On Similarity of Computational Kernels in our Codes and Proxies

Workshop on Reproducible and Sustainable HPC Benchmarking | ICS 2026

CASC
Center for Applied Scientific Computing

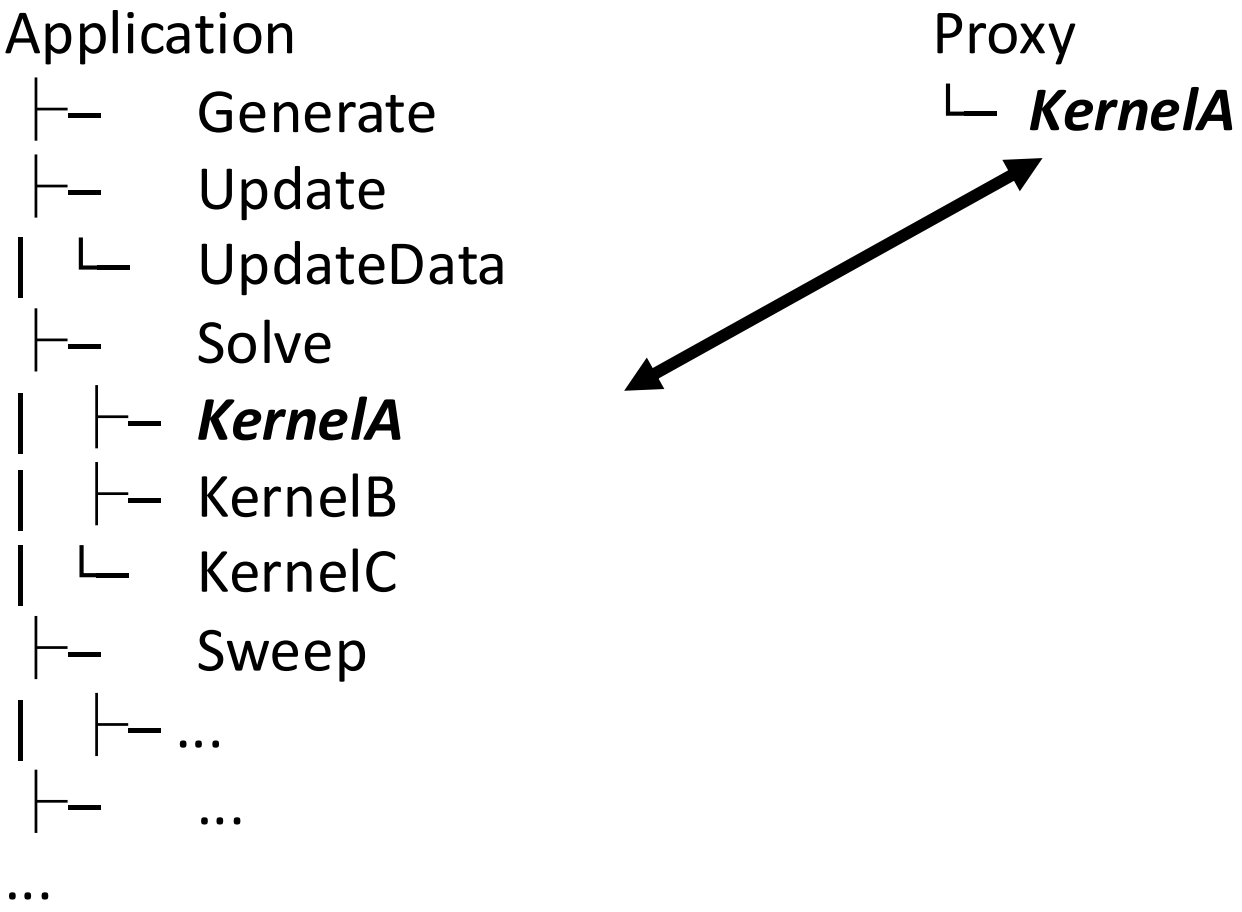
July 6, 2026

Michael McKinsey, Stephanie Brink, **Olga Pearce**
pearce8@llnl.gov

LLNL-CFPRES-2008398
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

Lawrence Livermore National Laboratory

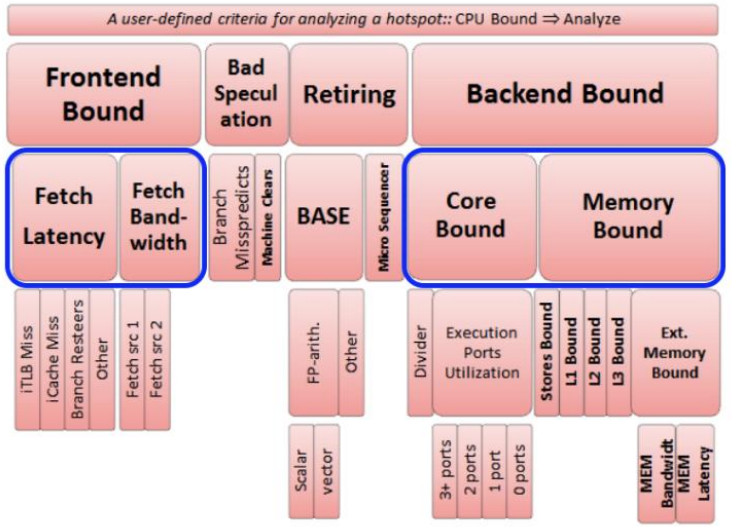
Use benchmarks to evaluate hardware and procure systems



Proxies enable running subsets of computational kernels from complex applications.

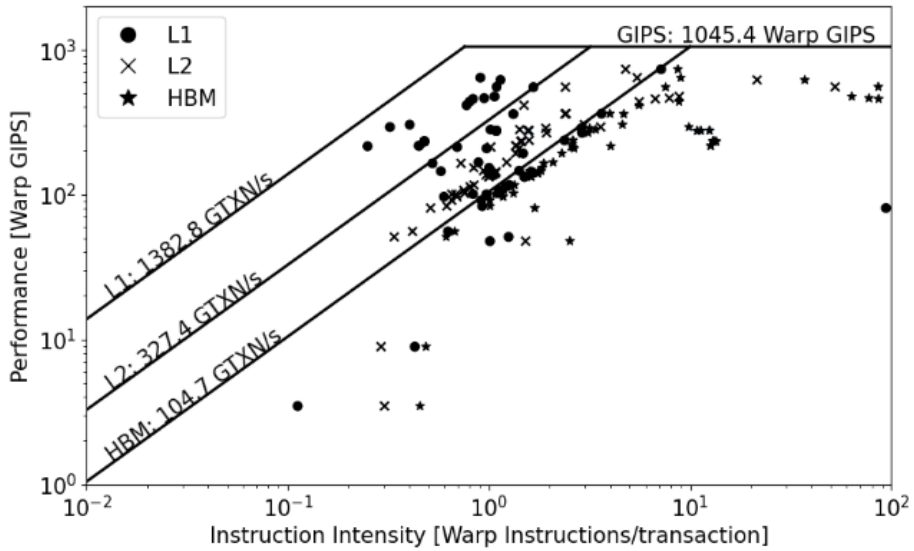
Use hardware metrics to evaluate similarity (runtime sensitive to problem size)

