Power Signatures of High-Performance Computing Workloads

Jacob Combs, Jolie Nazor, Rachelle Thysell, Fabian Santiago,
Matthew Hardwick, Lowell Olson, Suzanne Rivoire
Department of Computer Science
Sonoma State University
Rohnert Park, CA, USA

Chung-Hsing Hsu and Stephen W. Poole
Oak Ridge National Laboratory
Oak Ridge, TN, USA

***Abstract*—Workload-aware power management and scheduling techniques have the potential to save energy while minimizing negative impact on performance. The effectiveness of these techniques depends on the stability of a workload's power consumption pattern across different input data, resource allocations (e.g. number of cores), and hardware platforms.**

In this paper, we show that the power consumption behavior of HPC workloads can be accurately captured by concise signatures built from their power traces. We validate this approach using 255 traces collected from 13 high-performance computing workloads on 4 different hardware platforms. First, we use both feature-based and time-series-based distance metrics to cluster our traces, and we quantitatively show that feature-based clusterings segregate traces by workload just as effectively as the more compute- and space-intensive time-series-based clusterings. Second, we demonstrate that unlabeled traces can be classified by workload with over 85% accuracy, based only on these concise statistical signatures.

I. Introduction

High-performance computing systems are increasingly power-constrained, requiring careful management to maximize the amount of computation done within a system's power budget [1]. Particularly for systems that host a limited number of workloads, workload-aware power management and scheduling techniques have the potential to maximize work done within a power budget or to save energy without compromising performance [2]–[4]. However, workload-aware techniques implicitly assume that a workload's power consumption behavior is relatively stable across different input data. Furthermore, the practical applicability of these techniques is limited unless workloads also exhibit consistent power consumption behavior when run with different resource allocations (e.g. number of cores) and across hardware platforms.

Our goal is to automatically identify workload-specific power consumption patterns based on quantitative measures, or *power signatures*, for HPC applications. The central questions we explore are

1) Is it possible to automatically, quantitatively summarize the power behavior of an HPC workload or class of workloads into a distinctive power signature?
2) To what extent is this signature consistent across runs, input data, hardware configurations, and hardware platforms?
3) Quantitatively, how well does this signature distinguish this workload or class of workloads from others?

To answer these questions, we characterize a set of 255 power consumption traces collected from 13 HPC workloads on 4 different hardware platforms. Each workload was run multiple times, in different configurations, on multiple platforms. We begin by experimenting with different notions of the pairwise distance between traces, which we use as input to standard clustering algorithms. The goals of this initial step are twofold. First, we look for groups of applications that are consistently clustered together. The presence of such groups would indicate that it is possible to construct a single power signature for the entire group, but maybe not for individual workloads within the group. Second, we quantitatively evaluate the ability of the different distance metrics to yield clusterings that are cleanly separated by workload. We use the most successful metrics from this initial clustering step to construct *power signatures* for our traces. Finally, we train a classifier based on these power signatures and evaluate its accuracy in identifying unlabeled traces.

This paper makes the following contributions:

- **Building power signatures:** We use information-theoretic measures of clustering quality to evaluate the ability of time-series-based and feature-based distance metrics to cleanly separate power traces that came from the same workload. Our results show that simple statistical feature-based distance metrics yield clusterings that are just as good as the computation- and space-intensive time-series metrics.
- **Consistency of signatures:** We quantitatively verify that power signatures are more consistent within a single hardware platform, even when input data and resource allocation vary, than across platforms. However, we show that it is still possible to generate accurate clusters for power traces taken from multiple platforms.
- **Distinctiveness of signatures:** We train a classifier on simple feature-based signatures, and we show that it is possible to identify unlabeled traces with over 85% accuracy, even across platforms and configurations.

The rest of this paper is organized as follows. Section II covers related work on identifying workload-specific power

consumption patterns. Section III describes our different methods of constructing power signatures, and Section IV explains how we quantitatively evaluate these signatures using clustering. Section V presents the clustering results, and Section VI evaluates the performance of a classifier that uses our best-performing signatures. Section VII concludes the paper.

