Preparing for the Sustainable Delivery of the DOE Exascale Software Stack

Michael A. Heroux, Sandia National Laboratories, Director of Software Technology
Rajeev Thakur, Argonne National Laboratory, Programming Models and Runtimes Area Lead
Jeffrey Vetter, Oak Ridge National Laboratory, Development Tools Area Lead

Advanced Scientific Computing Advisory Committee
September 25, 2020
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 high-performance numerical software, including development of PETSc and leadership of multi-institutional work toward sustainable scientific software ecosystems.

**Rajeev Thakur, Programming Models and Runtimes (2.3.1)**
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 (2.3.2)**
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.

**Xaiyou (Sherry) Li, Math Libraries (2.3.3)**
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 (2.3.4)**
Jim is a senior research scientist at the Los Alamos National Laboratory and an expert in data science at scale. He started and actively contributes to many open-source data science packages including ParaView.

**Kathryn Mohror, NNSA ST (2.3.6)**
Kathryn is Group Leader for the CAS analysis and tuning, fault tolerance

**Todd Munson, Software Ecosystem and Delivery (2.3.5)**
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.

Many thanks to Rob Neely! Rob was part of ECP leadership from the very beginning.
<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></td>
<td></td>
</tr>
<tr>
<td>2.3.1</td>
<td>Programming Models &amp; Runtimes</td>
<td>Thakur, Rajeev</td>
<td></td>
</tr>
<tr>
<td>2.3.1.01</td>
<td>PMR SDK</td>
<td>Shende, Sameer</td>
<td>Shende, Sameer</td>
</tr>
<tr>
<td>2.3.1.07</td>
<td>Exascale MPI (MPICH)</td>
<td>Balaji, Pavan</td>
<td>Guo, Yanfei</td>
</tr>
<tr>
<td>2.3.1.08</td>
<td>Legion</td>
<td>McCormick, Pat</td>
<td>McCormick, Pat</td>
</tr>
<tr>
<td>2.3.1.09</td>
<td>PaRSEC</td>
<td>Bosilica, George</td>
<td>Carr, Earl</td>
</tr>
<tr>
<td>2.3.1.14</td>
<td>Programs: UPC++/GASNet for Lightweight Communication and Global Address</td>
<td>Hargrove, Paul</td>
<td>Hargrove, Paul</td>
</tr>
<tr>
<td></td>
<td>Space Support</td>
<td>Lang, Michael</td>
<td>Vigil, Britney</td>
</tr>
<tr>
<td>2.3.1.17</td>
<td>OMPI-X</td>
<td>Bernholdt, David</td>
<td>Grundhofer, 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>Vetter, Jeff</td>
<td></td>
</tr>
<tr>
<td>2.3.2.01</td>
<td>Development Tools Software Development Kit</td>
<td>Miller, Barton</td>
<td>Tim Haines</td>
</tr>
<tr>
<td>2.3.2.06</td>
<td>Exa-PAPI++: The Exascale Performance Application Programming Interface</td>
<td>Dongarra, Jack</td>
<td>Jagode, Heike</td>
</tr>
<tr>
<td></td>
<td>with Modern C++</td>
<td>Mellor-Crummey, John</td>
<td>Meng, Xiaozhu</td>
</tr>
<tr>
<td>2.3.2.08</td>
<td>Extending HPCToolkit to Measure and Analyze Code Performance on</td>
<td>Vetter, Jeff</td>
<td>Glassbrook, Dick</td>
</tr>
<tr>
<td></td>
<td>Exascale Platforms</td>
<td>Chapman, Barbara</td>
<td></td>
</tr>
<tr>
<td>2.3.2.10</td>
<td>PROTEAS-TUNE</td>
<td>Turn, John</td>
<td></td>
</tr>
<tr>
<td>2.3.2.11</td>
<td>SOLLVE: Scaling OpenMP with LLVm for Exascale</td>
<td>Munson, Todd</td>
<td>Munson, Todd</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>Li, Sherry</td>
<td></td>
</tr>
<tr>
<td>2.3.3.01</td>
<td>Extreme-scale Scientific xSDK for ECP</td>
<td>Yang, Ulrike</td>
<td>Yang, Ulrike</td>
</tr>
<tr>
<td>2.3.3.06</td>
<td>Preparing PETSc/TAO for Exascale</td>
<td>Munson, Todd</td>
<td>Munson, Todd</td>
</tr>
<tr>
<td>2.3.3.07</td>
<td>STRUMPACK/SuperLU/FFTX: sparse direct solvers, preconditioners, and</td>
<td>Li, Sherry</td>
<td>Li, Sherry</td>
</tr>
<tr>
<td></td>
<td>FFT libraries</td>
<td>Woodward, Carol</td>
<td>Woodward, Carol</td>
</tr>
<tr>
<td>2.3.3.12</td>
<td>Enabling Time Integrators for Exascale Through SUNDIALS/ Hypre</td>
<td>Dongarra, Jack</td>
<td>Carr, Earl</td>
</tr>
<tr>
<td>2.3.3.13</td>
<td>CLOVER: Computational Libraries Optimized Via Exascale Research</td>
<td>Dongarra, Jack</td>
<td></td>
</tr>
<tr>
<td>2.3.3.14</td>
<td>ALEXa: Accelerated Libraries for Exascale/ForTrilinos</td>
<td>Turn, John</td>
<td></td>
</tr>
<tr>
<td>2.3.4</td>
<td>Data and Visualization</td>
<td>Ahrens, James</td>
<td></td>
</tr>
<tr>
<td>2.3.4.01</td>
<td>Data and Visualization Software Development Kit</td>
<td>Atkins, Chuck</td>
<td>Bagha, Neelam</td>
</tr>
<tr>
<td>2.3.4.09</td>
<td>ADIOS Framework for Scientific Data on Exascale Systems</td>
<td>Klassy, Scott</td>
<td>Grundhofer, Alicia</td>
</tr>
<tr>
<td>2.3.4.10</td>
<td>DataLib: Data Libraries and Services Enabling Exascale Science</td>
<td>Ross, Rob</td>
<td>Ross, Bob</td>
</tr>
<tr>
<td>2.3.4.13</td>
<td>ECP/VTK-m</td>
<td>Moreland, Kenneth</td>
<td>Moreland, Kenneth</td>
</tr>
<tr>
<td>2.3.4.14</td>
<td>Veloc: Very Low Overhead Transparent Multilevel Checkpoint/Restart/Sz</td>
<td>Cappello, Franck</td>
<td>Ehling, Scott</td>
</tr>
<tr>
<td>2.3.4.15</td>
<td>ExaIO - Delivering Efficient Parallel I/O on Exascale Computing Systems</td>
<td>Byna, Suren</td>
<td>Bagha, Neelam</td>
</tr>
<tr>
<td></td>
<td>with HDF5 and Unify</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.4.16</td>
<td>ALPINE: Algorithms and Infrastructure for In Situ Visualization and</td>
<td>Ahrens, James</td>
<td>Turton, Terry</td>
</tr>
<tr>
<td></td>
<td>Analysis/ZFP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.3.5</td>
<td>Software Ecosystem and Delivery</td>
<td>Munson, Todd</td>
<td></td>
</tr>
<tr>
<td>2.3.5.01</td>
<td>Software Ecosystem and Delivery Software Development Kit</td>
<td>Willenbring, James M</td>
<td>Willenbring, James M</td>
</tr>
<tr>
<td>2.3.5.09</td>
<td>SW Packaging Technologies</td>
<td>Gamblin, Todd</td>
<td>Gamblin, Todd</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>Becky 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>
Software Technology Tracking: KPP-3
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).
KPP-3 Dashboard (2020-09-24)

