What is a proxy app and why should I care?

IEEE Cluster 2017

David Richards
PI ECP Proxy App Project

September 7, 2017

LLNL-PRES-739582

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

What is the Exascale Computing Project (ECP)?

- As part of the National Strategic Computing Initiative, ECP was established to accelerate delivery of a capable exascale computing system that integrates hardware and software capability to deliver approximately 50 times more performance than today's 20-petaflops machines on mission critical applications.
  - DOE is a lead agency within NSCI, along with DoD and NSF
  - Deployment agencies: NASA, FBI, NIH, DHS, NOAA
- The ECP is a coordinated effort to support US leadership in achieving next-generation HPC
- ECP's work encompasses
  - applications,
  - system software,
  - hardware technologies and architectures, and
  - workforce development to meet scientific and national security mission needs.

The ECP Plan of Record

- A 7-year project that follows the co-design approach, which runs through 2023 (including 12 months of schedule contingency)
- Enable an initial exascale system based on advanced architecture and delivered in 2021
- Enable capable exascale systems, based on ECP R&D, delivered in 2022 and deployed in 2023 as part of an NNSA and SC facility upgrades
- Acquisition of the exascale systems is outside of the ECP scope, will be carried out by DOE-SC and NNSA-ASC supercomputing facilities

The ECP Plan of Record

- A 7-year project that follows the co-design approach, which runs through 2023 (including 12 months of schedule contingency)
- Enable an initial exascale system based on advanced architecture and delivered in 2021
- Enable capable exascale systems, based on ECP R&D, delivered in 2022 and deployed in 2023 as part of an NNSA and SC facility upgrades
- Acquisition of the exascale systems is outside of the ECP scope, will be carried out by DOE-SC and NNSA-ASC supercomputing facilities

"No battle plan survives contact with the enemy."
- Helmuth von Moltke

EXASCALE COMPUTING PROJECT

What is a Capable Exascale Computing System?

- Delivers 50× the performance of today's 20 PF systems, supporting applications that deliver high-fidelity solutions in less time and address problems of greater complexity
- Operates in an efficient and affordable power envelope
- Is sufficiently resilient (perceived fault rate: ≤1/week)
- Includes a software stack that supports a broad spectrum of applications and workloads

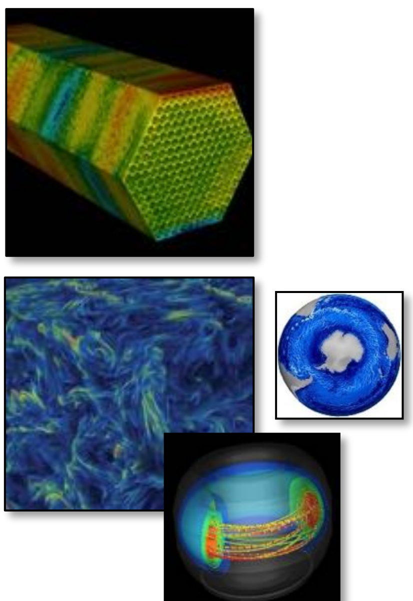
This ecosystem will be developed using a co-design approach to deliver new software and hardware technologies and new science, energy, & national security applications at heretofore unseen scale.

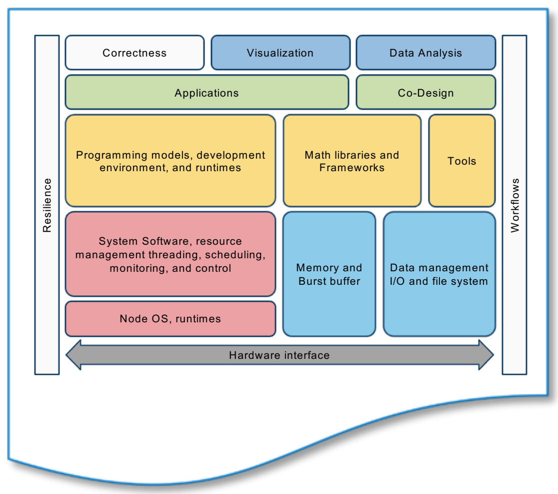
A capable exascale system will be

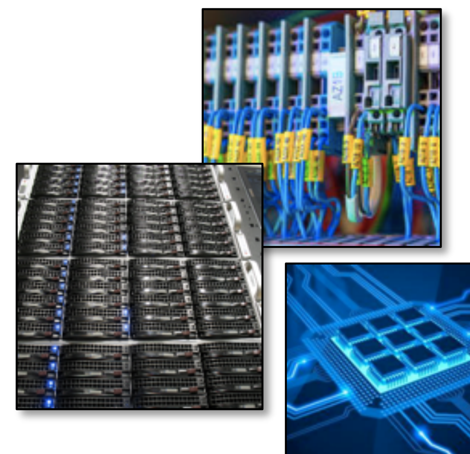
- Affordable
- Usable
- Useful

Achieving capable exascale requires co-design across the entire computing ecosystem

Application Development	Software Technology	Hardware Technology
Science and mission applications	Scalable and productive software stack	Hardware technology elements