Intel's Top-down analysis [1]
Intel Sapphire Rapids CPU



- Core bound – stalls from execution starvation
- Memory bound – stalls related to the memory subsystem
- Fetch Latency – corresponds to instruction cache misses
- Fetch Bandwidth – inefficiency in the instruction decoders

Roofline analysis [2]
NVIDIA H100 GPU

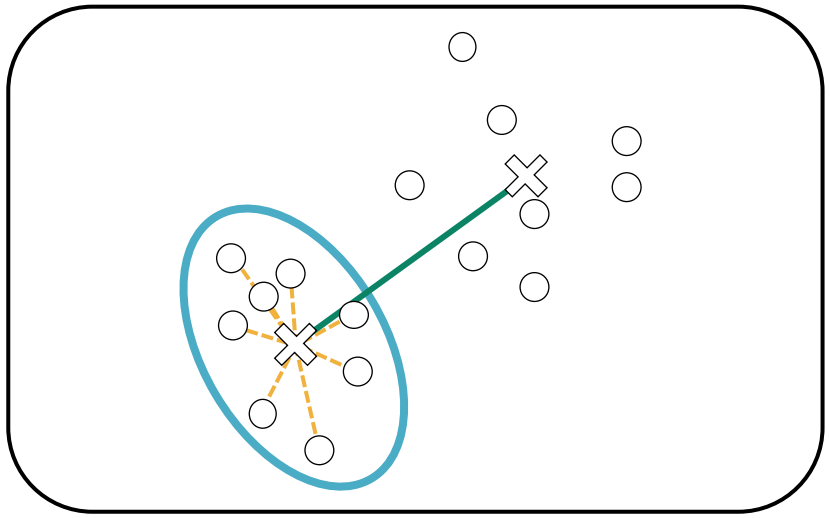


- L1 Transaction Rate = $L1_{Transactions,total}$ / $Time_{GPU}$
- L2 Transaction Rate = $L2_{Transactions,total}$ / $Time_{GPU}$
- HBM Transaction Rate = $HBM_{Transactions,total}$ / $Time_{GPU}$
- Performance (IPS) = $Instructions_{warp}$ / $Time_{GPU}$

[1] Ahmad Yasin. 2014. A Top-Down Method for Performance Analysis and Counters Architecture. doi:10.1109/ISPASS.2014.6844459
[2] Nan Ding and Samuel Williams. 2019. An Instruction Roofline Model for GPUs. doi:10.1109/PMBS49563.2019.00007

Does the proxy represent computational kernels from the application?

$$d(p, q) = \sqrt{(p_{corebound} - q_{corebound})^2 + \cdots}$$



Euclidean distance $d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$

where p_i is the value of the *i*-th metric in the hardware metrics list for kernel *p*

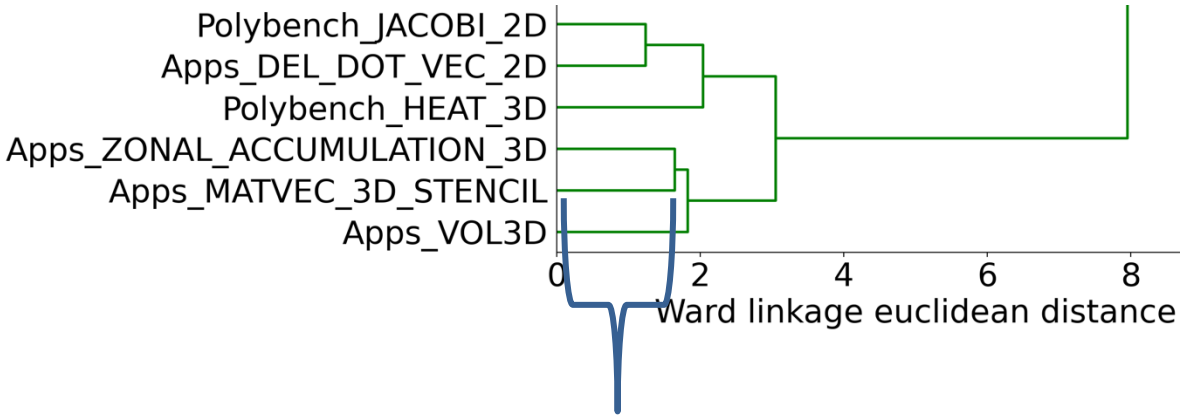
1. Partitioning – are kernels in the "correct" cluster
2. Compactness – The average *Euclidean distance* of each kernel to the cluster centroid
3. Separation – The average *Euclidean distance* between cluster centroids

Quantify similarity using partitioning, compactness, and separation

We use clustering to group kernels with similar performance

Agglomerative (ward linkage)

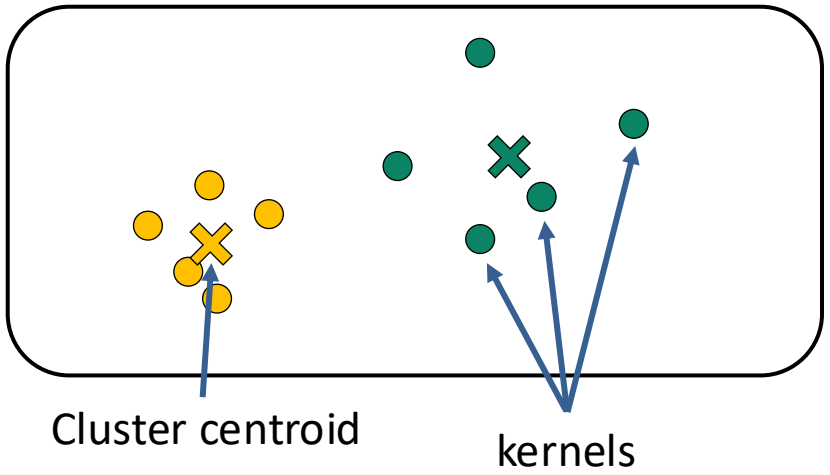
Create clusters “bottom-up” minimizing within-cluster variance



Distance at which “clusters” are merged. The first merge level indicates the two most similar kernels.