- Manage KPP-3 progress in Jira
- Special issue type
- Present values updated at least twice a year
- Updates include uploaded artifacts supporting scores
- Measuring tentative present values now
- Confirmed values possible when Aurora, Frontier arrive
Key Takeaways for ECP ST Progress Tracking

• KPP-3 measures the key value proposition of ECP ST activities:
  – The sustainable integration of capabilities demonstrated on the exascale platforms

• Final KPP-3 definition benefitted tremendously from review teams input

• Use of Jira enable real-time tracking of integration progress:
  – At any time, we can see status
  – We collect artifacts as we go
  – Currently 21 of 63 (33%) have achieved tentative passing value
  – Require 32 of 63 (>50%) achieving confirmed passing value to achieve threshold for KPP-3

• The KPP-3 approach of sustainable capability integration is applicable to other settings:
  – Other reusable software projects
  – Future DOE projects
The Extreme-Scale Scientific Software Stack (E4S):

A collaborative HPC Linux Ecosystem
Delivering an open, hierarchical software ecosystem

<table>
<thead>
<tr>
<th>Levels of Integration</th>
<th>Product</th>
<th>Source and Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECP ST Individual Products</td>
<td>ST Products</td>
<td>Source: ECP L4 teams; Non-ECP Developers; Standards Groups &lt;br&gt; Delivery: Apps directly; spack; vendor stack; facility stack</td>
</tr>
</tbody>
</table>

- Standard workflow
- Existed before ECP
Delivering an open, hierarchical software ecosystem

Levels of Integration | Product | Source and Delivery
---|---|---
ECP ST Individual Products | ST Products | Source: ECP L4 teams; Non-ECP Developers; Standards Groups
Delivery: Apps directly; spack; vendor stack; facility stack

- Standard workflow
- Existed before ECP

SDKs | Source: ECP SDK teams; Non-ECP Products (policy compliant, spackified)
Delivery: Apps directly; spack install sdk; future: vendor/facility

- Group similar products
- Make interoperable
- Assure policy compliant
- Include external products

E4S | Source: ECP E4S team; Non-ECP Products (all dependencies)
Delivery: spack install e4s; containers; CI Testing

- Build all SDKs
- Build complete stack
- Containerize binaries

ECP ST Open Product Integration Architecture
E4S Components

- E4S is a curated release of ECP ST products based on Spack [http://spack.io].
- E4S Spack cache to support bare-metal installs at facilities and custom container builds:
  - x86_64, ppc64le, and aarch64
- Container images on DockerHub and E4S website of pre-built binaries of ECP ST products.
- Base images and full featured containers (GPU support).
- GitHub recipes for creating custom images from base images.
- e4s-cl for container launch and for replacing MPI in application with system MPI libraries.
- Validation test suite on GitHub provides automated build and run tests.
- Automates build process via GitLab Continuous Integration to ensure packages can be built.
- E4S Doc Portal aggregates and summarizes documentation and metadata by raking product repos.
- E4S VirtualBox image with support for Docker, Shifter, Singularity, and Charliecloud runtimes.
- AWS image to deploy E4S on EC2.

https://e4s.io
E4S Community Policies

Community-driven Quality Commitments
xSDK compatible package: Must satisfy mandatory xSDK policies:

M1. Support xSDK community GNU Autoconf or CMake options.
M2. Provide a comprehensive test suite.
M3. Employ user-provided MPI communicator.
M4. Give best effort at portability to key architectures.
M5. Provide a documented, reliable way to contact the development team.
M6. Respect system resources and settings made by other previously called packages.
M7. Come with an open source license.
M8. Provide a runtime API to return the current version number of the software.
M9. Use a limited and well-defined symbol, macro, library, and include file name space.
M10. Provide an accessible repository (not necessarily publicly available).
M11. Have no hardwired print or IO statements that cannot be turned off.
M12. For external dependencies, allow installing, building, and linking against an outside copy of external software.
M13. Install headers and libraries under <prefix>/include/ and <prefix>/lib/.
M14. Be buildable using 64 bit pointers. 32 bit is optional.
M15. All xSDK compatibility changes should be sustainable.
M16. The package must support production-quality installation compatible with the xSDK install tool and xSDK metapackage.

Also recommended policies, which currently are encouraged but not required:

R1. Have a public repository.
R2. Possible to run test suite under valgrind in order to test for memory corruption issues.
R3. Adopt and document consistent system for error conditions/exceptions.
R4. Free all system resources it has acquired as soon as they are no longer needed.
R5. Provide a mechanism to export ordered list of library dependencies.
R6. Document versions of packages that it works with or depends on, preferably in machine-readable form
R7. Have README, SUPPORT, LICENSE, and CHANGELOG files in top directory.

xSDK member package: Must be an xSDK-compatible package, and it uses or can be used by another package in the xSDK, and the connecting interface is regularly tested for regressions.

We welcome feedback. What policies make sense for your software?

https://bssw.io/blog_posts/building-community-through-software-policies
E4S Community Candidate Policies V 1.0 Beta

- **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.

- **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.

- **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.

- **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.

- **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.

- **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.

- **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.

- **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/SDK Policy Initiative Status

- Community policies are important for several reasons:
  - Commitment to quality
  - Membership criteria for the future
  - Community discussion

- Each SDK community developing policies like Math Libs (xSDK).

- Policies common to all SDKs will be promoted to E4S level

- Policies will determine:
  - Quality label
  - Membership in E4S and the SDKs

- Version 1.0 of policies due by end of 2020
E4S DocPortal

A Single Portal with Redirect to Product Documentation
Product Documentation

Challenges: User Perspective

- Finding info for specific product
  - What it does
  - License
  - Support
  - Contact info
  - More ...

- Finding new products
  - What can solve my problem

- Trusting accuracy of information
  - Up to date
  - Complete

- Hierarchical
  - Summary to deep dive
Product Documentation Challenges: Developer Perspective

- Efficient and Effective generation and maintenance
- Getting noticed by new users
- Conveying summary information \textit{and} details
E4S Documentation Portal Strategy

All content resides in product repositories

- Use open source community approach of specially-name files in software repositories.
- Adopt commonly used file names when available.
- ID new information items not already being requested.

Documentation portal provides single point of access

- Web-based raking tool capture information from product repositories and present in summary form.
- Aggregates and summarizes documentation and metadata for E4S products
- Regularly updates information directly from product repositories
- Location: https://e4s-project.github.io/DocPortal.html
E4S DocPortal

- The DocPortal is live!
- Summary Info
  - Name
  - Functional Area
  - Description
  - License
- Searchable
- Sortable

https://e4s-project.github.io/DocPortal.html
Goal: All E4S Product Documentation Accessible from single portal on E4S.io (Working Mock Webpage below)

https://e4s-project.github.io/DocPortal.html
Q: What do we need for adding a product to the DocPortal?
A: A repo URL + up-to-date meta-data files

Takeaway: Adding new products is very straightforward!
E4S Spack Build Cache and Container Build Pipeline
E4S: Spack Build Cache at U. Oregon

E4S Build Cache for Spack 0.15.0

To use this build cache, just add it to your Spack

```bash
spack buildcache keys --trust --install
spack mirror add E4S https://cache.e4s.io/e4s
```

Click on one of the packages below to see a list of all available variants.