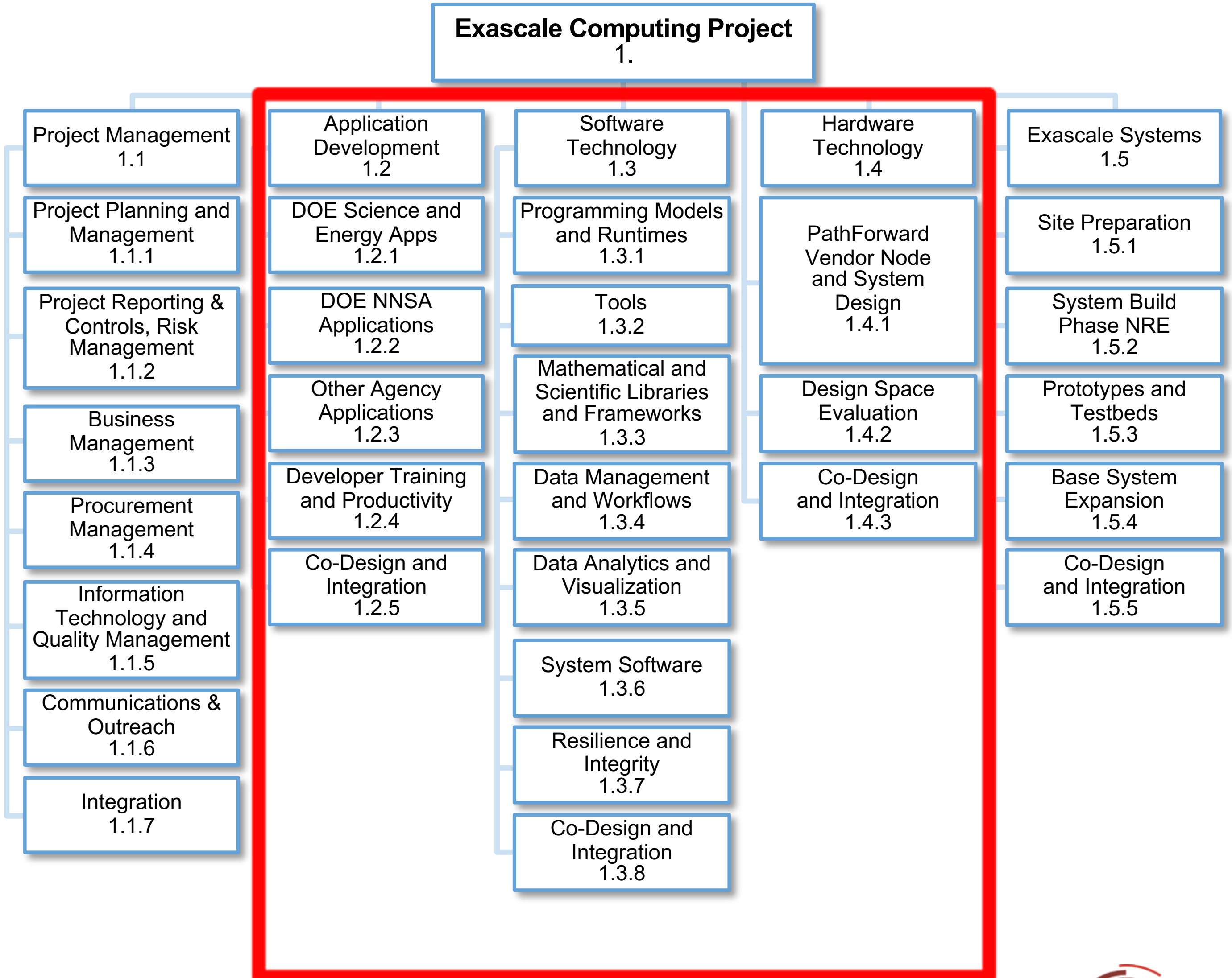






EXASCALE COMPUTING PROJECT

The ECP Work Breakdown Structure (WBS)



EXASCALE COMPUTING PROJECT

ECP Outcomes

Applications

- Addressing major challenges
  - National security
  - Energy assurance
  - Economic competitiveness
  - Scientific discovery
- Many important applications running at exascale in 2021, producing useful results
- Full suite of mission and science applications ready for 2023 exascale systems
- Industry and mission critical applications have been prepared for a more diverse and sophisticated set of computing technologies

Programming Models and Environments

- Systems usable by scientists across DOE and other agencies (not just a few “hero” programmers)
- Scalable software that provides effective tools for science and improves resilience
- New tools that shorten the development cycle for tackling new national challenges

Hardware Technology

- Advances in architecture, resilience, and power efficiency that will be incorporated into smaller systems & product roadmaps
  - Reducing energy consumption
  - Accelerating U.S. science
- Pathway for application development and resources for science breakthroughs

EXASCALE COMPUTING PROJECT

Proxy apps are models for one or more features of the parent application

- Proxy apps come in various sizes
  - Kernels, skeleton apps, mini apps
- Proxies can be models for
  - Performance critical algorithms
  - Communication patterns
  - Programming models and styles
- Like any model, proxies can be misused beyond their regime of validity
- When feasible, use full applications instead of proxies

Proxy apps will be used extensively in ECP co-design

Why create a proxy for your application?

- Can not share application with collaborator
  - OUO, Export Control, Classified, etc.
  - Proxy apps were key drivers of vendor interactions during FastForward and DesignForward
- Need a more nimble code to prototype and test ideas
  - Smaller code base that still captures key issue being explored
  - Easier to build and work with

Weight is wrong, but aerodynamics can scale

Proxies are most useful when created by or in collaboration with domain experts

Why you should care about proxy apps #1

Proxy apps are used to select which supercomputer your center will purchase

The CORAL benchmarking suite included proxy apps (hint: CORAL 2 will too)

Categories	Scalable Science	Throughput	Data Centric	Skeleton
Marquee (TR-1)	LSMS QBOX NEKbone HACC	CAM-SE UMT2013 AMG2013 MCB	Graph500 Int sort Hashing	CLOMP IOR CORAL MPI Memory CORAL loops
Elective (TR-2)		QMCPACK NAMD LULESH SNAP miniFE	SPECint_ peak2006	Pynamic HACC I/O FTQ XSBench miniMADNESS
Elective Micro-Benchmarks (TR-3)	NEKbonemk HACCmk	UMTmk AMDmk MILCmk GFMCmk		

A proxy app becomes a benchmark when it is matched with:

- A Figure of Merit (FOM)
- A set of run rules
  - Problem size
  - Code version
  - Etc.

The FOM and rules must be carefully chosen or the benchmark is meaningless

What is a Figure of Merit?

- A FOM is a measure of application throughput performance
- Good FOMs usually scale with performance
  - 2X problem run 2X faster (than 1X problem on old platform) = 4X FOM
  - 1X problem run 4X faster = 4X FOM
  - FOM may need to consider application algorithm scaling with system size

Vendors bid an FOM and must later meet a target FOM at system acceptance

The figure of merit can profoundly impact design

These vehicles are exquisitely tailored to satisfy a particular figure of merit

What you ask for is what you will get

The figure of merit can profoundly impact design

What you ask for is what you will get

The figure of merit for the “F1 Benchmark”

FOM: Time to complete a series of complex race courses

Benchmarks need rules:
A very small selection of the 2017 FIA technical regulations

- The internal combustion engine of a Formula One car must 1.6-litres in capacity and rev-limited to 15,000rpm.
- The engine must also have six cylinders arranged in a 90-degree formation, with two inlet and two exhaust valves per cylinder and a single turbocharger.
- Engines exhaust systems must have a single tailpipe for the turbine and either one or two tailpipes for the wastegate.
- Fuel flow to the engine is limited to 100 kilograms/hour.
- The use of any device, other than the engine and one MGU-K, to propel the car, is not permitted.
- The overall weight of the power unit must be a minimum of 145kg. The Energy Store must be installed wholly within the survival cell and must weigh between 20kg and 25kg.
- The only means by which the driver may control acceleration torque to the driven wheels is via a single chassis mounted foot (accelerator) pedal.
- The crankcase and cylinder block of the engine must be made of cast or wrought aluminium alloys - the use of composite materials is not allowed. The crankshaft and camshafts must be made from an iron-based alloy, pistons from an aluminium alloy and valves from alloys based on iron, nickel, cobalt or titanium.
- The MGU-H must be solely mechanically linked to the exhaust turbine of the pressure charging system. The MGU-K must be solely and permanently mechanically linked to the powertrain before the main clutch.
- A maximum of 4MJ per lap can be transferred from the ES to the MGU-K (and then in turn to the drivetrain).

Complete regulations are 102 pages

Benchmarks need rules:
A very small selection of the 2017 FIA technical regulations

- The internal combustion engine of a Formula One car must 1.6-litres in capacity and rev-limited to 15,000rpm.
- The engine must also have six cylinders arranged in a 90-degree formation, with two inlet and two exhaust valves per cylinder and a single turbocharger.
- Engines exhaust systems must have a single tailpipe for the turbine and eithe[illegible] the wastegate.
- Fuel flow to the engine is limited to 100 kilograms/hour.
- The use of any device, other than the engine and one MGU-K, to propel the car, is not permitted.
- The overall weight of the power unit must be a minimum of 145kg. The Energy Store must be installed wholly within the survival cell and must weigh between 20kg and 25kg.

