Beyond Guess and Check: Quantifying the Fidelity of Proxy Applications

Si Chen[1], Simon Garcia de Gonzalo[2], Omar Aaziz[2], Jeanine Cook[2], **Avani Wildani**[1,3]

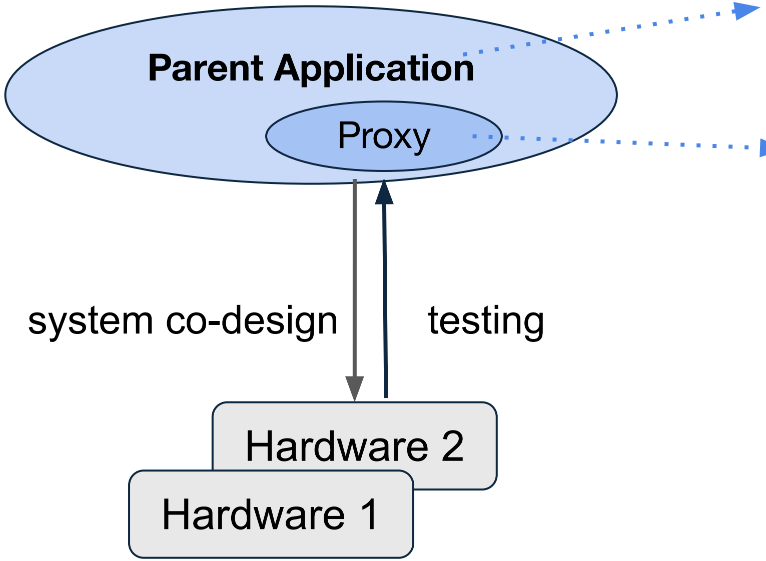
PMBS'25, held in conjunction with SC'25
November 2025

[1] Emory University

[2] Sandia National Laboratories

[3] Cloudflare

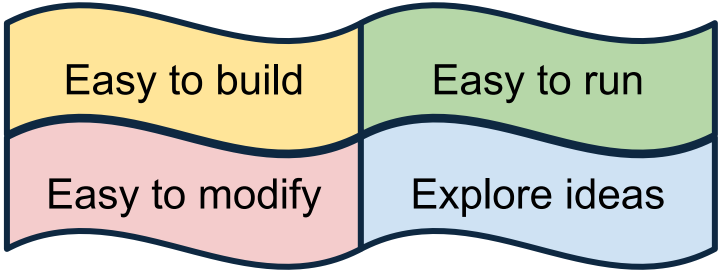
Background: Proxy Applications for co-design



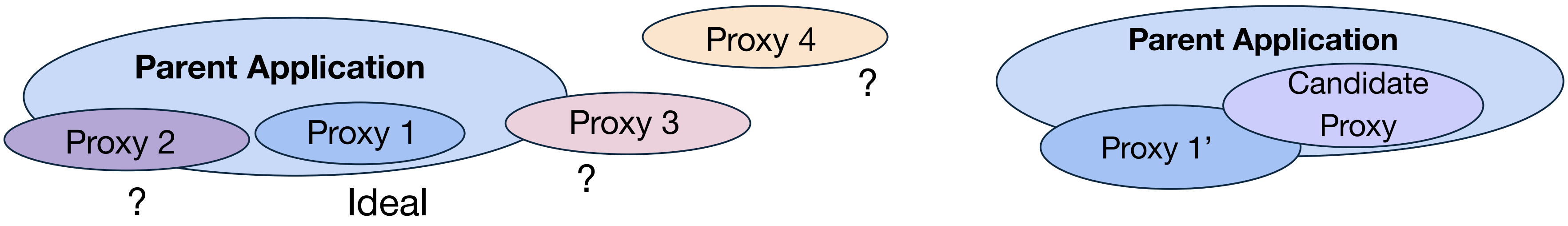
Full code bases are large, complex, private

Small, self-contained, simplified code

Advantages of Proxy Applications



Motivation: Fidelity of Proxy Applications



Fidelity: **the degree of accuracy** with which a proxy application mimics the behavior and performance characteristics of its parent application.

Why quantify the fidelity? Poor fidelity lead to suboptimal architectural decisions

Motivation

Previous Work:

- Limited number of applications
- Limited features or metrics
- Qualitative analysis
- Lack of standard metrics for proxy

Three research questions:

How to quantify **similarity**?

How to **select** a representative set of proxies?

How to effectively identify the **discrepancies**?