- All Architectures
- PPC64LE
- X86_64
- All Operating Systems
- Centos 7
- Centos 8
- RHEL 7
- RHEL 8
- Ubuntu 18.04

Last updated: 06-25-2020 19:30 PDT

11370 Spack packages

Search

- adiak@0.1.1
- adios2@2.5.0
- adios2@2.6.0
- adios@1.13.1
- adlbx@0.9.2
- aml@0.1.0
- ant@1.10.0
- ant@1.10.7
- argbots@1.0
- argbots@1.0rc1
- argbots@1.0rc2
- arpack-ng@3.7.0
- autoconf@2.69
- automake@1.16.1
- automake@1.16.2
- axl@0.1.1
- axl@0.3.0
- axiom@0.3.3
- bdfstopcf@1.0.5
- binutils@2.31.1
- binutils@2.32
- binutils@2.33.1
- binutils@2.34

- 10,000+ binaries
- S3 mirror
- No need to build from source code!

https://oaciss.uoregon.edu/e4s/inventory.html
WDMApp: Speeding up bare-metal installs using E4S build cache

The E4S project has created a build cache for Rhea. This provides many packages as precompiled binaries, so will reduce the installation time. To use it:

```bash
$ wget https://oaciss.uoregon.edu/e4s/e4s.pub
$ spack gpg trust e4s.pub
$ spack mirror add E4S https://cache.e4s.io/e4s
```

### Building WDMapp

You should be able to just follow the generic instructions from Building WDMAPP.

### Using E4S WDMapp docker container

Alternatively, the E4S project has created a docker image that mirrors the Rhea environment, which can be used for local development and debugging. To run this image, you need to have docker installed and then do the following:

- E4S Spack build cache
- Adding E4S mirror
- WDMApp install speeds up!
Pantheon and E4S support end-to-end ECP examples

Overview: The Exascale Computing Project (ECP) is a complex undertaking, involving a myriad of technologies working together. An outstanding need is a way to capture, curate, communicate and validate workflows that cross all of these boundaries.

The Pantheon and E4S projects are collaborating to advance the integration and testing of capabilities, and to promote understanding of the complex workflows required by the ECP project. Utilizing a host of ECP technologies (spack, Ascent, Cinema, among others), this collaboration brings curated workflows to the fingertips of ECP researchers.

Contributions
- Curated end-to-end application/in-situ analysis examples can be run quickly by anyone on Summit. ([https://github.com/pantheonscience/ECP-E4S-Examples](https://github.com/pantheonscience/ECP-E4S-Examples))
- Pantheon/E4S integration speeds up build/setup times over source builds due to cached binaries (approx. 10x speed up).

Special Thanks to David Rogers, Jim Ahrens, Sameer Shende

Instructions page for (top) Nyx, Ascent and Cinema workflow repository, and (bottom) Cloverleaf3d, Ascent, Cinema workflow. These curated workflows use Pantheon, E4S and spack to provide curated workflows for ECP.
## E4S Summary

### What E4S is not
- A closed system taking contributions only from DOE software development teams.
- A monolithic, take-it-or-leave-it software behemoth.
- A commercial product.
- A simple packaging of existing software.

### What E4S is
- Extensible, open architecture software ecosystem accepting contributions from US and international teams.
- Framework for collaborative open-source product integration.
- A full collection if compatible software capabilities and a manifest of a la carte selectable software capabilities.
- Vehicle for delivering high-quality reusable software products in collaboration with others.
- The conduit for future leading edge HPC software targeting scalable next-generation computing platforms.
- A hierarchical software framework to enhance (via SDKs) software interoperability and quality expectations.
The Second Extreme-scale Scientific Software Stack Forum (E4S Forum)  
September 24th, 2020, Workshop at EuroMPI/USA'20

- E4S: The Extreme-scale Scientific Software Stack for Collaborative Open Source Software, Michael Heroux, Sandia National Laboratories
- Title: Practical Performance Portability at CSCS, Ben Cumming, CSCS
- Title: An Overview of High Performance Computing and Computational Fluid Dynamics at NASA, Eric Nielsen, NASA Langley
- Towards An Integrated and Resource-Aware Software Stack for the EU Exascale Systems, Martin Schulz, Technische Universität München
- Spack and E4S, Todd Gamblin, LLNL
- Advances in, and Opportunities for, LLVM for Exascale, Hal Finkel, Argonne National Laboratory
- Kokkos: Building an Open Source Community, Christian Trott, SNL
- Experiences in Designing, Developing, Packaging, and Deploying the MVAPICH2 Libraries in Spack, Hari Subramoni, Ohio State University
- Software Needs for Frontera and the NSF Leadership Class Computing Facility – the Extreme Software Stack at the Texas Advanced Computing Center, Dan Stanzione, TACC
- Building an effective ecosystem of math libraries for exascale, Ulrike Yang
- Towards Containerized HPC Applications at Exascale, Andrew Younge, Sandia
- E4S Overview and Demo, Sameer Shende, University of Oregon
- The Supercomputer “Fugaku” and Software, programming models and tools, Mitsuhisa Sato, RIKEN Center for Computational Science (R-CCS), Japan

- Presenters from 11 institutions, 6 non-DOE
- 70 participants
  - DOE Labs, NASA
  - AMD
  - HLRS, CSCS
Vision for E4S Now and in the Future

• E4S has emerged as a new top-level component in the DOE HPC community, enabling fundamentally new relationships

• E4S has similar potential for new interactions with other US agencies, US industry and international collaborators. NSF and UK are examples

• The E4S portfolio can expand to include new domains (ML/AI), lower—level components (OS), and more.

• E4S can provide better (increased quality), faster (timely delivery of leading edge capabilities) and cheaper (assisting product teams)
E4S/SDK Summary

- E4S/SDK Software: Curated release of complete production-quality HPC Linux software stack:
  - Latest ECP-developed features for 50+ products.
  - Ported and validated regularly on all common and emerging HPC platforms.
  - Single DocPortal access to all product documentation.
  - Collaborative development communities around SDKs to build culture of quality.
  - Policies for SW and user experience quality.
  - Containers, build caches for (dramatic) reduction in build time and complexity.