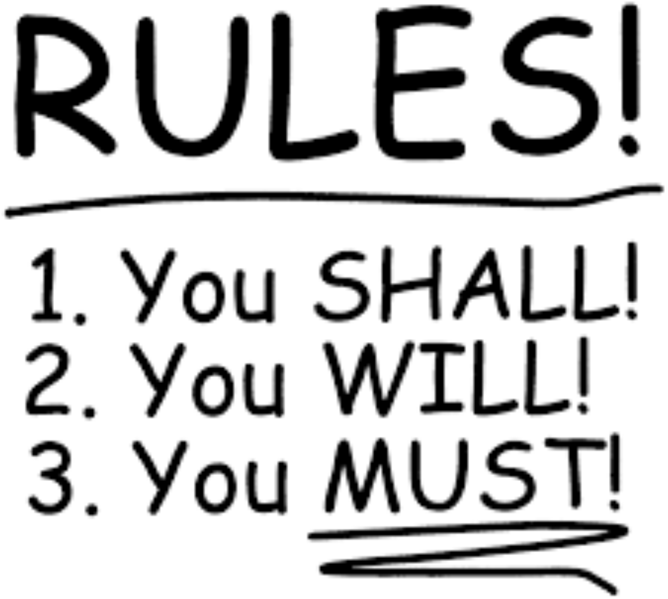
Accelerators

- The only means by which the driver may control acceleration torque to the driven wheels is via a single chassis mounted foot (accelerator) pedal.
- The crankcase and cylinder block of the engine must be made of cast or wrought aluminium alloys - the use of composite materials is not allowed. The crankshaft and camshafts must be made from an iron-based alloy, pistons from an aluminium alloy and valves from alloys based on iron, nickel, cobalt or titanium.
- The MGU-H must be solely mechanically linked to the exhaust turbine of the pressure charging system. The MGU-K must be solely and permanently mechanically linked to the powertrain before the main clutch.
- A maximum of 4MJ per lap can be transferred from the ES to the MGU-K (and then in turn to the drivetrain).

Complete regulations are 102 pages

Typical procurement rules

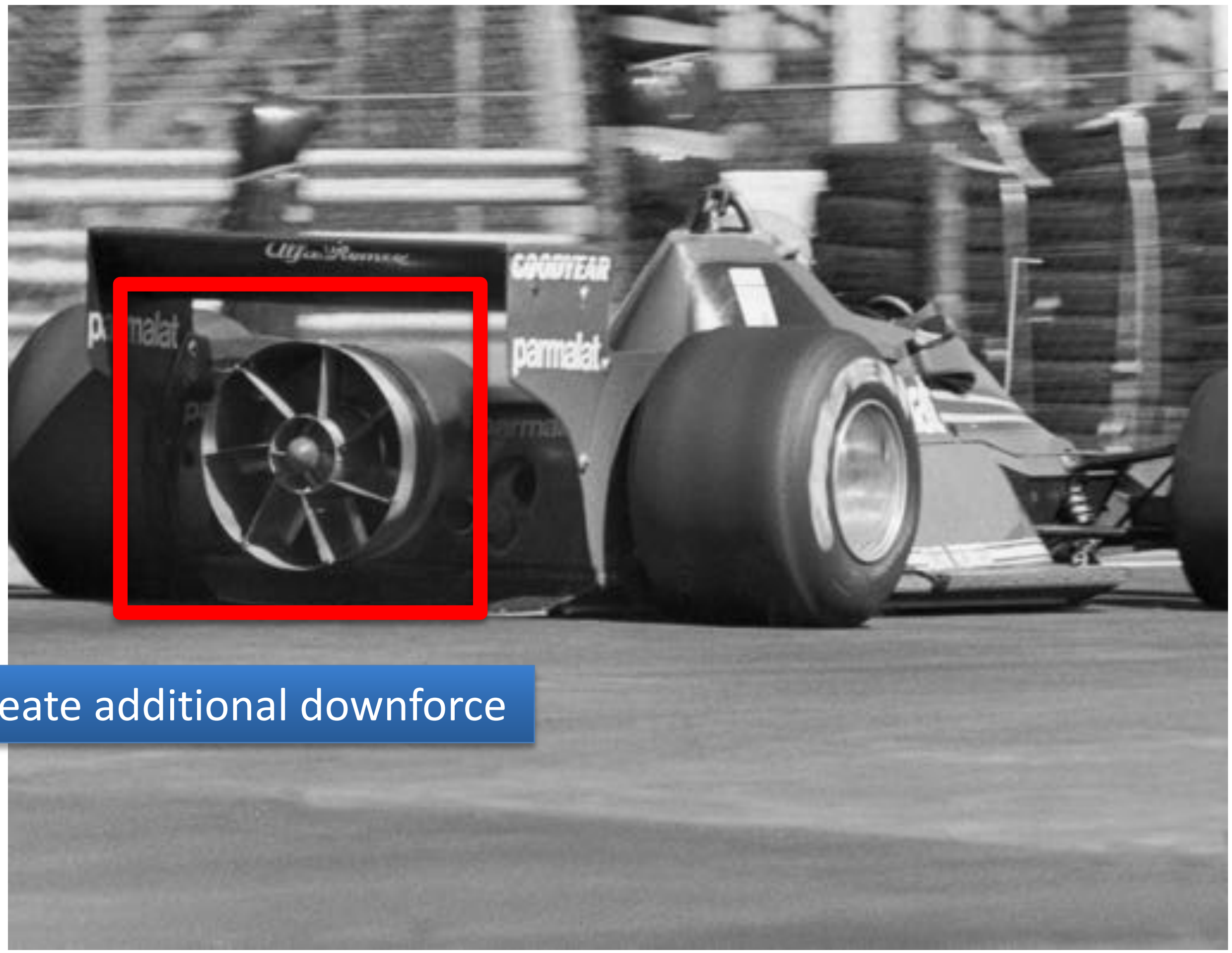
- Expected weak and strong scaling to meet target FOM
- Allowed code modifications
- Memory per MPI rank
- Node count(s) to run jobs on



DOE FOMs typically emphasize throughput

Benchmark participants will ~~cheat~~ bend the rules

Benchmark participants will ~~cheat~~ bend the rules



Huge fan to create additional downforce

What could go wrong in a procurement?

- Poorly defined FOMs
- Buggy code
- Unrepresentative Problems
- Missing key hardware stressor in benchmark suite

Vendors will look for any angle that is left open.
Heads they win tails you lose.

Why you should care about proxy apps #2

Proxy apps are major drivers
of co-design research with vendors

Proxy apps have (at least) four use cases

- Procurement benchmarks
- Vendor Co-design
- Basic Research
- Application Prototyping

DOE selected 6 vendors for PathForward projects

- $430 million (40% from vendors)
- AMD, Cray, HPE, IBM, Intel, Nvidia
- Awards focus on accelerating the development of hardware technology
- Vendor contracts require interaction with DOE application teams
- At least some of these interactions will be through proxy apps

Vendor co-design and proxies

- DOE funds multiple vendor R&D projects
  - FastForward, DesignForward, PathForward
  - CORAL non recurring engineering
- Proxies play a critical role in these efforts
  - Used to evaluate the success of deliverables
  - Communication vehicle for DOE concerns
  - Start conversations about code constraints and flexibility

Not meant to be static codes, but start a give and take process towards mutual understanding.