II. Related Work

The quantitative study of power signatures is new in HPC. However, researchers have used the term to refer to qualitative patterns observed in workloads' power consumption. For example, Song et al. found, by visual inspection, that an application's power consumption looks similar from run to run but varies more across machines [5], and Laros et al. showed an example of two similar-looking power consumption traces generated by the same application on two different hardware platforms [6]. Subramaniam et al. similarly used the term "power signature" to talk about observed patterns, or the lack thereof, in HPC applications and in their individual phases [7]. By contrast, Kamil et al. visually examined the power traces of multiple HPC workloads. They found that some classes of workloads can be distinguished from each other but that a broad class of CPU-intensive workloads was indistinguishable from Linpack [8].

More quantitatively, Hsu et al. developed signatures of the power-load relationship of aggressively power-managed servers, based on published SPECpower_ssj2008 results [9]; however, these signatures were not used to distinguish one workload from another. More recently, Hsu et al. developed power signatures based on short statistical summaries of applications' power traces [10]. However, they evaluated them only qualitatively, by visual inspection of the clusterings they yielded for a small dataset, and they did not attempt to compare them to results based on time-series metrics or signatures with more complex features.

Power signatures have a longer history in other subfields of computing, particularly in security. In 1999, Kocher et al. showed that an attacker could identify the encryption algorithm used by an embedded device based on visual inspection of its power traces and could uncover secret keys via statistical analysis [11]. Along similar lines, Clark et al. recently showed that it is possible to automatically determine what website a machine is visiting based only on a frequency-domain representation of its power traces [12]. They argued that advances in energy-proportional computing [13], which strengthen the relationship between a system's load and its power consumption, are making it easier to construct such power signatures. While their goal of identifying workloads based on power signatures is similar to ours, their domain of interest (identifying websites loaded on a laptop) and their high-resolution power measurements (250 KHz) make frequency analysis more promising in their context. Power signatures can also help to maintain system integrity; Kim et al. found that anomalies in average power consumption can indicate the presence of malware on mobile devices [14].

Our work, and the papers discussed so far, all use power consumption traces as the basis for constructing power signatures, whether by visual inspection or by more quantitative means. A related area of research is building signatures based on code instrumentation or performance counters. Several studies have used this approach to identify patterns in applications' power consumption [15]–[17] or resource usage [18]–[21] or to detect anomalous behavior [22]. They build these signatures from traces of performance counter values rather than power consumption, and performance counters can be collected much more frequently than the standard 1 Hz for system-level power measurements in HPC [23]. However, to construct and validate the signatures, some of these studies use similar clustering and classification techniques to the ones we employ.

Finally, some papers have applied time-series analysis to power consumption traces, generally in the context of datacenters. Wang et al. [24] looked at patterns in datacenter workloads across spatial and temporal scales at an even coarser granularity than this work. They found high self-similarity in datacenter power demands. Herbst et al. [25] also apply time-series analysis to power traces, but in the context of online, longer-term workload forecasting in datacenters. Long-term datacenter loads tend to vary predictably with the time of day and the day of the year, so they are able to heavily rely on traditional time series metrics like seasonality and periodicity. By contrast, any periodic behavior in the power traces we examine is due to the application itself rather than to predictable seasonal fluctuations. Furthermore, our focus is on distinguishing power traces from different workloads rather than on forecasting future loads.

III. Constructing Power Signatures

What is a power signature? In the context of our work, a power signature is a representation of a power trace—or family of traces—that preserves information about workload-specific power behavior. We evaluate two approaches to constructing these signatures:

1) Using traces themselves, in their entirety, as "signatures." This approach certainly preserves information, but it is space-inefficient. If we store the entire trace as a signature, the method used to compute the distance between a pair of traces takes on primary importance. We explore two such methods.
2) More compactly, we can use statistical summaries of power traces, or feature vectors, as signatures. This technique maps traces onto a feature space in which standard distance metrics (e.g. Euclidean or Manhattan) may be used.

A. Time-Series Representation

The original time series representation of a power trace may constitute a distinctive, though space-intensive, power signature. To investigate the distinctiveness of this "signature," we consider two time-series methods of measuring distance between power traces.

