SAND2022-9501C

Sandia National Laboratories

The ECP Proxy App Project: Highlights and Lessons Learned

Omar Aaziz

Jeanine Cook

PES GM Panel

U.S. DEPARTMENT OF ENERGY | National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Project Primary Goals

Compose and release a proxy application suite to represent the ECP application complex

- Release on a yearly cadence to track changes in application space
- Suite must be easy to use to ensure community adoption

Assess the fidelity of these proxy applications compared to their respective parents

- Since proxies are used in co-design and acquisition, we must know if and how they are/are not representative of parent behavior

Proxies used in Co-Design

Software development, used to evaluate

- Algorithmic options
- Code optimizations
- Programming models
- Performance portability

Proxies must represent behavior of applications we care about

Proxies must be easy to obtain, build, and execute across platforms

Hardware/system development, used to evaluate

- New architectural and system components
- Algorithms that drive behavior of hardware components

System Acquisition

- Performance/benchmarking

Project Highlights

ECP Proxy App Suite Release 4.0

Currently 64 proxies in ECP Proxy App Catalog

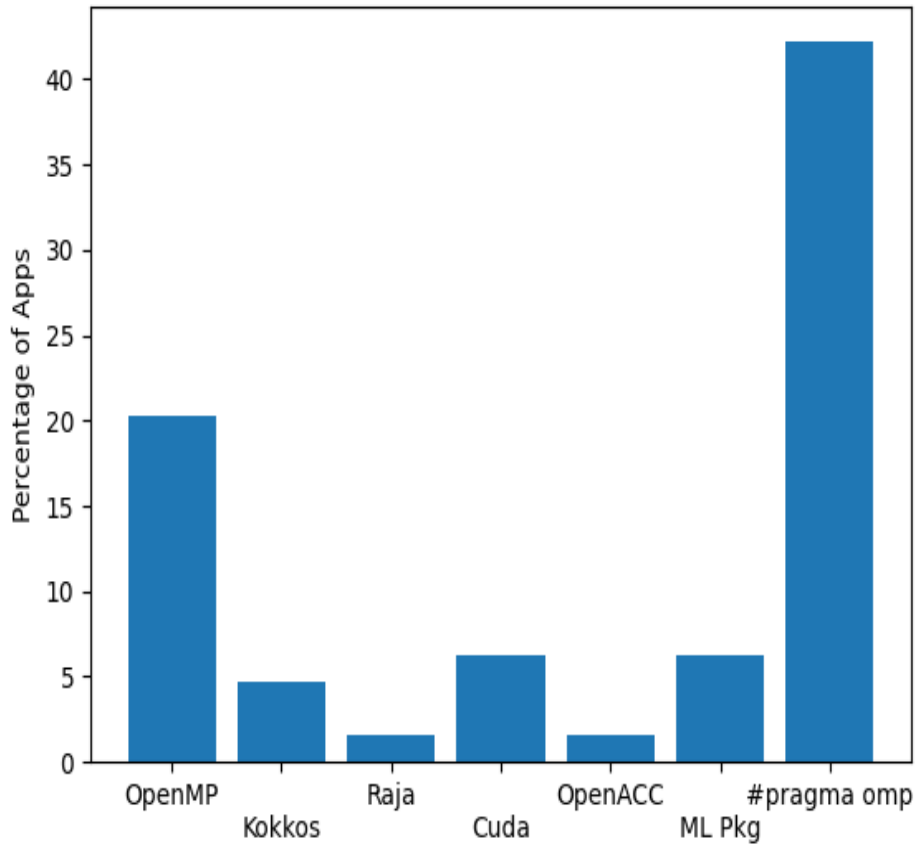
- https://proxyapps.exascaleproject.org/app/
- Organized by programming model
  - All programming models of interest to DOE

Implemented as spack packages for easier build and portability

- spack info ecp-proxy-apps
- spack install ecp=proxy-apps@4.0
- spack list –tag proxy-app

