# Best Practices for Using Proxy Apps as Benchmarks

Approved for public release

ECP Annual Meeting

February 5, 2020

David Richards, LLNL

Jeanine Cook, SNL

Oscar Hernandez, ORNL

Hal Finkel, ANL

Joe Glenski, HPE/Cray

LLNL-PRES-805269

# Today's Agenda is Packed with Proxy App Goodness

**10:30-10:45 David Richards**

- Introduction. When is a proxy app also a benchmark? What makes a good benchmark? Examples from Quicksilver.

**10:45-11:00 Jeanine Cook**

- Evaluating fidelity of proxy apps.

**11:00-11:15 Oscar Hernandez**

- How facilities assemble benchmark suites and the considerations for what is and is not included.

**11:15-11:30 Hal Finkel**

- Uses of proxy apps for software-technology project development (LLVM, profiling tools, etc.). Experiences working with vendors with proxy apps.

**11:30-11:45 Joe Glenski**

- How vendors view our benchmark suite including what is and is not effective

**11:45-12:00 Feedback Session**

- How to improve the usefulness of the ECP Proxy App Suite for Benchmarking

# 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

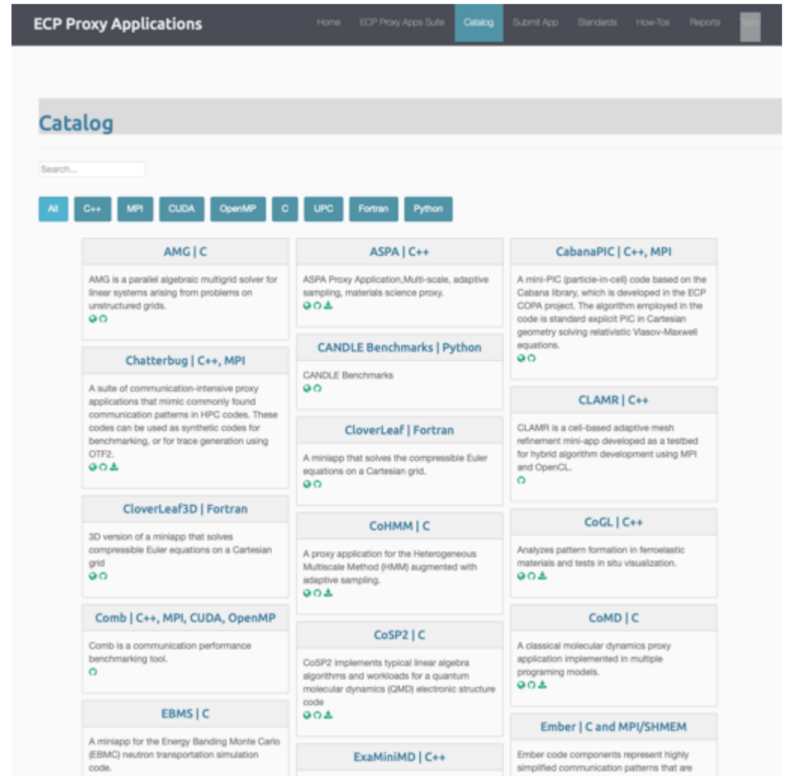
**All benchmarks are proxy apps.**
**Proxy apps are not automatically good benchmarks.**

# When proxies go out into the wild…

- The collection of proxy apps is large and growing
- 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 proxies in their papers

However

- Proxy authors often fail to anticipate possible uses
- Proxy users aren't always familiar with caveats and limitations of proxies




The ECP Proxy App Catalog lists over 50 proxy apps

**Sometimes this works out well and sometimes it does not**

# Proxy apps are models.  Models are easy to mis-use

- "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.
- Many papers and reports present proxy app performance information without describing input parameters.  Sensitivity analysis is rare.
- Vendor reports often contain similar errors to research papers.

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 widely published even when they are 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 which proxies make good benchmarks (and what inputs to use)

- Writing code is fun
  Writing documentation is not

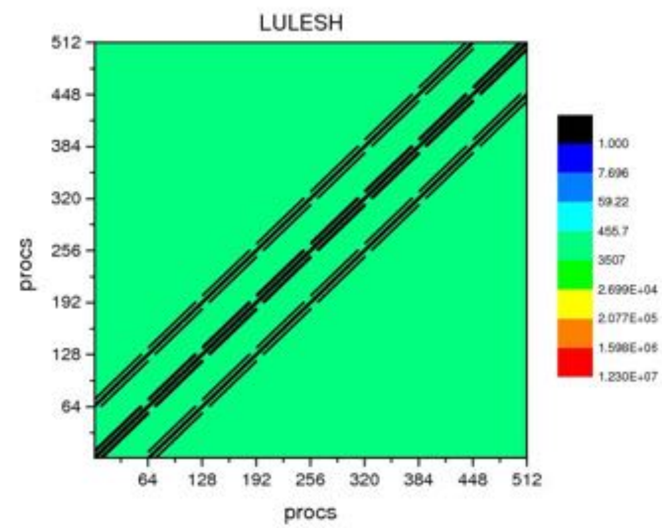



Image from a DOE website showing LULESH communication pattern. LULESH is good for many things but
it is **not** representative of unstructured codes' communication patterns.

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

## A Figure of Merit (FOM)

- An 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

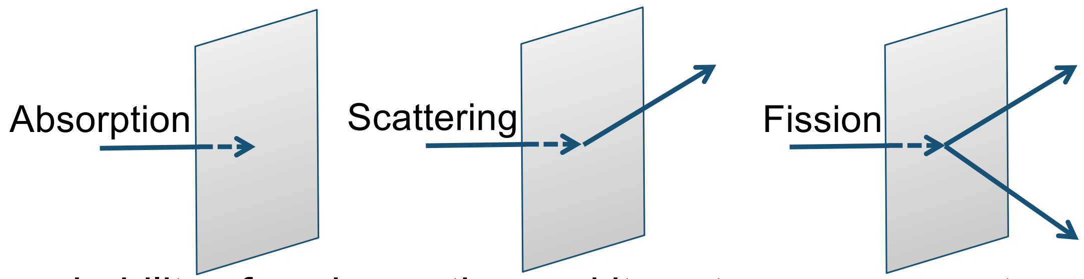
## A Set of Run Rules

- Run rules may include:
  - Problem specification
  - Code version
  - Weak or strong scaling constraints
  - Allowable code modifications
  - Wall time constraints
  - Misc limits such as memory per MPI rank, node count(s) to run jobs on, etc.

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

# Quicksilver is a proxy for Mercury (Monte Carlo 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)

**Quicksilver attempts to capture these key traits of Mercury**

# Defining a good Quicksilver benchmark problem is very challenging

## Challenges

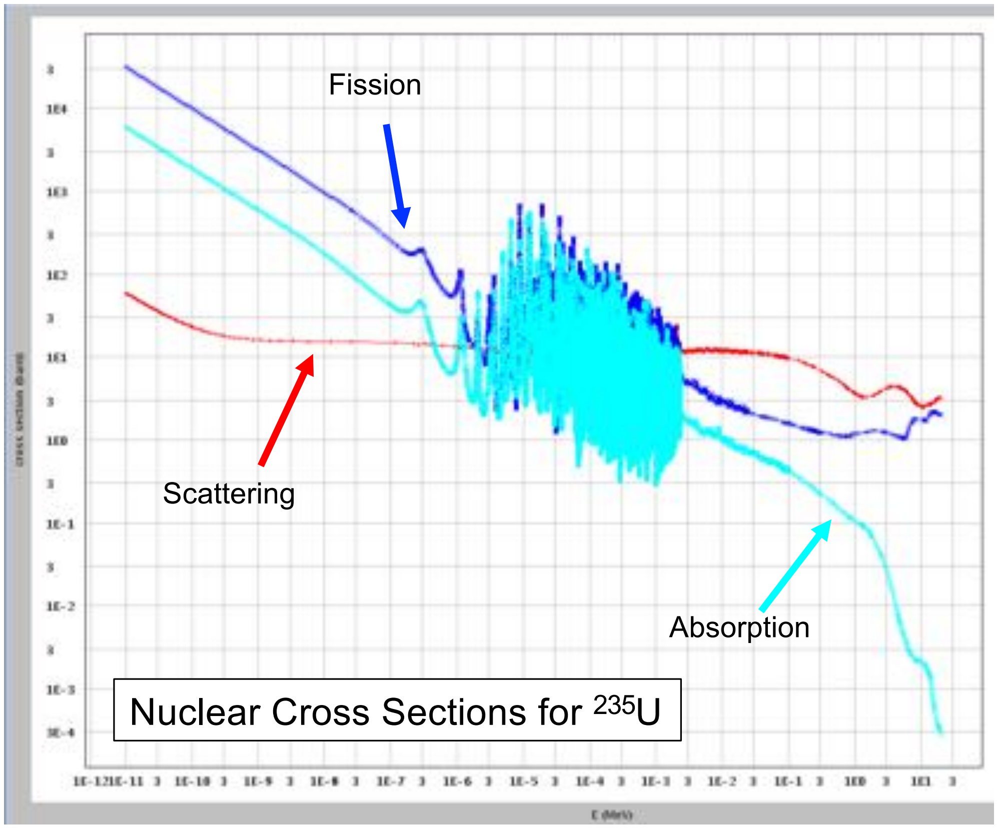
- **Huge variation in scale:**
  Benchmark must be equally valid on 1 node or 10,000 nodes.
- **Simulation geometry:**
  Any geometry that resembles production use will be difficult to scale.
- **Realistic behavior:**
  Production behavior arises from complex geometry and multiple materials.
- **Load Balance:**
  Imbalanced load distorts performance.