A strawman approach to comparing two time series is to simply calculate mean-squared-difference (MSD) between pairs of corresponding observations. This method requires that the two power traces be of equal length; given two unequal-length traces, we downsample the longer one. If S and T have lengths $|S|$ and $|T|$ with $|S| < |T|$, we create T' by sampling every $|T|/|S|$ points from T so that $|T'| = |S|$; then

TABLE I. SUMMARY FEATURES FOR TIME SERIES

Feature	Calculation
Normalized Max	$\max(T)/\min(T)$
Normalized Median	$\text{median}(T)/\min(T)$
Standard Deviation	$\text{sd}(T)$
Skewness	$\left\lvert \frac{1}{\text{sd}(T)^3 \lvert T\rvert} \sum_{t \in T} (t - \text{mean}(T))^3 \right\rvert$
Kurtosis	$\frac{1}{\text{sd}(T)^4 \lvert T\rvert} \sum_{t \in T} (t - \text{mean}(T))^4$
Serial Correlation	See Section III
Nonlinearity	See Section III
Self-similarity	Hurst Exponent [27], [28]
Chaos	Lyapunov Exponent [29]
Trend	$1 - \text{var}(T')/\text{var}(T)$
DC Skewness	Skewness(T')
DC Kurtosis	Kurtosis(T')
DC Serial Correlation	Serial Correlation(T')
DC Nonlinearity	Nonlinearity(T')

T is the original power trace. T' is the decomposed (DC) power trace.

we define $\text{MSD}(S, T)$ to be $\text{MSD}(S, T')$. This metric is close to Euclidean distance.

MSD, however, fails to capture our intuitive notions of similarity for some types of traces. For example, if two traces both alternate between peak and idle power, we would like to say that the similarity between the two traces is very high (or, equivalently, that the distance is very low). However, if one trace is offset by just one sample relative to the other, the mean squared difference will be at its maximum possible value. An ideal metric would be shift-invariant.

Therefore, we also examine a more sophisticated approach to time series comparison: Dynamic Time Warping (DTW) [26], which is a well-known algorithm for identifying similarities between two time series, irrespective of offsets in time or differences in frequencies of periodic behavior. DTW builds a sequence of ordered pairs called a "warping path" that represents an alignment of carefully chosen corresponding points in the two traces. This warping path may be subject to constraints to ensure certain alignment properties, such as the requirement that observations be matched in a strictly increasing order with respect to time (monotonicity).

B. Feature-Vector Representation

An alternative to storing entire traces as "signatures" is to summarize each trace as a vector of features extracted from the raw data. We evaluate two approaches to feature-vector-based signatures.

First, we consider the two-element feature vector MaxMed, which consists of a power trace's maximum and median values, each normalized to the minimum trace power.

$$\left\langle \frac{\text{max}}{\text{min}}, \frac{\text{median}}{\text{min}} \right\rangle$$

We choose these two features as our initial signature, following the work of Hsu et al. [10], which showed qualitatively that combining two location parameters seemed to yield good clusterings for power traces.

Second, we augment this feature vector with twelve additional features: the standard deviation of the observations, and eleven features that Wang et al. found useful in time-series clustering [27]. Table I lists these features. In particular:

- *Skewness* measures lack of symmetry in a distribution.
- *Kurtosis* measures the heaviness of the tails of a distribution, with higher kurtosis corresponding to a sharper peak (lighter tails).
- *Serial correlation* is calculated using the Box-Pierce test statistic implemented in R[1].
- *Nonlinearity* is measured by the Teräsvirta test statistic implemented in the R `tseries` package[2].
- *Self-similarity* refers to long-range dependence in a time series. We use the Hurst exponent [27], [28], estimated by the R `fracdiff` package[3].
- *Chaos* refers to the tendency of time series observations to diverge from initial values. We measure chaos using the Lyapunov exponent [29].
- *Trend* refers to changes in the mean of a series over time. We measure this by "decomposing" a power trace, which involves subtracting out trend and seasonality components. A preliminary analysis revealed that our power traces do not exhibit significant seasonal behavior, so the process for decomposing them consists of identifying and removing trend, using a penalized regression spline as outlined by Wang et al. [27]. Once we have obtained the detrended time series, we calculate the *Trend* feature as shown in Table I.
- Features marked "DC" are statistical summaries of the decomposed power trace.