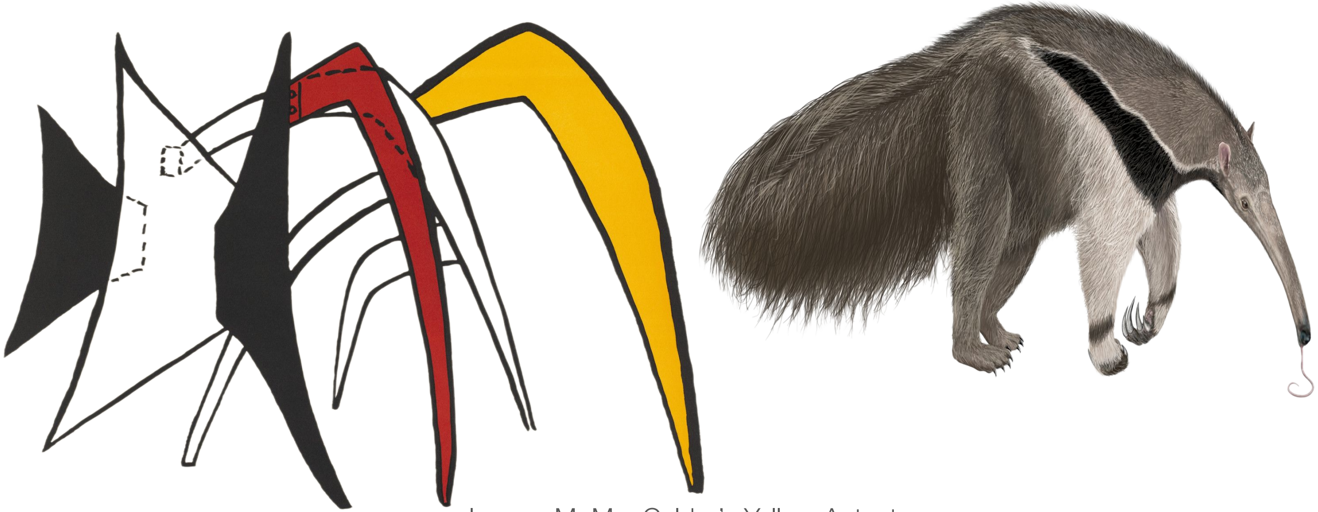


Image: MoMa, Calder’s Yellow Anteater

We introduce **Calder**, a tool kit that implements **similarity algorithms** to compare application behaviors, using **Laplacian scores** and a correlation filter to select the **most important** application behavior features.