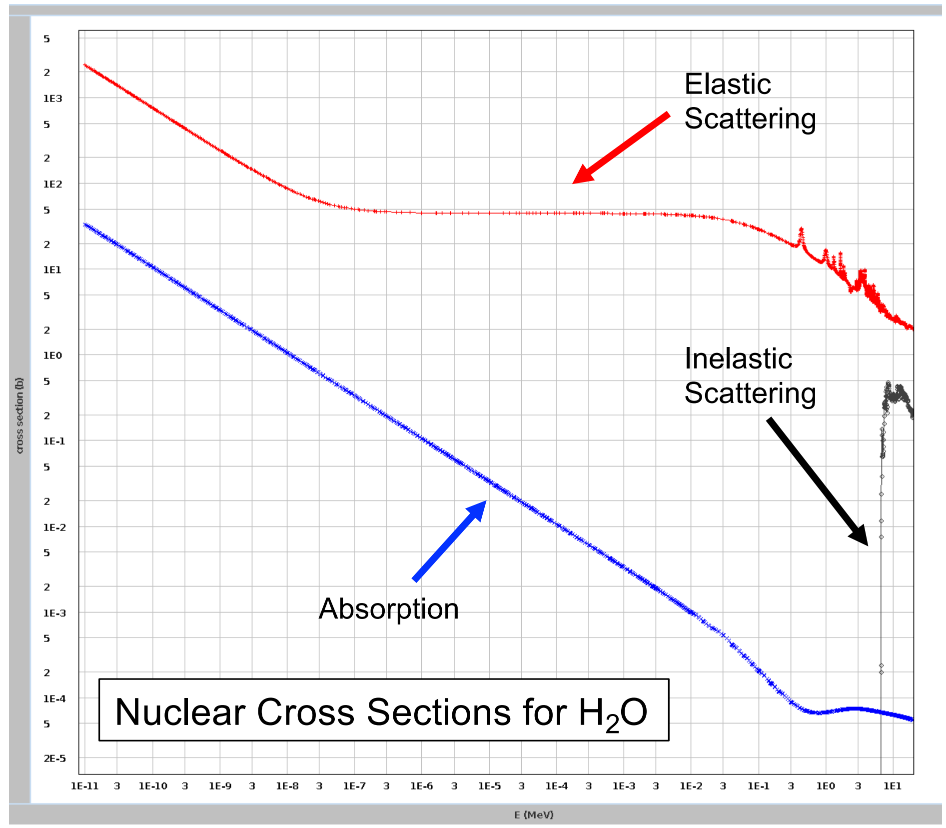
## Solutions

- **Homogeneous single material geometry:**
  Trivially scalable and load balanced.
- **Run rules to constrain problem:**
  Fixed mesh size and elements per node. Also set target range for wall time per step.
- **Made-up Materials:**
  Material properties tailored to interact with simplified physics to produce desired behavior. Blend of real materials.

# Simplified physics can drastically alter program behavior

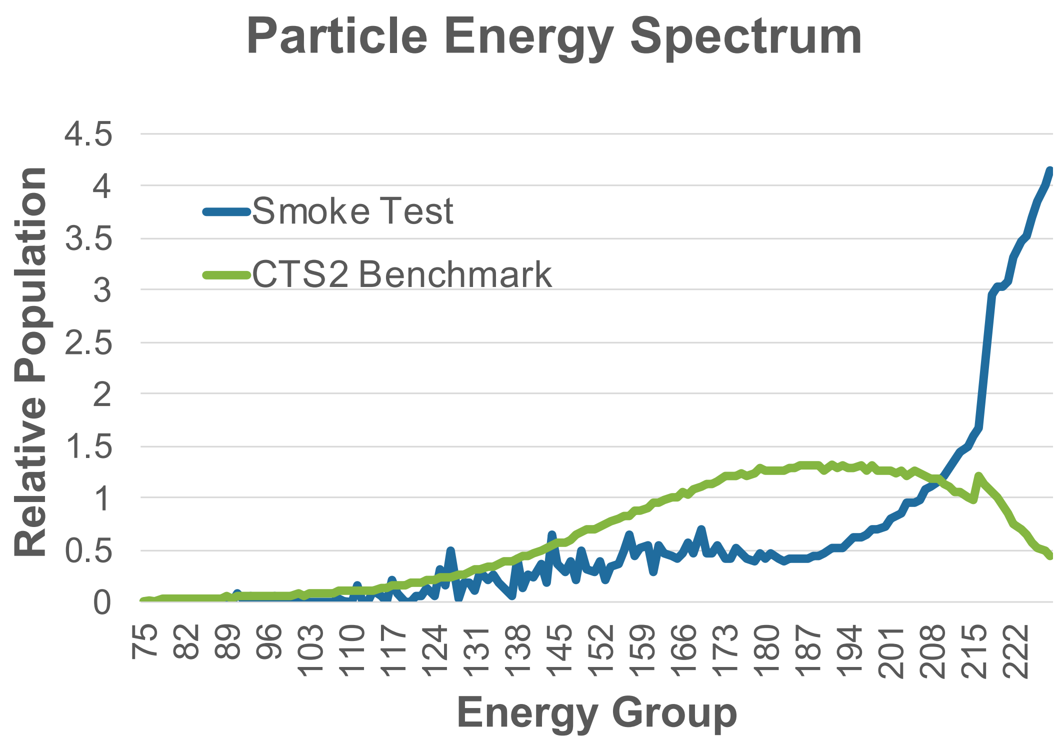
## Quicksilver’s synthetic cross sections struggle to match this complexity



# The Quicksilver CTS2 benchmark problem represents memory access patterns more accurately than the default problem

- The default Quicksilver problem is only a “smoke test” intended for developers
- Energy spectrum determines memory access pattern for cross section lookups
- Smoke test overpopulates high energies compared to intended benchmark
- **Moral:** Beware default problems unless you know they are intended to be representative



# Please remember these key take-aways:

- All benchmarks are proxy apps.  Proxy apps are not automatically good benchmarks
- An understanding of what you are using and why its important are essential when using proxy apps
- Benchmarks have well-defined run rules and a figure of merit
- Good benchmark problems can be hard to design.  Must address issues of scalability, fidelity, ease of use, etc.
- DOE system procurement suites can be a good place to look for benchmark problems

LLNL-PRES-804121

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

EXASCALE COMPUTING PROJECT

# Using Cosine Similarity to Quantify Representativeness of ECP Proxy Apps

Approved for public release

EXASCALE COMPUTING PROJECT

Jeanine Cook (SNL) and Jeffery Kuehn (LANL)
Omar Aaziz (SNL)
Courtenay Vaughan (SNL)

National Nuclear Security Administration

SAND2020-1245 PE

U.S. DEPARTMENT OF ENERGY | Office of Science

# Motivation for Examining Representativeness

- Proxy applications used for
  - Long term vendor collaboration projects (e.g., PathForward)
  - Procurements (benchmarking/performance estimation)
  - Testing new systems/architectures
- Incentive to limit the number of proxy codes
  - Constrained on staff and time (labs & vendors)
  - Vendors have limited time & staff to respond to RFPs
- Qualitatively down-select number of project codes
  - Debate among team of SMEs about perceived relevance
  - Choices often advocated based on familiarity, ease, etc

Strategy: Add quantitative support to balance qualitative inputs

# Insights

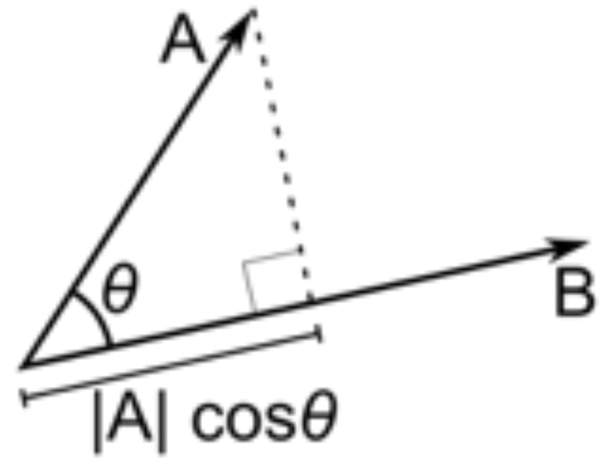
- Performance is interaction of workload with set of design constraints imposed by a particular system
  - Manner and proportion that design constraints affect particular workload becomes the workload fingerprint
- Similar workload fingerprints mean workload responds similarly to particular design constraint and to changes in that particular constraint
  - E.g. Expect codes with similar dependence/bottleneck on memory bandwidth to derive similar benefit from memory bandwidth improvement
- Workload fingerprints must be easy and fast to collect
  - Not through detailed simulators!