IV. METHODOLOGY

A. Experimental Setup

Our collection of power traces contains 255 power traces from 13 benchmarks run on 4 machines. We ran each benchmark multiple times, with at least two different data inputs, and with different configurations (e.g. number of cores) whenever possible.

The benchmarks are

1) *baseline:* Active-idle mode
2) *nsort:* The *nsort* sorting software, doing an external sort [30]
3) *p95:* A Mersenne prime finder [31]
4) *linpack* [32]
5) *calib:* The Mantis calibration suite, which stresses the CPU, memory, and/or I/O subsystems to varying degrees [33]
6) *stream:* The *STREAM* memory-bandwidth benchmark [34]
7) *graph500* [35]
8) *sb-tilt:* TILT [36]
9) *sb-fft1d:* FFT1D [36]
10) *sb-fft2d:* FFT2D [36]
11) *sb-dgemm:* DGEMM [36]
12) *sb-gups:* GUPS [36]
13) *scublas.dgemm:* DGEMM-CPU and cuBLAS-GPU

[1]R code: `Box.stat(T)$statistic`
[2]R code: `terasvirta.test(T)$statistic`
[3]R code: `fracdiff(T, 0, 0)$d + 0.5`

TABLE II. FOUR SYSTEMS USED IN DATA COLLECTION

Machine	RR	OC	LC	RF
CPU	AMD Athlon X2 4800+	Intel Core i5-750	Intel Core i5-750	Intel Core i7-3770
Cores	2	4	4	4
GHz	1.0–2.5	1.2–2.67	1.2–2.67	1.6–3.4
RAM	4 GB	8 GB	8 GB	8 GB
GPU	GeForce 9800gt	GeForce GTX 285	GeForce GTX 650 Ti 1GB	GeForce GTX 670 2GB
Power	115–195 W	120–226 W	85–252 W	74–309 W
Traces	14	16	111	114

Benchmark names with the *sb* prefix come from the SystemBurn software tool [36]. SystemBurn allows users to methodically create a maximal system load for testing and validation purposes, and the benchmarks listed are among the sample workloads it provides. The scublas.dgemm workload consists of DGEMM running on the CPU while cuBLAS runs on the GPU.

Table II shows the configurations of our 4 test machines, which all run the Linux operating system. The *ondemand* frequency scaling governor was turned on during data collection, and Intel Turbo Boost was not used. To monitor power consumption, we used a Watts Up? PRO power meter, which reports instantaneous power consumption once per second. This sampling rate is typical in HPC environments [23], [37]. The accuracy of power measurement is $\pm 1.5\%$ plus 3 counts of the reported value.

B. Clustering Algorithms

The popular approaches for cluster analysis include hierarchical clustering, centroid-based clustering, and density-based clustering; we consider one from each category. All of these approaches require a notion of distance between a pair of points in the feature space, as described in the previous section.

1) **hclust** [38] (hierarchical clustering): We use agglomerative ("bottom-up") hierarchical clustering, which begins by assigning each data point to its own cluster and incrementally joins the two closest clusters together until all of the data points are in a single cluster. The algorithm requires a definition of the distance between two clusters; we use the complete-linkage distance, which is the largest distance between any two points in the two different clusters. This choice tends to result in compact clusters of points that are all relatively close to each other. A nice feature of hclust is that the technique produces a tree showing which smaller clusters merged at each step, which provides more qualitative insight than a single clustering. To get a clustering, we can cut the tree at any height (or distance threshold); the number of clusters will depend on the height at which we cut the tree.
2) **dbscan** [39] (density-based clustering): The dbscan algorithm searches the feature space for regions of unusually high density. Its notion of density-reachability requires two parameters, a distance ϵ and *MinPts*. It groups chains of directly or indirectly density-reachable points into a cluster. This algorithm does not require a pre-specified number of clusters, and it does not force all of the points to belong to some cluster. However, the value of ϵ must be chosen with care, and it is possible for the desired clusters to have such varying densities that there is no reasonable choice of ϵ.
3) **k-means** [40] (centroid-based clustering): The idea of k-means is to partition all the points into k clusters, in which each point belongs to the cluster with the nearest mean. This partition effectively divides the feature space into k pieces. The required parameter for the technique is the desired number of clusters, k. While k-means is straightforward, it has some disadvantages. First, it only works on Euclidean distance, although variants of k-means use other distance metrics. Second, k-means tends to perform poorly if the desired clusters have significantly different sizes or densities.

