Los Alamos
NATIONAL LABORATORY

EST. 1943

NNSA
National Nuclear Security Administration
Operated by Los Alamos National Security, LLC for the U.S. Department of Energy's NNSA

Los Alamos
NATIONAL LABORATORY
EST. 1943

Delivering science and technology
to protect our nation
and promote world stability

LA-UR-20-26663

A Cosine Similarity Methodology to Characterize Proxy-Parent Application Correspondence

Jeffery A. Kuehn (LANL)
Jeanine Cook (SNL)
Omar Aaziz (SNL)
Courtney Vaughan (SNL)
Jonathan Cook (NMSU)

Sept 1, 2020

Why Proxy Applications?

Real Applications	Proxy Applications
100 KLOC – 1 MLOC	1 KLOC – 100 KLOC
Large number of library dependencies	Minimize library dependencies
High Code Complexity	Simpler - Captures key kernels
May contain proprietary information	No protected IP
Licensing / Export / Classification	Freely distributable by design
Exactly what is/will be run	An approximation to the real app
Staff intensive to work with	Easier to work with

DOE Proxy Applications

- **Currently a set of ~60 proxy applications**
- **~12.5 M Lines-of-Code (LOC)**
- **Mixed languages (C/C++/Fortran/Python)**
- **Used for**
  - System acquisition benchmarks
  - Exemplars for collaborative research contracts (e.g. PATHFORWARD)
  - System Testing
  - Figure-of-Merit for comparison exercises
- **Issues**
  - Poorly understood correspondence between proxies and real app
  - Proxy may represent only selected features of the real app
  - A lot of code – down-select is required for most exercises
  - Current down-select process is heavily subjective

Pick Your Proxies Carefully

- **Imagine two vendors offering the following alternatives to your current system:**
  - Offer #1: has twice the peak floating point performance, but is otherwise similar to your current system
  - Offer #2: has twice the memory bandwidth, but is otherwise similar to your current system
- **The benchmarks you choose to compare these alternatives will determine which system your purchase.**
  - Dense linear algebra will select Offer #1 because it has a higher peakFP, but you already knew that
  - Streaming benchmarks will select Offer #2 because it has a higher memBW, but you already knew that
  - The wrong choice of benchmark could cost you 2x in capability
    - Performance-metric space is two-dimensional (peakFP & memBW)
- **Which is more similar to your WORKLOAD?**

How to down-select in a more data-driven way?

- **Codes: Large in number, huge in size, byzantine in complexity**
- **Limited resources skilled in performing the analysis**
- **Deep analysis and simulation efforts are both time and labor intensive**
  - These will still be needed but they need to be focused, preferably on a smaller amount of code
- **How to quickly determine which proxies are most similar to the workload?**
  - Insight: Think of “performance” as the interaction between a workload and a particular device’s unique set of resource constraints
    - **The manner and proportion to which those resource constraints are exercised by a particular workload becomes a “fingerprint” for that workload**
    - **It follows that workloads with similar fingerprints will respond similarly to small relaxations of the resource constraints**
      - e.g. similarly memory bandwidth intensive codes will respond similarly to a memory bandwidth change (one-dimensional performance-metric space)

Approach

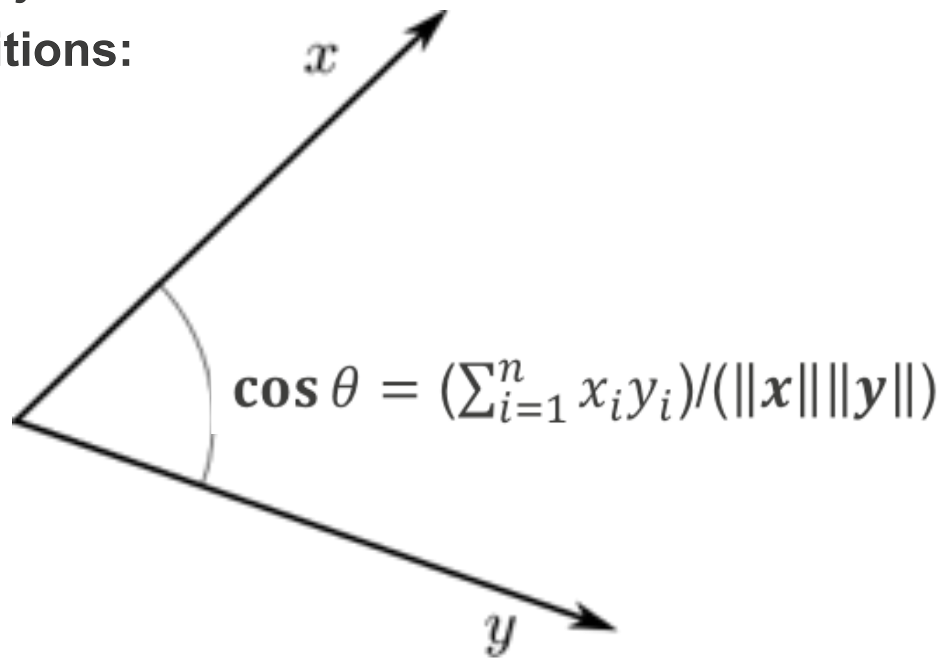
- **We rely on two elements as the building-blocks/tools:**
  - The ability to collect ”fingerprint” for a code
  - The ability to quantify a similarity comparison of two “fingerprints”

- **Desirable features for the component metrics and comparison method:**
  - Should be related to hardware constraints (limitations / rooflines / bottlenecks)
  - Should be automatically forgiving of extraneous, redundant, or missing characteristics
  - Should be raw metrics and minimum analysis
    - Help to ***focus*** the analyst’s time rather than merely ***consuming*** it

- **Both capabilities are relatively easy to provide**
  - Construct fingerprint from aggregation of characteristic metrics (e.g. processed hardware counters)
  - Comparison: treat metrics as components of a vector in a high dimensionality space (10’s to 100’s of metrics) and compare the angle between these vectors
  - Can extend to include addt’l counters, different hardware, different compilers, etc

What is Cosine Similarity?