# Approach

- Rely on two-elements as building-blocks/tools
  - Ability to collect fingerprint for a code
  - Ability to quantify a similarity comparison between two fingerprints
- Fingerprint construction
  - Aggregation of set of metrics relevant to system design constraints
    - Hardware performance counters/events grouped by design constraints
      - E.g., Processor frontend, execution, backend, cache/memory hierarchy
- Cosine similarity comparison
  - Compares vectors of performance counter events in high dimensional space

# Cosine Similarity

- A property of the inner (dot) product in vector spaces of two or more dimensions
  - Think: "Projection of **A** in the direction of **B**"
- Uses cos$\theta$ as an angular distance metric
  - Quantifies the distance between A and B independent of their magnitude




$$\boldsymbol{A} \cdot \boldsymbol{B} \equiv \sum_{i=1}^{n} a_i b_i = \| \boldsymbol{A} \| \| \boldsymbol{B} \| \cos\theta$$

$$\therefore \cos\theta = \frac{\left(\sum_{i=1}^{n} a_i b_i\right)}{\left(\| \boldsymbol{A} \| \| \boldsymbol{B} \|\right)}$$

# Performance Counter Events & Selectivity

| Cache | Selectivity | Pipeline | Selectivity |
|---|---|---|---|
| MEM_LOAD_UOPS_L3_HIT_RETIRED.XSNP_HIT | 2.721 | FP_ASSIST.ANY | 3.162 |
| MEM_LOAD_UOPS_L3_HIT_RETIRED.XSNP_HITM | 2.213 | FP_ASSIST.X87_INPUT | 3.162 |
| MEM_LOAD_UOPS_L3_HIT_RETIRED.XSNP_MISS | 2.178 | MEM_UOPS_RETIRED.STLB_MISS_LOADS | 2.839 |
| L2_LINES_IN.I | 1.531 | MEM_UOPS_RETIRED.STLB_MISS_STORES | 2.577 |
| MEM_LOAD_UOPS_RETIRED.L3_MISS | 1.482 | LD_BLOCKS.STORE_FORWARD | 2.212 |
| L2_RQSTS.RFO_HIT | 1.410 | UOPS_ISSUED.SINGLE_MUL | 2.114 |
| L2_RQSTS.CODE_RD_MISS | 1.406 | LD_BLOCKS.NO_SR | 2.039 |
| MEM_LOAD_UOPS_RETIRED.L2_MISS | 1.383 | UOPS_ISSUED.FLAGS_MERGE | 1.977 |
| MEM_LOAD_UOPS_L3_HIT_RETIRED.XSNP_NONE | 1.305 | ILD_STALL.LCP | 1.796 |
| MEM_LOAD_UOPS_RETIRED.L3_HIT | 1.305 | DSB2MITE_SWITCHES.PENALTY_CYCLES | 1.777 |
| L2_LINES_IN.S | 1.267 | DSB2MITE_SWITCHES | 1.777 |
| ICACHE.MISSES | 1.131 | MISALIGN_MEM_REF.STORES | 1.656 |
| L2_RQSTS.ALL_CODE_RD | 1.073 | LSD.CYCLES_4_UOPS | 1.650 |
| L2_TRANS.CODE_RD | 1.070 | LSD.UOPS | 1.608 |
| MEM_LOAD_UOPS_L3_MISS_RETIRED.LOCAL_DRAM | 1.067 | LSD.ACTIVE | 1.580 |
| ICACHE.HIT | 1.023 | ARITH.FPU_DIV_ACTIVE | 1.551 |
| L2_RQSTS.DEMAND_DATA_RD_HIT | 1.018 | UOPS_DISPATCHES_CANCELLED.SIMD_PRF | 1.434 |
| L2_RQSTS.DEMAND_DATA_RD_MISS | 0.999 | BACLEARS.ANY | 1.358 |

EXASCALE COMPUTING PROJECT

| BROADWELL | ExaMiniMD | LAMMPS | MiniQMC | QMCPack | sw4lite | sw4 | SWFFT | HACC | pennant | snap |
|---|---|---|---|---|---|---|---|---|---|---|
| ExaMiniMD | 0.00 | 10.24 | 84.61 | 83.55 | 61.94 | 64.17 | 86.71 | 85.58 | 75.88 | 44.50 |
| LAMMPS | 10.24 | 0.00 | 75.12 | 73.95 | 53.63 | 56.50 | 79.66 | 78.51 | 70.97 | 34.97 |
| MiniQMC | 84.61 | 75.12 | 0.00 | 5.97 | 42.91 | 47.75 | 51.57 | 51.28 | 66.16 | 43.41 |
| QMCPack | 83.55 | 73.95 | 5.97 | 0.00 | 37.71 | 42.28 | 45.85 | 45.52 | 60.31 | 40.89 |
| sw4lite | 61.94 | 53.63 | 42.91 | 37.71 | 0.00 | 6.47 | 27.99 | 26.86 | 30.17 | 24.55 |
| sw4 | 64.17 | 56.50 | 47.75 | 42.28 | 6.47 | 0.00 | 23.59 | 22.42 | 23.83 | 29.89 |
| SWFFT | 86.71 | 79.66 | 51.57 | 45.85 | 27.99 | 23.59 | 0.00 | 1.22 | 18.65 | 51.79 |
| HACC | 85.58 | 78.51 | 51.28 | 45.52 | 26.86 | 22.42 | 1.22 | 0.00 | 18.14 | 50.70 |
| pennant | 75.88 | 70.97 | 66.16 | 60.31 | 30.17 | 23.83 | 18.65 | 18.14 | 0.00 | 51.63 |
| snap | 44.50 | 34.97 | 43.41 | 40.89 | 24.55 | 29.89 | 51.79 | 50.70 | 51.63 | 0.00 |

| SKYLAKE | ExaMiniMD | LAMMPS | MiniQMC | QMCPack | sw4lite | sw4 | SWFFT | HACC | pennant | snap |
|---|---|---|---|---|---|---|---|---|---|---|
| ExaMiniMD | 0.00 | 8.97 | 81.96 | 68.83 | 38.66 | 39.55 | 28.51 | 37.76 | 43.58 | 22.20 |
| LAMMPS | 8.97 | 0.00 | 81.38 | 68.47 | 38.60 | 39.33 | 29.50 | 38.49 | 42.40 | 20.45 |
| MiniQMC | 81.96 | 81.38 | 0.00 | 16.35 | 47.28 | 47.63 | 58.78 | 49.85 | 46.58 | 65.55 |
| QMCPack | 68.83 | 68.47 | 16.35 | 0.00 | 36.05 | 36.40 | 46.19 | 37.82 | 36.33 | 53.30 |
| sw4lite | 38.66 | 38.60 | 47.28 | 36.05 | 0.00 | 4.05 | 20.56 | 17.09 | 12.89 | 21.69 |
| sw4 | 39.55 | 39.33 | 47.63 | 36.40 | 4.05 | 0.00 | 19.82 | 15.87 | 11.91 | 22.79 |
| SWFFT | 28.51 | 29.50 | 58.78 | 46.19 | 20.56 | 19.82 | 0.00 | 10.33 | 24.49 | 21.44 |
| HACC | 37.76 | 38.49 | 49.85 | 37.82 | 17.09 | 15.87 | 10.33 | 0.00 | 19.92 | 26.67 |
| pennant | 43.58 | 42.40 | 46.58 | 36.33 | 12.89 | 11.91 | 24.49 | 19.92 | 0.00 | 25.00 |
| snap | 22.20 | 20.45 | 65.55 | 53.30 | 21.69 | 22.79 | 21.44 | 26.67 | 25.00 | 0.00 |