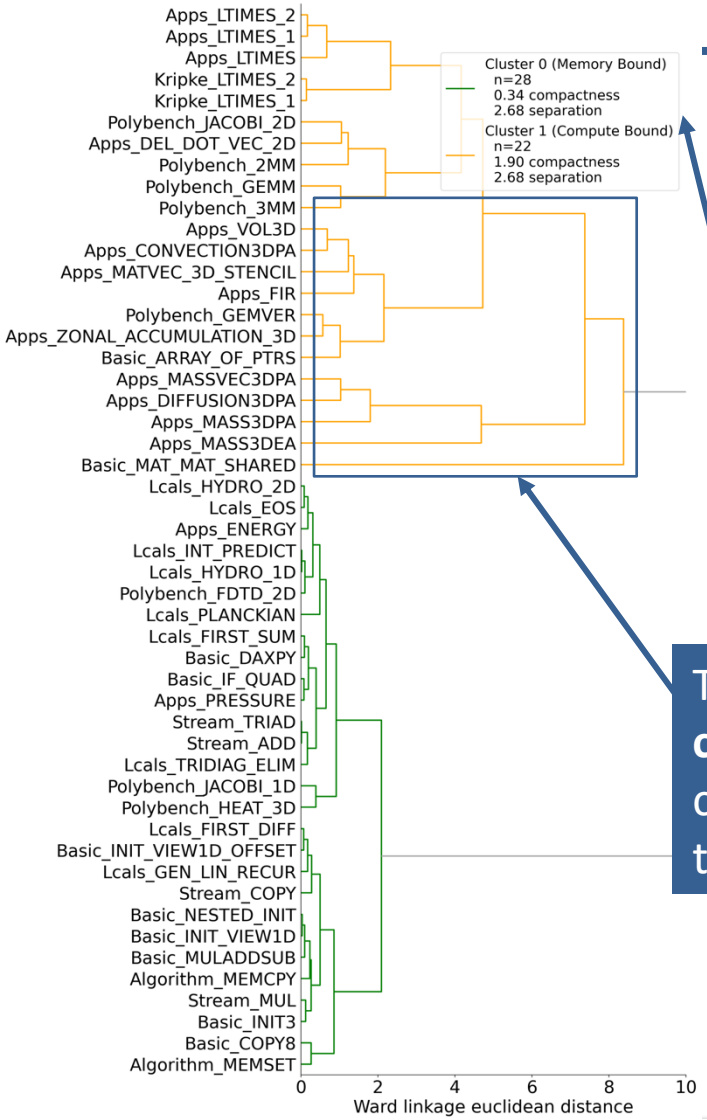
K-means (Lloyd’s algorithm)

Create clusters iteratively updating centroids to minimize within-cluster variance



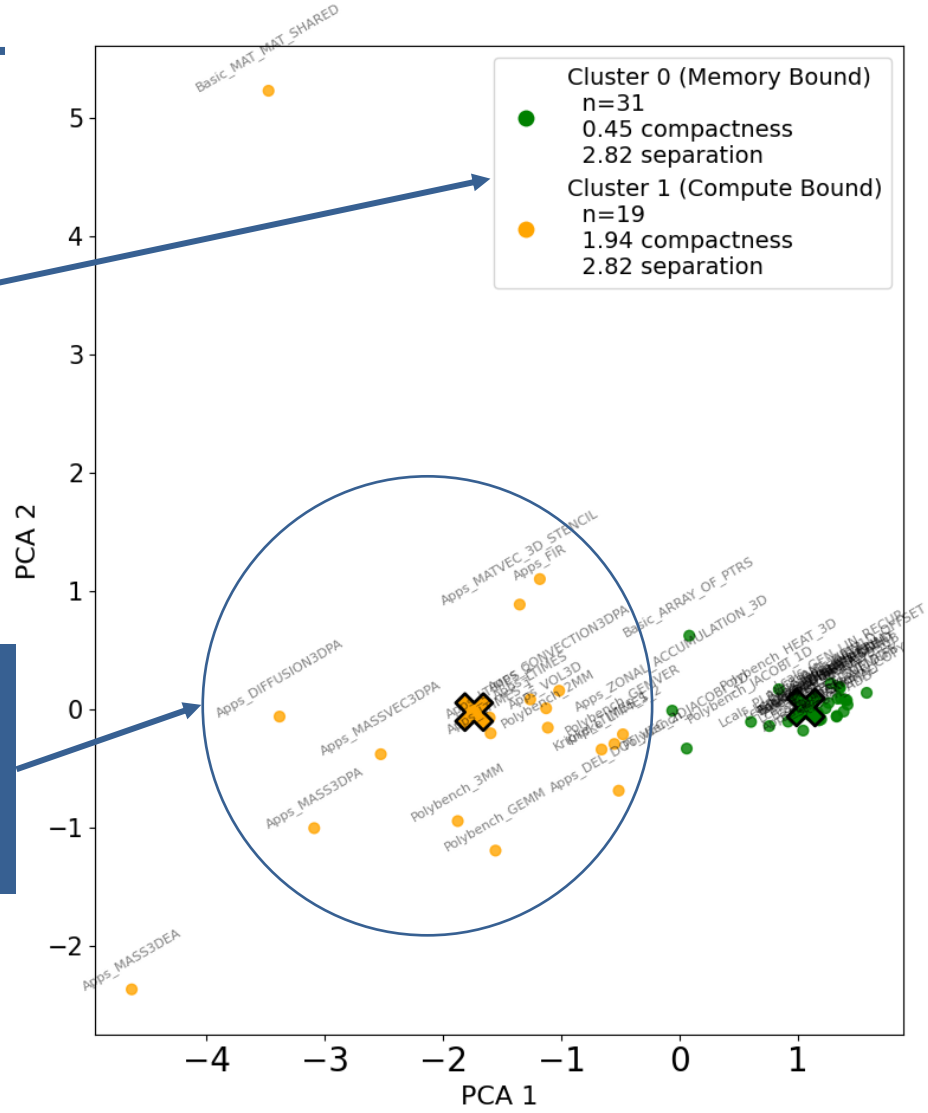
Do different clustering algorithms group kernels the same way?

Performance similarity on CPUs: Memory bound vs. Compute (core) bound

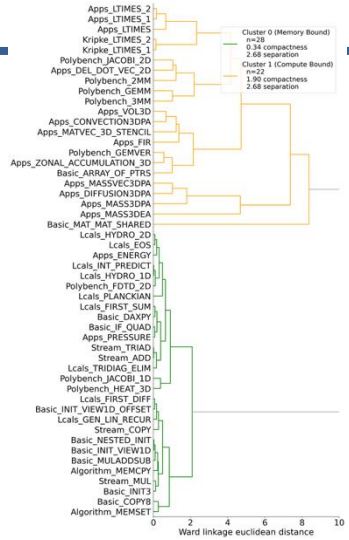


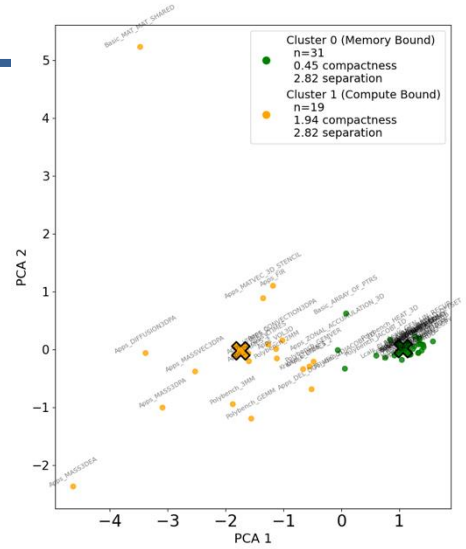
Partitioning of kernels is different for each clustering method, indicating that this subset of metrics alone may not be sufficient to categorize all the kernels.