- E4S: A new member of the HPC ecosystem:
  - A managed portfolio of HPC software teams and products.
  - Extensible to new domains: AI/ML.
  - A new way of delivering reusable HPC software with ever-improving quality and functionality.

https://e4s.io
Next

- Rajeev Thakur will highlight progress in programming models, highlighting efforts in MPICH, Kokkos, RAJA
- Jeff Vetter will describe efforts to integrate ECP ST work into the LLVM ecosystem
Programming Models and Performance Portability

Rajeev Thakur
Argonne National Laboratory

September 25, 2020
## DOE HPC Roadmap to Exascale Systems

<table>
<thead>
<tr>
<th>FY 2012</th>
<th>FY 2016</th>
<th>FY 2018</th>
<th>FY 2021</th>
<th>FY 2022</th>
<th>FY 2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titan</td>
<td>ORNL</td>
<td>ORNL</td>
<td>ORNL</td>
<td>ORNL</td>
<td>ORNL</td>
</tr>
<tr>
<td>Cray/AMD/NVIDIA</td>
<td>IBM/NVIDIA</td>
<td>IBM/NVIDIA</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
<td>HPE/AMD</td>
</tr>
<tr>
<td>Mira</td>
<td>ANL</td>
<td>ANL</td>
<td>Aurora</td>
<td>Aurora</td>
<td>Aurora</td>
</tr>
<tr>
<td>IBM BG/Q</td>
<td>Cray/Intel KNL</td>
<td>Cray/Intel KNL</td>
<td>ANL</td>
<td>Intel/HPC</td>
<td>Intel/HPC</td>
</tr>
<tr>
<td>Theta</td>
<td>LBNL</td>
<td>Perlmutter</td>
<td>LBNL</td>
<td>LBNL</td>
<td>LBNL</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td>HPE/AMD/NVIDIA</td>
<td>HPE/AMD/NVIDIA</td>
<td>HPE/TBD</td>
</tr>
<tr>
<td>Sequoia</td>
<td>LLNL</td>
<td>Sierra</td>
<td>Sierra</td>
<td>ORNL</td>
<td>LLNL</td>
</tr>
<tr>
<td>IBM BG/Q</td>
<td>LLNL</td>
<td>LLNL</td>
<td>LLNL</td>
<td>ORNL</td>
<td>LLNL</td>
</tr>
<tr>
<td>Trinity</td>
<td>LANL/SNL</td>
<td>LANL/SNL</td>
<td>LANL/SNL</td>
<td>LANL/SNL</td>
<td>LANL/SNL</td>
</tr>
<tr>
<td>Cray/Intel Xeon/KNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td>Cray/Intel Xeon/KNL</td>
<td>HPE/AMD/NVIDIA</td>
<td>HPE/TBD</td>
<td>HPE/AMD</td>
</tr>
</tbody>
</table>

To date, only NVIDIA GPUs

GPUs from three different vendors
Trends in Internode Programming

• Individual compute nodes are becoming very powerful because of accelerators

• As a result, fewer total number of nodes are needed

• MPI will continue to serve as the main internode programming model

• Nonetheless, improvements are needed in the MPI Standard and in MPI implementations
  – Hybrid programming (integration with GPUs & GPU memory and with the node programming model)
  – Scalability, low-latency communication, optimized collective algorithms
  – Overall resilience and robustness
  – Optimized support for exascale interconnects and lower-level communication paradigms (OFI, UCX)
  – Scalable process startup and management

• PGAS models (e.g., UPC++) are also available to be used by applications that rely on them, and they face similar challenges as MPI on exascale systems
Trends in Intranode Programming

• Main challenge for exascale is in achieving performance and portability for intranode programming

• Vendor-supported options for GPUs
  – NVIDIA: CUDA, OpenACC
  – Intel: SYCL/DPC++ (C++ abstractions on top of OpenCL)
  – AMD: HIP (similar to CUDA)

• OpenMP (portable standard)
  – Supports accelerators via the target directive (since OpenMP version 4.0, July 2013)
  – Subsequent releases of OpenMP (4.5 and 5.0) have further improved support for accelerators
  – Supported by vendors on all platforms

• Kokkos and RAJA (developed at SNL and LLNL)
  – Portable, heterogenous-node programming via C++ abstractions
  – Support complex node architectures with multiple types of execution resources and multilevel memories
  – Many ECP applications use Kokkos and RAJA to write portable code for a variety of CPUs and GPUs
<table>
<thead>
<tr>
<th>Application project</th>
<th>Code</th>
<th>Main language</th>
<th>GPU programming model</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExaStar</td>
<td>FLASH</td>
<td>Fortran</td>
<td>OpenMP</td>
</tr>
<tr>
<td>ExaStar</td>
<td>CASTRO</td>
<td>Fortran, C++</td>
<td>OpenMP, OpenACC</td>
</tr>
<tr>
<td>EQSIM</td>
<td>SW4</td>
<td>C++</td>
<td>RAJA</td>
</tr>
<tr>
<td>ExaSky</td>
<td>HACC</td>
<td>C++</td>
<td>CUDA, OpenCL</td>
</tr>
<tr>
<td>ExaSky</td>
<td>CRK-HACC</td>
<td>C++</td>
<td>CUDA, OpenCL</td>
</tr>
<tr>
<td>ExaSky</td>
<td>Nyx</td>
<td>C++</td>
<td>AMReX</td>
</tr>
<tr>
<td>Subsurface</td>
<td>Chombo-Crunch</td>
<td>C++</td>
<td>PROTO, UPC++</td>
</tr>
<tr>
<td>Subsurface</td>
<td>GEOSX</td>
<td>C++</td>
<td>RAJA</td>
</tr>
<tr>
<td>E3SM-MMF</td>
<td>E3SM</td>
<td>Fortran</td>
<td>OpenACC, moving to OpenMP</td>
</tr>
<tr>
<td>Combustion-PELE</td>
<td>PeleC</td>
<td>Fortran</td>
<td>CUDA, OpenACC</td>
</tr>
<tr>
<td>Combustion-PELE</td>
<td>PeleLM</td>
<td>Fortran</td>
<td>CUDA, OpenACC</td>
</tr>
<tr>
<td>WarpX</td>
<td>WarpX + PICSAR</td>
<td>C++</td>
<td>AMReX abstractions</td>
</tr>
<tr>
<td>ExaSMR</td>
<td>Nek5000</td>
<td>Fortran</td>
<td>OpenACC</td>
</tr>
<tr>
<td>ExaSMR</td>
<td>NekRS</td>
<td>Fortran</td>
<td>libParanumal (OCCA)</td>
</tr>
<tr>
<td>ExaSMR</td>
<td>OpenMC</td>
<td>C++</td>
<td>OpenMP, OpenCL or SYCL</td>
</tr>
<tr>
<td>ExaSMR</td>
<td>Shift</td>
<td>C++</td>
<td>CUDA</td>
</tr>
<tr>
<td>WDMApp</td>
<td>GENE</td>
<td>Fortran</td>
<td>OpenMP</td>
</tr>
<tr>
<td>WDMApp</td>
<td>GEM</td>
<td>Fortran</td>
<td>OpenACC</td>
</tr>
<tr>
<td>WDMApp</td>
<td>XGC</td>
<td>Fortran</td>
<td>OpenMP, OpenACC</td>
</tr>
<tr>
<td>MFIX-Exa</td>
<td>MFIX-Exa</td>
<td>C++</td>
<td>AMReX abstractions</td>
</tr>
<tr>
<td>ExaWind</td>
<td>Nalu-Wind</td>
<td>C++</td>
<td>Kokkos</td>
</tr>
<tr>
<td>ExaWind</td>
<td>OpenFAST</td>
<td>Fortran 90</td>
<td>N/A</td>
</tr>
<tr>
<td>Project</td>
<td>Library/Model</td>
<td>Language</td>
<td>Programming Models</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
<td>----------</td>
<td>--------------------</td>
</tr>
<tr>
<td>ExaBiome</td>
<td>MetaHipMer</td>
<td>C++</td>
<td>UPC++</td>
</tr>
<tr>
<td>ExaBiome</td>
<td>GOTTCHA</td>
<td>C++</td>
<td>OpenMP, HIP, SYCL</td>
</tr>
<tr>
<td>ExaBiome</td>
<td>HipMCL</td>
<td>C++</td>
<td>OpenMP, HIP, SYCL</td>
</tr>
<tr>
<td>ExaFEL</td>
<td>M-TIP</td>
<td>C++</td>
<td>CUDA, HIP, OpenCL</td>
</tr>
<tr>
<td>ExaFEL</td>
<td>PSANA</td>
<td>C++</td>
<td>Legion</td>
</tr>
<tr>
<td>CANDLE</td>
<td>CANDLE</td>
<td>Python</td>
<td>TensorFlow, PyTorch</td>
</tr>
<tr>
<td>ExaSGD</td>
<td>GridPACK</td>
<td>C++</td>
<td></td>
</tr>
<tr>
<td>ExaSGD</td>
<td>PIPS</td>
<td>C++</td>
<td>RAJA or Kokkos</td>
</tr>
<tr>
<td>ExaSGD</td>
<td>StructJuMP</td>
<td>Julia</td>
<td></td>
</tr>
<tr>
<td>QMCPACK</td>
<td>QMCPACK</td>
<td>C++</td>
<td>OpenMP</td>
</tr>
<tr>
<td>ExaAM</td>
<td>MEUMAPPS-SS</td>
<td>Fortran</td>
<td>OpenMP, OpenACC</td>
</tr>
<tr>
<td>ExaAM</td>
<td>ExaConstit</td>
<td>C++</td>
<td>MFEM</td>
</tr>
<tr>
<td>ExaAM</td>
<td>TruchasPBF</td>
<td>Fortran</td>
<td>AMReX</td>
</tr>
<tr>
<td>ExaAM</td>
<td>Diablo</td>
<td>Fortran</td>
<td>OpenMP</td>
</tr>
<tr>
<td>ExaAM</td>
<td>ExaCA</td>
<td>C++</td>
<td>Kokkos</td>
</tr>
<tr>
<td>NWChemEx</td>
<td>NWChemEx</td>
<td>C++</td>
<td>CUDA, Kokkos</td>
</tr>
<tr>
<td>LatticeQCD</td>
<td>Chroma</td>
<td>C++</td>
<td>Kokkos</td>
</tr>
<tr>
<td>LatticeQCD</td>
<td>CPS</td>
<td>C++</td>
<td>GRID library</td>
</tr>
<tr>
<td>LatticeQCD</td>
<td>MILC</td>
<td>C</td>
<td>GRID library</td>
</tr>
<tr>
<td>GAMESS</td>
<td>GAMESS</td>
<td>Fortran</td>
<td>libcchem, libaccint</td>
</tr>
<tr>
<td>GAMESS</td>
<td>libcchem</td>
<td>C++</td>
<td>libaccint</td>
</tr>
<tr>
<td>EXAALT</td>
<td>ParSplice</td>
<td>C++</td>
<td>N/A</td>
</tr>
<tr>
<td>EXAALT</td>
<td>LAMMPS</td>
<td>C++</td>
<td>Kokkos</td>
</tr>
<tr>
<td>EXAALT</td>
<td>SNAP</td>
<td>C++</td>
<td>Kokkos</td>
</tr>
</tbody>
</table>
# ECP Programming Models and Runtimes Portfolio

