Everything You Always Wanted to Know About Proxy Apps
(But Were Afraid to Ask)

ECP Annual Meeting 2018

David Richards (LLNL)
Christoph Junghans (LANL)
Hal Finkel (ANL)
Jeanine Cook (SNL)

Knoxville, TN
February 8, 2018

Prepared by LLNL under Contract DE-AC52-07NA27344. LLNL-PRES-746247

Agenda

- (20 min) David Richards — Project Overview
- (10 min) Christoph Junghans — Web Site & Spack
- (15 min) Hal Finkel — Proxy App Use Cases
- (15 min) Jeanine Cook — Proxy Characterization
- (5 min) David Richards — Wrap Up

Please feel free to ask questions at any time
(don’t be afraid)

ECP Proxy App Project: Objectives and Scope

Why does this project exist?

- 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.

Proxy App Project Team

- ANL: Hal Finkel, Tom Uram, Summer students
- LANL: Christoph Junghans, Robert Pavel
- LBNL: Peter McCorquodale
- LLNL: David Richards, Abhinav Bhatele, Nikhil Jain
- ORNL: Bronson Messer, Tiffany Mintz, Shirley Moore
- SNL: Omar Aaziz, Jeanine Cook, Courtenay Vaughan

Proxy applications are models for one or more features of a parent application

- Proxy apps omit many features of parent apps
- 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

Why create a proxy for your application?

- Application cannot be shared with collaborator
  – OUO, Export Control, Classified, etc.
- Application is too large and complex for collaborator to understand or use
- 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 close collaboration with a code team.

Vendors use proxy apps for co-design

- Proxy apps communicate DOE concerns
  - PathForward, CORAL NRE, etc.
  - Vendors are asking us for codes
- The proxy app team has a dedicated POC for each PathForward vendor
  - AMD Jeanine Cook
  - Cray Christoph Junghans
  - HPE David Richards
  - IBM Shirley Moore
  - Intel Hal Finkel
  - Nvidia Tom Uram

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

Proxy apps play a prominent role in procurement benchmarks

- CORAL: Nekbone, LULESH, SNAP, miniFE, XSBENCH
- APEX: MILC, miniDFT, miniPIC, PENNANT, SNAP, UMT
- CORAL 2: Nekbone, Kripke, Quicksilver, PENNANT
- Not to mention numerous microkernels, and example suites
  - DGEMM, IOR, Mdtest, RAJA Performance Suite, Stream, Stride, etc.

How are proxy apps different from benchmarks

Proxy Application

- Generic modeling and co-design tool
- Unconstrained observables
- Intended to be modified (within limits)
- Flexible problem definitions

Benchmark

- Measurement of a specific observable
- Defines a Figure of Merit
- Defines rules and restrictions
- Frozen code version
- Constrained problem definition

Proxy apps only become benchmarks
when sufficient definitions and constraints are applied.

How to solve problems with proxy apps

- Wrong question: “Which proxy app should I use”
- Right Question(s):
  - What is the problem/question to be investigated
  - What are the resources? Will this be run on a simulator? Real hardware?
  - What is a reasonable model to address the question within the available resources?
  - What other investigations are necessary to establish reasonable model parameters?
- Example: Use a communication simulator to examine on-node vs off-node traffic
  - Look at real applications, especially in a strong scaling regime, to find parameters of communication models that are both realistic and representative of a real world use case.
- Bad example: Run an MD code (or proxy) with many particles per rank using a communication simulator.
  - Communication fraction is so low that improvements won’t impact overall performance.

Proxies are models.
This is a modeling exercise

Reporting results obtained with proxy apps

- Bad: table or graph with (proxy) app names and “performance” results
- Bad: optimize the proxy instead of the application
- Good:
  - Specify the problem parameters for each (proxy) app
  - Explain why the app should (or should not) be sensitive (performance or otherwise) to the topic being explored.
  - Explain why the apps were chosen and why they advance understanding of the topic.
    - Bad: These are DOE proxies that I found on a web site

WARNING: The default problems for many proxy apps are little more than “smoke tests”.
They have little relevance to the production workload

The ECP Proxy App Suite (v1.0)
New releases planned every 6 months

- AMG
- CANDLE Benchmarks
- CoMD
- Ember
- Laghos
- MACSio
- miniAMR
- miniFE
- miniTri
- Nekbone
- SW4lite
- SWFFT

You can help!!
We are looking for proxies created by ECP projects

We may define other specialized collections of proxies

Takeaways

- Proxy apps are models
  - They only represent parts of the parent app
- Proxy apps have many use cases
  - Prototyping, co-design, procurement, etc.
- Using a proxy app is a modeling exercise
  - Proxies and benchmarks are different and should not be confused
- The proxy app project will help the external community use proxies successfully

If you are not sure about if you are using a proxy correctly, e-mail the developer.

Questions?

ECP Proxy Apps
Website and Spack

Christoph Junghans & Robert Pavel (LANL)

08-Feb-2018

LA-UR 18-20872

EXASCALE COMPUTING PROJECT

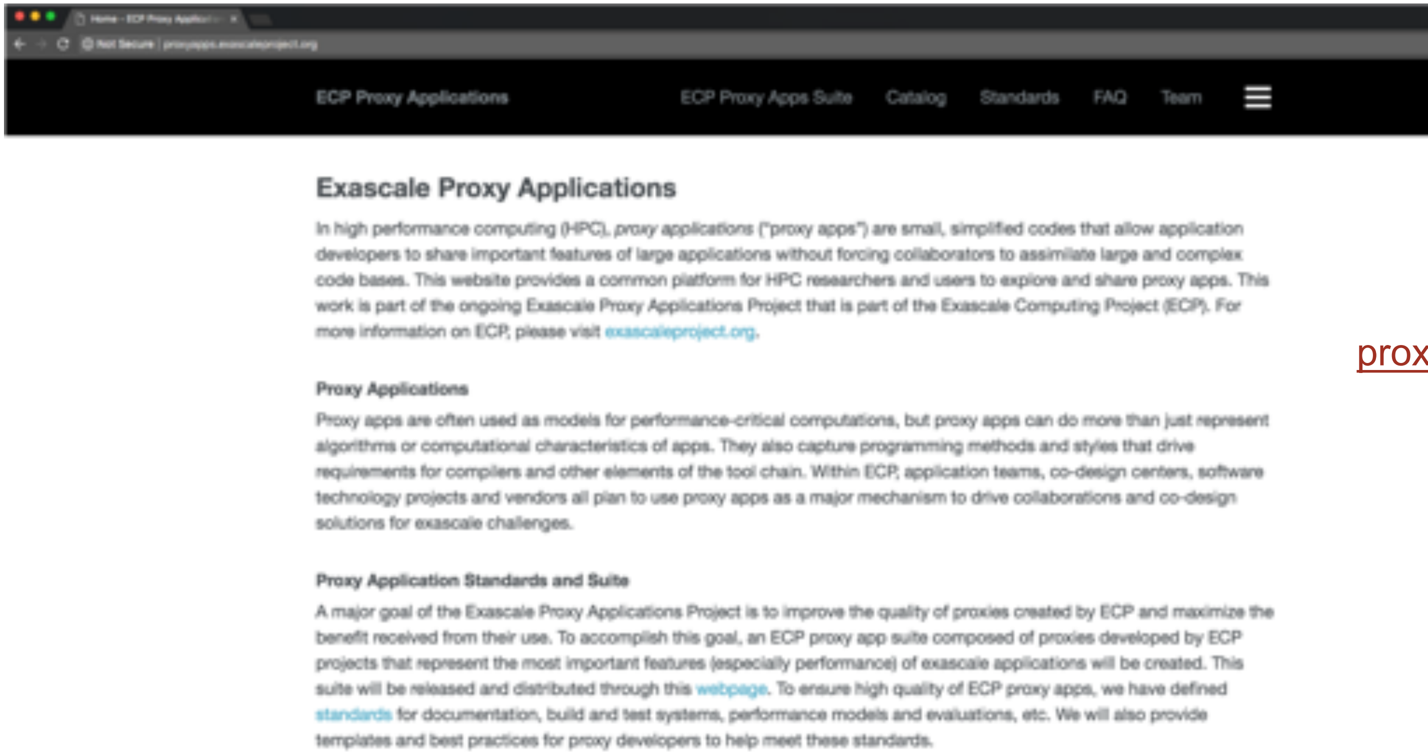
National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

