – Transition our efforts to a sustainable organization and model for
    • Continued development and delivery of future capabilities, including
      – Incorporation of new scientific software domains, and
        • Expansion of the contributor and user communities

• LSSw is key component toward sustainability
LSSw Mission

• LSSw is dedicated to
  – Building community and understanding around the
    • Development and sustainable delivery of
      – Leadership scientific software
  
• Development
  – Portfolio-driven approach
  – Co-design with hardware, system software, applications

• Sustainable
  – Organizational stability
  – Emphasis on quality
  – Broad accessibility
Leadership Scientific Software (defn)

• Libraries, tools and environments that
  – Contribute to scientific discovery and insight in
    • New and emerging computing environments

• Are end-user applications within scope?
  – Yes, as stakeholders in the effort
  – Goal is to provide
    • High-priority functionality not available elsewhere
    • Portable performance on leading edge and emerging platforms
    • A sustainable turnkey software ecosystem
Leadership Scientific Software (defn)

- Push the boundary of feasibility
  - Enabling
    - Larger scale, higher fidelity and greater integration of
      - Advanced computing ecosystems

- Does “leadership” limit the scope of discussion?
  - Yes, we are directly focused on non-commodity environments, but:
    - Still use laptops, desktops, CPU clusters as part of our development efforts
    - Many of our tools and libraries need to be available everywhere
    - Non-commodity focus does not mean we work only on non-commodity systems

- Focus is on efforts that include co-design of
  - **Computing platforms**: Modeling & simulation, AI/ML, edge: at scale
  - **System software**: Collaborative co-design with vendors
  - **Science-specific tools and libraries**: What we are developing for users
ECP Efforts

• ECP is a notable project:
  – Stable, sustained funding of a national project with clear goals
  – Infrastructure to innovate and establish new collaborative work

• ECP enables tremendous opportunities to:
  – Create a new generation of scientific software
  – Provide a curated portfolio of reusable software products for apps
  – Qualitatively change how we plan, develop and deliver leadership SW
Join the conversation

- **https://lssw.io**: Main portal for the LSSw community
- **LSSw Town Hall Meetings**:
  - 3rd Thursday each month, 3 – 4:30 pm Eastern US time
- **Slack**: Share your ideas interactively
- **White Papers**: Written content for LSSw conversations
  - We need your ideas
  - 2 – 4 page white paper
  - Submit via GitHub PR or attachment to contribute@lssw.io
- **References**:
  - Help us build a reading list
  - Submit via GitHub PR or email to contribute@lssw.io
Summary & Next Steps

• Scientific software capabilities and complexity are increasing
• Computing systems are becoming more diverse
• A portfolio approach to planning and delivering is attractive
• ECP provides a working example to address complexity:
  – ECP ST lifecycle enables coordinated planning, executing, tracking and assessing
  – E4S and SDKs provide a scalable software architecture and portfolio for “turnkey” software stack
  – The IDEAS project and BSSw provide community building for scientific software developers
  – Goal: Better, faster and cheaper

• We believe the next steps require broad community engagement:
  – What are other fundamental requirements for improving leadership scientific software?
  – How can we collaborate as a broad community in development and use?
  – Are there other working software ecosystems we should learn from?
  – What topics are missing from the conversation?

• We need your engagement in this effort!
Thank you to all collaborators in the ECP and broader computational science communities. The work discussed in this presentation represents creative contributions of many people who are passionately working toward next-generation computational science.