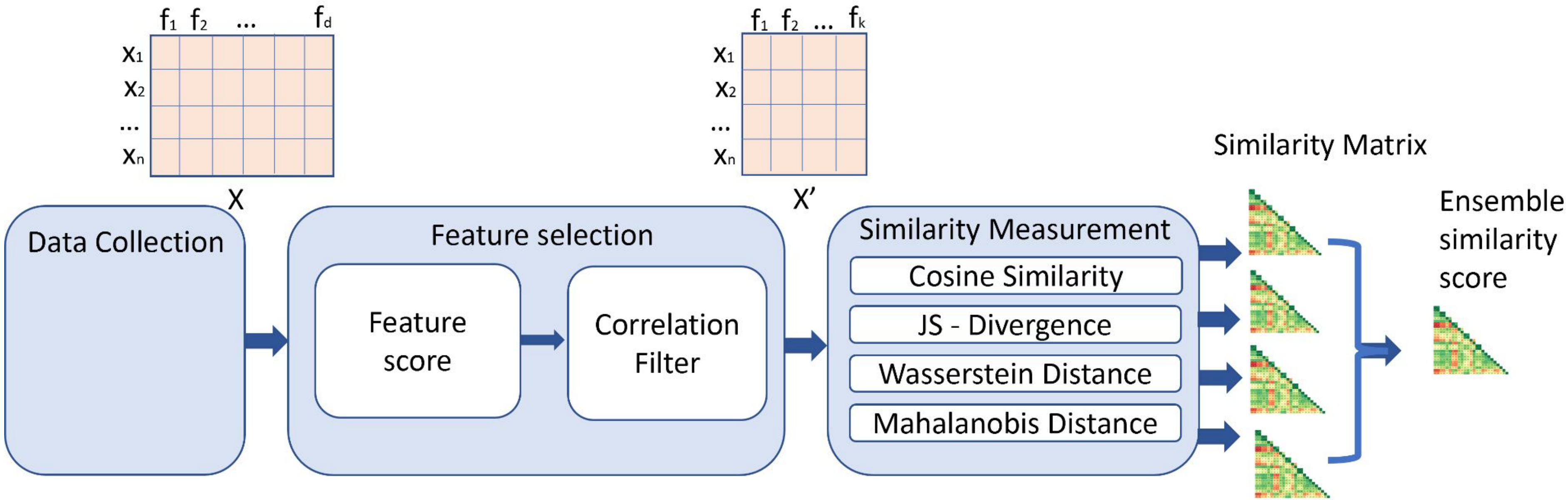
Key Contributions

- **Quantitative characterization** of fidelity across a broad range of proxy and parent application pairs.
- Advanced **feature selection** techniques that reduce data size by up to 95%.
- Usage of kernel function similarity as ground truth to **define similarity** threshold.

Outline

- Background and Motivation
- Calder Architecture
- How to quantify the similarity?
- How to select a representative set of proxies?
- How to effectively identify the discrepancies?
- Conclusion

Calder Architecture



"Features" in this talk are hardware performance counters

Data Collection

- PAPI events collected by **LDMS** infrastructure.
- 600+ hardware events
- 15 subgroups according to event categories (*e.g.*, Dispatch Pipeline, Instruction Cache)
- Input size: same problem or 50% memory usage
- MPI-only mode, 128 ranks on 4 nodes average, end time point accumulated value
- Normalized with CPU cycles

- **Hardware Platforms**
  - Intel Skylake
  - IBM Power9

To ensure robustness and reliability, each subgroup was run five times for each application, totaling over 3,000 runs.

Data Collection: Applications

Table 1: Proxy/Parent and Control Apps

Proxy	Parent	Other apps
ExaMiniMD	LAMMPS	AMG2013
miniQMC	QMCPACK	Castro
miniVite	Vite	Laghos
Nekbone	Nek5000	PENNANT
PICSARlite	PICSAR	SNAP
SW4lite	SW4	HPCG benchmark
SWFFT	HACC	HPCC benchmark
XSBench	OpenMC	

Data Collection: Applications

An application is represented by a long **vector** with multiple events.

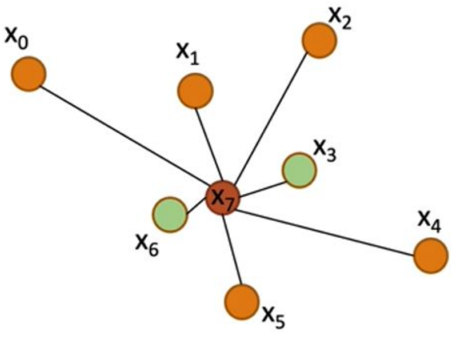
	BR_INST_RETIRED:ALL_BRANCHES	BR_INST_RETIRED:CONDITIONAL	BR_INST_RETIRED:FAR_BRANCH
ExaMiniMD	0.102941117	0.084736321	3.04E-06
LAMMPS	0.166040571	0.132466767	5.41E-06
sw4lite	0.066925525	0.042952159	1.14E-05
sw4	0.060973991	0.039039782	1.05E-05

…

…

Feature Selection: Laplacian Score

Laplacian score builds a **nearest neighbor graph** for application points and seeks those features that respect **local** graph structure to **preserve the similarity structure**.



k=2

Step 1: Derive the nearest neighbor graph

	X0	X1	X2	X3	X4	X5	X6	X7
X0								0
X1								0
X2								0
X3								$S_{3.7}$
X4								0
X5								0
X6								$S_{6.7}$
X7	0	0	0	$S_{7.3}$	0	0	$S_{7.6}$	0

Step 2: Calculate the Laplacian

$$L_r = \frac{\texttt{neighbor difference for feature } r}{\texttt{Variance of feature } r}$$