Why you should care about proxy apps #3

Proxy apps can be useful for basic research

Research: What happens when proxies go out into the wild

- Proxies are relatively easy to use and build
- They are rightly viewed as more realistic than benchmark suites (e.g. NAS, Rhodinia, etc)
- Many researchers use them for their papers
- Proxy authors often fail to anticipate possible uses

Sometimes this works out well and sometimes it does not

Research: What happens when proxies go out into the wild

- Proxies are relatively easy to use and build
- They are rightly viewed as more

WARNING: Cape does not enable wearer to fly

- Many researchers use them for their papers
- Proxy authors often fail to anticipate possible uses

Sometimes this works out well and sometimes it does not

Proxy apps are models

Models are easily misused

- "To make LULESH go through the polyhedral compilation procedure, we modified LULESH by resolving all indirect array accesses. Although doing this oversimplified LULESH, it allows us to study the energy and time relationship of polyhedral compilation techniques with LULESH."
- Many papers use skeleton benchmarks (MPI only) out of context and draw networking conclusions
- Vendor reports often contain similar errors to research codes

An understanding of what you are using and why its important are essential when using proxy apps

Proxy app authors are not blameless

We have made some of these mistakes ourselves

- Proxies are often originally intended for internal use
- Better documentation that is easier to digest is usually needed to help guide researchers
- We need to be more clear what is a proxy and what is a benchmark
- Writing code is fun.
  Writing documentation is not.

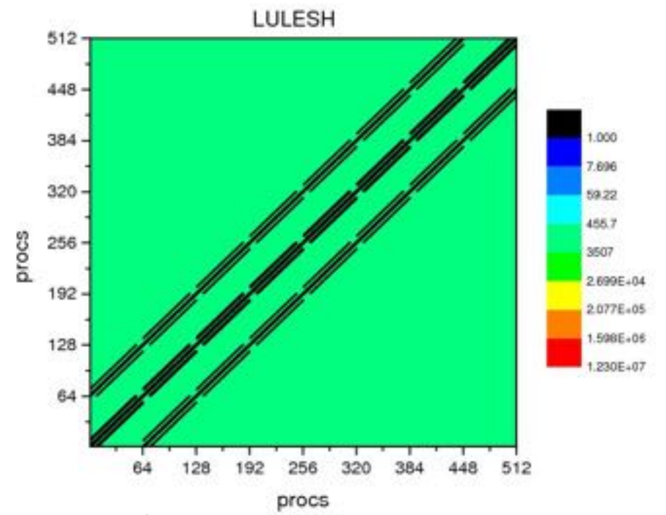


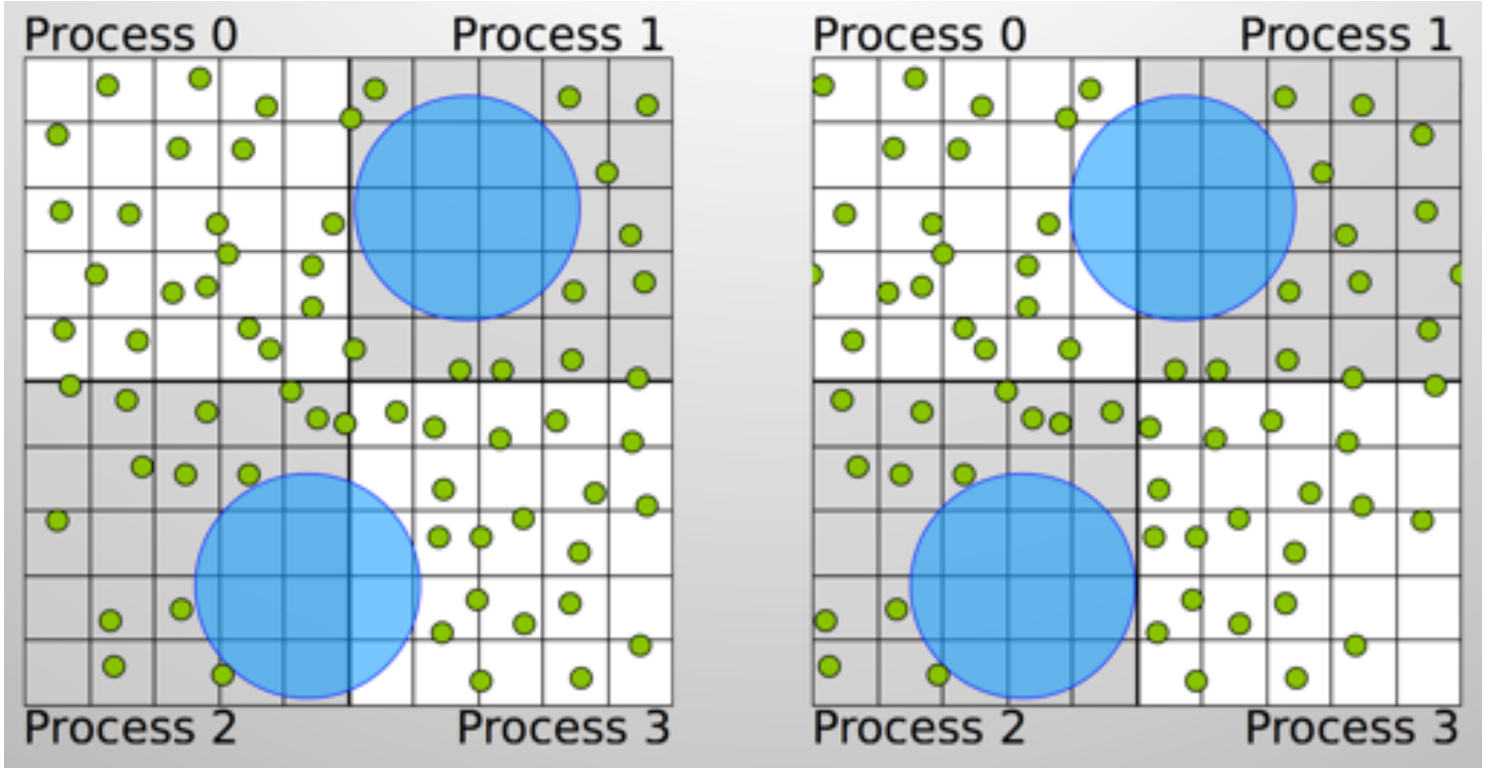
Image from a DOE website showing LULESH communication pattern. LULESH is good for many things, but is **not** representative of unstructured codes communication patterns.

Sometimes we succeed

- Implementing LULESH in the domain specific language Liszt helped us identify limitations and missing language features
- AMD paper on multi-level memory taught us that number of accesses to memory is not all that matters. Understanding cache behavior does too
- HPCToolKit paper from Rice University showed a malloc/free issue in LULESH about the same time we discovered it ourselves

These efforts featured connections and collaborations between proxy authors and the researchers using the proxy

The CoMD proxy app was modified to study load imbalance



Spherical voids are randomly introduced during problem setup to form “Swiss cheese”.

Adding a center of mass velocity makes load imbalance dynamic.