- **Term is taken from the ML community, but really just a property of dot (inner) product in vector spaces in 2 or more dimensions**
  - Think: "Projection of $\boldsymbol{x}$ in the direction of $\boldsymbol{y}$"
- **From the two complementary definitions:**
  - Algebraic: $\boldsymbol{x} \cdot \boldsymbol{y} = \sum_{i=1}^{n} x_i y_i$
  - Geometric: $\boldsymbol{x} \cdot \boldsymbol{y} = \|\boldsymbol{x}\| \|\boldsymbol{y}\| \cos\theta$
  - $\cos\theta = (\sum_{i=1}^{n} x_i y_i) / (\|\boldsymbol{x}\| \|\boldsymbol{y}\|)$
- **The included angle tells us about:**
  - Similarity in the direction of the vectors
    - Not their magnitudes
  - $\cos(\theta) = 0.0$ – orthogonal (never...)
    - 1st "quadrant" – non-negative components
  - $\cos(\theta) = 1.0$ – same
  - Non-unique/Non-orthogonal basis



Advantages of Cosine Similarity

- **Mathematically ”forgiving” of missing, extraneous, or redundant characteristics**
  - Non-distinguishing components are naturally suppressed (outside the plane of θ)
  - Avoids the need for a “perfect” principle component analysis.
  - A wide net can be cast, capturing a broad variety of components without fear of corrupting the results
- **Easily Extended to:**
  - Different Kinds of Metrics (Time, memory, calls, samples, etc)
  - Multiple processors (mpi or openmp)
  - Different hardware
  - Different software stacks
  - Large numbers of component metrics
  - Assess similarity across different configurations of an application or proxy

Identify how proxies or apps cluster and which proxies best represent sets of apps

	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
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
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
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
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
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
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
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
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
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
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
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
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
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
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

Clustering the Results:

- None of the proxies exhibit high similarity to App2
  - ***REPRESENTATION GAP***
- But several of the proxies exhibit high similarity to each other
  - 4,5,8,11,1 (&10,2)
  - 7,9
  - 6,12
  - ***REDUNDANCY***
- For these Apps in this basis set
  - 1 (2?) proxy was ”useful”
  - 4-6 proxies, not 12.
- More isn’t better
- ***Significant down-select possible***

- For reference:
  – $\arccos 0.28 \approx 74°$
  – $\arccos 0.90 \approx 25°$
  – $\arccos 0.98 \approx 12°$
  – $\arccos 0.99 \approx 8°$

Proxy – Parent Correspondence

Intel Skylake: Full Node Behavior

	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

Intel Skylake: Cache Behavior

	ExaMiniMD	LAMMPS	MiniQMC	QMCPack	sw4lite	sw4	SWFFT	HACC	pennant	snap
ExaMiniMD	0.00	0.29	58.53	20.78	8.33	8.19	6.30	7.66	8.39	3.67
LAMMPS	0.29	0.00	58.51	20.76	8.36	8.23	6.14	7.52	8.38	3.59
MiniQMC	58.53	58.51	0.00	39.30	51.41	51.60	56.63	54.44	51.33	55.20
QMCPack	20.78	20.76	39.30	0.00	15.18	15.33	19.19	17.91	14.56	17.76
sw4lite	8.33	8.36	51.41	15.18	0.00	0.39	10.47	9.66	3.89	6.04
sw4	8.19	8.23	51.60	15.33	0.39	0.00	10.52	9.78	3.76	6.00
SWFFT	6.30	6.14	56.63	19.19	10.47	10.52	0.00	3.14	10.42	5.18
HACC	7.66	7.52	54.44	17.91	9.66	9.78	3.14	0.00	9.82	5.26
pennant	8.39	8.38	51.33	14.56	3.89	3.76	10.42	9.82	0.00	6.01
snap	3.67	3.59	55.20	17.76	6.04	6.00	5.18	5.26	6.01	0.00

- Most proxies of high fidelity representation of parent app
  - Exception: MiniQMC is weaker overall, particularly on cache behavior
  - Exception: SWFFT is weaker overall, but represents cache behavior
- Gap: representation of HACC & QMCPack is weak, particularly QMCPack's cache behavior
- Redundancy: All proxies have similar cache behavior to parent EXCEPT MiniQMC/QMCPACK
- Working set: MD and QMC are fairly different from the rest of the tested DOE apps/miniapps
  - Be careful using MD/QMC to characterize a system for other apps

Impacts

- **Identify Redundancies in the Proxy App Suite / Benchmark Suites**
  - ➔ Lower software maintenance burden
  - ➔ Easier to work with
  - ➔ Better understanding of what aspect(s) of the parent workload the proxy represents
- **Deliver proxies that are more representative of real app behaviors to vendors**
  - ➔ systems better optimized for our apps
- **Define minimal proxy suite that covers all parent behavior**
  - ➔ faster design-space exploration
- **Can be broadly used to understand performance differences in compiler and application optimizations, application inputs/problems, kernels and systems, etc**

End