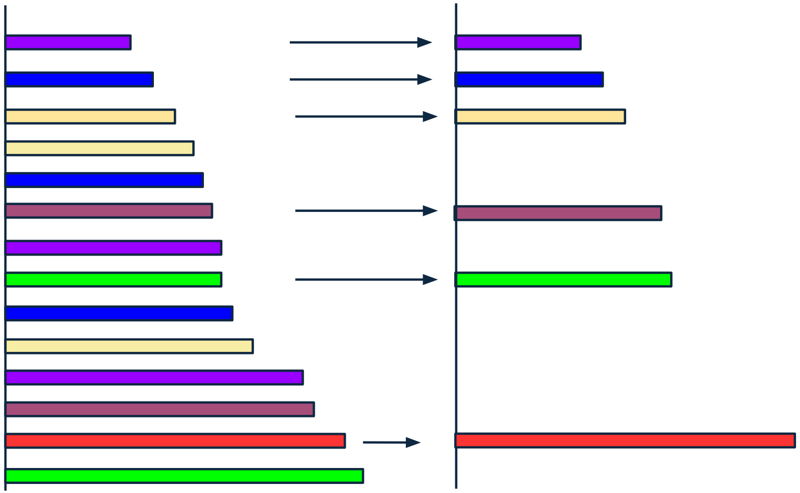
The **smaller** the Laplacian score is, the **more important** the feature is!

Feature Selection: Correlated Features

L ranks features by importance 🙂 ...but does not consider the relationship between features. 🙁

Pearson Correlation Coefficient (PCC) measures the linear correlation between two variables:

$$\rho_{X,Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$$



Ranked features with Laplacian scores

Ranked uncorrelated features

Similarity and Distance

To evaluate the similarity between applications, we calculate the pairwise distance for each application pair using four representative similarity measurement methods.

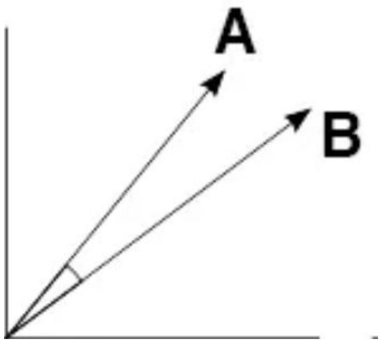
- Cosine similarity
- Jensen-Shannon (JS) divergence
- Wasserstein Distance (WD)
- Mahalanobis Distance (MaD)

Cosine Similarity

Intel Skylake

Similarity score

0	
<30	0



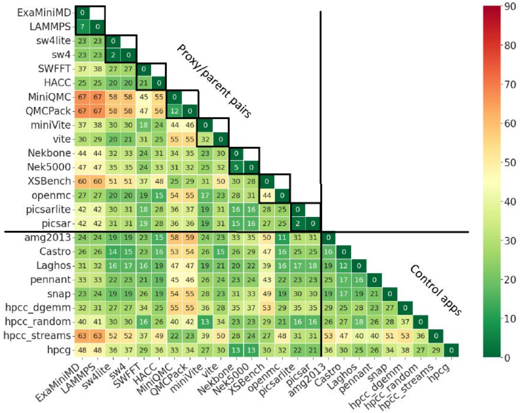
Proxy/parent pairs

Control apps

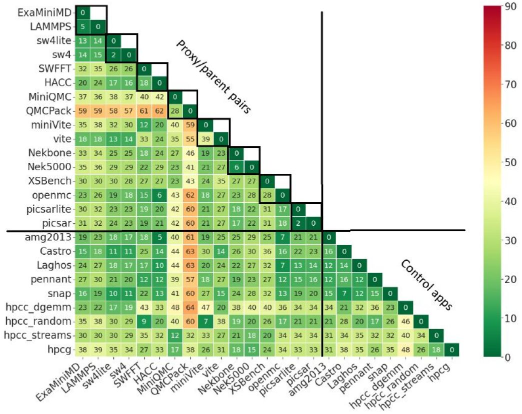
	ExaMiniMD	LAMMPS	sw4lite	sw4	SWFFT	HACC	MiniQMC	QMCPack	miniVite	vite	Nekbone	Nek5000	XSBench	openmc	picsarlite	picsar	amg2013	Castro	Laghos	pennant	snap	hpcc_dgemm	hpcc_random	hpcc_streams	hpcg
ExaMiniMD	0																								
LAMMPS	4	0																							
sw4lite	17	17	0																						
sw4	16	16	2	0																					
SWFFT	34	35	26	26	0																				
HACC	15	15	15	14	23	0																			
MiniQMC	83	83	73	73	58	73	0																		
QMCPack	84	84	74	75	60	75	11	0																	
miniVite	33	33	28	28	20	24	62	66	0																
vite	36	35	25	26	31	30	58	60	35	0															
Nekbone	56	56	45	46	35	47	31	35	40	30	0														
Nek5000	62	62	50	51	39	52	27	30	44	35	6	0													
XSBench	73	73	65	65	48	64	30	36	45	59	37	36	0												
openmc	17	17	14	14	23	10	72	75	19	28	46	51	61	0											
picsarlite	47	47	37	38	23	37	40	42	28	31	22	24	33	35	0										
picsar	47	47	37	38	24	37	39	42	29	31	21	24	33	36	2	0									
amg2013	16	16	14	14	25	10	75	77	23	28	47	53	65	6	38	39	0								
Castro	24	23	13	15	23	16	65	67	26	15	36	41	60	15	30	31	15	0							
Laghos	29	30	17	18	18	21	57	60	23	16	29	34	51	19	22	23	20	10	0						
pennant	32	32	23	24	20	22	55	59	28	19	30	34	53	24	26	26	26	16	13	0					
snap	18	18	15	15	26	16	68	70	28	23	42	47	63	16	35	35	16	14	17	20	0				
hpcc_dgemm	26	26	24	25	33	23	70	72	34	31	46	51	65	25	40	40	25	24	27	26	24	0			
hpcc_random	39	40	32	32	18	30	52	55	15	36	34	37	37	27	19	19	30	30	22	28	31	38	0		
hpcc_streams	79	79	68	69	51	69	18	21	58	56	29	24	32	68	38	37	71	61	53	51	65	68	49	0	
hpcg	66	66	55	56	46	58	31	36	51	35	16	15	43	56	34	33	57	45	39	39	51	55	45	31	0