# Gaps & Redundancy

| | Average App1& App2 | App1 | App2 | Proxy 10 | Proxy 04 | Proxy 05 | Proxy 08 | Proxy 11 | Proxy 01 | Proxy 02 | Proxy 07 | Proxy 09 | Proxy 03 | Proxy 06 | Proxy 12 | Exclusive Sum across Proxies | min | Avg | Exclusive Average across Proxies |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| App1 | 0.97 | 1.00 | 0.94 | 0.98 | 0.95 | 0.92 | 0.91 | 0.91 | 0.89 | 0.90 | 0.94 | 0.91 | 0.78 | 0.72 | 0.67 | 11.41 | 0.67 | 0.89 | 0.88 |
| App2 | 0.97 | 0.94 | 1.00 | 0.88 | 0.89 | 0.85 | 0.85 | 0.84 | 0.84 | 0.88 | 0.82 | 0.78 | 0.81 | 0.53 | 0.48 | 10.40 | 0.48 | 0.81 | 0.8 |
| Proxy10 | 0.93 | 0.98 | 0.88 | 1.00 | 0.98 | 0.96 | 0.94 | 0.95 | 0.94 | 0.91 | 0.93 | 0.90 | 0.73 | 0.72 | 0.68 | 11.50 | 0.68 | 0.89 | 0.88 |
| Proxy04 | 0.92 | 0.95 | 0.89 | 0.98 | 1.00 | 0.99 | 0.99 | 0.99 | 0.99 | 0.96 | 0.83 | 0.80 | 0.76 | 0.58 | 0.51 | 11.22 | 0.51 | 0.87 | 0.86 |
| Proxy05 | 0.89 | 0.92 | 0.85 | 0.96 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 0.96 | 0.80 | 0.76 | 0.72 | 0.55 | 0.47 | 10.97 | 0.47 | 0.86 | 0.84 |
| Proxy08 | 0.88 | 0.91 | 0.85 | 0.94 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 0.96 | 0.77 | 0.74 | 0.73 | 0.50 | 0.43 | 10.83 | 0.43 | 0.84 | 0.83 |
| Proxy11 | 0.88 | 0.91 | 0.84 | 0.95 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 0.95 | 0.78 | 0.75 | 0.72 | 0.53 | 0.46 | 10.89 | 0.46 | 0.85 | 0.84 |
| Proxy01 | 0.87 | 0.89 | 0.84 | 0.94 | 0.99 | 1.00 | 1.00 | 1.00 | 1.00 | 0.96 | 0.76 | 0.72 | 0.72 | 0.49 | 0.41 | 10.71 | 0.41 | 0.84 | 0.82 |
| Proxy02 | 0.89 | 0.90 | 0.88 | 0.91 | 0.96 | 0.96 | 0.96 | 0.95 | 0.96 | 1.00 | 0.73 | 0.69 | 0.88 | 0.42 | 0.35 | 10.56 | 0.35 | 0.83 | 0.81 |
| Proxy07 | 0.88 | 0.94 | 0.82 | 0.93 | 0.83 | 0.80 | 0.77 | 0.78 | 0.76 | 0.73 | 1.00 | 0.99 | 0.62 | 0.90 | 0.86 | 10.74 | 0.62 | 0.84 | 0.83 |
| Proxy09 | 0.85 | 0.91 | 0.78 | 0.90 | 0.80 | 0.76 | 0.74 | 0.75 | 0.72 | 0.69 | 0.99 | 1.00 | 0.59 | 0.92 | 0.89 | 10.45 | 0.59 | 0.82 | 0.8 |
| Proxy03 | 0.80 | 0.78 | 0.81 | 0.73 | 0.76 | 0.72 | 0.73 | 0.72 | 0.72 | 0.88 | 0.62 | 0.59 | 1.00 | 0.33 | 0.28 | 8.68 | 0.28 | 0.69 | 0.67 |
| Proxy06 | 0.63 | 0.72 | 0.53 | 0.72 | 0.58 | 0.55 | 0.50 | 0.53 | 0.49 | 0.42 | 0.90 | 0.92 | 0.33 | 1.00 | 0.96 | 8.17 | 0.33 | 0.65 | 0.63 |
| Proxy12 | 0.57 | 0.67 | 0.48 | 0.68 | 0.51 | 0.47 | 0.43 | 0.46 | 0.41 | 0.35 | 0.86 | 0.89 | 0.28 | 0.96 | 1.00 | 7.44 | 0.28 | 0.60 | 0.57 |

# Performance Group Breakdown: Cache

| | ExaMiniMD | LAMMPS | MiniQMC | QMCPack | sw4lite | sw4 | SWFFT | HACC | pennant | snap |
|---|---|---|---|---|---|---|---|---|---|---|
| **ExaMiniMD** | 0.00 | 5.02 | 54.54 | 38.73 | 11.70 | 12.49 | 6.58 | 6.38 | 13.21 | 7.13 |
| **LAMMPS** | 5.02 | 0.00 | 54.69 | 38.62 | 15.66 | 16.27 | 4.87 | 6.38 | 13.60 | 10.88 |
| **MiniQMC** | 54.54 | 54.69 | 0.00 | 17.15 | 47.12 | 46.08 | 50.02 | 48.98 | 42.16 | 49.15 |
| **QMCPack** | 38.73 | 38.62 | 17.15 | 0.00 | 32.64 | 31.67 | 33.92 | 32.94 | 26.29 | 33.78 |
| **sw4lite** | 11.70 | 15.66 | 47.12 | 32.64 | 0.00 | 1.15 | 13.41 | 11.40 | 11.15 | 5.07 |
| **sw4** | 12.49 | 16.27 | 46.08 | 31.67 | 1.15 | 0.00 | 13.74 | 11.70 | 10.69 | 5.69 |
| **SWFFT** | 6.58 | 4.87 | 50.02 | 33.92 | 13.41 | 13.74 | 0.00 | 2.24 | 9.09 | 8.80 |
| **HACC** | 6.38 | 6.38 | 48.98 | 32.94 | 11.40 | 11.70 | 2.24 | 0.00 | 7.86 | 6.87 |
| **pennant** | 13.21 | 13.60 | 42.16 | 26.29 | 11.15 | 10.69 | 9.09 | 7.86 | 0.00 | 9.37 |
| **snap** | 7.13 | 10.88 | 49.15 | 33.78 | 5.07 | 5.69 | 8.80 | 6.87 | 9.37 | 0.00 |

# Performance Differences with Different Inputs

| | Angular difference in signatures for clamr_mpiopenmponly -n_4000_-i_100_-t_600 | | | | | |
|---|---|---|---|---|---|---|
| | regular-grid | regular-grid-by-faces | face-in-place | cell | face | cell-in-place |
| regular-grid | 0.00 | 0.15 | 0.19 | 0.12 | 0.28 | 0.27 |
| regular-grid-by-faces | 0.15 | 0.00 | 0.13 | 0.16 | 0.20 | 0.19 |
| face-in-place | 0.19 | 0.13 | 0.00 | 0.19 | 0.18 | 0.19 |
| cell | 0.12 | 0.16 | 0.19 | 0.00 | 0.27 | 0.25 |
| face | 0.28 | 0.20 | 0.18 | 0.27 | 0.00 | 0.14 |
| cell-in-place | 0.27 | 0.19 | 0.19 | 0.25 | 0.14 | 0.00 |
| sum | 0.99 | 0.83 | 0.87 | 0.98 | 1.06 | 1.03 |
| | | **Best representatives** | | | **Worst representative** | |

EXASCALE COMPUTING PROJECT

# How Might this be Used?

- Identify **<u>gaps/artifacts</u>** in representation for set of proxies
  - Artifacts – proxy behaviors that do not appear in workload
  - Gaps – workload behaviors that do not appear in proxies
- Identify **<u>redundancies</u>** in set of proxies
- Quantify **<u>similarities</u>** between proxies and parents or workloads
  - Infer relationships between proxy and workload performance
  - Infer relationships for particular proxy/parent with varying problem/input
- Apply these three properties to:
  - Provide feedback to proxy developers to improve representativeness
  - Help procurement/project teams to better identify minimum spanning sets
  - Identify workload-platform mappings by similarity
    - Identify workloads that are favorable candidates to port to GPU
    - Steer application workloads toward favorable architectures

# Future Work

- Infer error bounds on similarity-based proxy performance projections
- Validation
  - Correlate results with additional performance data
- Examine network and I/O behavior similarity
- Determine which applications optimally map to which architectures based on similarity
- Predict porting effort to target architectures
  - Quantify code differences in application ports to target architectures
  - Use application similarity to predict potential code effort
- Guide optimization efforts

# THANKS!

Approved for public release

EXASCALE COMPUTING PROJECT

National Nuclear Security Administration

SAND2020-1245 PE

U.S. DEPARTMENT OF ENERGY | Office of Science

# Perspectives on Application Benchmarking at the Oak Ridge Leadership Computing Facility (OLCF)

Oak Ridge Leadership Computing Facility
National Center for Computational Sciences
Oak Ridge National Laboratory

Contributors: Verónica Vergara, Reuben D. Budiardja, Bronson Messer, Jack Wells, Wayne Joubert, Swen Boehm and Oscar Hernandez



This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725

# OLCF Use Cases for Benchmarking

- Program Development and Marketing
- Application Development and Performance Readiness for Future platform
- Procurements
- User Program Management
- Programming Models (PM) Development

# Program Development and Marketing

We build supercomputers for science!