Proxy App Website

A page to find them all

http://proxyapps.exascaleproject.org/



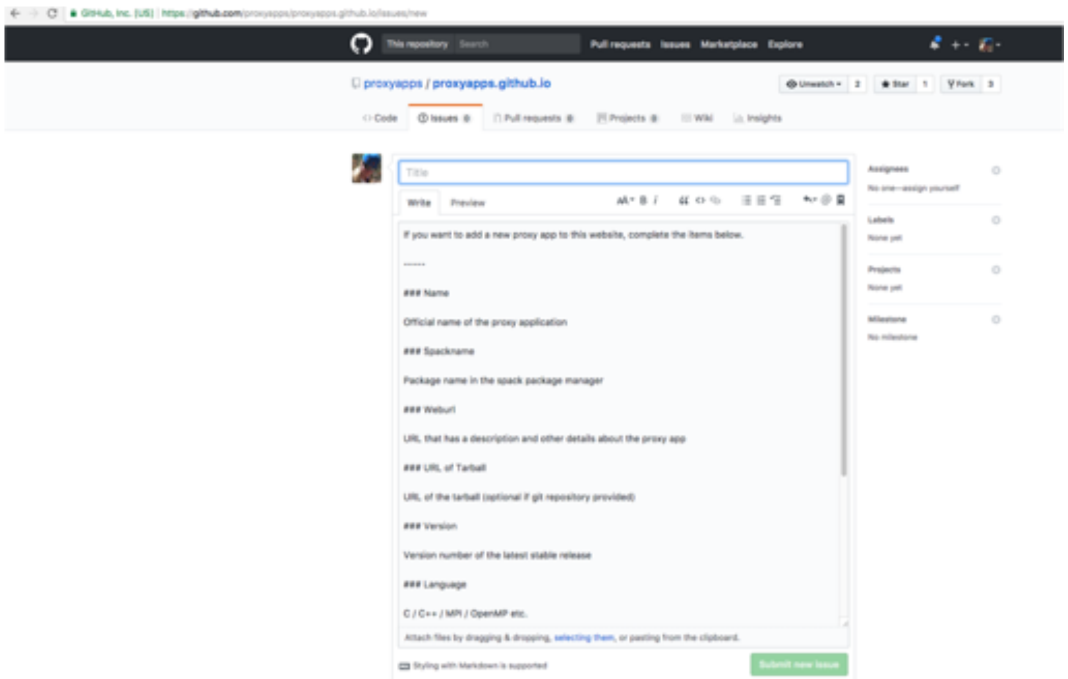
Something missing?

File an issue:
https://github.com/proxyapps/proxyapps.github.io

Proxy App Website

Add a proxy app

Under FAQ -> “Getting a proxy application added to this website” -> https://github.com/proxyapps/proxyapps.github.io/issues/new



Proxy Suite in Spack

A command to get them all

- Spack: A flexible package manager endorsed by ECP
- Get the Proxy Suite with one command
  `$ spack install --source ECP-Proxy-Suite@1.0`
  (might take a while!)
- How to get Spack (see SPACK breakout session for more details)
  ```
  $ git clone https://github.com/spack/spack.git
  $ cd spack/bin
  $ ./spack install <Some Package>
  ```
- See the packages of the proxy suite:
  `$ spack list -t ecp-proxy-app`

Random Tricks for Spack

Things you might need along the way

- Using your system libraries instead of building them:
  http://spack.readthedocs.io/en/latest/build_settings.html#external-packages
  (need to edit your `~/.spack/packages.yml`)
- Build proxy app with certain compiler:
  `$ spack install <app> %gcc@7.0`
- Build proxies with certain compiler flags:
  `$ spack install <app> cflags='-O2 -g'`
- Build proxies with certain MPI:
  `$ spack install <app> ^openmpi@4.0`
- Don't build proxy app, just get me the code:
  `$ spack install --source --only package <app>`

Example Spack package.xml

Also see the Spack documentation

Under FAQ -> "How do I download the ECP Proxy Apps suite?" -> http://proxyapps.exascaleproject.org/download/

Spack-related tips and tricks

If you want to assist spack in finding compilers, read this.

If you want to use a system installed MPI and prevent spack from building it, read this. A sample `~/.spack/packages.yaml` file is below:

```
packages:
  all:
    providers:
      mpi: [mvapich2]
  mvapich2:
    paths:
      mvapich2@system: <path to mvapich2>
    version: [system]
    buildable: False
```

If you do not want a long hash in the installed proxy apps' directories, read this. A sample `~/.spack/config.yaml` file is below:

```
config:
    install_path_scheme: '${ARCHITECTURE}/${COMPILERNAME}-${COMPILERVER}/${PACKAGE}-${VERSION}'
```

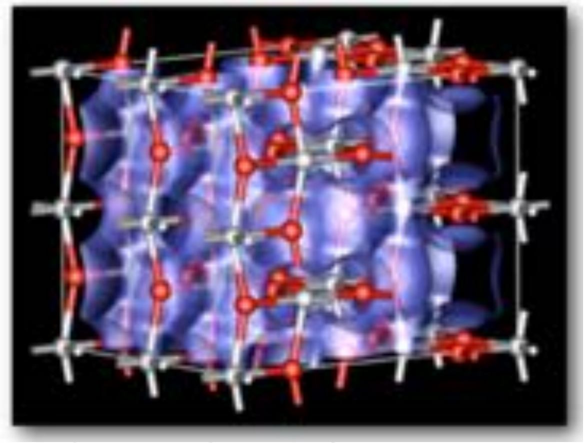
Something wrong with Spack?