90
80
70
60
50
40
30
20
10
0

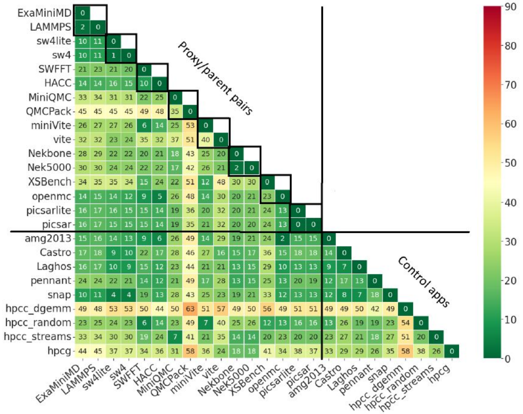
How to quantify **similarity**?



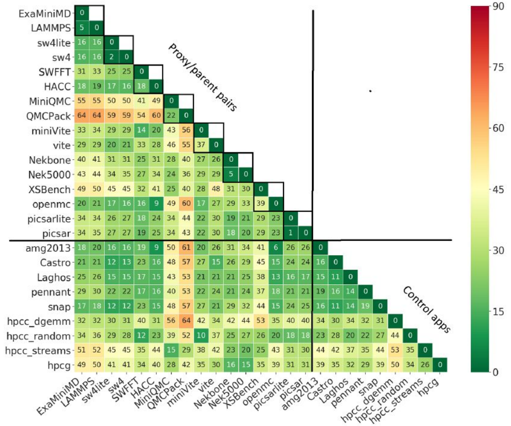
(a) JS divergence (100×)



(b) Wasserstein distance (1000×)



(c) Mahalanobis distance (10×)



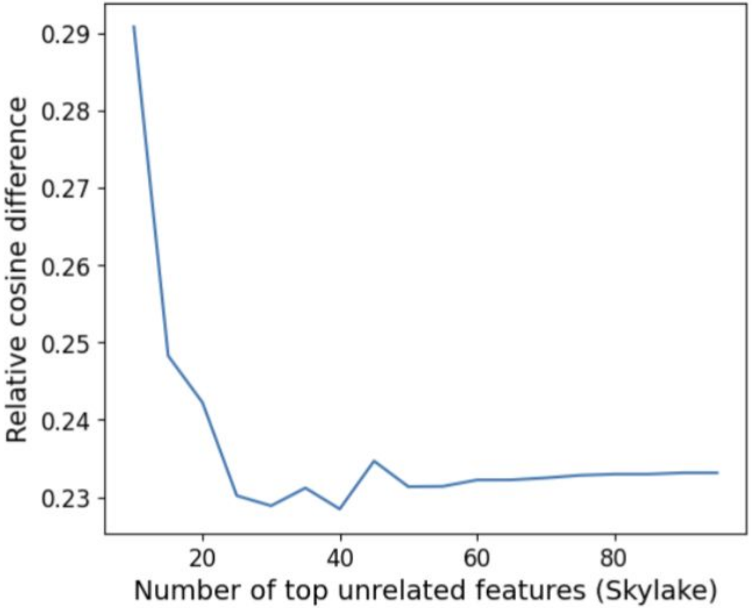
(d) Similarity for ensemble methods

Similar proxy/parent pairs maintain consistency across similarity algorithms.