Top500.org has been a success in marketing HPC




OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

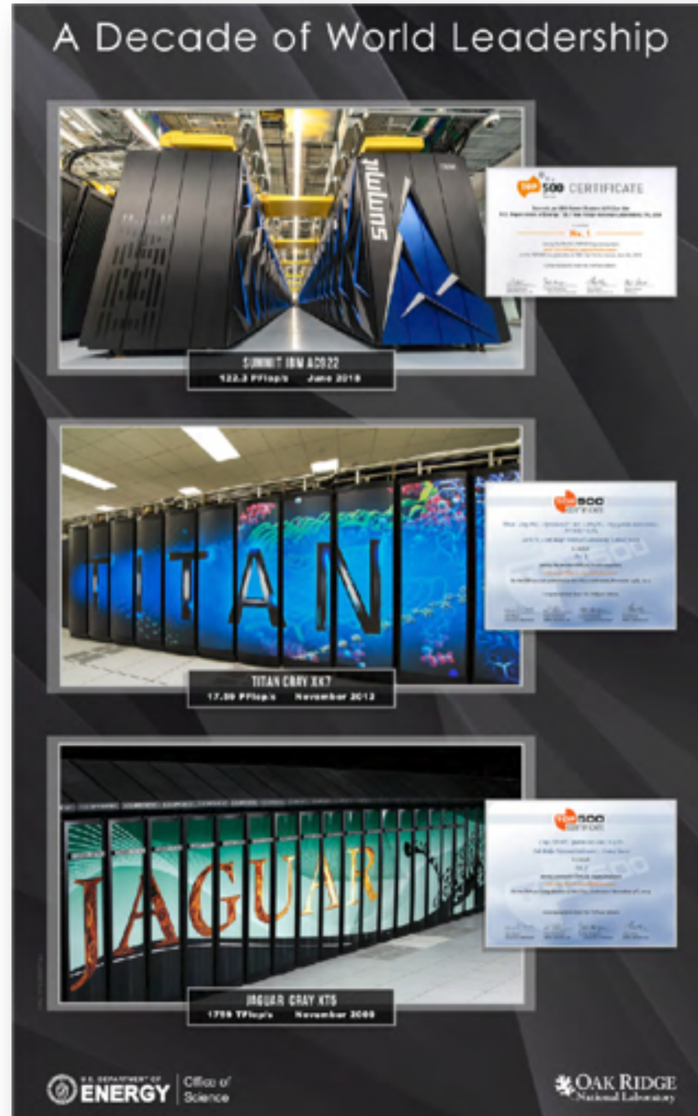
# Summit is latest DOE #1 system on Top500


**143.5 PF HPL**
**Shows math performance**


**2.9 PF HPCG**
**Shows fast data movement**


**14.7 GF/W**
**Shows energy efficiency**


**#1 on the IO-500**
**Shows file system performance**



# CORAL Procurements

**Objective -** Procure 3 leadership computers to be sited at Argonne, Oak Ridge and Lawrence Livermore in 2017 *(CORAL2, 2021-2022)*.

***Current DOE Leadership Computers***

**Titan (ORNL)**
**2012 - 2017**

**Sequoia (LLNL)**
**2012 - 2017**

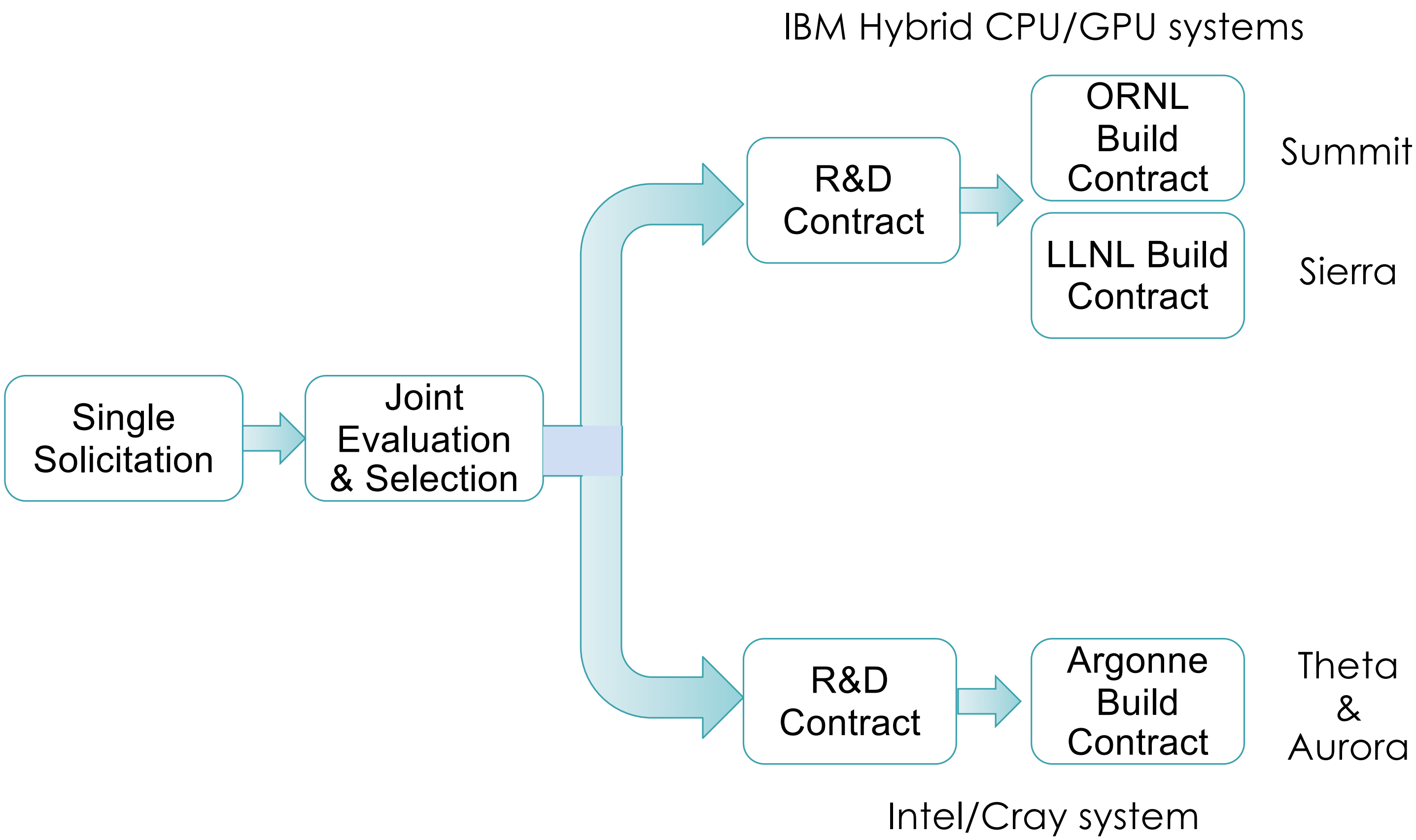
**Mira (ANL)**
**2012 - 2017**

**Leadership Computers** RFP requested >100 PF, 2 GB/core main memory, local NVRAM, and science performance 4x-8x Titan or Sequoia (CORAL2: 50x)

**Approach**

- Competitive process - one RFP (issued by LLNL) leading to 2 R&D contracts and 3 computer procurement contracts
- For risk reduction and to meet a broad set of requirements, 2 architectural paths were selected and Oak Ridge and Argonne must choose different architectures
- Multi-year Lab-Awardee relationship to co-design computers
- Both R&D contracts jointly managed by the 3 Labs
- Each lab manages and negotiates its own computer procurement contract, and may exercise options to meet their specific needs
- Understanding that long procurement lead-time may impact architectural characteristics and designs of procured computers

# CORAL (I) Results



# Wants and constraints

- CORAL benchmarks should
  - Span the breadth of the NNSA (LLNL) workload
  - Span the time-dependent(!) and much broader space of LCF workloads
  - Span co-spaces of algorithms, implementations, and use cases
  - Provide adequate drivers for system SW and library development
- CORAL benchmarks must
  - ..not be so numerous that vendors cannot provide sophisticated analyses on O(weeks) time scale
    - Significant challenge to cover/span the breadth of concerns, while not being onerous on vendors.
  - …not encumber application developers with 24-7 support responsibilities during those weeks
  - …use proxies for NNSA apps

# CORAL-2 Benchmark Codes

- Scalable Science Benchmarks: HACC, Nekbone, QMCPACK, LAMMPS
- Throughput Benchmarks: AMG, Kripke, Quicksilver, PENNANT
- Data Science and Deep Learning Benchmarks:
  - Big Data Analytic Suite
    - [Schmidt, et al., "Defining Big Data Analytics Benchmarks for Next Generation Supercomputers," https://arxiv.org/abs/1811.02287]
  - Deep Learning Suite
- Skeleton Benchmarks
- Microkernel Benchmarks

https://asc.llnl.gov/coral-2-benchmarks/

# Application Development and Performance Readiness

**CAAR** (Center for Accelerated Application Readiness) Goals and Anticipated Outcomes:

- Primary OLCF means to ensure application readiness
- Scalable, accelerated science applications at the start of Frontier operation
- CAAR experience is translated to robust training program, "Best Practices" papers / documentation, report to ASCR
- Close collaboration with Programming Environment and Tools Team
- Further hardening of the system at scale with a broader set of applications
- Build staff expertise to enable a smooth transition and effective support of user programs