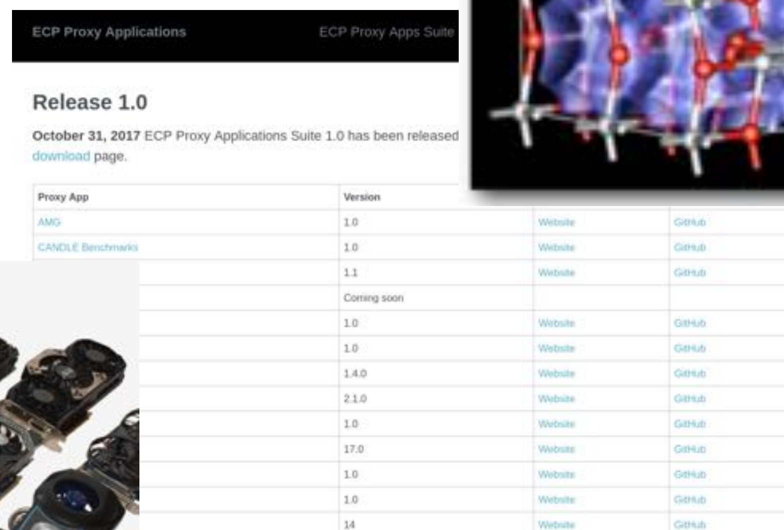
File an issue:
https://github.com/spack/spack

Questions?

Proxy Apps – How Do We Use These Things?

- ✓ Proxy apps as prototype for application design change
- ✓ Proxy apps and ST projects
- ✓ Proxy apps and hardware evaluation





ECP Proxy Applications ECP Proxy Apps Suite

Release 1.0

October 31, 2017 ECP Proxy Applications Suite 1.0 has been released

download page.

Proxy App	Version		
AMG	1.0	Website	GitHub
CANDLE Benchmarks	1.0	Website	GitHub
	1.1	Website	GitHub
	Coming soon		
	1.0	Website	GitHub
	1.0	Website	GitHub
	1.4.0	Website	GitHub
	2.1.0	Website	GitHub
	1.0	Website	GitHub
	17.0	Website	GitHub
	1.0	Website	GitHub
	1.0	Website	GitHub
	14	Website	GitHub

(picture from: https://cryptosrus.com/best-gpu-for-mining/)

Modeling Changes to Applications

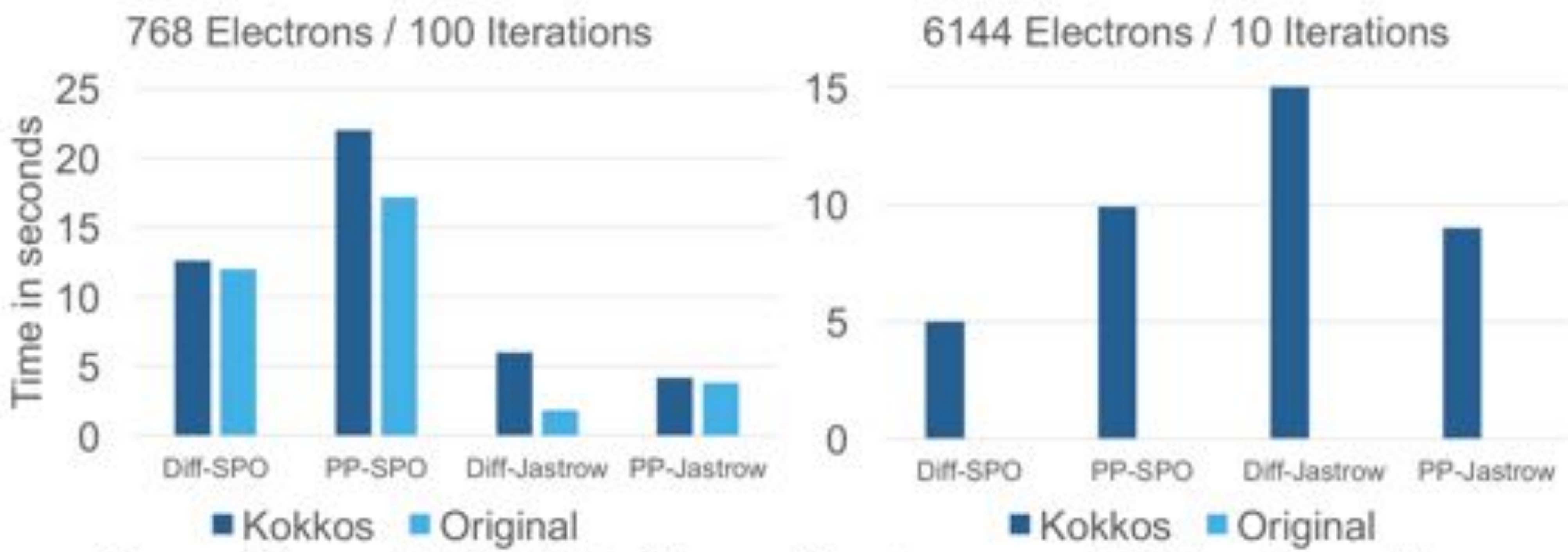
QMCPACK http://qmcpack.org, https://github.com/QMCPACK

QMCPACK is a production level open source QMC code for computing the electronic structure of atoms, molecules, and solids.

- Used for INCITE, CORAL, OLCF CAAR & ALCF ESP.
- C++11, MPI, OpenMP, CUDA, Boost, Parallel HDF5, libXML2.
- O(400K) code lines, estimated O(100K) for core functionality.
- 10+ year history, 7500+ commits, 35 contributors.
- Today: Largely incompatible OpenMP & CUDA execution pathways.

Performance Comparisons: *Many Core Intel KNL*

Performance comparable to original reference code for several kernels



Bowman: 68 cores Intel KNL, Using 64 cores, 2 threads per core, 1 / 32 threads per walker

Exploring Kokkos using miniQMC (work exploring OpenMP offloading also in progress).