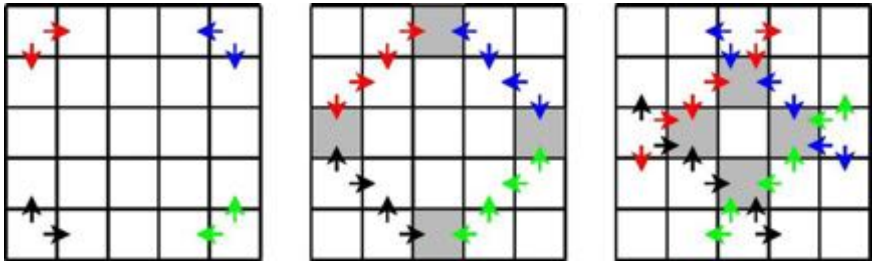
Small modifications to existing proxies can allow exploration of questions the proxy author didn’t intend

Why you should care about proxy apps #4

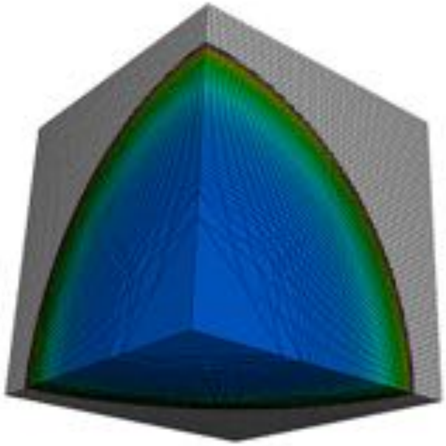
Proxy apps support rapid prototyping for proposed design changes

Prototyping successes with proxies at LLNL

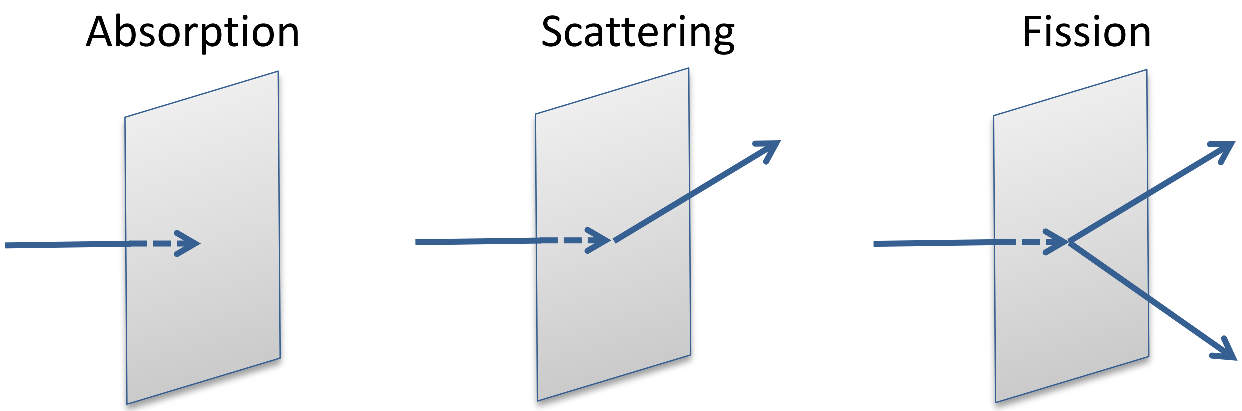
- Kripke
  - Tested out Tloops constructs
  - Led to RAJA ForALLn development
  - Code changes being ported back into Ardra



- LULESH
  - Learned the significant codes of malloc/free on some systems
  - Led to adoption by Ares of tcmalloc on BG/Q



Quicksilver is a proxy for Mercury.
Both calculate Monte Carlo Particle Transport



- Particles interact with matter by a variety of "reactions".
- The probability of each reaction and its outcomes are captured in experimentally measured "cross sections". (Latency bound table lookups)
- Follows many particles (millions or more) and uses random numbers to sample the probability distributions. (Very branchy, divergent code)
- Particles contribute to diagnostic "tallies". (Potential data races)
- The result is a statistically correct representation of the physical system.

Quicksilver and Mercury are hostile to the typical GPU fine-grained threading approach

- loop over cycles (time steps)
  - cycle_init
    - source in new particles
    - population control
  - cycle_tracking
    - loop over particles
      - until census
        - find distance to census (end of time step)
        - find distance to material boundary (mesh facet)
        - find distance to collision (reaction)
        - select reaction and update particle
  - cycle_finalize

This is 1000s (or 10,000s) of lines of code

Majority of cross section look ups are in here

Quicksilver and Mercury are hostile to the typical GPU fine-grained threading approach

- loop over cycles (time steps)
  - cycle_init
    - source in new particles
    - population control
  - cycle_tracking
    - loop over particles
      - until census
        - find distance to census (end of time step)
        - find distance to material boundary (mesh facet)
        - find distance to collision (reaction)
        - select reaction and update particle
  - cycle_finalize

Coarse threading strategy:

- Each thread gets its own "vault" of particles
- Tally and buffer data structures are replicated to avoid races
- Works great on CPU platforms!

This is 1000s (or 10,000s) of lines of code

Majority of cross section look ups are in here

How do you write this code for GPUs?

Quicksilver and Mercury are hostile to the typical GPU fine-grained threading approach

- loop over cycles (time steps)
  - cycle_init
    - source in new particles
    - population control
  - cycle_tracking
    - loop over particles
      - until census
        - find distance to census (end of time step)
        - find distance to material boundary (mesh facet)
        - find distance to collision (reaction)
        - select reaction and update particle
  - cycle_finalize

This is 1000s (or 10,000s) of lines of code

Coarse threading strategy:
- Each thread gets its own "vault" of particles
- Tally and buffer data structures are replicated to avoid races
- Works great on CPU platforms!

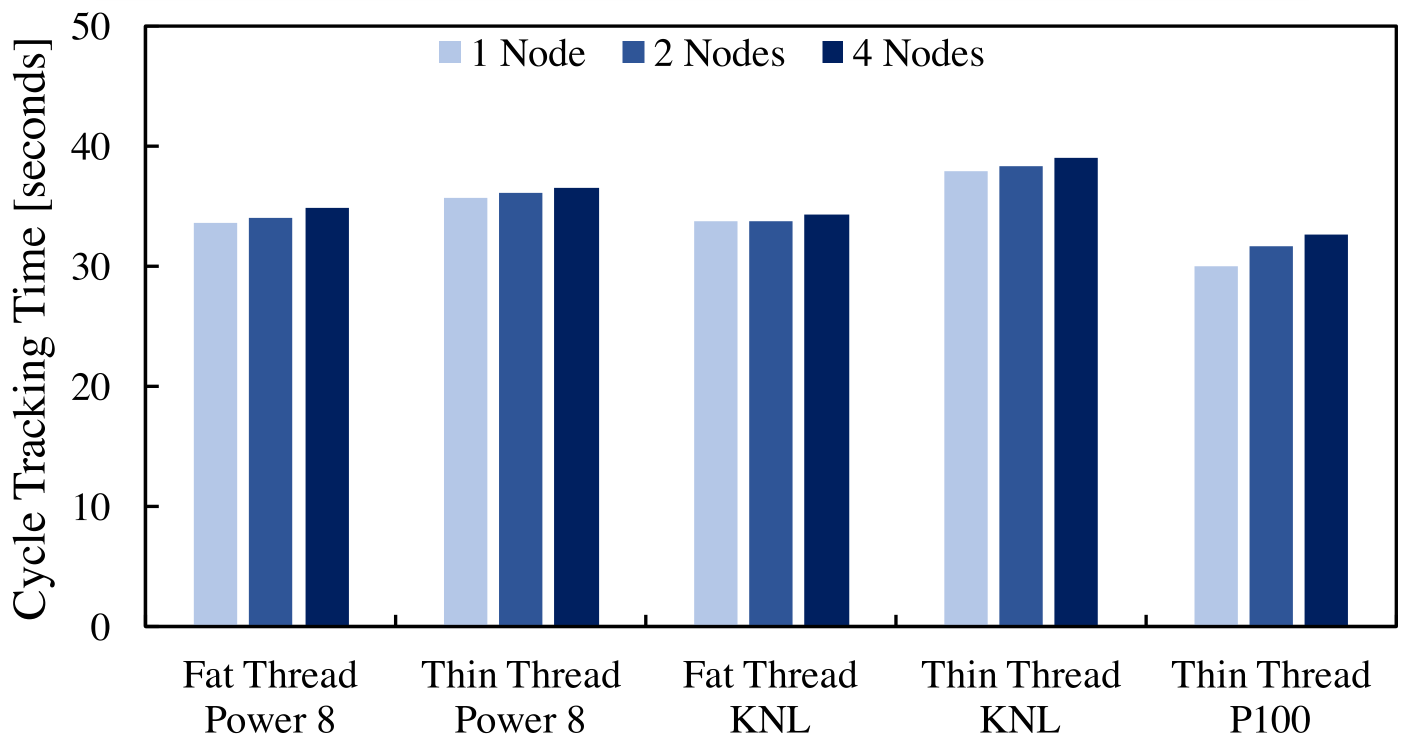
Majority of cross section look ups are in here

Make this a kernel!