# "CAAR for Frontier" Selection Criteria

| Category | Description |
|---|---|
| Science | • Compelling scientific vision alignment with Nation's science needs<br>• Broad coverage of science domains |
| Implementation (models & algorithms) | • Broad coverage of relevant programming models, environment, languages, implementations<br>• Broad coverage of relevant algorithms and data structures |
| Development Plan | • Feasibility: measure of success is "Figure of Merit" compared to Summit<br>• Clear challenge problem for execution on Frontier |
| Development Team | • Commitment from development team<br>• Plan for integration with other active development directions<br>• OLCF liaison domain-specific skills and expertise with the application<br>• Engagement with Vendor Center of Excellence |

https://www.olcf.ornl.gov/caar/Frontier-CAAR/

SHARE

4

LOVE

Eight projects to gain early access to the Frontier supercomputer

n preparation for the Frontier supercomputer, the US Department of Energy's (DOE's) Oak Ridge Leadership Computing Facility (OLCF) has selected eight research projects to participate in its Center for Accelerated Application Readiness (CAAR) program.

WEEK

ORNL Suicid



OAK RIDGE National Laboratory | FACILITY

# Application Readiness: Community Effort

- Readiness applications are drawn from CAAR, ECP engagement applications, as well as INCITE and ALCC projects on Summit
- CAAR provides the primary risk mitigation strategy for meeting the application readiness KPP
- CAAR is also the vanguard for the broader application readiness ecosystem and for future science
  - Development of training and documentation
  - Knowledge development for staff
  - Improvements to the software stack robustness and performance

## Application Readiness

OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

CAAR

U.S. DEPARTMENT OF ENERGY INCITE LEADERSHIP COMPUTING

ECP EXASCALE COMPUTING PROJECT

ALCC ASCR Leadership Computing Challenge

# Acceptance Testing (AT)

- Main objectives of the AT:
  - Verify correct functionality of the OLCF system and its programming environment
  - Evaluate the system to ensure it meets the functionality, performance, and stability requirements outlined in the contract
  - Demonstrate the usability of the system by the broad scientific user community represented at the OLCF
- Acceptance Test Elements: hardware, functionality, performance, and stability tests
- Tests are selected from applications from the production portfolio

# Acceptance Tests Selection

- Review applications used by active projects on production systems
- Compile list of features, programming languages, libraries, etc.
- Select a subset from the OLCF portfolio of application that provides the highest coverage
- In some cases, no applications are available to use a new technology/upcoming feature
  - Use codes in active development? Not ideal, we want a frozen source
  - Use mini-apps and benchmarks for these cases

# Acceptance Tests Selection (cont'd)

- Summit AT included applications, mini-apps, and benchmarks

| | Languages | | | | Compilers | | | | Features / Prog. Models | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Full Applications** | FORTRAN | FORTRAN 2003 | C | C++ | XL | LLVM | GNU | PGI | GPU accelerated | Uses Multiple GPUs | Pthreads | C++11 threads | OpenMP 3.1 | OpenMP 4 | OpenACC | CUDA | Kokkos | MPI |
| CHROMA | | | x | x | x | x | x | | x | x | | | x | | | x | | x |
| NAMD | | | | x | | | | | x | | | | x | | | | | x |
| LAMMPS | | | | x | | | x | | x | x | | | x | | | x | x | x |
| LSMS | x | | | x | | | x | | x | x | | | x | | | x | | x |
| ACME | x | x | | | x | | x | x | x | | | | x | | x | | | x |
| GTC | x | | | | | | | x | x | x | | | x | | x | | | x |
| NWCHEM | x | | x | x | x | | x | x | x | | x | | x | | | x | | x |
| Profugus | | | | x | | x | x | | x | | | | | | | x | | |
| Additional benchmarks, kernels, and mini-apps | | | | | | | | | | | | x | | x | | | | |

| | Libraries | | | | | | | | | | | | Motifs | | | | | | | Other | | | | | I/O | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Full Applications** | BLAS | LAPACK | ESSL | Magma | cuBLAS | ScaLAPACK | cuFFT | FFTW | PETSc | Trilinos | Hypre | AmgX | Structured grids | Unstructured grids | Dense linear algebra | Sparse linear algebra | Particles | Monte Carlo | FFT | Burst Buffer | SCR | GPUDirect | UVM | GlobalArrays | HDF5 | HDF5 parallel | NetCDF | NetCDF parallel | ADIOS |
| CHROMA | x | | | x | x | | | | | | | | x | | | x | | x | | | | x | x | | | | | | |
| NAMD | | | | | | | x | x | | | | | x | | | | x | | x | | | | | | | | | | |
| LAMMPS | | | | | | x | | x | | | | | | | | | x | | x | | | | | | | | | | |
| LSMS | x | | | | x | | | | | | | | | | x | | | x | | | | | | | x | | | | |
| ACME | | | | | | | | | | | | | x | x | | | | | | | | x | | | | | x | x | |
| GTC | | | | | | | | | x | | x | x | x | x | | x | x | | | | | | | | | | | | x |
| NWCHEM | x | x | x | | x | x | | | | | | | | | x | | | | | | | | | x | | | | | |
| Profugus | | | | | | | | | | x | | | | | | | | x | | | | | | | | | | | |
| Additional benchmarks, kernels, and mini-apps | | | | | | | | | | | | | | | | | | | | x | x | | | | | x | | | |

# Acceptance Test Selection for Frontier

- Developing tests for new technologies requires porting of applications
- Mini-apps and benchmarks are easier to port
  - Usually a smaller source and simpler
  - Easier to debug when issues come up
- Rely on benchmarks that adequately represent real applications
  - CORAL benchmarks (1 & 2), Proxy Apps, standard benchmark suites used across centers

# User Program Management:
# *INCITE System Capability Metric (ISCM)*

- Challenges:
  - Ambiguity in many “allocation unit”
    - core-hours → node hours
  - Difficulty to compare relative performance across systems
    - e.g. OLCF  Summit’s node-hours vs Titan’s node-hours (== 30 x core-hours)

- Goals:
  - Develop a metric that more accurately reflects system capability for the execution of science applications
  - (Potentially) use metric for “currency unit” in user-program allocations.
  - Extendable, better longevity, and “workload agnostic”
- Initial focus on systems allocated under the INCITE program.

# User Program Management:
# *INCITE System Capability Metric (ISCM)*

- Benchmark selection criteria:
  - finer granularity than just two benchmarks (HPL and HPCG)
  - not using specific applications since workloads are apt to vary over time
  - representative of specific machine characteristics to give some visibility into machine characteristics being measured
  - concurrence with growth in capability of leadership class systems over time
  - alignment with an existing benchmark suite which has some level of community acceptance, to give some credibility to the choice.
- → Use a modified & extended version of the HPC Challenge benchmark suite (https://icl.utk.edu/hpcc/) to build a measure we call the INCITE System Capability Metric (ISCM).

R.D. Budiardja, W. Joubert, J. A. Harris, A. Tillack, T. L. Papatheodore, "ISCM: Towards a Comprehensive Metric For Comparative Evaluation of Leadership-Class System Capability for Scientific Applications" (unpublished, 2020)

1
8

# PM Development: SPEC HPG -www.spec.org/hpg

HPG develops benchmarks to represent high-performance computing applications for standardized, cross-platform performance evaluation.