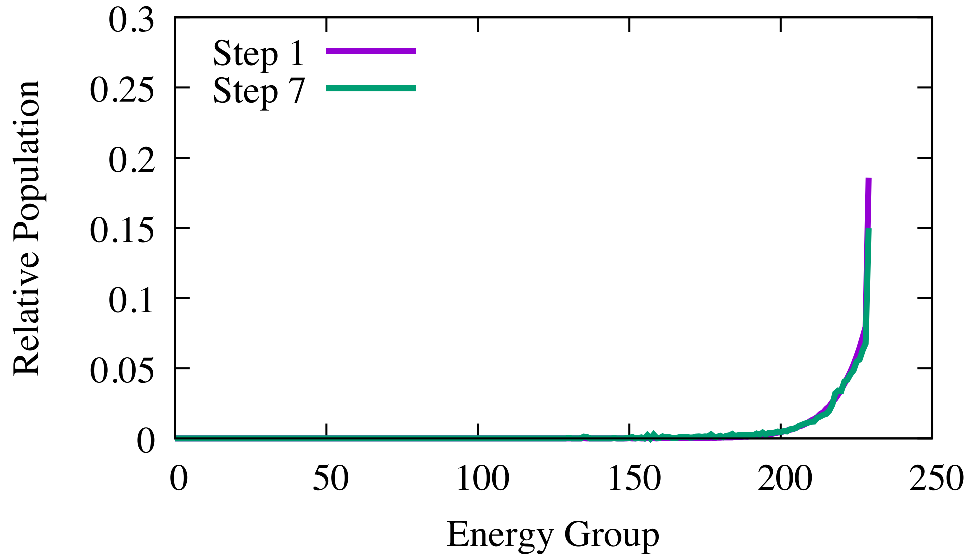
Creating Quicksilver (proxy for Mercury) required modeling choices

Proxy apps are models for one or more aspects of their parents

- Three specific uses:
  - A nimble prototype code for testing design or refactoring options for Mercury
  - An open source vehicle for co-design with outside partners
  - A benchmark code to replace our previous Monte Carlo benchmark code
- Overall goal was to approximate the overall application performance of Mercury
  - Control flow is dominated by branching due to the random sampling of reactions.
  - Memory access patterns associated with reading cross section tables tend to be latency-bound, small memory loads that are difficult or impossible to cache or coalesce.
  - Domain decomposition and internode communication to handle large problems.
- Major data structures intentionally similar to Mercury
- Flexible inputs to represent multiple common use modes

It is essential to to identify the key features of the parent app the proxy is intended to represent and include faithful models of those features

Is Quicksilver a good representation of Mercury? Initially no, but...
Deleting low-weight particles solves the problem



There is no such capability in the parent application,
but it makes Quicksilver a better model for Mercury

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

"Fat" 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?

To test big-kernel we wrote a proxy app for our proxy app

- Quicksilver was more complicated than we wanted to port GPU
  - MPI, variable particle count, etc.
- Quicksilver_lite is even more approximate than Quicksilver
  - Zero-D mesh, very simplified physics
- Quicksilver_lite maintains features most likely to impair GPU performance
  - Random table look-ups
  - Call stack depth in nuclear data look-ups
  - Branchy control flow and divergence

QS_lite run times (lower is better)

	Initialize	Compute
P8 CPU (10 threads)	0.27 sec	1.25 sec
P8 CPU (40 threads)	0.45 sec	0.72 sec
P-100 GPU	0.26 sec	0.45 sec

QS_lite provided our first evidence that the big-kernel approach might actually work

Mercury is employed for a very wide variety of problems

No single sample problem will represent all use cases

- By changing problem inputs we can adjust:
  - Fraction of particles that reach census
  - Ratio of facet crossings to reactions
  - Relative probabilities of different reaction types

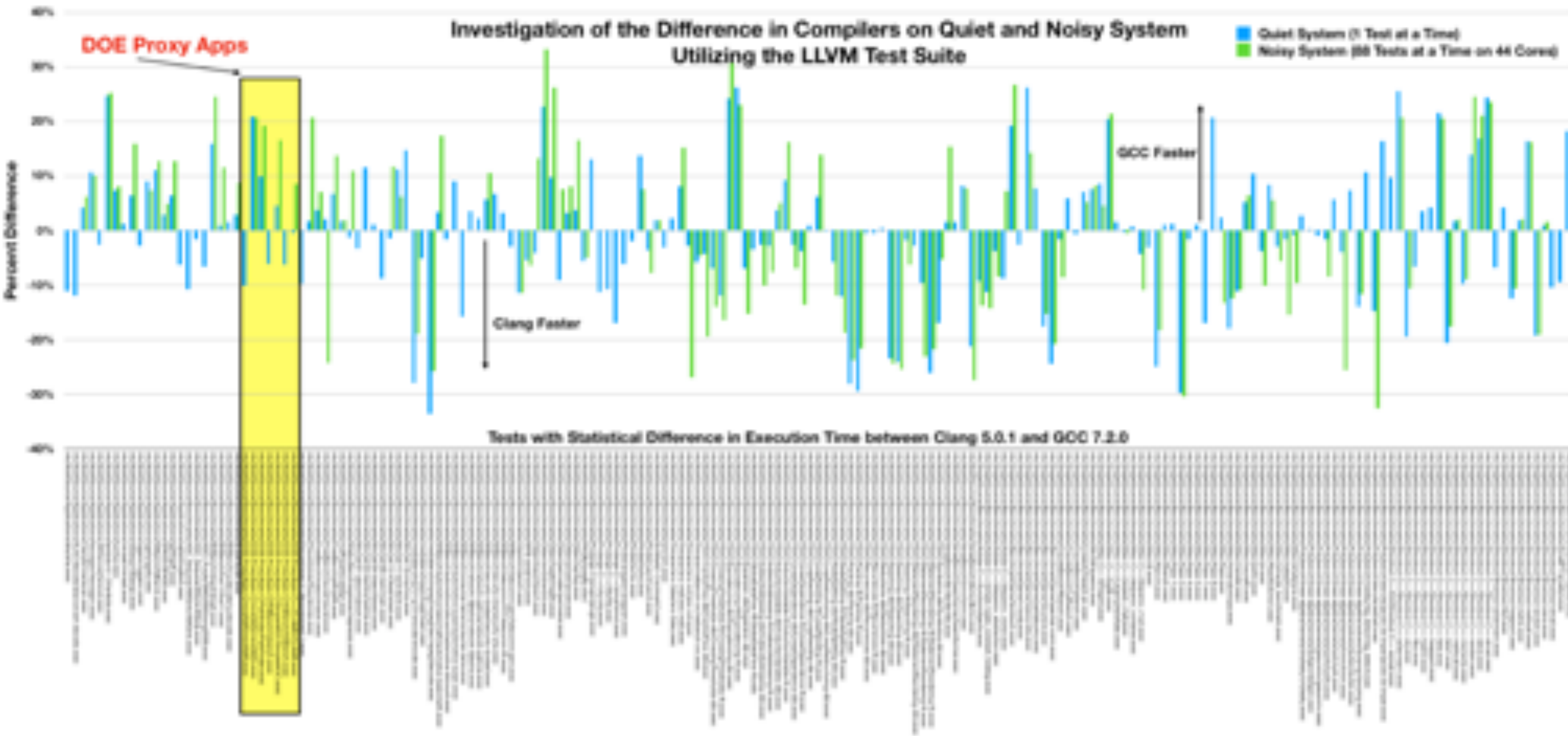
	Fat (CPU) seg/sec	Thin (GPU) seg/sec
Reaction Dominated	8.52e+06	1.15e+07
Balanced	1.35e+07	7.50+e06
Facet Dominated	2.24e+07	2.90+e07

Higher is better

GPUs and CPUs are similar, but with difference performance sensitivities.
GPUs are slowest in balanced case. Perhaps due to highest divergence?