Proxy App Quality Standards and Best Practices

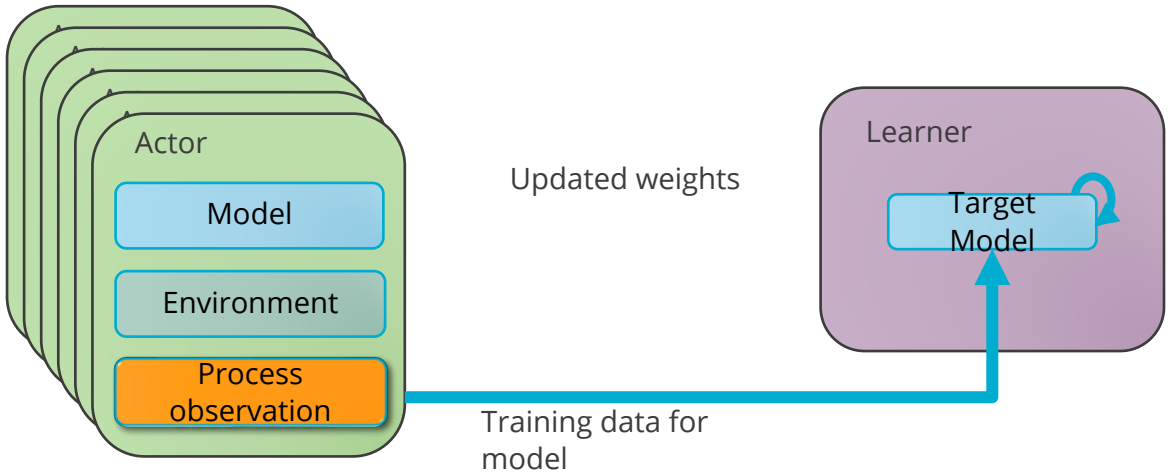
- https://proxyapps.exascaleproject.org/standards/



New ML Proxies

miniRL

- Is a reinforcement learning proxy app derived from EXARL
- Captures important communication and computational aspects
- Demonstrated using Deep Q-Network and a synthetic environment called ExaCartpole
- Provides modular agents, environments, and workflows



- miniTally
  - Implements tallying system from OpenMC
- Qtensor-mini
  - Tensor contraction simulator
- HyPar
  - Finite-difference framework to solve any general hyperbolic-parabolic PDEs on structured grids
- IMEXLBM
  - Lattice Boltzmann Method proxy app suite for heterogeneous systems

ECP Proxy Apps

POC: ECP Proxy Apps Team (ecp-proxy-apps@llnl.gov)

URL: https://proxyapps.exascaleproject.org/

Publications and reports: url

Ways to contribute?

- Submit proxy apps: https://proxyapps.exascaleproject.org/submit-app/
- Develop or improve spackage for proxy apps: https://github.com/spack/spack/

Proxy Application Fidelity Assessment

Develop method to assess fidelity of proxies that are used in hardware co-design

- Only use proxies that are designed to represent computational/memory/communication behavior of parent applications

Key insights

- An application generates a “fingerprint” during execution on a particular system
  - Reflects how the application behaves in response to system constraints
- Similar application fingerprints mean application responds similarly to particular design constraint and to changes in that particular constraint
  - Expect codes with similar dependence/bottleneck on memory bandwidth to derive similar benefit from memory bandwidth improvement
- Fingerprints must be easy and fast to collect

Method

- Evaluated several ML Similarity techniques
- Fingerprint is 500-700 (based on platform) performance counter events
- Used an average of (take from SC paper)

Experimental Platform

Application fingerprint from two different platforms (IBM & Intel)

- ~700 IBM events; ~500 Intel events
  - Can compare proxy/parent on single platform
  - Have to be careful across platforms
    - Cross-platform events can be very different ➔ group so that only similar events are compared

Used 8 proxy/parent pairs and 5 others that were either proxies or applications (not paired)

Run configuration fixed

- 4 nodes, 128 ranks

Normalize input/problem to consume ~50% of memory

~4500 total runs

- 5 for each application x # groups
- Average over 5 runs, then over ranks ➔ single value for each event