<table>
<thead>
<tr>
<th>Project Short Name</th>
<th>PI Name, Inst</th>
<th>Short Description/Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMR SDK</td>
<td>Shende, UOregon</td>
<td>Support the deployment, testing and usage of PMR products</td>
</tr>
<tr>
<td>Exascale MPI</td>
<td>Balaji, ANL</td>
<td>Enhancement of the MPI Standard and the MPICH implementation of MPI for exascale</td>
</tr>
<tr>
<td>Legion</td>
<td>McCormick, LANL</td>
<td>Task-based programming model</td>
</tr>
<tr>
<td>PaRSEC</td>
<td>Dongarra, UTK</td>
<td>Task-based programming model</td>
</tr>
<tr>
<td>UPC++/GASNet</td>
<td>Hargrove, LBNL</td>
<td>Partitioned Global Address Space (PGAS) programming model</td>
</tr>
<tr>
<td>SICM</td>
<td>Lang, LANL</td>
<td>Interface and library for accessing complex memory hierarchy</td>
</tr>
<tr>
<td>OMPi-X</td>
<td>Bernholdt, ORNL</td>
<td>Enhancement of the MPI Standard and the Open MPI implementation for exascale</td>
</tr>
<tr>
<td>Kokkos / RAJA</td>
<td>Trott, SNL</td>
<td>C++ abstractions for node-level performance portability</td>
</tr>
<tr>
<td>Argo</td>
<td>Beckman, ANL</td>
<td>Low-level resource management for the operating system and runtime</td>
</tr>
</tbody>
</table>

*OpenMP (SOLLVE) project moved to Development Tools area to keep all LLVM-related efforts in one area.*
Kokkos and RAJA

- C++ performance portability abstractions developed at Sandia and Livermore labs
- Primarily funded by NNSA
- ECP ST provides additional funding to develop optimized backends for Aurora and Frontier (OpenMP, SYCL/DPC++, HIP) and for outreach to ECP applications
- Organized as one project in ECP ST. The two teams collaborate on common features, C++ and backend support, and outreach activities
- The combined project involves six labs: SNL, LLNL, ANL, LANL, LBNL, and ORNL
  - PI: Christian Trott (SNL); Co-PIs: Rich Hornung (LLNL), Hal Finkel (ANL), Galen Shipman (LANL), Jack Deslippe (LBNL), Damien Lebrun-Grandie (ORNL)
Kokkos: Preparation for Exascale Platforms

• Kokkos provides a production-quality solution for C++ performance portability
  – Kokkos Core: C++ template-based library for the programming model
  – Kokkos Tools: Profiling, debugging and tuning tools that connect into Kokkos Core
  – Kokkos Kernels: Math libraries based on Kokkos Core

• Distributed development team: SNL, ORNL, ANL, LBNL, and LANL

• Long-term goal is alignment and extension of the ISO C++ standard
  – Fundamental capabilities from Kokkos are proposed for the C++ standard

• Kokkos is currently used on most DOE and many European production HPC systems
  – Numerous applications use Kokkos to run on Sierra, Summit, Astra, Trinity, Theta, PizDaint, and others

• Extensively used across ECP AD and ST projects
  – Exawind, EXAALT, WDMApp, ExaAM, SNL and LANL ATDM apps, ExaGraph, CoPA, ALExa (ArborX and DTK), Kokkos Kernels, Trilinos, FleCSI, …

• Numerous workshops and training events
  – Now available “The Kokkos Lectures”: 15 hours of recorded lectures https://kokkos.link/the-lectures.
Kokkos: SPARTA – An Example of Production Runs

- **Stochastic Parallel Rarefied-gas Time-accurate Analyzer**
- A direct simulation Monte Carlo code
- Developers: *Steve Plimpton, Stan Moore, Michael Gallis*
- Only application to have run on all of Trinity
- Benchmarked on 16K GPUs on Sierra