Uses by software-technology projects

Proxy Apps in the LLVM Test Suite



COMPUTING PROJECT

Initial Results of ROSE on ECP Proxy Apps

ROSE: 8 passing 100% out of 11 applications from spack/spack:
1 application had build system issue (TBD)

#	application	passes	failures	% passing
1	amg	80	0	100%
2	ECP CANDLE Benchmarks	(TBD)	(TBD)	(TBD)
3	CoMD	20	0	100%
4	Laghos	3	0	100%
5	MACSio	4	7	36%
6	MiniAMR	18	0	100%
7	MiniFE	7	0	100%
8	MiniTri	5	1	83%
9	Nekbone	60	6	91%
10	SW4lite	24	0	100%
11	SWFFT	5	0	100%
12	XSBench	6	0	100%

Our experiences have informed our documented best practices:
http://proxyapps.exascaleproject.org/standards/

- What does the proxy represent?
- How do you run and what do the options mean?

- How can we tell if the computation is sufficiently correct?
- How can we relate the performance to improvements in the real applications?

Three Expected Running Modes to be specified

Required: A quick test to verify correct compilation and, if applicable, track kernel performance. This should have an execution time on the order of seconds but can be larger if necessary. See Verification of Correctness below.

Recommended: A running mode (weak and strong scaling) representing a current workload.

Recommended: A running mode (weak and strong scaling) representing an exascale workload.

General Documentation

General Description of Proxy Application

Required: A general description of what the proxy is and which algorithms it comprises/comp ... represents.

Required: Proxy apps should have documentation of the general scope of the proxy app and ... the parent application correctly.

Intent and Limitations of Proxy Application with Respect to Parent

Recommended: Documented analysis of how the app proxy represents the parent application.

Recommended: Documented aspects of the proxy app that may look simple enough that some quick optimizations can be applied but **should not** be applied because they will not be feasible in the parent application.

Required: Documentation should also mention what aspect of the parent application the proxy is a proxy for (e.g. computation, memory, communication, I/O).

Building and running/testing

Required: Documentation should cover the build process (including dependencies) and how to test the final executable. See Verification of Correctness below.

Required: Documentation should cover all configuration options and command-line parameters a user is expected to use.

Recommended: Documentation should explain why the user might use each option, not just what each option enables.

Running mode documentation

Required: The running mode documentation should not only include input decks but should also include expected outputs for each of the input decks. See Verification of Correctness below.

Recommended: The documentation should discuss minimum/maximum reasonable problem sizes to prevent users running at extreme scaling points or running unrealistic cases.

Verification of Correctness

Required: Inclusion of an automatable regression test that checks the final output against a known good reference.

Recommended: Regression test that compares final output against a reference within an acceptable threshold based upon the science being simulated.

Recommended: Tests should cover most code paths provided by the proxy app. /the Proxy App team can help with measuring code coverage numbers (e.g., reported performance metrics/models).

Reported Performance metrics/models

Required: The proxy app should include a figure of merit (FOM) that vendors can use to compare performance under different conditions. This could be based on the science being simulated or what the proxy app is stressing (e.g., computational power, memory etc.). The FOM should be representative of the performance of the parent application.

Recommended: If applicable, include one or more timers around the most important code regions and an option to output the execution time of those regions.

Recommended: If applicable, include a performance model based on the resource the proxy app is stressing.

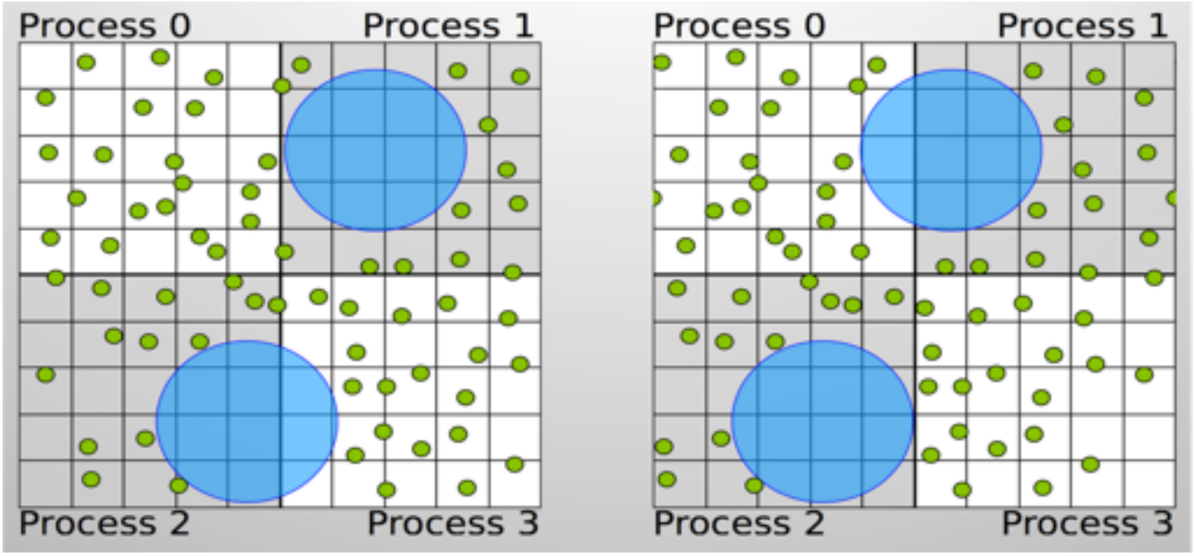
Uses for hardware evaluation and co-design

How Vendors Use Proxy Apps

- Better understand our workloads and applications
- Use with simulators for processors, memory, networks, and I/O
- Use to test new software technology and programming models (much as we do)
- Be mindful of over-optimization for current hardware: we want to know how to realistically want to write the software in the future.

As a result, we need to understand our proxy apps and applications well so that we can give vendors meaningful pieces of software!

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.

Often small modifications in how existing proxies are used can be helpful for exploring new questions

Questions?

Proxy App Assessment: Are They Representative?