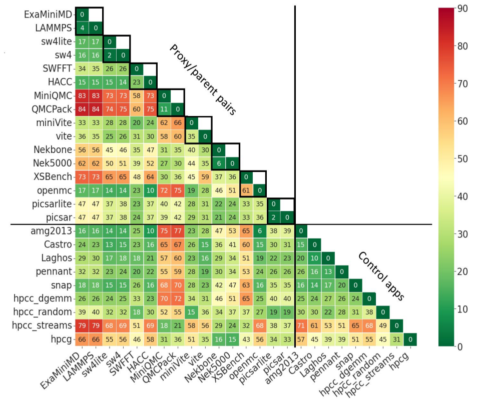
Dissimilarities are algorithm-dependent.

We select **cosine similarity** for validating fidelity.

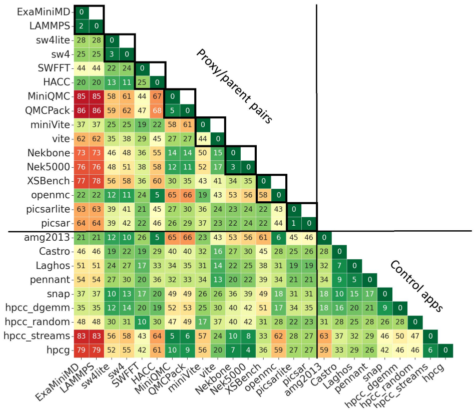
Feature sensitivity and Top Features



Cosine Similarity



All features

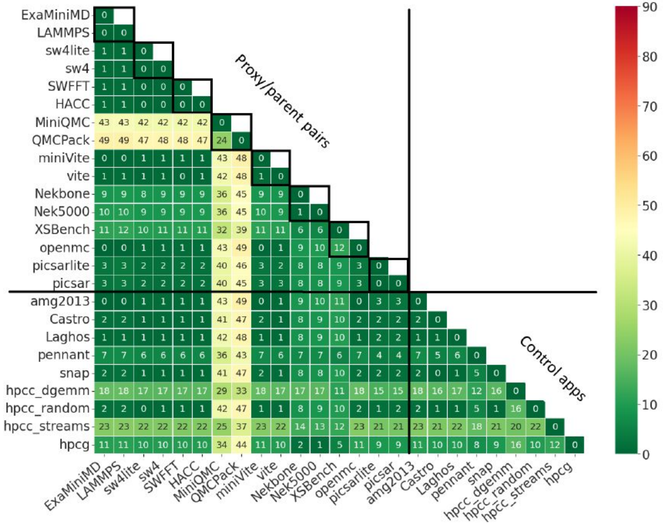


Top 25 unrelated features

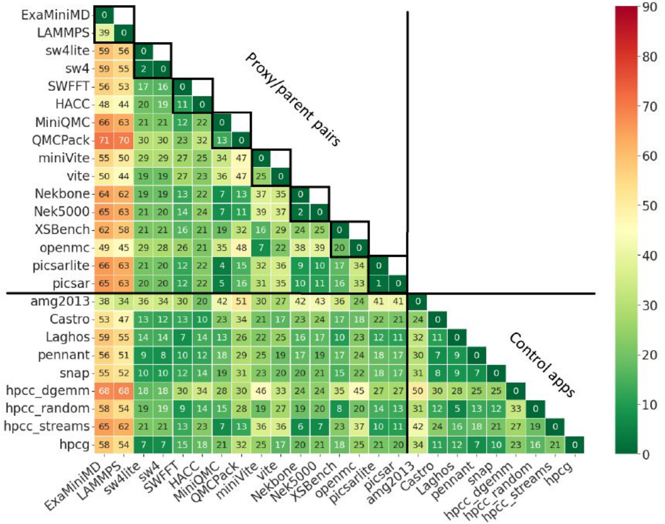
Subgroup Features Similarity

How to **select** a representative set of proxies?

Help proxy developers to observe the behavior differences to tackle code changes better and produce sounder proxies.



(a) L1 Cache



(b) Memory Pipeline

Feature Standard Deviation

How to effectively identify the **discrepancies**?