Can this "Big-Kernel" approach possibly perform well?

Will big kernel work?

Cycle tracking time for a reaction dominated problem
(lower is better)



In spite of adverse algorithmic characteristics, we are hopeful that Mercury will perform equally well on GPUs as CPUs. A potential 3-5x speedup compared to CPUs only

Common issues with proxies as design studies

- Forgetting that the proxy app is not a real application
- Tendency to optimize the proxy app instead of the application
- Using constructs, abstractions, language features, etc., that are cannot be integrated with the parent application

Ardra is LLNL's next generation deterministic transport code

- Solves the neutral particle Boltzmann Transport Equation in 1D, 2D, and 3D
- Solution is neutron energy, direction of flight, and spatial distribution
  - We have MPI parallelism in each of these dimensions
- 4 major kernels
  - LTimes, LPlusTimes, Sweep, Fission
- LTimes and LPlusTimes kernels are streaming kernels of the form phi(nm,g,z) += ell(nm,d) * psi(d,g,z)
  - z is stride 1
  - ell(nm,d) is invariant in inner loop
- Previous Kripke studies found LTimes & LPlusTimes (and all other kernels) are memory bound on GPU, but not elsewhere (less than 25% of B/W).

Intel VTune suggests Ardra is not B/W bound

DRAM B/W peaks
at 50-60 GB/sec
or ~50% of
130 GB/sec peak

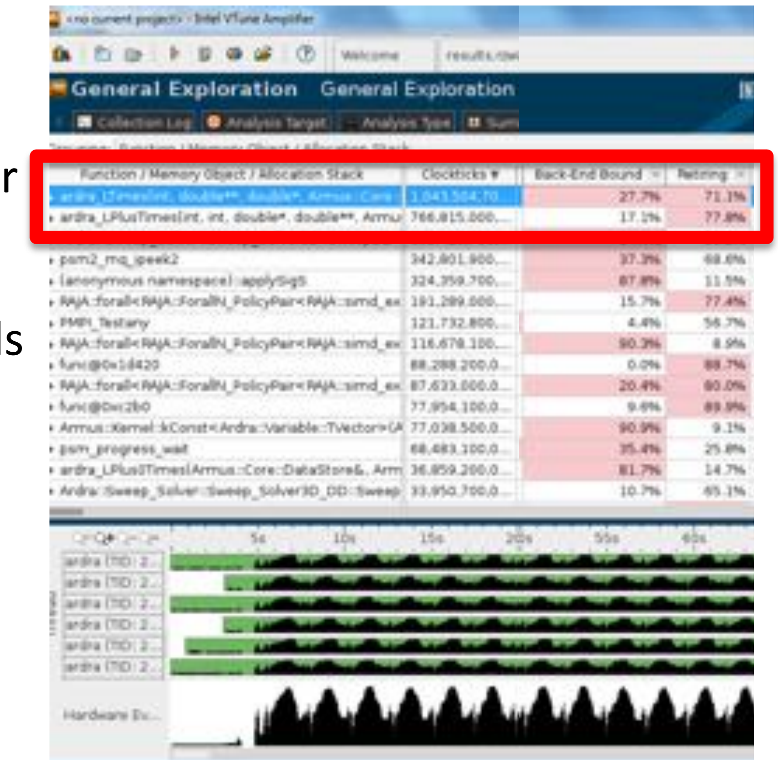
Intel VTune suggests Ardra is not B/W bound

This diagnostic is confusing and not consistent with developer intuition

FPU usage is uniformly low

Root cause analysis requires deduction and guesswork

- Low flop rate throughout code (2.1% of peak)
- Nearly all flops are scalar instructions
  - VTune reports 12:1 ratio for scalar to vector flops
  - Allinea Map reports 6:1 ratio
  - VTune shows no vectorization in key kernels
- VTune shows high “Retiring” in LTimes and LPlusTimes
- Intel optimization manual says high retirement
  means you should vectorize
  - Allows more operations to complete per instruction



After a lengthy saga, vectorization improved performance
(except when it didn't, and not the way we expected)

- RAJA reliance on lambdas is inhibiting vectorization on icc
  - Refactoring and refinements are in progress
  - Manual unrolling and intrinsics were used to obtain vectorized code
- Performance improved:
  - Kernel times are significantly faster
  - But DRAM B/W changed very little
- Performance impact depends on problem size. What's going on?
  - Small problem now operates at L2 B/W
  - Loads from L2 don't show up in DRAM B/W
  - Would blocking for L2 improve big problem performance?

Small Problem	scalar	vector
LTimes	15.53 sec	6.67 sec
DRAM B/W	60 GB/sec	68 GB/sec

When vectorized, pulls full L2 B/W

Big Problem	scalar	vector
LTimes	84.75 sec	82.43 sec
DRAM B/W	117 GB/sec	119 GB/sec

ECP Proxy App Project: Objectives and Scope

- Assemble and curate a proxy app suite that represents the most important features (especially performance) of exascale applications
- Improve the quality of proxies created by ECP and maximize the benefit received from their use.
  - Set standards for documentation, build and test systems, performance models and evaluations, etc.
- Collect requirements of app teams. Assess gaps between ECP applications and proxy app suite. Ensure proxy suite covers application motifs and requirements
- Coordinate use of proxy apps in the co-design process. Connect producers to consumers. Promote success stories and correct misuse of proxies.