**Exascale Support Status**

- HIP support in Kokkos 3.2 for AMD GPUs
  - Currently missing hierarchical parallelism and tasking due to compiler bug
  - Some apps already running (ArborX, some LAMMPS runs, ...)
- OpenMP Target support for Intel GPUs in Kokkos 3.2
  - Some MiniApps are already running
- Expect most functionality will be available for HIP and OpenMP in Kokkos 3.3 in Nov 2020
- SYCL/DPC++ port in progress. Encountered compiler/runtime bugs, which are getting fixed.
RAJA: Preparation for Exascale Platforms

- A suite of performance-portability tools developed at LLNL
  - RAJA (C++ kernel execution abstractions); CHAI (C++ array abstractions); Umpire (memory management)

- Used in a diverse set of ECP and LLNL mission applications and support libraries
  - Supports majority of production LLNL weapons program apps and apps in other LLNL programs
  - ECP: SW4 (EQSIM), GEOSX (Subsurface), MFEM (CEED), ExaSGD, SUNDIALS, DevilRay (Alpine)

- Exascale platform support
  - Full support for HIP back-end in public releases since January 2020. Key part of El Capitan CoE activities.
  - SYCL back-end in progress, including SW4 & RAJA Performance Suite running on pre-Aurora systems now

- ECP applications have demonstrated substantial GPU node (Sierra) vs. CTS-1 node speedups
  - SW4: 28X speedup on Sierra node (4 NVIDIA Volta GPUs) vs. CTS-1 CPU-only node (36 core Intel Xeon).
    - Hayward fault earthquake simulation of unprecedented resolution (26B grid points) runs in 10.3 hours on 256 nodes of Sierra (6% of machine). Same problem (grid size, run time) requires 8196 nodes of Cori-II (85% of machine)
  - GEOSX: 14X speedup on Sierra node vs. CTS-1 CPU-only node for explicit time integration solid mechanics model needed for ECP challenge problem
Exascale MPI (MPICH)

- PI: Pavan Balaji (ANL)
- MPICH has been a key influencer in the adoption of MPI
  - First or most comprehensive implementation of each new version of the MPI standard
  - Allows supercomputing centers to not compromise on what features they demand from vendors
- R&D 100 Award in 2005
- MPICH and its derivatives are the world’s most widely used MPI implementations on large-scale supercomputers
- MPICH will be the primary MPI implementation on Aurora, Frontier, and El Capitan (via Intel MPI and Cray MPI)

Ongoing activities (September 2020)
- MPI Forum and standardization efforts
- Addressing ECP application-specific issues
- Efficient and transparent support for GPUs from multiple vendors (Intel, AMD, NVIDIA)
- New library for efficient noncontiguous data communication (MPI derived datatypes)

https://www.mpich.org/
https://github.com/pmodels/mpich

MPICH is not just a software
It’s an Ecosystem
ECP Application Interactions, MPI Standardization Activities, and Vendor Collaboration

Interactions with ECP Applications
- Improvement on Key Technologies for ECP Applications
- Collaboration on Evaluation of Prototypes

Vendor Collaborations
- Weekly telecon for project updates
- Deep dives with individual teams to discuss development priorities and plans
- Hackathons: Week-long accelerated development for priority topics

Standardization Efforts for MPI-4.0
- Leading the Hybrid Working Group
- Participating in other Working Groups
  - Point-to-point Communication
  - Fault Tolerance
  - Remote Memory Access
  - Hardware-Topologies
  - I/O
GPU Support in MPICH

- Communication using GPU buffers
  - Support native RDMA on GPU buffers through Libfabric (OFI) and UCX
  - Fallback for the case where RDMA is not available with GPU
  - Support GPU intranode communication through GPU IPC
  - Datatype handling with GPU buffers

- Supporting noncontiguous datatypes with GPU
  - Yaksa: A high performance datatype engine
  - Multiple backends provide datatype support for CPUs and GPUs from different vendors

The GPU support in MPICH is developed in close collaboration with vendors (AMD, Cray, Intel, Mellanox, NVIDIA).

https://github.com/pmodels/yaksa
Improvements to MPI + Threads Support

Lock Optimization
- Global Locks
  - Replace global lock with per-object locks
  - Reduce lock scope

Granular Locks
- Replace global lock with per-object locks
- Reduce lock scope

Argobots Integration
- MPICH works with Argobots, a lightweight high-performance user-level thread library.
- Increased opportunities for computation and communication overlapping
- Reduced thread synchronization

Virtual Communication Interface (VCI)
- VCIs are independent sets of communication resources in MPICH. They can be mapped to network hardware contexts.
- Having multiple VCIs allows the communication from different threads to be handled by different MPI resources, therefore reducing contention in the MPI library.
- It also allows multithreaded MPI applications to fully utilize network hardware contexts.
- With reduce contention and full utilization of network hardware, MPI+Threads can achieve a performance close to MPI-only application which leads to significant improvement on strong scaling performance of MPI+Threads applications.
Brief updates on other PMR Projects
PMR Software Development Kit (SDK)

• PI: Sameer Shende (UOregon)

• Developed draft set of community policies for PMR projects

• Integrated and released PMR products in a containerized distribution with support for GPUs (ROCm 3.x and CUDA 10.1). Available for download from https://e4s.io or DockerHub for Linux x86_64 and ppc64le.

• E4S Spack build cache for bare-metal installation has over 20,000 binaries [http://oaciss.uoregon.edu/e4s/inventory.html] with support for the latest release of Spack. E4S is used for Spack pull request (PR) merge validation testing [cdash.spack.io].

• E4S validation testing [https://github.com/E4S-Project/testsuite] planned for early access systems and includes LLVM support for Shasta Testing Project (STP)

• Examples of E4S build cache being used in ECP applications:
  – Pantheon: http://pantheonscience.org/projects/e4s
Open MPI (OMPI-X)

- PI: David Bernholdt (ORNL); Co-PIs: Howard Pritchard (LANL), Ignacio Laguna (LLNL), Ron Brightwell (SNL), George Bosilca (UTK)
- Enhance the MPI Standard and the Open MPI implementation for exascale
- Open MPI (via IBM Spectrum MPI) is the default implementation on Summit and Sierra
- Two new features championed by the group have been officially accepted for the upcoming MPI-4 Standard
  - Partitioned point-to-point communication
  - MPI sessions
- These features are already available in Open MPI for ECP teams to try out
Legion & FlexFlow

- Task-based programming model
  - PI: Pat McCormick (LANL); Co-PIs: Alex Aiken (Stanford), Pavan Balaji (ANL)
  - Used in ExaFEL, S3D, CANDLE, NNSA PSAAP II & III, ...
  - Ports to exascale architectures in progress

- FlexFlow is a distributed deep learning framework
  - Built on Legion
  - Exploits Legion’s first-class data partitioning & distributed execution
  - Secret sauce: Automatic search to find a high-performance partitioning
  - Dramatically improves locality and scalability
  - Reduces CANDLE run times from days to hours
  - Portability interfaces for Keras and PyTorch in progress

The FlexFlow system is based on Legion and, in comparison to Google’s TensorFlow, can scale to 768 GPUs and reduces the per-epoch training time from 18 hours to 1.2 hours for CANDLE’s Uno benchmark.
Argo