The variability of hardware performance counters (features) during runtime

Substantial feature-level deviations often reflect deeper behavioral divergence.

Table 4: Dissimilarity Feature Source for Proxy/Parents Pairs

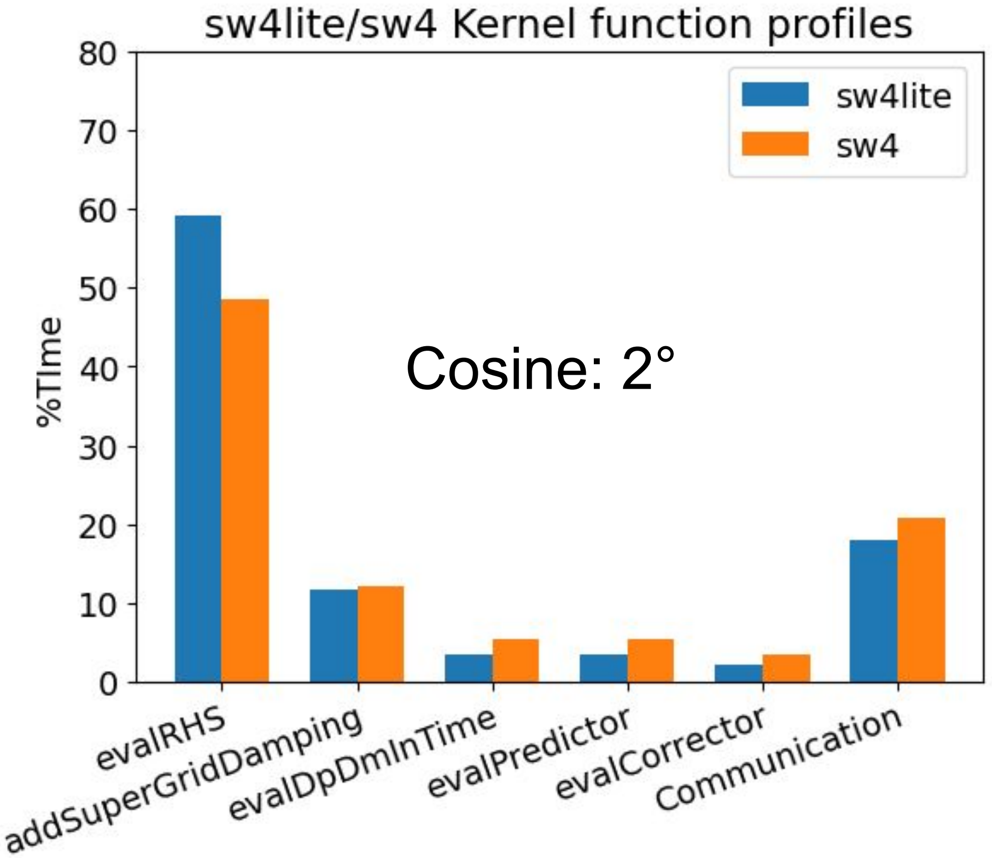
Proxy and Parent pairs	**>2std**	**>3std**	**>4std**	**>5std**
ExaMiniMD / LAMMPS	10	8	8	4
SW4lite / SW4	1	1	1	1
SWFFT / HACC	17	12	11	8
miniQMC / QMCPACK	13	10	8	6
miniVite / Vite	72	38	23	21
Nekbone / Nek5000	11	6	2	2
XSbench OpenMC	16	9	9	8
PICSARlite / PICSAR	1	1	1	0
Unique feature #s	99	64	49	38


SO?