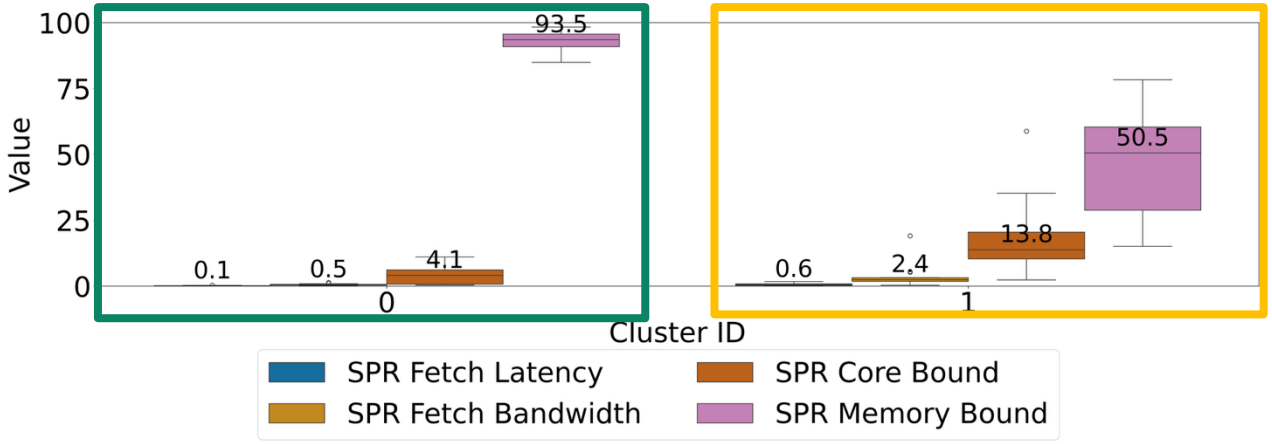
The compute bound cluster is ***not* compact**, indicating the kernels in this cluster are not as similar as the kernels in the memory bound cluster.