Use LDMS and PAPI sampler

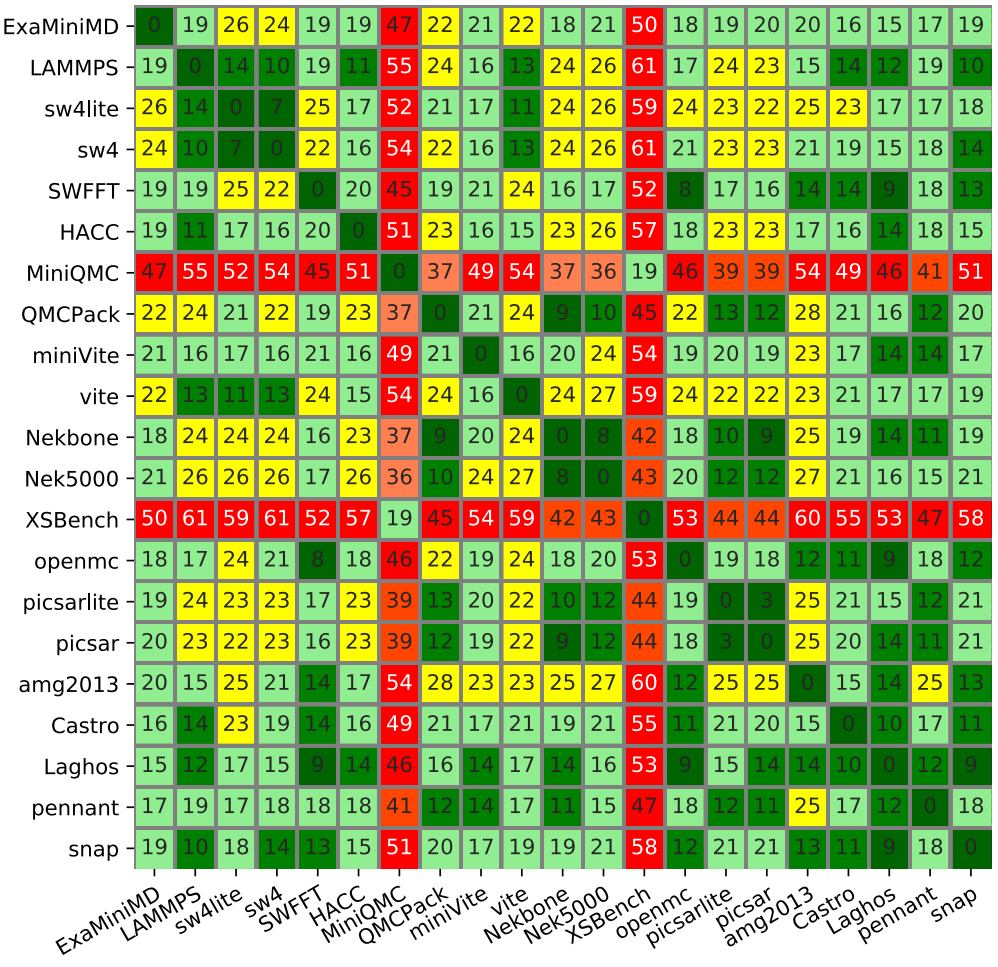
- Sample rate: 1 sec

Apply similarity

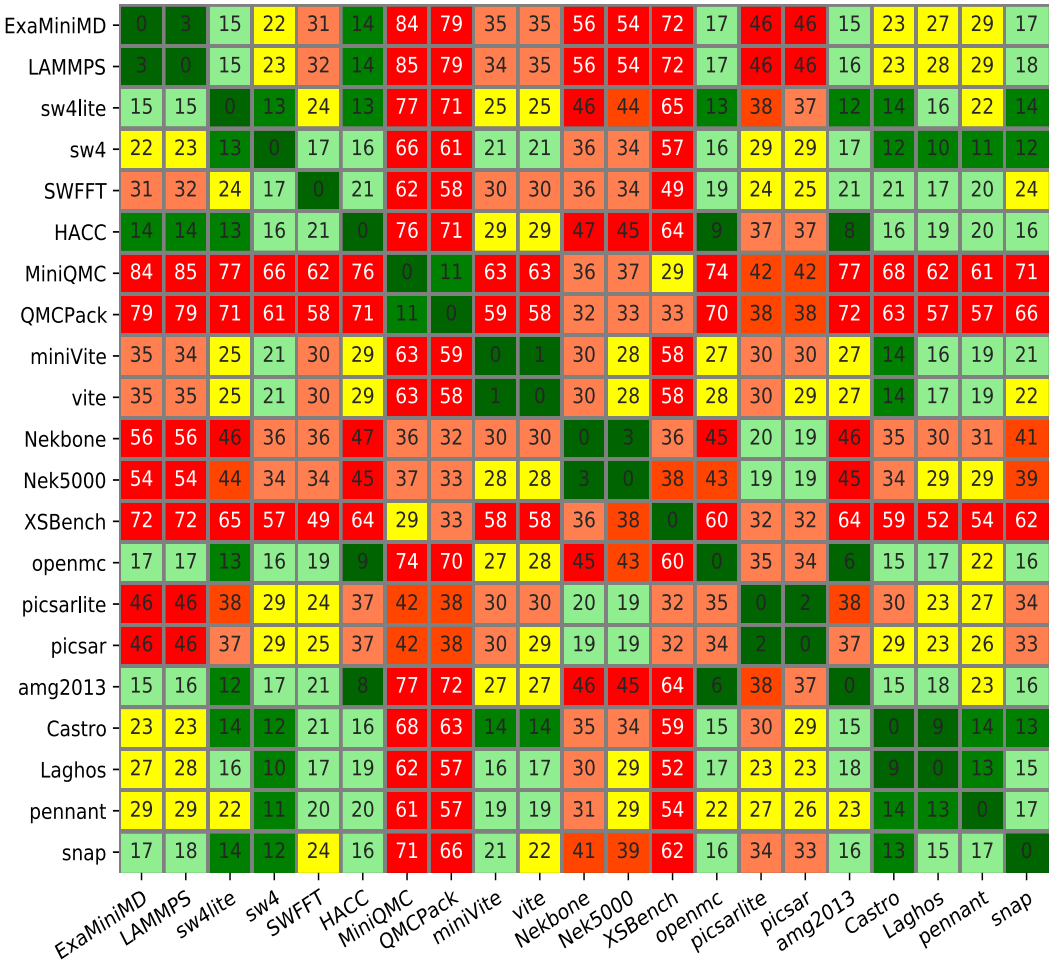
- LIST the techniques we used in SC paper

Entire Fingerprint

IBM P9



Intel SKX



Assessment Summary

ML methods can be used to understand proxy app fidelity

- Node and memory, and communication

COS on-going work

- Re-factoring event groupings to be more accurate within a single platform and to enable more accurate cross-platform comparisons
- Developing a method to determine ground truth
  - Unsupervised learning, ~recommendation system
- Examining ML techniques to reduce data collection
- Developing infrastructure to collect application fingerprints on ARM platforms

I/O assessment work

- Evaluation of MacSio (configurable to match various communication patterns)
- Explore the viability of creating a model to predict certain I/O behaviors of AMR-based applications based on combinations of input parameters and system variability
- Use this model to create a tool that will generate MACSio parameters from an AMReX input file (with placeholders for missing capabilities)

Big Lessons Learned

Assessing proxy fidelity is feasible using ML techniques and performance measurement
- Feedback must be given to proxy developers when proxy fidelity is low
  - Sometimes they haven't updated the proxy!

Curating a proxy app suite is more difficult than it sounds, but possible!
- In spite of 100's of proxies that exist, possible to
  - Organize, document, catalog for easy access
  - Spackify for ease of porting

Developers should understand motivation and use of proxy
- What is the proxy going to be used for?
  - Software optimization? Hardware co-design? Programming model exploration?
- What is it intended to represent?
  - Algorithm? Computational behavior? Programming model?

Proxy documentation must state intended use

Proxies must keep up with application space

Thanks!

The End