Validating Similarity: Root Cause Analysis

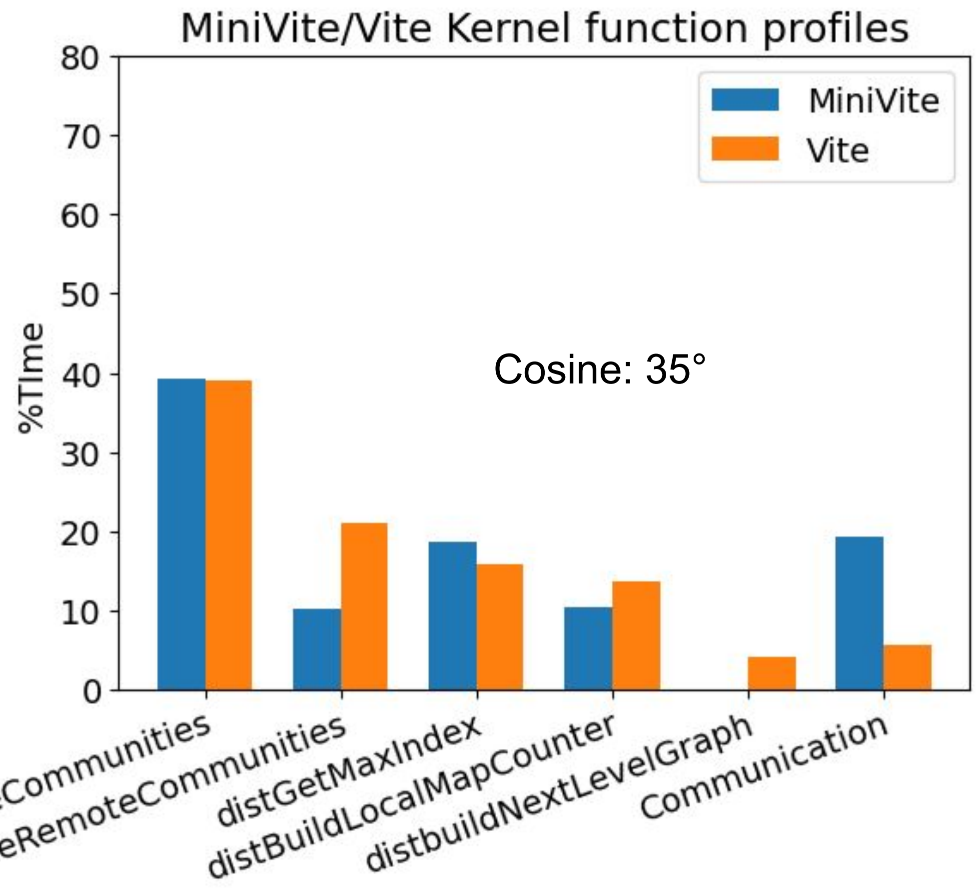
Ground truth for proxy-parent similarity: code base implementations

Kernel function profiles (normalized execution time) collected with gporf.



Less Rosy Root Cause Analysis

We recommend a general **similarity threshold** of **30°** for cosine similarity.



Conclusion

Calder **identifies** representative features, **minimizing** data collection overhead, and **highlights** key dissimilarities to **guide** future proxy co-design.

How to quantify **similarity**?

Cosine Similarity

How to **select** a representative set of proxies?

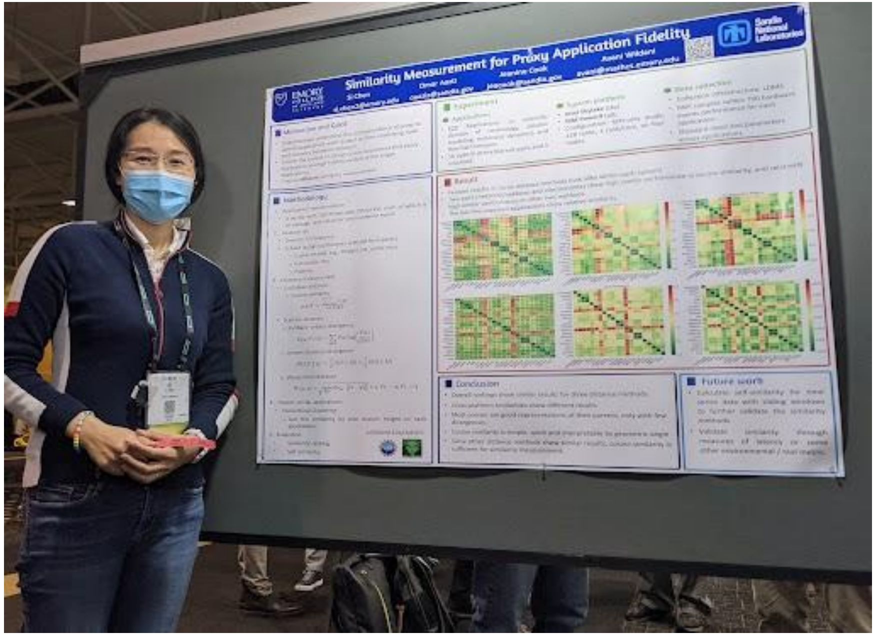
Subgroup Features

How to effectively identify the **discrepancies**?

Dissimilarity

Thank you!

GitHub



CV of Dr. Si Chen

meditates@gmail.com