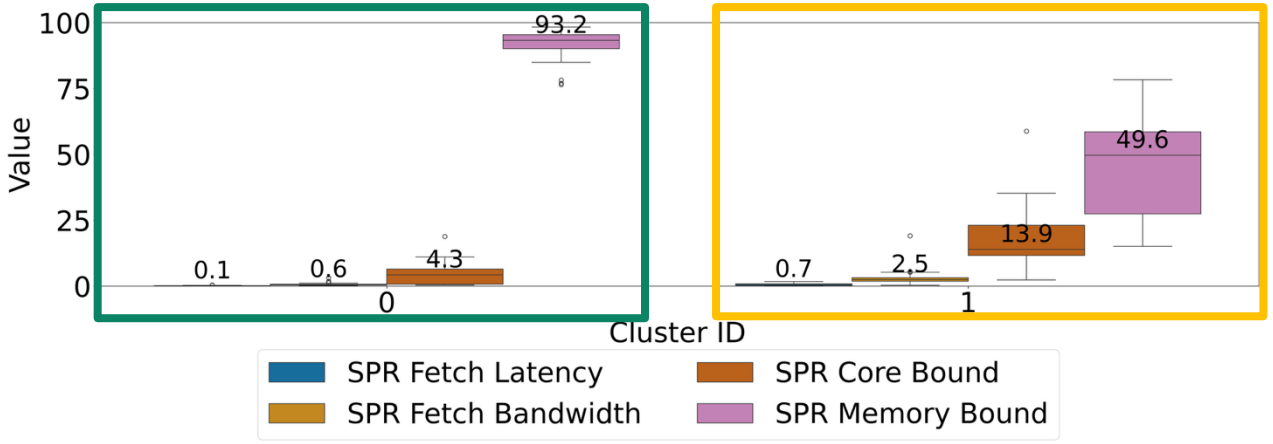


Performance similarity on CPUs: Memory bound vs. Compute (core) bound



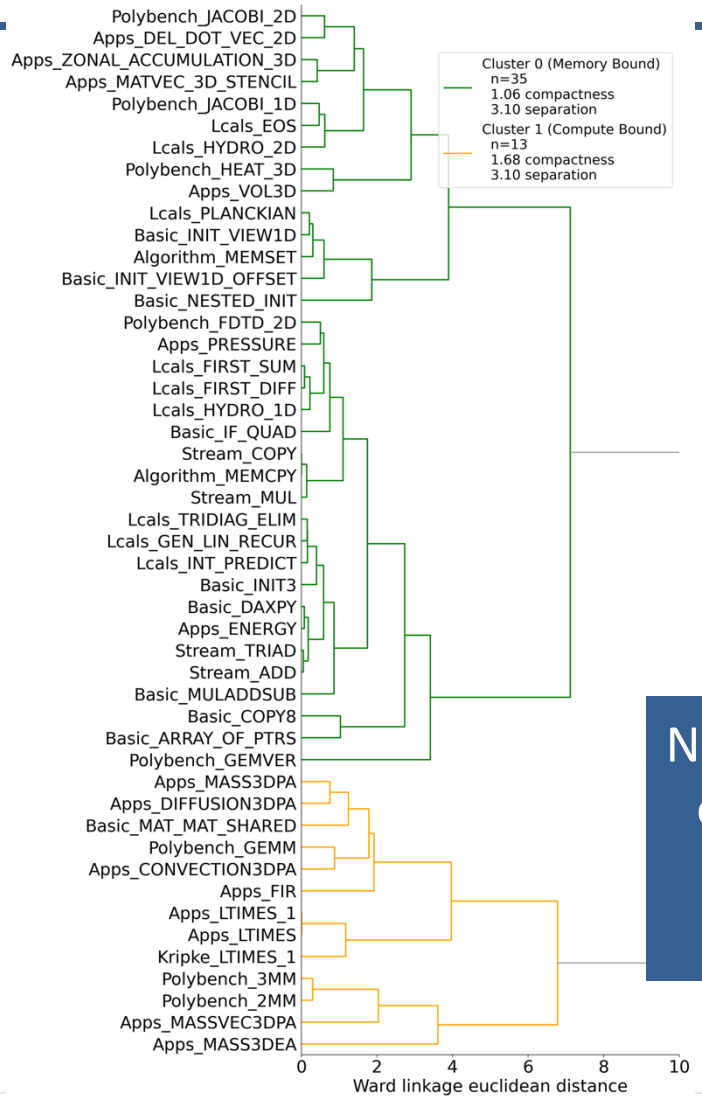






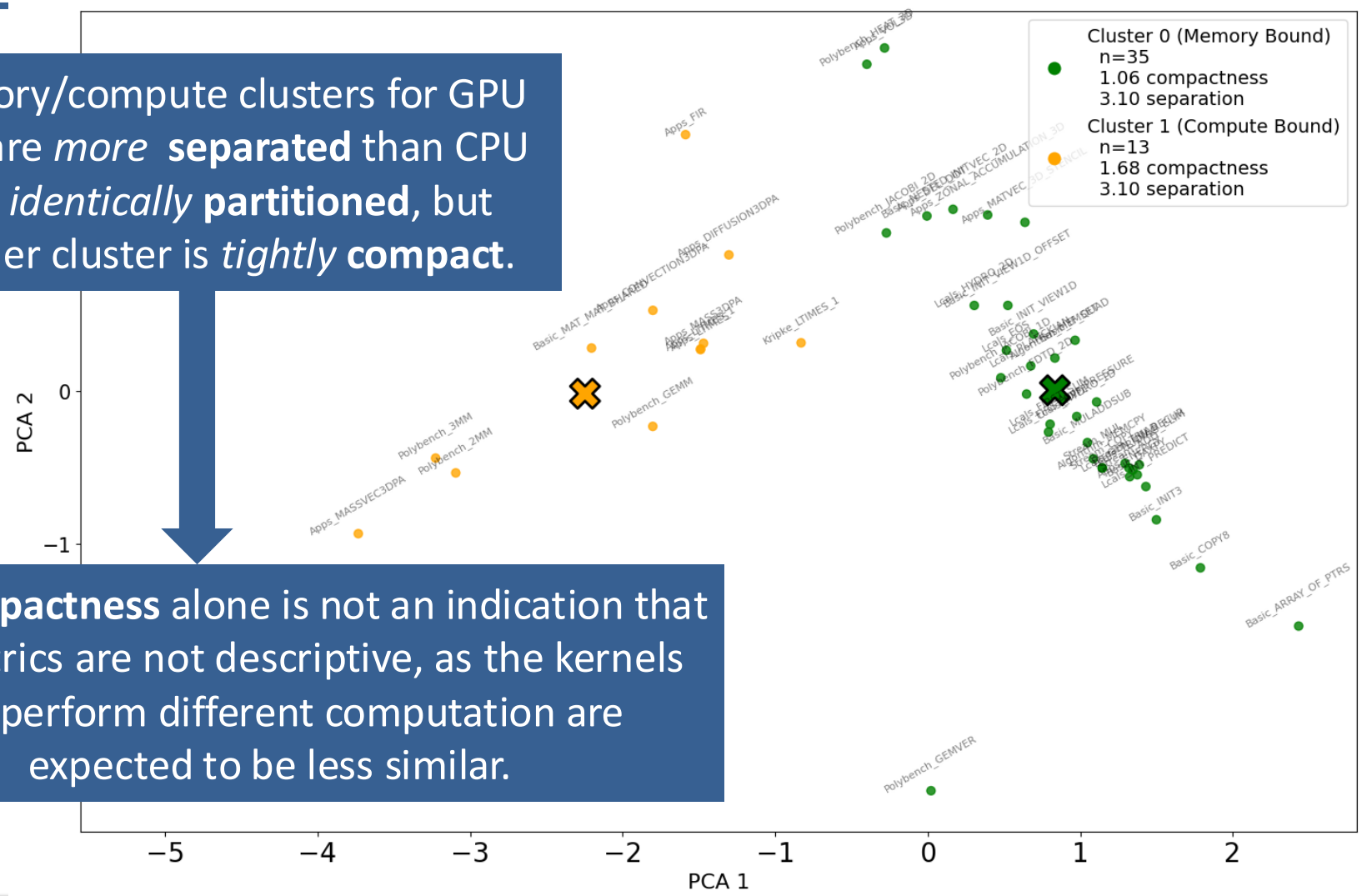
Memory bound kernels characterized by a high memory bound metric value (>90%). Compute bound kernels have significantly higher relative core bound values (~3x) along with higher fetch bandwidth/latency (~4-7x).

Performance similarity on GPU – memory (L2/HBM transaction rate) bound kernels versus compute (IPS) bound



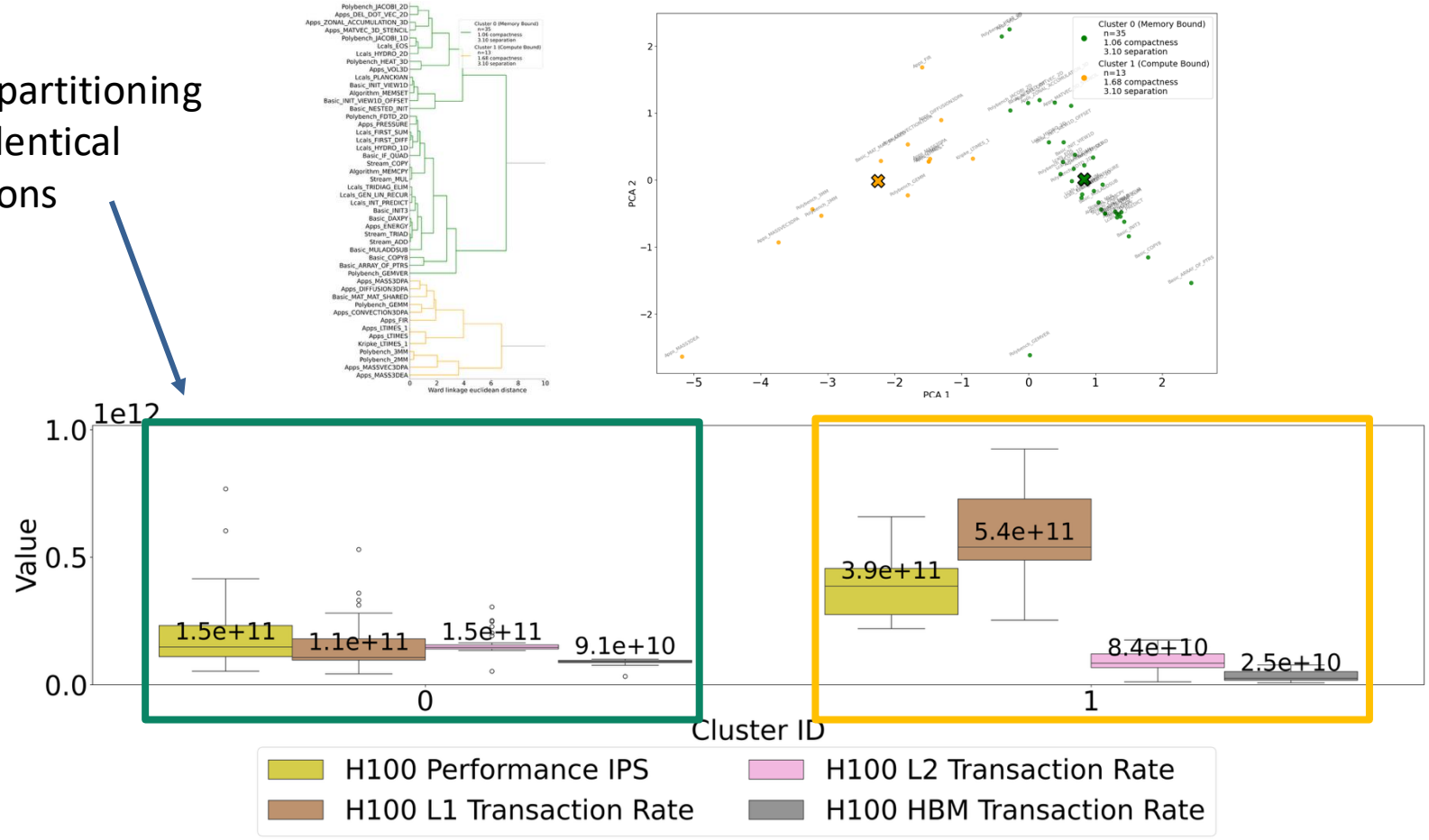
Memory/compute clusters for GPU data are *more* **separated** than CPU and *identically* **partitioned**, but neither cluster is *tightly* **compact**.