C. Quantitative Evaluation of Clustering Results

To quantitatively evaluate the "goodness" of a clustering, we look at its similarity to one or more reference clusterings. In our case, these clusterings represent the traces in the dataset grouped by workload or by workload class.

Numerous measures of clustering similarity have been proposed, falling into two main categories: pair-based measures such as the Rand index [41], and information-theoretic measures. The Rand index simply examines each pair of points in the dataset and checks whether the two clusterings agree or disagree on whether the pair should be in the same cluster: a Rand index of 0 means that the two clusterings do not agree on any pair of points, and a Rand index of 1 means that the two clusterings are identical. A variation of the Rand index, the Adjusted Rand Index (ARI), adjusts this index for chance; that is, two random partitionings would have an expected ARI of 0, and ARI can go below 0 if the degree of similarity is lower than this expected value [42].

However, in this work, we use the more complicated Adjusted Normalized Mutual Information (ANMI) [43] metric to quantify clustering similarity. We first explain this metric and then justify its use over the Adjusted Rand Index in this context.

Consider the two clusterings $U = \{U_1, U_2, \ldots, U_R\}$ and $V = \{V_1, V_2, \ldots, V_C\}$, and let a_i and b_j represent the number of data points in clusters U_i and V_j respectively. The similarity between two clusterings of N data points (traces) can be summarized by the $R \times C$ contingency table where each entry n_{ij} represents the number of data points common to clusters U_i and V_j.

Mutual information I and joint entropy H are given by

$$\mathrm{I}(U,V) = \sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}/N}{a_i b_j / N^2},$$

$$\mathrm{H}(U,V) = -\sum_{i=1}^{R}\sum_{j=1}^{C} \frac{n_{ij}}{N} \log \frac{n_{ij}}{N}.$$

A similarity metric, normalized mutual information (NMI), is

given by

$$\mathrm{NMI}(U,V) = \frac{\mathrm{I}(U,V)}{\mathrm{H}(U,V)}.$$

For any two clusterings U and V, $\mathrm{NMI}(U,V)$ is between 0 and 1, with higher values indicating more similar clusterings.

We might expect this similarity metric to have values close to 0 given two random clusterings as input, but this is not the case for NMI. To achieve this property, we adjust NMI as proposed by Vinh et al. [43], calculating the expected value $\mathrm{E}\left[\mathrm{NMI}(U,V)\right]$ by simulation.

This adjusted similarity metric, ANMI, is given by

$$\mathrm{ANMI}(U,V) = \frac{\mathrm{NMI}(U,V) - \mathrm{E}\left[\mathrm{NMI}(U,V)\right]}{1 - \mathrm{E}\left[\mathrm{NMI}(U,V)\right]}.$$

Despite the complexity involved in understanding and calculating the ANMI, it has one major advantage over the ARI for our purposes: its value is much less sensitive to the numbers of clusters in the two clusterings being compared. To develop intuition about this metric, we start with the combined 225 traces on our LC and RF machines, grouped into 13 clusters (one per workload). We then experiment with different modifications to this reference clustering and compute the resulting ANMI and ARI similarity measures:

- Splitting all clusters in half. In this analysis, we compare the original 13-clustering to a 26-clustering generated by splitting each of the original 13 clusters into two equal-sized clusters. The ANMI between these two clusterings is 0.75, and the ARI is 0.62. After performing a second split to compare the original 13-clustering to a 52-clustering, the discrepancy grows: the ANMI is 0.54, and the ARI is 0.33. Because we are open to the idea of grouping related workloads into a class, we prefer a metric that does not harshly penalize this type of split.
- Moving data points between clusters. In this analysis, we build 4 new clusterings by starting with the original 13-clustering and randomly moving 1, 2, 10, or 20 data points between clusters. When we compare each of the new clusterings with the original clustering, the ANMI value is less than the ARI value; that is, ANMI punishes movement between clusters more harshly than ARI. When we move 20 data points, or approximately 10% of the points, the ANMI drops to 0.75, while the ARI is 0.84.