31 Organizations
10 companies
21 academic

Benchmark selection, development and results are peer-reviewed by members

# PM Development: HPG Benchmarks – SPEC ACCEL

- SPEC Accel provides a comparative performance measure of
  - Hardware accelerator devices (GPU, Co-processors, etc.)
  - Supporting software tool chains (Compilers, Drivers, etc.)
  - Host systems and accelerator interface (CPU, PCIe, etc.)
- Computationally-intensive parallel HPC applications and mini-apps
- Portable across multiple accelerators
- Three distinct benchmarks, initially released in 2014, updated in 2017:
  - OpenCL v1.1 19 C/C++ applications
  - OpenACC v 1.0 15 Fortran/C applications
  - OpenMP v4.5, 15 Fortran/C applications
- Support for power measurement

spec®

# We use SPEC ACCEL benchmarks to develop compilers

https://procurement.ornl.gov/rfp/6400016227/

Solicitation No. 6400016227 : **GNU Compiler Collection**

| | Open ACC | | | | OpenMP | | | |
|---|---|---|---|---|---|---|---|---|
| | | | GNU 9.1.0 | PGI 19.5 | | | GNU 9.1.0 | XL 16.1.1-3 |
| | Benchmark | Reference time | Pass/Fail | Time | Benchmark | Reference time | Pass/Fail | Time |
| ostencil | 303.ostencil | 145 | | 12.1 | 503.postencil | 109 | | 10.2 |
| olbm | 304.olbm | 455 | | 36.3 | 504.polbm | 122 | | 19.9 |
| omriq | 314.omriq | 956 | | 35.5 | 514.pomriq | 621 | | 45.5 |
| md | 350.md | 252 | | 9.28 | 550.pmd | 241 | | 21.2 |
| palm | 351.palm | 370 | | 117 | 551.ppalm | 544 | | 203 |
| ep | 352.ep | 530 | | 45.8 | 552.pep | 231 | | 179 |
| clvrleaf | 353.clvrleaf | 445 | | 35.9 | 553.pclvrleaf | 1145 | | 55.5 |
| cg | 354.cg | 408 | | 31.2 | 554.pcg | 333 | | 72.8 |
| seismic | 355.seismic | 370 | | 26 | 555.pseismic | 282 | | 45.8 |
| sp | 356.sp | 276 | | 21.6 | 556.psp | 818 | | 29.3 |
| csp | 357.csp | 270 | | 19.5 | 557.pcsp | 859 | | 92.4 |
| miniGhost | 359.miniGhost | 369 | | 35.8 | 559.pmniGhost | 397 | | 41.5 |
| ilbdc | 360.ilbdc | 367 | | 27.3 | 560.pilbdc | 653 | | 30.5 |
| swim | 363.swim | 230 | | 34.2 | 563.pswim | 159 | | 28 |
| bt | 370.bt | 223 | | 9.37 | 570.pbt | 780 | | 75.7 |

Unofficial results: SPEC ACCEL 1.2 results – Academic use

Source: Swen Boehm, ORNL

## Summary & Discussion

- OLCF is engaged in a variety of mission-critical activities that require application and motif benchmarking.
- Flexibility is necessary in accomplishing activities.
  - "Different horses for difference courses".
- Sustainability and maintainability are key problems to address.
- ORNL participation in SPEC HPG provide real value to many mission-critical functions.
  - Investing in standards is a key strategy including benchmarking
  - Opens the door to engage rest of HPC community - researchers, vendors, HPC centers, etc.

# Use of Proxy Apps by Software-Technology Projects and Hardware Vendors

Approved for public release

Hal Finkel

Leadership Computing Facility

Argonne National Laboratory

National Nuclear Security Administration

U.S. DEPARTMENT OF ENERGY | Office of Science

# Proxy Apps Can Be Used By Software-Technology Projects In Many Different Ways

- Can be used to test new library implementations, including:
  - Math libraries
  - Communication libraries (e.g., MPI)
  - Support libraries (e.g., OpenMP's runtime library)
- Can be used to evaluate how new features in these libraries might be used in different kinds of applications.
- Can be used to test new programming-language/compiler features, including:
  - Compiler optimizations
  - Language constructs and extensions
  - Warnings and other programming aids
- Can be used to test the functionality of tools, including:
  - Profiling tools
  - Debuggers and testing frameworks

# An Example: DOE Proxy Apps in LLVM's Test Suite

## MultiSource/Benchmarks/DOE-ProxyApps-C[++]

- Pathfinder
- RSBench
- SimpleMOC
- XSBench
- MiniAMR
- miniGMG
- CLAMR
- HACCKernels
- HPCCG
- PENNANT
- miniFE

execution_time

| Name | Current | % |
|---|---|---|
| MultiSource/Benchmarks/DOE-ProxyApps-C++/CLAMR/CLAMR | 1.5408 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C++/HACCKernels/HACCKernels | 0.4073 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C++/HPCCG/HPCCG | 0.6024 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C++/miniFE/miniFE | 3.1404 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C++/PENNANT/PENNANT | 0.5841 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C/miniAMR/miniAMR | 0.6336 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C/miniGMG/miniGMG | 0.8441 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C/Pathfinder/PathFinder | 2.7840 | 1.81% |
| MultiSource/Benchmarks/DOE-ProxyApps-C/RSBench/rsbench | 0.5323 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C/SimpleMOC/SimpleMOC | 1.8238 | - |
| MultiSource/Benchmarks/DOE-ProxyApps-C/XSBench/XSBench | 2.3148 | -4.18% |
| Geometric Mean | 1.0396 | - |

LLVM is an open-source compiler infrastructure used by many parts of our exascale ecosystem...

# Proxy App Design vs. Use Cases For ST Development and Testing

- Can your proxy app be used as part of an automated test suite?
  - Does it produce non-deterministic output?
  - Does it require large input files or produce large output files? Must it run on many ranks? Use a lot of memory?
  - How portable is it? Does it use Linux-specific functionality?
  - Does it have a unique build system and/or depend on difficult-to-build libraries?
  - Remember that even debuggers and source-code analysis tools have test suites – it's not just proxies for which performance is meaningful.
- Does your proxy app use advanced programming-language features?
- Does your proxy app depend on a lot of other libraries (just like the real application)?
- How easy would it be to change the programming model in your proxy app? How easy would it be to change the data structures or data layout? Note that:
  - A proxy app can be a good representation of the use of a programming model
  - A proxy app can be a good representation of an algorithm independent of the programming model

# Proxy App Design vs. Use Cases For ST Development and Testing

- If I'm developing a new programming model (e.g., Kokkos, RAJA, OpenMP, OpenSomethingElse)
  - I would like a proxy app where it's easy to change the programming model.
- If I'm developing a compiler, profiling tool, etc.
  - I don't care about changing the programming model; I want the programming-model usage to be realistic.
- Note: These generally apply to the libraries on which your proxy app depends as well.
- If I'm developing for an existing hardware ecosystem (e.g., x86_64 + NVIDIA GPU)
  - I might not care what libraries you use or how
- If I'm developing for a new hardware ecosystem (e.g., NewFancyAccelerator)
  - Library dependencies might be very hard to deal with because of immature tools, hardware-specific code, etc.
- Enable your proxy app to run in a number of different modes:
  - A quick mode to test proper algorithmic functioning (many tools use cases need this).
  - Plus other modes which stress the machine in representative ways.

# On Build Systems...

Winston Churchill said:

"No one pretends that democracy is perfect or all-wise. Indeed it has been said that democracy is the worst form of Government except for all those other forms that have been tried from time to time…"

The same is true for build systems. Right now, CMake is our best approximation of democracy for build systems. Use CMake. Do this even if your real application doesn't (unless your making a proxy for your build system).

# Some Experience Working With Hardware Vendors With Proxy Apps...

1) Developing good proxy apps takes some time: don't wait to start developing them until the vendor engagement has already started.

2) Proxy apps need good documentation, but, direct interaction with the application team (or some person with sufficient application knowledge) is almost always essential.
   - Vendor needs to document how the proxy app was run (parameters, etc.) and these should be reviewed by knowledgeable people (to catch mistakes, miscommunication, etc.).
   - This is all too common: You: "That makes no physical sense!" Vendor: "Yay! It's faster!”

3) Especially for influencing hardware design, vendors want to connect each application analysis to money:
   - Apps used in procurement benchmarks are good.
   - Apps for which improvements can be directly translated to value are good.
   - Thus, the real app is almost always better (if it can be handled; might start with proxy and move to the real app later).
   - Additional proxies might add to test suites (which can be valuable), but don't add significant value for design work.

# A VENDOR VIEW ON BENCHMARKS IN HPC PROCUREMENTS

Joe Glenski

CRAY
a Hewlett Packard Enterprise company

glenski@hpe.com

# FORWARD LOOKING STATEMENTS

This presentation may contain forward-looking statements that involve risks, uncertainties and assumptions. If the risks or uncertainties ever materialize or the assumptions prove incorrect, the results of Hewlett Packard Enterprise Company and its consolidated subsidiaries ("Hewlett Packard Enterprise") may differ materially from those expressed or implied by such forward-looking statements and assumptions. All statements other than statements of historical fact are statements that could be deemed forward-looking statements, including but not limited to any statements regarding the expected benefits and costs of the transaction contemplated by this presentation; the expected timing of the completion of the transaction; the ability of HPE, its subsidiaries and Cray to complete the transaction considering the various conditions to the transaction, some of which are outside the parties' control, including those conditions related to regulatory approvals; projections of revenue, margins, expenses, net earnings, net earnings per share, cash flows, or other financial items; any statements concerning the expected development, performance, market share or competitive performance relating to products or services; any statements regarding current or future macroeconomic trends or events and the impact of those trends and events on Hewlett Packard Enterprise and its financial performance; any statements of expectation or belief; and any statements of assumptions underlying any of the foregoing. Risks, uncertainties and assumptions include the possibility that expected benefits of the transaction described in this presentation may not materialize as expected; that the transaction may not be timely completed, if at all; that, prior to the completion of the transaction, Cray's business may not perform as expected due to transaction-related uncertainty or other factors; that the parties are unable to successfully implement integration strategies; the need to address the many challenges facing Hewlett Packard Enterprise's businesses; the competitive pressures faced by Hewlett Packard Enterprise's businesses; risks associated with executing Hewlett Packard Enterprise's strategy; the impact of macroeconomic and geopolitical trends and events; the development and transition of new products and services and the enhancement of existing products and services to meet customer needs and respond to emerging technological trends; and other risks that are described in our Fiscal Year 2018 Annual Report on Form 10-K, and that are otherwise described or updated from time to time in Hewlett Packard Enterprise's other filings with the Securities and Exchange Commission, including but not limited to our subsequent Quarterly Reports on Form 10-Q. Hewlett Packard Enterprise assumes no obligation and does not intend to update these forward-looking statements.