Non-**compactness** alone is not an indication that our metrics are not descriptive, as the kernels that perform different computation are expected to be less similar.



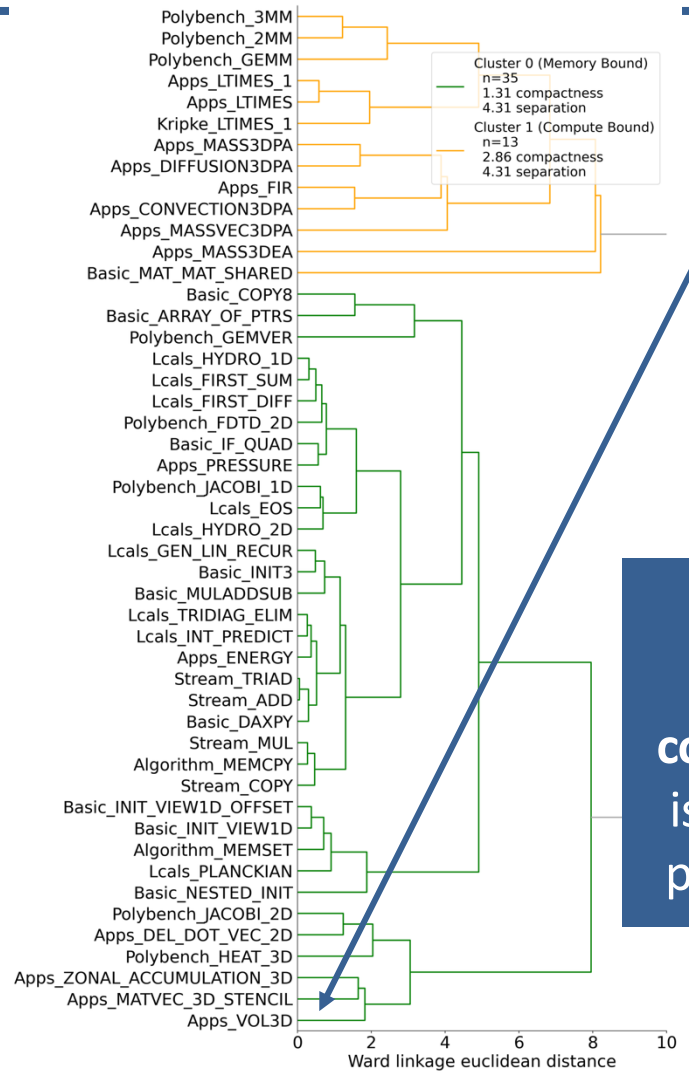
Performance similarity on GPU – memory (L2/HBM transaction rate) bound kernels versus compute (IPS) bound

identical partitioning implies identical distributions



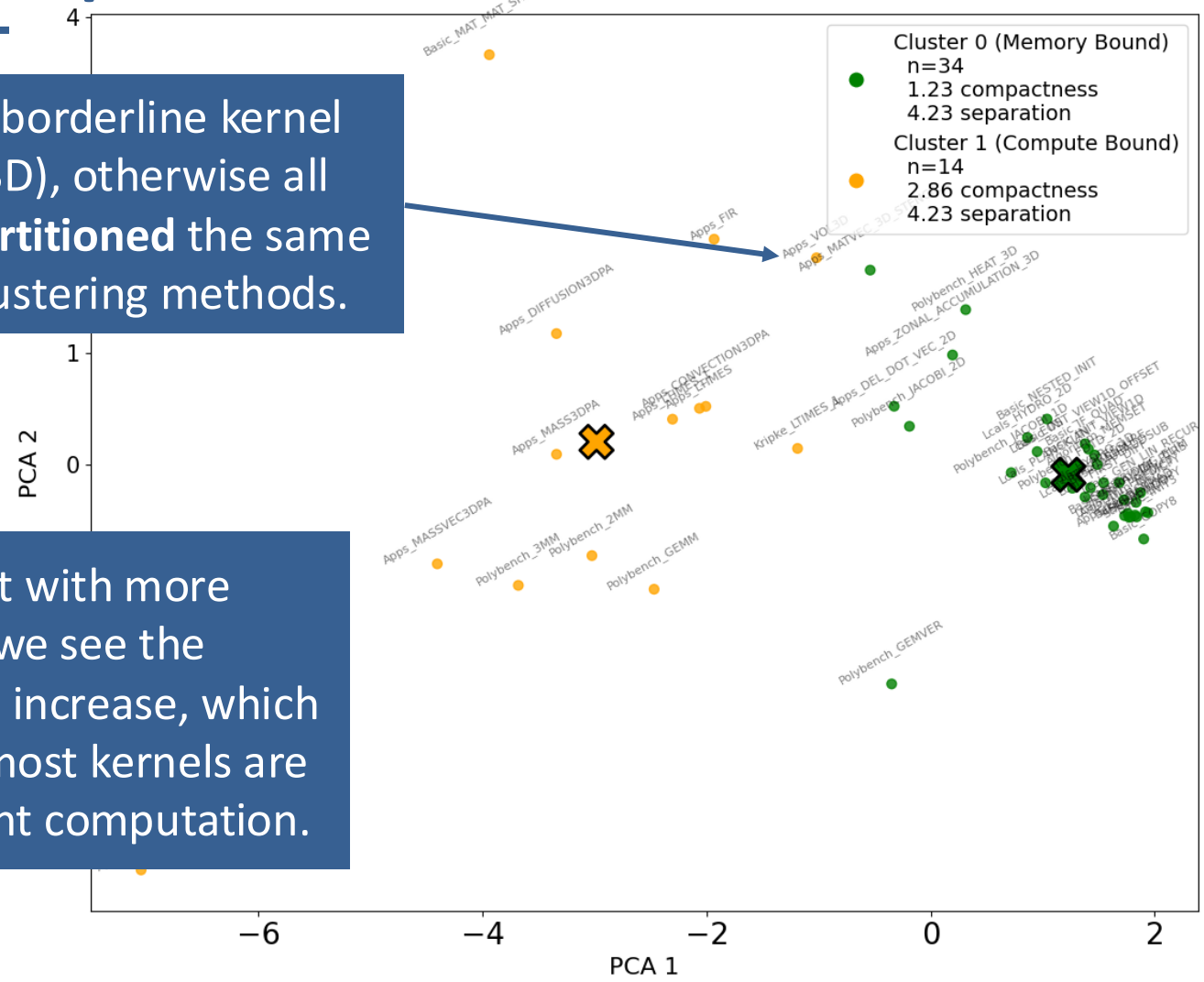
Memory bound kernels characterized by high L2 and HBM transaction rates (2-3x), where the compute bound kernels have high IPS and L1 transaction rates (2-5x).