Jeanine Cook, Omar Aaziz, Tanner Juedeman (SNL)
Hal Finkel, Brian Homerding (ANL)
Shirley Moore, Tiffany Mintz (ORNL)
Peter McCorquodale (LBNL)

How do we Determine if Proxies are Representative of Full Apps? (1)

Success is highly dependent on coordination with the Application Development Teams

- Qualitative
  - Determine what each proxy is intended to represent
    - Memory behavior?
    - Computation?
    - Communication?
    - Programming model exploration?
  - Determine the mapping of proxies to parent apps
    - Some are generic and meant to represent a large problem space so map back to multiple apps
    - Some map to a specific app (but right now, these are mostly not in the ECP App suite)
  - Determine representative scaling configurations
  - Determine representative problems and sizes
  - Determine consistent (across labs involved) platform for experimentation
    - Architecture and compilers/optimizations need to be held constant

How do we Determine if Proxies are Representative of Full Apps? (2)

Success is highly dependent on Application Assessments

- Quantitative
  - First steps
    - Profiling
      - Mostly to understand proxy composition
    - Basic characterization
      - % of theoretical peak
      - Fraction of memory BW used
      - Fraction of cache BW used at various levels
      - FLOPS/core
      - IntOPS/core
      - Arithmetic intensity (FLOPS/DRAM bytes)
      - Computation vs communication time

How do we Determine if Proxies are Representative of Full Apps? (3)

Success is highly dependent on coordination with the Application Assessment Team

- Quantitative
  - Comparison Methodology
    - Statistical – uses PCA and K-means Clustering (with Manhattan distance)
    - Key is to determine metrics that distinguish behavior of proxies/apps
      - Many proxies/apps have very similar behavior at the hardware level for basic characteristics
        - In some cases, not easy to identify metrics that distinguish behavior
  - Can easily do for proxies, but what about applications?
    - Collaboration and coordination with the Application Assessment Team a must
      - Agree on which data to collect, which platform/compilers/optimizations, which tools to use so we can fairly compare/use data
    - Many of the ECP applications are in various stages of development
      - Continuous cycle of measurement, analysis….

To Date

- Qualitative
  - Mostly understand what each proxy is supposed to represent (mostly obtained from documentation)
  - Have problem size and scaling info that maps back to parent app for about 90% of proxies (will release to community in upcoming report)
  - Don't fully understand mapping of proxies back to parents for all proxies
    - Because they aren't ECP application specific, they can make back to many parents
      - We see this in comparison data
  - Have consistent experimental platform across labs (Haswell), but unfortunately its performance monitoring unit known to have issues
- Initial quantitative characterization/comparison
  - Profiling ➔ mostly done
  - Characterization ➔ in progress
  - Proxy/parent comparison ➔ Lots of work on methodology, but still not sure correct
- First Milestone on this work due in March

EXASCALE COMPUTING PROJECT

Proxy App	Description
miniFE	Unstructured implicit finite element or finite volume
SWFFT	Fast Fourier transform which distributes data between ranks in a 3D cartesian grid communicator, and then re-distributed across three 2D pencil dsitributions to compute the DFFTs along each dimension.
XSbench	Computationally intensive part of a typical MC transport algorithm - the calculation of macroscopic neutron cross sections
miniTri	Triangle enumeration with a calculation of specific vertex and edge properties. Key uses include dense subgraph detection, characterizing graphs, improving community detection, and generating graphs.
CoMD	Reference implementation of classical molecular dynamics algorithms and workloads as used in materials science
miniMD	Parallel molecular dynamics simulation of a Lennard-Jones or a EAM system
nekbone	Solves a standard Poisson equation using a conjugate gradient iteration with a simple or spectral element multigrid preconditioner on a block or linear geometry
SW4lite	Bare bone version of SW4; 3D seismic modeling

Profiling and Characterization: Initial Investigation Workflow

- Document build, optimization and run options
- Look into how the application scales across threads
- Find and focus on sections of the code where the application is spending its time.
- Gather some high level performance measures with PAPI counter data using HPCToolkit. Eg. Cache Miss Rates, IPC (compare to the baseline from hardware documentation)
- Additionally, investigate how the important parts of the code are stressing the memory hierarchy by combining performance counter data with measurements from the Empirical Roofline Tool.
- From these results begin to determine how application is stressing the hardware. Eg. Computation, memory bandwidth, memory latency.

Initial Findings on Haswell

- miniFE (openMP version with 256 size parameters):
  - Memory latency bound from indirect memory access.
- SWFFT(1 node with 720 grid vertexes along one side):
  - Split between computation and communication.
- XSBench (default with lookups increased):
  - Experiences memory latency issues from going to DRAM while also using around 50% of DRAM bandwidth.
- miniTri(openMP version using Email-enron dataset):
  - Spends majority of its time in the STL and allocating memory.

Application Characterization Guide

Utilizing Observations to Create Application Characterization

Workflow:

1. Gather Context starting with recording build options and run options.
2. Decide on Important part(s) of Application to Analyze.
3. Gather Observations with HPCToolkit *(Short HPCToolkit Guide at Bottom)* alongside Context.
4. Perform Analysis.

Gather Application Context

1. Build
   - Record Compiler Version: `gcc -v`
   - Record Optimization Flags: eg. `-O3 -ffast-math -march=native`
   - Record Dependent Library Versions?: `ldd -v [.exe]`
   - Check for Compiler Specific Source Code Pragmas
     - Check to see what compile definitions in the build system are doing (-DINTEL)

       ```
       #ifdef INTEL
       #pragma simd
       #endif
       ```

     - Look Directly for `#pragma`'s in Source Code: `grep -B 3 -A 3 "#pragma" ./*`
2. Run
   - Record Number of Threads
   - Record Application Run Options Used

Create Application Characterization from Observations

1. Scaling
2. Instructions per Cycle
3. Memory Characteristics

```
In [28]: # Data From Roofline # B/W with 72 Threads
Roofline_Bandwidth = {}
Roofline_Bandwidth['L1'] = 2399.9 # GB/s
Roofline_Bandwidth['L2']  = 784.0 # GB/s
Roofline_Bandwidth['L3']  = 564.8 # GB/s
Roofline_Bandwidth['DRAM']  = 60.4 # GB/s
```

Collect Additional System Data

```
[43]: # Collect information from other sources
BYTES_per_CACHELINE = 64
Threads = 72
```

2. Measure the Application with HPCToolkit

Choose Desired Events and Measure

```
hpcrun -e PAPI_L1_DCM -e PAPI_L2_DCM -e PAPI_L2_ICM -e PAPI_L3_TCM \
    -o BWMeasurements <.exe> [application options]
hpcrun -e PAPI_TOT_CYC -e PAPI_LD_INS -e PAPI_SR_INS \
    -o BWMeasurements <.exe>
hpcstruct <.exe>
hpcprof -S <.exe>.hpcstruct -I /Path/to/Src/'+' -o BWDatabase BWMeasurements
hpcviewer BWDatabase
```

3. Analyze the Data

```
[60]: # Measurements from HPCToolkit
BW_Used = {}
```


EXASCALE COMPUTING PROJECT

Proxy Analysis Report

miniFE

MiniFE is an proxy application for unstructured implicit finite element codes. It is similar to HPCCG and pHPCCG but provides a much more complete vertical covering of the steps in this class of applications. MiniFE also provides support for computation on multicore nodes, including pthreads and Intel Threading Building Blocks (TBB) for homogeneous multicore and CUDA for GPUs. Like HPCCG and pHPCCG, MiniFE is intended to be the "best approximation to an unstructured implicit finite element or finite volume application, but in 8000 lines or fewer."