- PI: Pete Beckman (ANL); Co-PIs: Tapasya Patki and Maya Gokhale (LLNL)
- **Deep Memory**: Enable applications to improve usage of new memory types
  - AML: Application-aware management of deep memory systems
    - New: Enhanced memory topology API for complex multi-CPU, multi-GPU exascale nodes; ongoing integration effort with OpenMC
  - UMap: User-level mapping of NVRAM/SSD into the memory hierarchy
    - New: UMap handler can access memory from other nodes, e.g., a dedicated memory server or data producer nodes
- **Power Management**: Dynamically manage power to improve energy usage and performance
  - PowerStack: Global management through power-aware job scheduling
    - New: Variorum + GEOPM integration, initial power management support for Kokkos tools
  - NRM: Node-level power management infrastructure
    - New: ML-based power management controller
- **Co-Design**: Working with vendors, improve interfaces to exploit new hardware capabilities in HPC and ML
- **Platform Readiness**
  - NRM runs on Theta; UMap runs on Sierra, AMD systems; PowerStack is deployed in TOSS; ongoing work on using ECP-CI to test AML on Theta and Summit
Pagoda (UPC++, GASNet-EX)

- PI: Paul Hargrove (LBNL)
- GASNet-EX: Portable high-performance networking layer for PGAS runtimes
- UPC++: C++ template library providing asynchronous one-sided RMA and RPC for rapid development of PGAS applications, implemented using GASNet-EX
- ECP use cases: ExaBiome, ExaGraph, NWChemEx, Legion
- Recent Activities
  - Semi-annual software releases of UPC++, GASNet-EX
  - Optimizations for GPUs in progress (efficient GPU-GPU memory transfers)
  - ECP training event for UPC++ in May
  - SC20 tutorial accepted
2.3.2 Development Tools (and Compilers)

Jeffrey Vetter, Oak Ridge National Laboratory

ASCAC Meeting
25 Sep 2020
## WBS 2.3.2 Development Tools: Context for the portfolio

<table>
<thead>
<tr>
<th>Vision</th>
<th>Enable exascale computing by providing architecture-ready compilers and programming tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenges</td>
<td>Complex and heterogeneous node architectures with immature software systems; exact system architectures not known until early systems arrive; diverse application requirements</td>
</tr>
<tr>
<td>Mission</td>
<td>Develop compilers and programming tools that enable applications teams to achieve high performance while providing basic portability across diverse exascale architectures</td>
</tr>
<tr>
<td>Objective</td>
<td>Produce production-ready compilers and programming tools, tuned for application needs, deployed at facilities via vendors or as part of SDK</td>
</tr>
<tr>
<td>Starting Point</td>
<td>Existing heterogeneous architecture programming models including CUDA, OpenACC, OpenMP, OpenCL, and various research efforts</td>
</tr>
<tr>
<td>Portfolio Goals</td>
<td><strong>Compilers</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Prog Tools</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Memory</strong></td>
</tr>
</tbody>
</table>
2.3.2 Development Tools Portfolio

<table>
<thead>
<tr>
<th>Project Short Name</th>
<th>PI Name, Inst</th>
<th>Short Description/Objective</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK</td>
<td>Miller, UW Madison</td>
<td>SDK: System-facing delivery of software products, APIs, and vendor integration</td>
</tr>
<tr>
<td>EXAPAPI</td>
<td>Dongarra, UTK</td>
<td>EXAPAPI: The Exascale Performance Application Programming Interface</td>
</tr>
<tr>
<td>HPCToolkit</td>
<td>Mellor-Crummy, Rice</td>
<td>Extending HPCToolkit to Measure and Analyze Code Performance on Exascale Platforms</td>
</tr>
<tr>
<td>PROTEAS-TUNE</td>
<td>Vetter, ORNL</td>
<td>PROTEAS-TUNE: Programming, Autotuning, and Optimization Toolchain for Emerging Architectures and Systems</td>
</tr>
<tr>
<td>SOLLVE</td>
<td>Chapman, Stony Brook/BNL</td>
<td>SOLLVE: OpenMP for LLVM</td>
</tr>
<tr>
<td>Flang</td>
<td>McCormick, LANL</td>
<td>Flang: Fortran for LLVM</td>
</tr>
</tbody>
</table>
Development Tools: LLVM Ecosystem

Deep Dive
LLVM is an infrastructure for creating compilers

- Features: LLVM has become well known for an important set of features:
  - LLVM is a liberally-licensed(*) infrastructure for creating compilers, other toolchain components, and JIT compilation engines.
  - A modular, well-defined IR allows use by a lot of different languages (C, C++, Fortran, Julia, Rust, Python (e.g., via Numba), Swift, ML frameworks (e.g., TensorFlow/XLA, PyTorch/Glow), and many others.
  - A backend infrastructure allowing the efficient creation of backends for new (heterogeneous) hardware.
  - A state-of-the-art C++ frontend, CUDA support, scalable LTO, sanitizers and other debugging capabilities, and more.
  - High code-quality community standards and review process

[Diagram showing LLVM ecosystem with connections to Intel, ARM, IBM, NVIDIA (and PGI), AMD, Apple, Google, Cray, and Academia, Labs, etc.]

*Note: (*) indicates that the license is under active discussion and may be updated.
Next week’s LLVM Dev Meeting Brings Together (virtually) over 500+ Developers from around the World including ECP Staff

https://llvm.org/devmtg/2020-09/

- The Present and Future of Interprocedural Optimization in LLVM - J. Doerfert; B. Homerding; ...
- Proposal for A Framework for More Effective Loop Optimizations - M. Kruse; H. Finkel
- Changing Everything With Clang Plugins; A Story About Syntax Extensions, Clang’s AST, and Quantum Computing = H. Finkel; A. Mccaskey
- (OpenMP) Parallelism-Aware Optimizations - J. Doerfert; S. Stipanovic; H. Mosquera; J. Chesterfield; G. Georgakoudis; J. Huber
- A Deep Dive into the Interprocedural Optimization Infrastructure - J. Doerfert; B. Homerding; S. Baziotis; S. Stipanovic; H. Ueno; K. Dinel; S. Okumura; L. Chen
- From Implicit Pass Dependencies to Effectiveness Prediction - H. Ueno; J. Doerfert; E. Park; G. Georgakoudis; T. Jayatilaka; S. Badruswamy
- OpenACC support in Flang with a MLIR dialect - V. Clement; J. Vetter
- Flang Update - S. Scalpone
- Code Feature Analysis, Tracking, and Future Usage - T. Jayatilaka; J. Doerfert; G. Georgakoudis; E. Park; H. Ueno; S. Badruswamy
- Loop Optimization BoF - M. Kruse; K. Barton
ECP is Improving the LLVM Compiler Ecosystem