# Outline

- The ~~big headache~~ Challenge of Writing RFPs
- How are benchmarks used in typical RFPs?
- Evaluation Metrics
- Projections and Estimates
- Optimization
- Suggestions from Benchmarkers

*Special thanks to Tricia Balle, who provide ideas and material for this presentation*

# The challenge of writing RFPs

- **Identify desired system characteristics and ensure the RFP requirements reflect them**
  - How to eliminate what you don’t want and ensure what you do want is scored appropriately?
  - How to easily compare vendor offerings?
- **Ensure the document is clear and unambiguous**
  - Lack of clarity -> questions
  - Questions -> time wasted -> delays in procurement schedule -> installation delays / risk of loss of funding
- **Allow vendors time to ask questions and share most questions and responses**
  - Clarification questions can identify issues that will affect all vendors
  - Releasing benchmarks early can shake out problems before official RFP release
  - Do allow for vendor-specific queries to be kept confidential if at all possible!
- **Beware of the law of unintended consequences**
  - A requirement for more HPL performance than budget supports can lead to trouble if vendors bid what you didn’t actually want

# Why use Benchmarks in RFPs?

**Basic Aim: To measure the vendors' proposed machine capabilities in comparison to the customer's workload requirements.**

**Basic Requirement: Understand what you value and how you will score proposals, then provide the smallest set of benchmarks necessary to compare performance.**

- Keep expectations of the vendors in proportion to value of the deal

**Common Scenarios for Benchmark Use in RFPs:**

- As **a hurdle** to limit responses from non-HPC savvy vendors
- To **enable evaluation** of offered systems and their capability to handle expected workloads.
    - Sometimes just a simple evaluation of performance of proposed hardware
    - If optimizations are allowed, can also evaluate vendors' support capabilities with eye to support post-delivery
- To **design and size the system** required to run the workload

# Evaluation Methods and Metrics

- **A clearly defined evaluation metric is important so we understand where to target performance and what you value**
- **Also important to understand how highly benchmarks are weighted in overall scoring**
  - Are benchmarks a very small proportion of the total final score?
  - Will HPL Rmax determine system size, regardless of benchmark performance?

- *Beware of benchmark requirements that have nothing to do with the purpose of the machine (e.g., if you need a lot of network, don't just use low node count benchmarks).*
- *If the workload is known to be memory bandwidth limited, maybe include codes similar to STREAM (or weight them highly) and exclude things like SPEC (mostly clock bound).*
- *Consider a benchmark such as GPCNet to get a measure of ability of system to handle congestion on the network*

# Evaluation Metrics – common scenarios

- Simply **run and report performance** (often used as a barrier to entry)
- Run each benchmark test in **under a specified target time** (makes most sense in cases such as operational weather with predefined constraints)
- **Evaluate applications individually** (relative to each vendor) Often includes an evaluation of scaling performance up to system size or scaling limit
- **Throughputs**
  - A well thought out throughput mix can be a useful tool and help evaluate I/O performance
  - Throughput metrics are tough for vendors to model and require additional work, so should ideally displace other benchmarks
- **Weighted metrics** (often referred to as SSI or SSP – sustained system performance)
  - Bundle mix of applications and kernels (don't just use small kernels)
  - Weight each one appropriately for workload priorities
  - Create single metric for easier evaluation (often done with Geomeans)
  - Can allow variation within mix at acceptance- especially good for future hardware

# Projections and Estimates

- **Projections are essential for any system with hardware not yet available or for system sizes beyond what is available**
- **How to ensure vendors know what they are doing?**
  - Prior record
  - Good explanation of methodology (but don't expect full details)
  - Good relationships
  - Full commitments to proposed performance
- **Decide whether to allow processor or interconnect vendors to supply benchmark results**
  - This can lead to identical results submitted by multiple OEMs
  - Requiring that OEM runs benchmarks can demonstrate potential for support in the future
  - Who will estimate future system performance and commit to it?
- **Be careful of applications that have RNG or iterative solvers**
  - Need iteration counts to be consistent from run to run
  - If have to scale out to higher core counts, must know number of iterations for reliable projection

# Optimization

- **Best to allow optimization with guidelines,** such as:
  - Specify types of optimizations allowed (I/O, communications., OpenMP, etc.)
  - Specify that scientific validity of results should not change
  - Don’t allow optimizations that are specific to benchmark problem itself
  - Require vendor to supply full details of all optimizations made
  - Retain ability to reject optimizations if are too complicated etc.
- **Legacy apps often just don’t scale up efficiently without being adapted to current or future hardware** (processor types, node counts, and networks)
- **Optimizations allows ability to evaluate full potential of system hardware, compiler, libraries etc.**
- **Also allows ability to evaluate vendor skill** (important if collaboration is rated)

# What benchmarkers like (and don’t like) to see…

# Do….

- **First, figure out what you want**, e.g., "the fastest running job, no matter how many nodes it takes", or "maximum number of jobs on the system"?
- **Make benchmark instructions clear**
  - Check that README does not conflict with main document
  - Get directions and files tested by people not involved in the benchmark preparation before you release them to vendors.
  - Remember that your working directory is not a benchmark distribution!
- **Supply validation requirements and make sure they are also clear**
  - e.g. “WRF output should match to within 5%” is not clear
- **Watch run length!!!** A good benchmark will run for 5 to 60 minutes.
  - Under 1 hour allows us more time to debug, optimize and find the best way to run your applications. But….sub-10 second runs aren’t very useful ☺
  - If you shorten a run, consider evaluating only the post-initialization portion
  - Decent problem sizes will differentiate vendors better

# More Do…

- **Set an appropriate deadline for getting results returned**
  - Allow enough time for the vendor to do the work
  - More complicated RFPs take more time
  - If the time is too short, the quality of response goes down
- **Remember the impact of year end holidays**
  - Releasing an RFP in early December and asking for response in early January will not get you good results
- **Make sure any penalties around missing performance targets are clearly defined in the RFP document** (we need to understand risks)
- **At Acceptance, be pragmatic about meeting targets**
  - If the system hardware was not yet in production when estimates were made, must expect some variation in actual performance. Measures like SSP help with this.

# Don't…

- **Don't add too many requirements that restrict how benchmarks can be run**
  - For example, don't specify number of MPI ranks / OpenMP threads to be used
  - Allow vendor flexibility to demonstrate best way to run app on proposed architecture
  - Don't assume anything about numbers of CPUs, cores, accelerators per node (unless they are mandatory requirements for system). This often occurs when too focused on existing system
  - Allow the use of multiple compilers/MPIs etc.
- **Don't ask for large numbers of commitments for no clear purpose**
  - Only ask for numbers that are clear to interpret and are useful
  - Is easy to ask for results for a huge variety of MPI tests, but hard to understand what the results mean for the real work. And hard for the vendor to provide them

# More Don't…

- **Don't expect output to be bit identical to that from another system**
  - How much precision do you really need in your results? If input data are based on measurements with 3 significant digits, don't ask for 14 digits of accuracy in comparison to data from original system. Determine what a scientifically valid result is and ask for that.
  - If identical runs must give identical output, say so. If runs must give identical output across all rank and thread counts, say so.
    - Code must be written to be bit reproducible in the first place
    - This can limit optimizations possible
- **Don't require huge amounts of output data to be returned**
  - Will you really look at all of it? Can you look at output from just the final step/iteration?
  - Can you provide a tool that can postprocess the data before return?
  - Large return data requirements can add up to a week to write a drive then ship, which leads to requests for extension or less time available to dedicate to actual benchmarking work

# In Conclusion

- Define your workload before designing the minimal set of benchmark tests to reflect that workload
- Write the RFP benchmark requirements as clearly as you can, and get them tested before releasing to vendors
- Define a clear evaluation metric to enable valid comparison among vendors and to ensure you end up with the system you want
- Allow vendors to show what their proposed system can do to help your scientific workloads perform as well and as efficiently as possible

QUESTIONS?

a Hewlett Packard Enterprise company