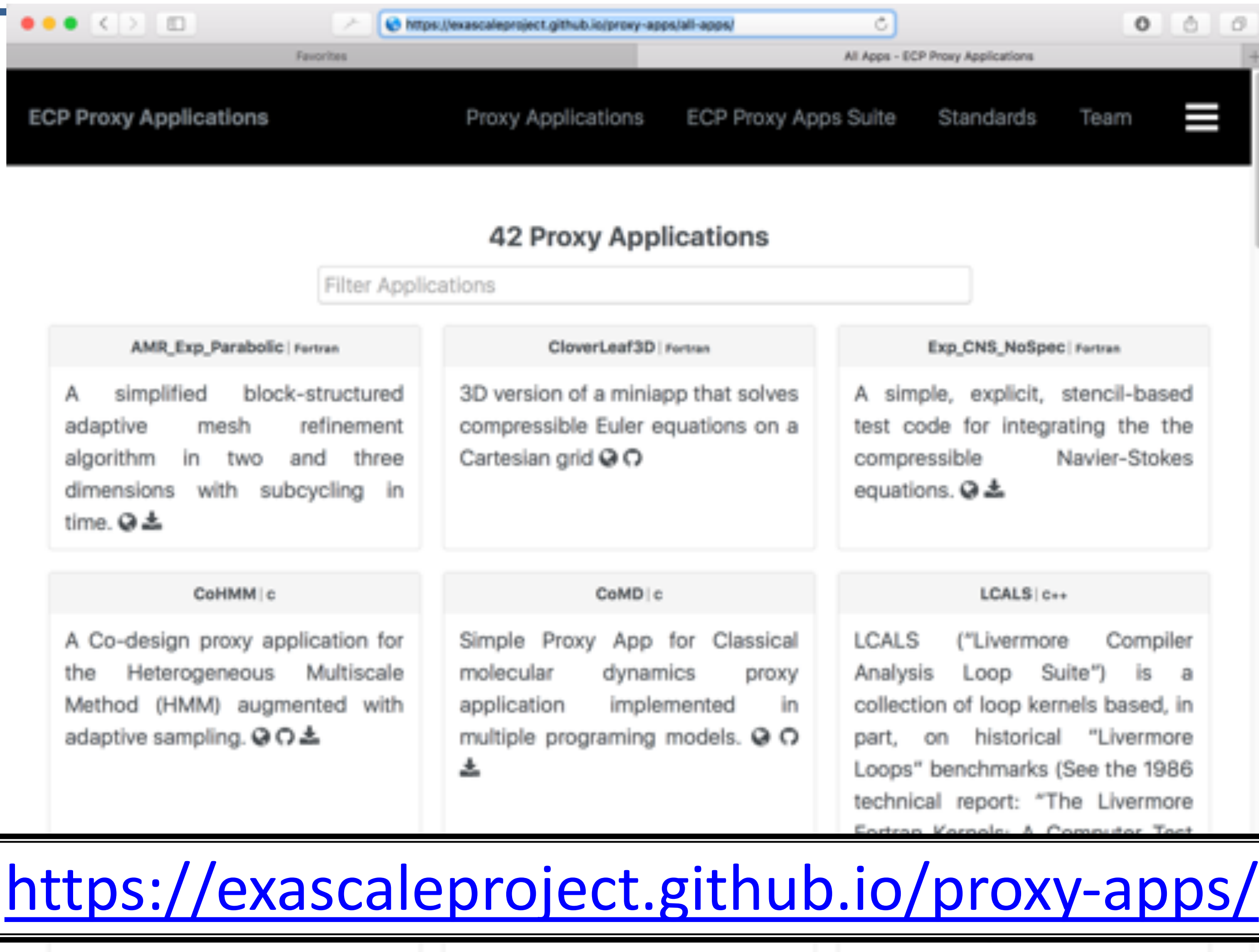
ECP Proxy App Project: Links to Other Projects

- **Application Assessment Project:** Cooperatively assess and quantitatively compare applications and proxy apps.
- **Design Space Evaluation Team:** Will need proxies specially adapted for hardware simulators.
- **Path Forward Vendors:** Evaluate needs and provides proxies & support. Review proxy app usage and results.
- **Application Development Projects & Co-Design Centers:** Producers of proxy apps. Close the loop with lessons learned from proxies.
- **Software Technology Projects:** Consumers of proxy apps. Use proxies to understand app requirements and to test and evaluate proposed ST offerings.

ECP Proxy App Project: Development Plan

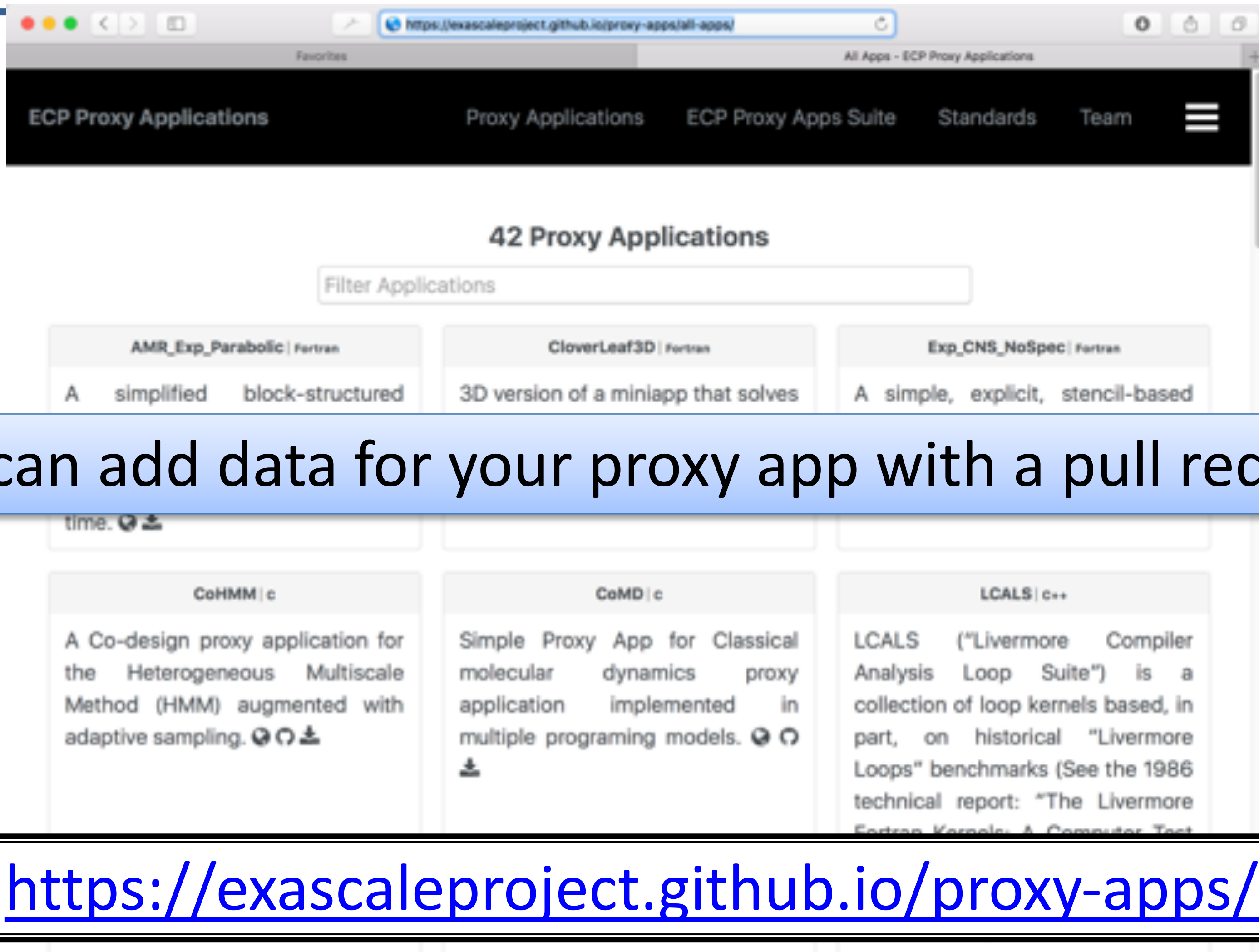
- Release updated versions of the proxy app suite every six months. This cadence allows for improved coverage and changing needs while maintaining needed stability.
- Annually update guidance on quality standards. Increase rigor of standards.
- Meet with each application project to maintain a catalog of their requirements, proxies, and key questions for which they are seeking assistance.
- Publish annual proxy app producer report with requirements and assessment of proxies in comparison to parent apps.
- Publish annual proxy app consumer report with success stories, surveys of how proxy consumers are using proxies, and plans to satisfy any unmet needs.