In short, we use ANMI because some workloads may be indistinguishable from other workloads of the same "class" based on their power traces. With both of the adjusted indices, it is important to remember that a value of 0, not 0.5, corresponds to the similarity of two random partitions of the dataset; values of 0.5 would actually indicate a strong similarity between the clusterings, as suggested by the examples above.

D. Classification Algorithm

Based on our clustering results, as we discuss in Section V, we conclude that it is possible to categorize traces based on their workload, rather than simply as members of a class of indistinguishable workloads. The next step is to train a classifier on a set of $\langle \mathrm{signature}, \mathrm{workload} \rangle$ pairs and evaluate its accuracy identifying the workload of unlabeled traces based on their power signatures.

We use random forest [44] as our classification algorithm. This technique builds a large number of randomized decision trees and chooses the mode of their predictions. We choose this algorithm for several reasons:

- It works well on smaller datasets, using the statistical technique of bagging to guard against overfitting.
- It is intuitive and interpretable compared to other similarly accurate classifiers, since it is based on decision trees.
- It provides estimates of the relative importance of the different variables used to construct the trees.
- It is easy to tune, with only two parameters: the number of trees created, and the number of predictors to sample at each node of a tree.
- Its training phase is fast and parallelizable, since the trees can be generated independently. Prediction is also fast.

V. Clustering Results

We divide our collection of 255 traces into two data sets, which we label OCRR and LCRF. The OCRR data set consists of 30 power traces corresponding to 6 workloads run with different input configurations on the two systems OC and RR. The remaining 225 traces make up the LCRF data set, which contains traces of 13 workloads run on machines LC and RF.

To build intuition about our power signatures, we evaluate clusterings of OCRR traces, manually checking for "successful" categorization by workload. Dendrograms are useful for visualizing hierarchical clustering of the smaller OCRR trace set using the MSD metric, and we present such a dendrogram in Figure 1. Of the two main clusters we see here, it is worth noting that one contains only traces from the workloads that are compute-intensive on our systems.

Hierarchical clustering using a more sophisticated signature—the Stat14 feature vector—produces the clustering in Figure 2. This clustering separates the workloads into four categories: calib, baseline, nsort, and the compute-intensive workloads. It appears to be more difficult to distinguish the compute-intensive workloads from one another, and this finding is consistent with prior work [8]. These preliminary results indicate that the Stat14 feature vector representation of a trace may be more effective than using MSD to identify similar traces.

For the LCRF data set, visually checking the 225-leaf dendrogram is impractical. Therefore, we use ANMI, the clustering similarity measure that we introduced in Section IV.

Recall that an ANMI value is associated with a "ground truth" reference clustering. For this, we consider an ideal clustering to be one that groups together traces from the same workload. However, we observe that clustering traces by workload may be more difficult when there is data from multiple hardware platforms. For this reason, we investigate

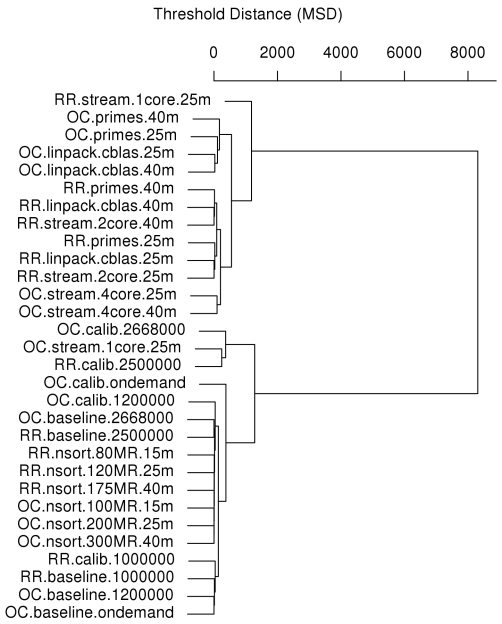

Fig. 1. Dendrogram: Hierarchical clustering of OCRR data ($n = 30$) using the MSD distance metric. The 2-clustering segregates the compute-intensive workloads (the top cluster).