<table>
<thead>
<tr>
<th>LLVM</th>
<th>+SOLLVE</th>
<th>+PROTEAS-TUNE</th>
<th>+FLANG</th>
<th>+HPCToolkit</th>
<th>+ATDM</th>
<th>Vendors</th>
</tr>
</thead>
</table>
| • Very popular open source compiler infrastructure  
  • Easily extensible  
  • Widely used and contributed to in industry  
  • Permissive license  
  • Used for heterogeneous computing | • Enhancing the implementation of OpenMP in LLVM  
  • Unified memory  
  • OMP Optimizations  
  • Prototype OMP features for LLVM  
  • OMP test suite  
  • Tracking OMP implementation quality | • Core optimization improvements to LLVM  
  • OpenACC capability for LLVM  
  • Clacc  
  • Flacc  
  • Autotuning for OpenACC and OpenMP in LLVM  
  • Integration with Tau performance tools | • Developing an open-source, production Fortran frontend  
  • Upstream to LLVM public release  
  • Support for OpenMP and OpenACC  
  • Recently approved by LLVM | • Improvements to OpenMP profiling interface OMPT  
  • OMPT specification improvements  
  • Refine HPCT for OMPT improvements | • Enhancing LLVM to optimize template expansion for FlexCSi, Kokkos, RAJA, etc.  
  • Flang testing and evaluation | • Increasing dependence on LLVM  
  • Collaborations with many vendors using LLVM  
  • AMD  
  • ARM  
  • Cray  
  • HPE  
  • IBM  
  • Intel  
  • NVIDIA |

Active involvement with broad LLVM community: LLVM Dev, EuroLLVM
Leveraging LLVM Ecosystem to Meet a Critical ECP (community) need: FORTRAN

- Fortran support continues to be an ongoing requirement
- Flang project started in NNSA funding NVIDIA/PGI to open source compiler frontend into LLVM ecosystem
- SOLLVE is improving OpenMP dialect, implementation, and core optimizations
- PROTEAS-TUNE is creating OpenACC dialect and improving MLIR
- ECP projects are contributing many changes upstream to LLVM core, MLIR, etc
- Many others are contributing: backends for processors, optimizations in toolchain, ...
  - Google contributed MLIR
PROTEAS-TUNE: Clacc – OpenACC in Clang/LLVM

• Develop production-quality, standard-conforming traditional OpenACC compiler and runtime support by extending Clang and LLVM
  – Build on existing OpenMP infrastructure

• Enable research and development of source-level OpenACC tools
  – Design compiler to leverage Clang/LLVM ecosystem extensibility
  – E.g., Pretty printers, analyzers, lint tools, and debugger and editor extensions

• Actively contribute improvements to the OpenACC specification

• Actively contribute upstream all Clang and LLVM improvements that are mutually beneficial
  – Many contributions are already in LLVM

• Open-source with multiple collaborators (vendors, universities)
Speculative Loop Transformation Representation

for (int i = c-2; i < c+255; i+=2)

normalization

for (int i = 0; i < 128; i+=1)

#pragma clang loop vectorize(enable)
for (int i = 0; i < 128; i+=1)

legality check

#pragma clang loop distribute(enable)
for (int i = 0; i < 128; i+=1)

legalitv check

#pragma omp target parallel for
for (int i = 0; i < 128; i+=1)

legalitiy check

Profitability Heuristics

- Best practice library
- Execution time model
- User-Directives / Application-provided
- Autotuning
- Machine learning

Codegen

Runtime Version Selection
ECP LLVM Integration and Deployment

- Develop an integrated ECP LLVM distribution
  - Integrating different ECP projects using LLVM
  - CI on target architectures
  - Shared vehicle for improvements in LLVM
  - Increased collaboration within ECP

- Operations
  - ECP LLVM distro will be closely maintained fork of LLVM mono repo
  - Multiple branches of individual ECP projects will exist at git branches
  - Branches will be integrated into ECP LLVM

- Periodic upstreaming and patching of LLVM monorepo

https://github.com/llvm-doe-org/llvm-project
Questions?
Bonus Slides
Example Integration Activities

PROTEAS/Kokkos

• Performance profiling of Kokkos applications
• Tau and Kokkos team collaborating on initial interface to allow transparent performance profiling
• Impact on application performance and user productivity

ExaPAPI/NWChemEx

• Co-design a prototype implementation of a PAPI component for NWChemEx-related software-defined events (SDE).
• Development of a proof-of-concept PAPI component that supports the performance metrics the NWChemEx team requires

ExaPAPI/NWChemEx Performance Counter | Counter Description
--- | ---
ade::NWChem::t28_chain_cnt | Total Number of chains with sequential DGEMMs in CCSD kernel v2.80
ade::NWChem::t28_max_chain_length | Maximum number of sequential DGEMMs per chain in CCSD kernel v2.80
ade::NWChem::t28_dgemm_cnt | Total number of DGEMMs in CCSD kernel v2.80
ade::NWChem::t28_flops_cnt | Total number of floating-point operations in CCSD kernel v2.80

Flang//LLVM Community

• Developing an open source, production quality Fortran front-end for LLVM
• Work in progress for several years including rewriting major portions of front-end in order to merge successfully with LLVM design and standards
• Received approval from LLVM leadership for upstreaming of final Flang implementation
Scope and objectives

- Monitor the costs of data movement.
- Identify latency and bandwidth bound code regions and quantify data movement associated with individual variables.
- Support accurate, low-overhead measurement of a broad range of parallel applications.

Project accomplishment

- Added a feature in HPCToolkit to monitor both dynamic allocation of memory and static variables in a load module or executable.
- Added a feature in the HPCToolkit user interface to present the cost of data movement for important variables.

Impact

- HPCToolkit is installed on many DOE systems.
- HPCToolkit supports low overhead measurement and analysis of performance for a wide range of applications.
- Application, library and tool developers can use HPCToolkit to analyze the performance of their software.

Pinpoint and quantify data movement

HPCToolkit utilizes the additional information from Intel PEBS and IBM Marked events to directly measure memory access latency to both variables and instructions.

HPCToolkit’s Github at https://github.com/HPCToolkit/hpctoolkit/tree/datacentric
PaRSEC

• PIs: Jack Dongarra and George Bosilca (UTK)
• Data-centric programming environment based on asynchronous tasks executing on a heterogeneous distributed environment
• ECP use cases: SLATE, NWChemEx, GAMESS
• Recent Activities
  – Redesigning the communication infrastructure to support one-sided communication
  – New Domain Specific Languages targeting developer productivity and facilitating the expression of parallel/distributed algorithmic constructs (including collective behaviors)
  – Initial support for ARM processors (atomic operations, gather/scatter, SVE)
  – Improved interoperability with DPLASMA, ScaLAPACK, and SLATE
    • Automatic support for heterogeneous environments
  – Heterogeneous-memory-aware algorithms and integration with MPQC (NWChemEx)
  – Large scale performance evaluation on all pre Exascale platforms
Simplified Interface to Complex Memory (SICM)

• PI: Mike Lang (LANL); Co-PIs: Terry Jones (ORNL), Maya Gokhale (LANL), Michael Jantz (UTK), Frank Mueller (NC State)

• Portable low-level interface targeting Intel KNL, IBM Power9+Volta, Intel Optane DC PMM, Frontier, and Aurora

• High-level interface that makes reasonable decisions for applications

• Meta allocator for persistent memory focusing on graph applications

• ECP use cases: SOLLVE (OpenMP), OMPI-X, Umpire, Kokkos, NaluWind, Trilinos

• Recent Activities
  – Integration as an allocator in CLANG/OpenMP
  – Integration into Kokkos
  – Integration into NaluWind
Concluding Remarks

• ECP Programming Models and Runtimes projects are actively working on performance tuning for exascale platforms and working closely with applications, vendors, and facilities

• There isn’t one single intranode programming model for meeting the performance portability needs of applications

• Instead, a variety of models are available, and applications are using them

• At the end of ECP, we will have a lot of useful data on what works well (and what doesn’t), which will help achieve convergence