Multi-platform performance similarity – CPU and GPU patterns align by memory bound and compute bound

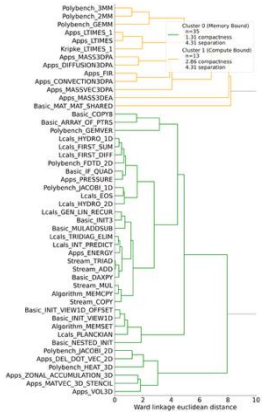


There is one borderline kernel (Apps_VOL3D), otherwise all kernels are **partitioned** the same across the clustering methods.

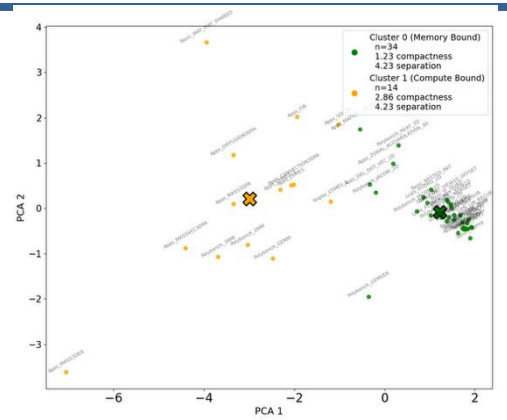
Using a dataset with more information, we see the **compactness** value increase, which is expected since most kernels are performing different computation.

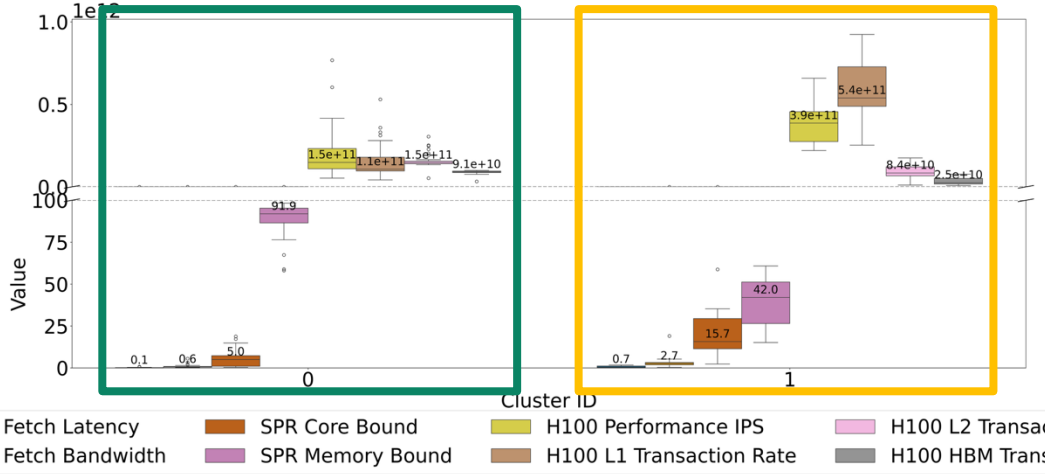


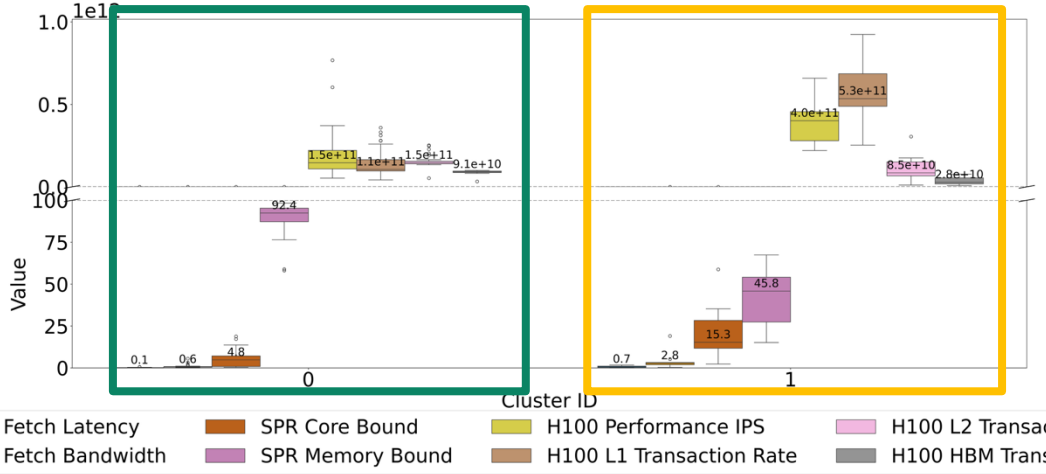
Multi-platform performance similarity – CPU and GPU patterns align by memory bound and compute bound



The memory/compute bound patterns matching on CPU and GPU indicates that a kernel that is memory bound on CPU is likely to be memory bound on GPU and vice-versa







SPR Fetch Latency | SPR Fetch Bandwidth | SPR Core Bound | SPR Memory Bound | H100 Performance IPS | H100 L1 Transaction Rate | H100 L2 Transaction Rate | H100 HBM Transaction Rate

Memory bound kernels characterized by high memory bound, L2 and HBM transaction rates, where the compute bound kernels have high core bound, IPS and L1 transaction rates.