Parameters

```
Compiler = 'clang 5.0.1'

Build_Flags = '-g -O3 -march=native -fopenmp -DMINIFE_SCALAR=double
-DMINIFE_LOCAL_ORDINAL=int -DMINIFE_GLOBAL_ORDINAL=int -DMINIFE_CSR
_MATRIX -DMINIFE_INFO=1 -DMINIFE_KERNELS=0'

Run_Parameters = '-nx 256 -ny 256 -nz 256'
```

Compiler Specific Pragmas

`#pragma loop_count(15)` -- intel

Scaling

```
In [1]: import matplotlib
        import numpy as np
```

jupyter miniFE (autosaved)

Logout

View Insert Cell Kernel Widgets Help

Trusted

Python 3

Markdown

```
struct matvec_std {
void operator()(MatrixType& A,
            VectorType& x,
            VectorType& y)
{
        exchange_externals(A, x);

        typedef typename MatrixType::ScalarType ScalarType;
        typedef typename MatrixType::GlobalOrdinalType GlobalOrdinalType;
        typedef typename MatrixType::LocalOrdinalType LocalOrdinalType;

        const MINIFE_GLOBAL_ORDINAL rows_size     = A.rows.size();
        const LocalOrdinalType* const Arowoffsets = &A.row_offsets[0];
        const GlobalOrdinalType* const Acols      = &A.packed_cols[0];
        const ScalarType* const Acoefs            = &A.packed_coefs[0];
        const ScalarType* const xcoefs            = &x.coefs[0];
        ScalarType* ycoefs                        = &y.coefs[0];
        const ScalarType beta                     = 0;

```

.omp_outlined..143	CPUTIME %	Inst/Cycle per Core	L1 DC Miss %	L2 DC Miss %	L3 Miss %	L1 Loads/Cycle per Core	L2 B/W Used	L3 B/W Used	DRAM B/W Used
72	41.2 %	0.49	8.2%	68.9%	9.6%	0.27	15.1%	14.4%	12.9%

```
        #pragma omp parallel for
        for(MINIFE_GLOBAL_ORDINAL row = 0; row < rows_size; ++row) {
                const MINIFE_GLOBAL_ORDINAL row_start = Arowoffsets[row];
                const MINIFE_GLOBAL_ORDINAL row_end   = Arowoffsets[row+1];

                MINIFE_SCALAR sum = 0;

                #pragma loop_count(15)
                for(MINIFE_GLOBAL_ORDINAL i = row_start; i < row_end; ++i) {
                        sum += Acoefs[i] * xcoefs[Acols[i]];
                }

```

EXASCALE COMPUTING PROJECT

Proxy to Parent Comparison

- Map proxies in current ECP suite to applications that comprise some of the ECP apps
  - Developing methodology, so this is OK from now
  - As ECP apps come available in varying degrees of completion, would like to use those
    - Will work with Application Assessment project
- Hardware performance counter and mpiP data ➔ PCA ➔ Clustering
  - Key is to collect data/metrics that actually distinguish applications
    - Have collected more performance data than any human would ever want to look at!
  - Using equivalent problems on proxy/parent apps, but probably not representative of ECP problems of interest
    - OK for now, since developing methodology
- Still tweaking methodology
- Milestone due in March

Experimental Methodology

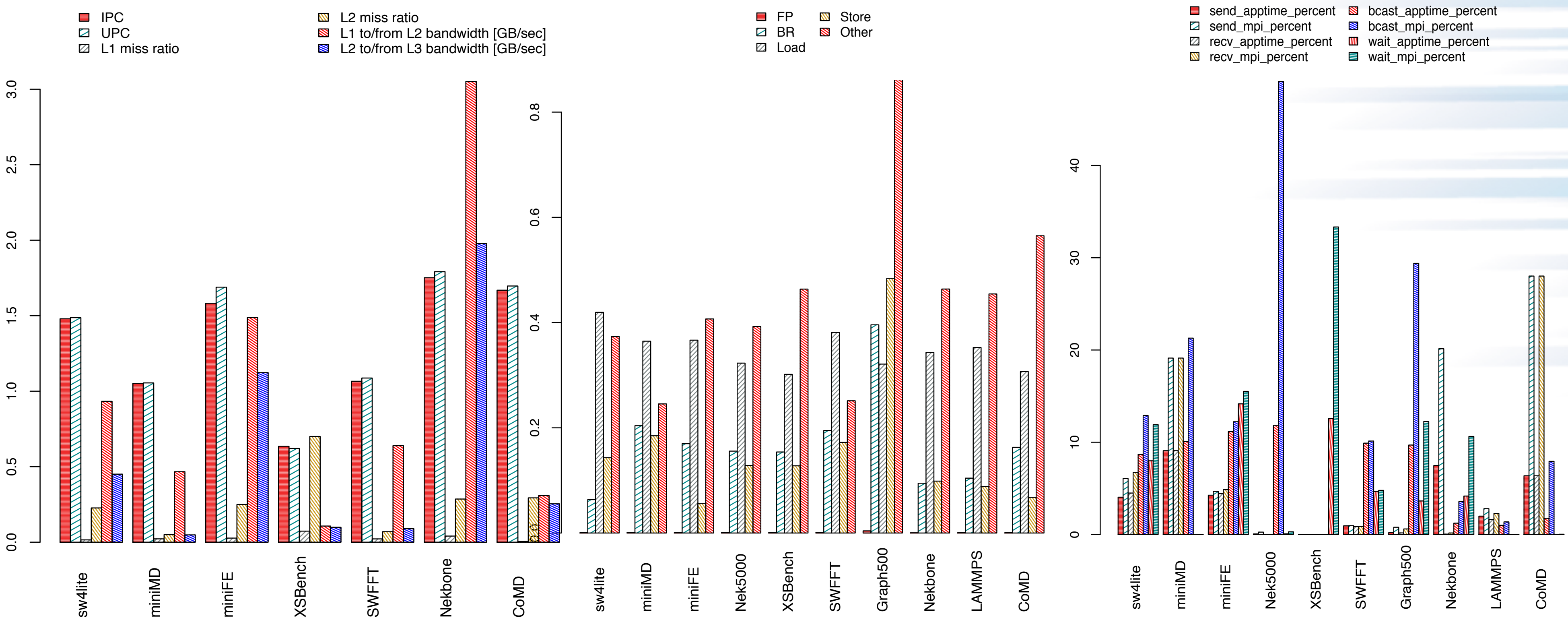
- Haswell
- LDMS application samplers for collecting hardware counter and mpiP data
  - Take sample every 1sec
- Application/proxy mapping: miniMD➔ LAMMPS; Nekbone ➔ Nek5000
  - Then used a bunch of different proxies
  - Used Graph 500 as outlier