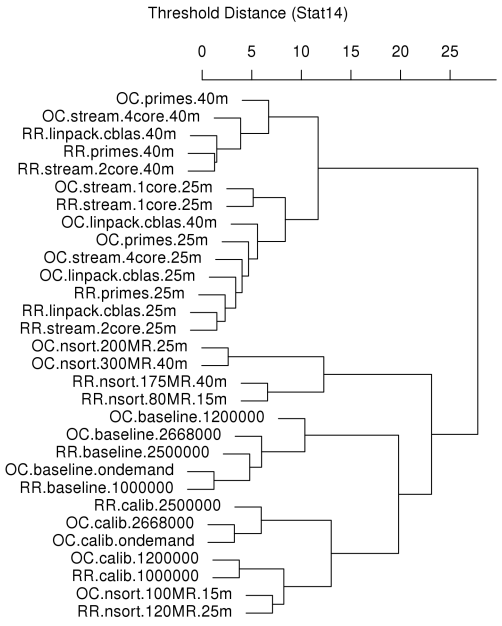

Fig. 2. Dendrogram: Hierarchical clustering of OCRR data ($n = 30$) using Manhattan distance between Stat14 feature vectors. The 4-clustering roughly groups the compute-intensive workload traces, nsort traces, baseline traces, and calib traces separately.

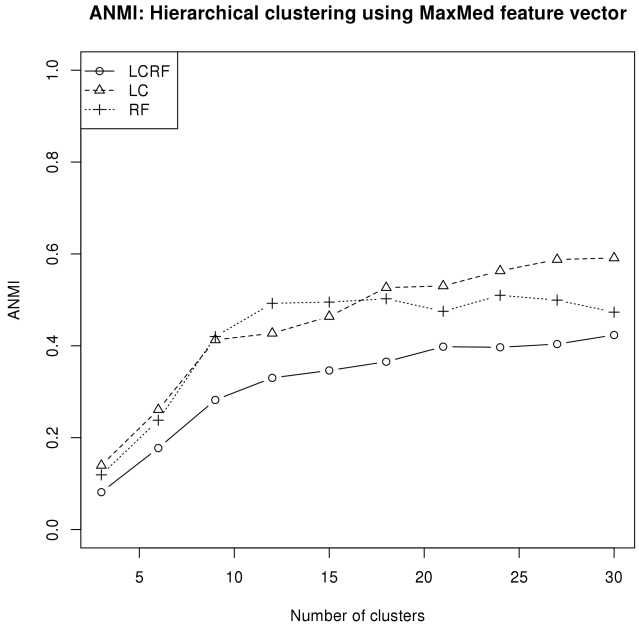

Fig. 3. ANMI plot based on hierarchical clustering of LCRF ($n = 225$), LC ($n = 111$), and RF ($n = 114$) data using Manhattan distance between MaxMed feature vectors. Higher ANMIs show that the clusterings do a better job of distinguishing workloads on a single machine than across platforms.

clusterings on each per-machine subset of the LCRF data set (with separate reference clusterings for each subset), as well on the whole data set. In all cases, the reference clustering has 13 clusters (one per workload).

First, we cluster these three datasets based on the MaxMed feature vector (see Section III-B), using hierarchical clustering. By cutting the tree at different heights, we vary the resulting number of clusters, as shown on the x-axis of Figure 3. The y-axis shows the ANMI of each of these clusterings as compared to the appropriate reference 13-clustering. The ANMI for the cross-platform dataset is substantially lower than for the per-platform datasets. This result indicates that, as expected, it is more difficult to build unified power signatures across hardware platforms than within a platform. Our conjecture is that this is because the workload's behavior actually varies across the two platforms, rather than because the appropriate normalization has not yet been identified. Normalizing the power traces based on the system's dynamic range did not improve the results.