Similarity of multiple implementations of the LTIMES kernel

Calltrees

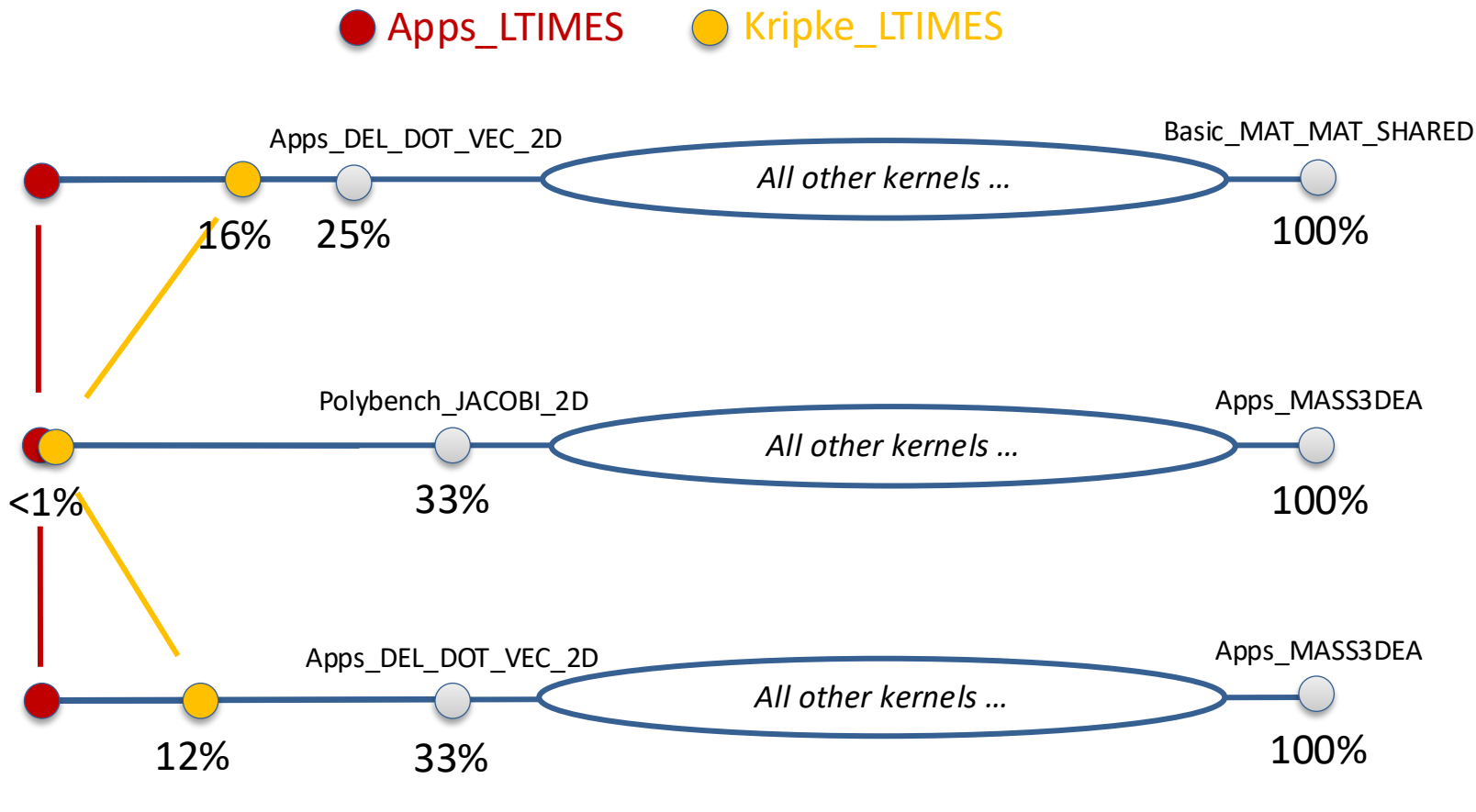
RAJAPerf
└ ***Apps_LTIMES***

Kripke
├– Generate
├– Solve
| └ solve
| └ ***LTIMES***
├– …
…

CPU Similarity

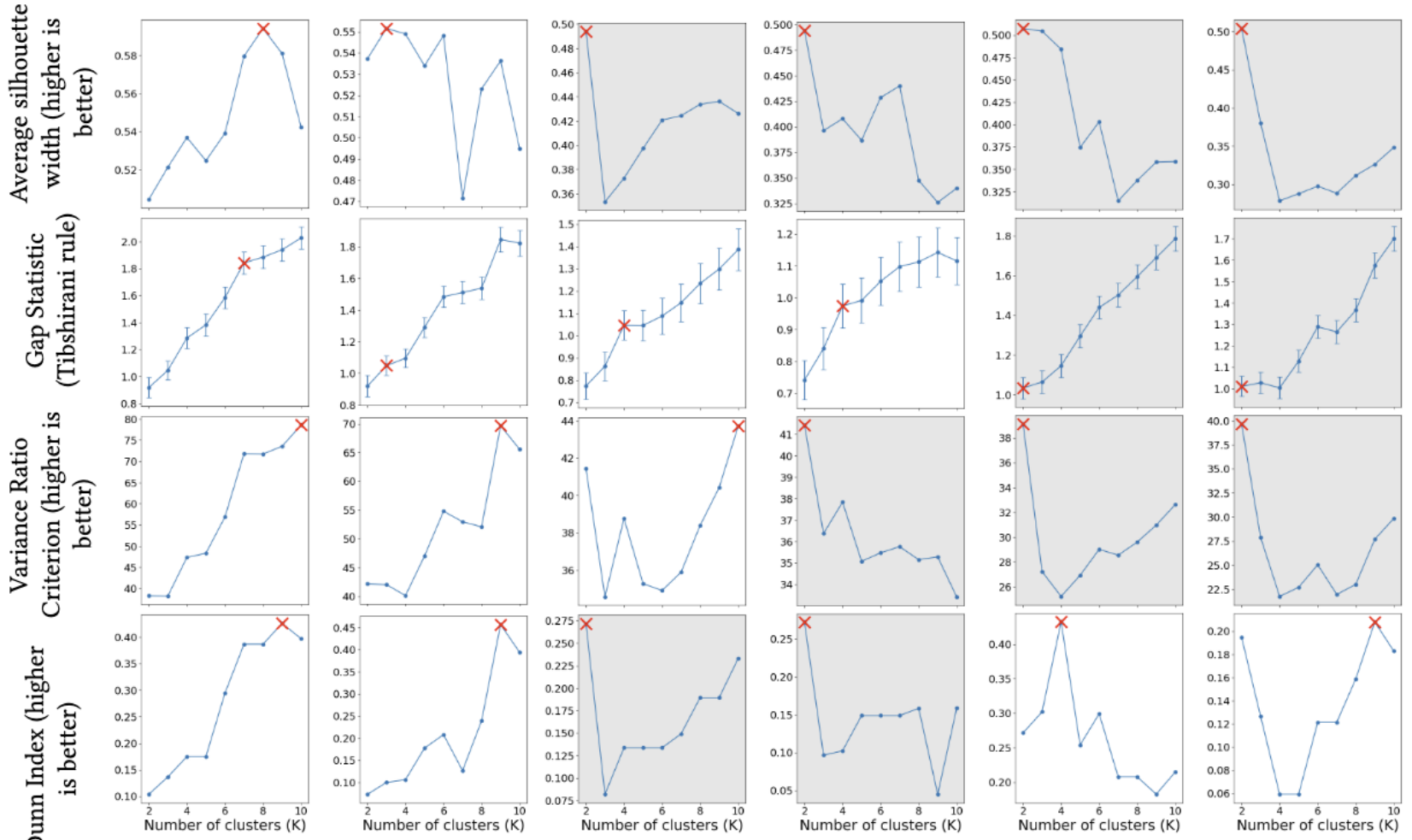
GPU Similarity

Multi-platform Similarity



Kripke LTIMES is consistently the most similar kernel to RAJAPerf LTIMES

Selecting the number of clusters



(a) Agglom. clustering, Top-down metrics (b) K-means clustering, Top-down metrics (c) Agglom. clustering, NCU metrics (d) K-means clustering, NCU metrics (e) Agglom. clustering, Top-down + NCU (f) K-means clustering, Top-down + NCU

Takeaways

- It is possible to classify kernels into memory and compute bound categories across multiple architectures with a handful of performance metrics.
- We can define performance similarity metrics to evaluate the similarity of many kernels across different applications.
- We can leverage performance data to evaluate whether a given kernel in a benchmark is representative of a kernel in an application.

CASC

Center for Applied
Scientific Computing