Problem Size

Applications	Problem size
Sw4lite	grid h=10 nx=512 ny=512 nz=512 time steps=440
miniMD	Units=metal force style=lj size of problem=20x20x20 Timesteps=1200000 timestep size=0.005
miniFE	nx=1152 ny=1152 nz=1152
Nek5000	Type=ethier numSteps = 150000 [PRESSURE] preconditioner = semg_xxt # residualTol = 1e-08 residualProj = no [VELOCITY] residualTol = 1e-12 residualProj = no density = 1 viscosity = -10
CoMD	-i 16 -j 4 -k 2 --nx=400 --ny=400 --nz=400 • xproc: 16 yproc: 4 zproc: 2

Applications	Problem size
XSBench	-s large -l 140000000 -G unionized • -s <size> Size of H-M Benchmark to run (small, large, XL, XXL) • -l <lookups> Number of Cross-section (XS) lookups • -G <grid type> Grid search type (unionized, nuclide). Defaults to unionized.
SWFFT	n_repetitions = 250000 ngx = 16 • <n_repetitions> is the number of times to run the complete forward and backward test (in case there are memory-system effects that make timing information different after the first repetition • <ngx> is the number of grid vertexes along one side of the entire 3D grid volume.
Graph500	Recursive MATrix (R-MAT) scale = 28
Nekbone	Range of number of elements per proc = 0 250 1
LAMMPS	nx, ny, nz = 100 Timestep = 0.005 Run = 12000

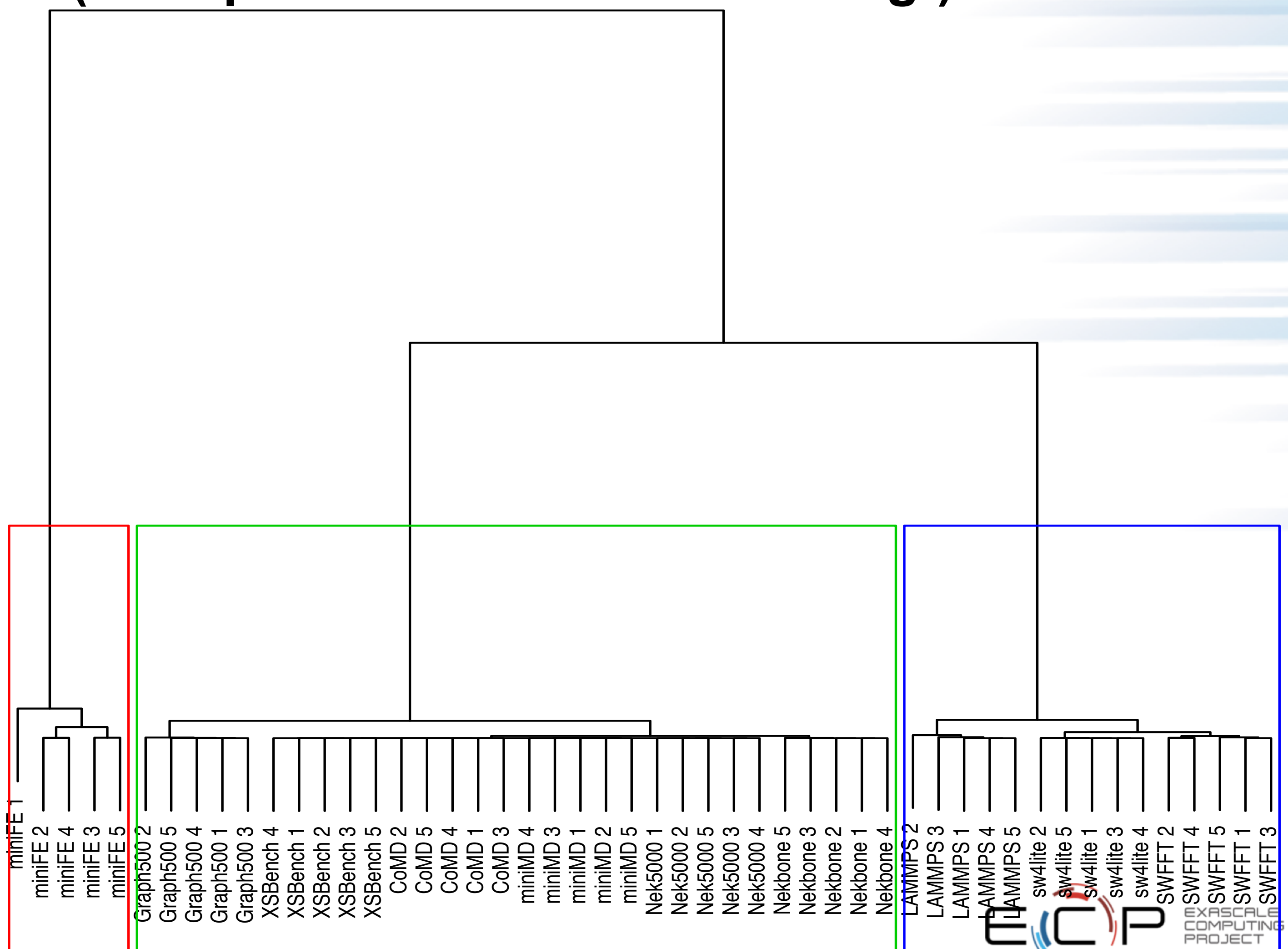
Snapshot of Characterization Data from Comparison Study



Haswell only has AVX event (limited FP); miss ratio events may be unreliable (don't trust denominators)

Clustering Analysis (Unsupervised Machine Learning!)

- After PCA, K-means clustering using Manhattan distance
  - Use average of hardware counter rates and mpiP data over 5 runs for each proxy/parent
  - A single run configuration (8 nodes, 16 processes/node, 1 rank/core)
  - Choose data for PCA randomly from 8 ranks, one of those is always rank 0



Summary

- Still have much work to do
  - Need to understand clustering data, use appropriate problem sizes, add more apps and more real app/proxy pairs
  - Haven't even started with measurement of accelerators yet!
- Need input from apps teams (through Apps Assessment project)
  - Problem sizes, scaling
  - What data would you like to see? Where to put instrumentation?
- Will share our data and lessons learned with vendors
  - Maybe through the new working group interactions (?)

Questions?

Wrap Up / Summary

Questions

1. Where can I find a proxy with <insert desired feature>
   Answer: Try our catalog at http://proxyapps.exascaleproject.org/
   Or, ask our team.
2. Will you add my proxy to the ECP suite?
   Answer: Maybe. But even if we don't you can add it to our catalog
3. What is the best way to create a proxy?
   Answer: Its complicated. Start from scratch or cut from existing app.
4. Can my proxy have <insert feature>
   Answer: Maybe. But simple is best.
5. How do I create a good proxy?
   Answer: See our standards and practices document
   Also, find a way to address a unique niche

EXASCALE COMPUTING PROJECT

Thank you!