We are creating a proxy app portal



https://exascaleproject.github.io/proxy-apps/

We are creating a proxy app portal



You can add data for your proxy app with a pull request

https://exascaleproject.github.io/proxy-apps/

Spack Integration

A simple way to get the ECP Proxy suite

- Spack: A flexible package manager endorsed by ECP
- **Aim:** getting the Proxy Suite with one command
  `$ spack install --source ECP-Proxy-Suite@1.0`
  adding (`--fake`) will not build it.
- Missing Features:
  - Support for installing source – CJ submitted pull request #4102
  - Support for Meta packages (a package, which pull in many sub packages, but doesn't install anything itself)
  - Support for handing over flags to make (e.g. "`make CC=icc`") – CJ submitted pull request #4704
- First 40+ proxy apps have spack packages (spackages): https://github.com/hfinkel/proxy-apps-spack

We are adding proxy apps to the LLVM test suite

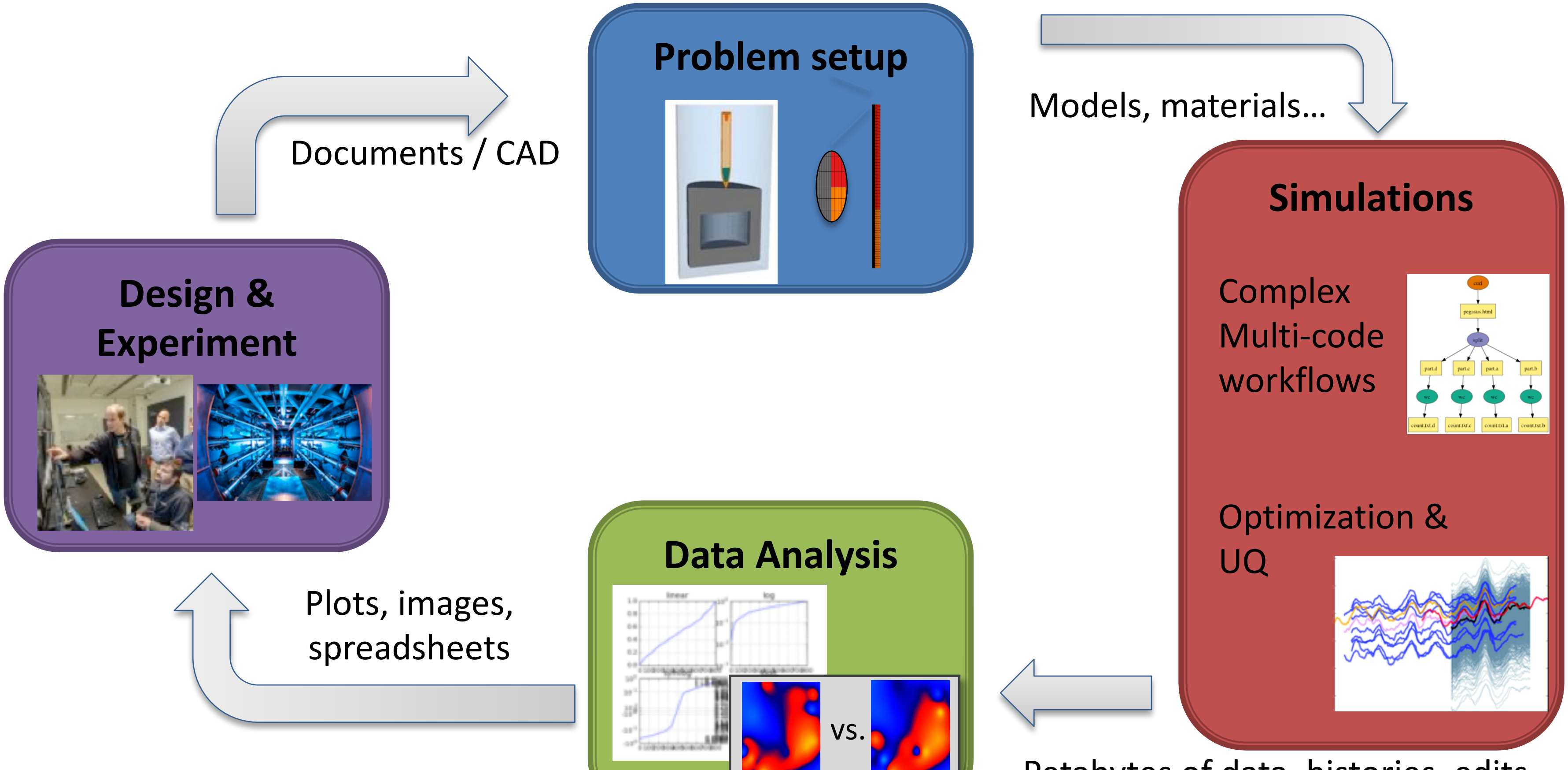
- LLVM's test suite is a compile-independent framework for tracking correctness and performance of applications over time.
- Providing a uniform way to compile **and run** our proxy apps allows ST projects to test on applications relevant to our workloads.
- 12 added so far (https://github.com/hfinkel/test-suite-with-proxy-apps)

XSBench is part of LLVM's test suite. More to come!

Performance Improvements - Execution Time	Δ (B)	Baseline	Current	σ (B)	Δ	σ
SingleSource/Benchmarks/McGill/chomp	-18.57%	1.1200	0.9120	0.0047	0.00%	0.0047
MultiSource/Benchmarks/FreeBench/analyzer/analyzer	-10.71%	0.1120	0.1000	0.0020	0.00%	0.0020
MultiSource/Benchmarks/Fhourstones/fhourstones	-10.43%	1.1120	0.9960	0.0146	-0.40%	0.0146
SingleSource/Benchmarks/Misc/flops	-1.10%	6.8840	6.8080	0.0217	-1.22%	0.0217
MultiSource/Benchmarks/DOE-ProxyApps-C/XSBench/XSBench	-1.08%	3.6960	3.6560	0.0067	0.00%	0.0067

With great power comes great complexity

Workflow is the next frontier in co-design



Dan Laney, Workflow Project Lead

More powerful systems enable UQ and large ensemble calculations. Co-design will ensure future systems support the entire simulation workflow

Takeaways

- Proxy apps are tools that can be used to learn something
  - But they are only models for the parent app
  - Consider learning more about the parent app and how the proxy is different
- Proxy apps have many use cases that you probably care about
  - Procurement, co-design, research, prototyping
- Proxies and benchmarks are different and should not be confused
- The ECP Proxy App Project was created to help producers and consumers successfully develop and use proxies

If all else fails, contact the developer!

Acknowledgements

- Ian Karlin, Doug Kothe
- Proxy App Team: Abhinav Bhatele, Jeanine Cook, Hal Finkel, Nikhil Jain, Christoph Junghans, Peter McCorquodale, Bronson Messer, Tiffany Mintz, Shirley Moore, Robert Pavel, Courtenay Vaughan
- Quicksilver Team: Ryan Bleile, Patrick Brantley, Shawn Dawson, Scott McKinley, Matt O'Brien
- Ardra Team: Teresa Bailey, Adam Kunen, Bujar Tagani, John Loffeld
- ProTools/AAPS: Matt Legendre, David Poliakoff
- *This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering and early testbed platforms, in support of the nation's exascale computing imperative.*

Lawrence Livermore
National Laboratory