In Figure 4, we compare the four notions of distance between time series discussed in Section III, using hierarchical clustering on LC-only data. We find that the DTW and MaxMed methods cluster traces by workload more effectively than Stat14 or MSD. Furthermore, DTW and MaxMed yield clusterings of similar quality. Results are similar for RF data.

We exhibit this comparison across all LCRF data in Figure 5. While no method appears to outperform clustering data separately by machine, we see similar results: DTW and MaxMed are superior for clustering time series by workload.

It is surprising that Stat14 performs so poorly, because preliminary results (dendrograms) suggested otherwise. The

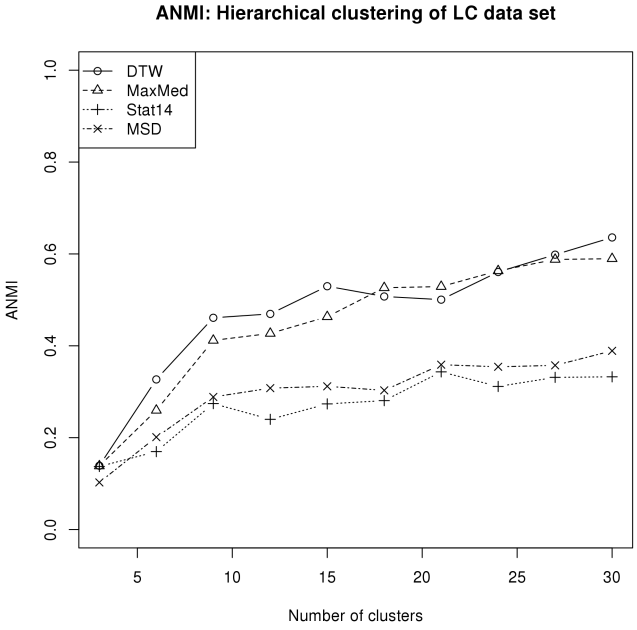

Fig. 4. ANMI plot based on hierarchical clustering of LC data ($n = 111$): Comparison of DTW, MaxMed, Stat14, and MSD notions of distance. DTW and MaxMed outperform Stat14 and MSD.

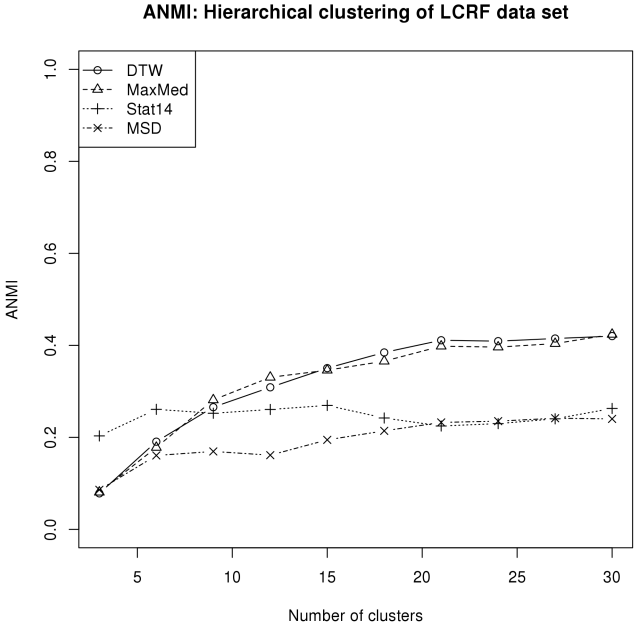

Fig. 5. ANMI plot based on hierarchical clustering of LCRF data ($n = 225$): Comparison of DTW, MaxMed, Stat14, and MSD notions of distance. DTW and MaxMed outperform Stat14 and MSD.

time series characteristics proposed by Wang et al. [27] work well for certain kinds of time series data, but are of limited usefulness in this domain, making the much higher dimensionality of this vector a liability. We use the Manhattan distance to mitigate the problems associated with Euclidean distance for high-dimensional data, but to little avail.

We also experiment with using k-means and dbscan as clustering algorithms, to see if they generate different results for our dataset. Recall that complete-linkage hierarchical clustering (hclust) favors compact clusters, while dbscan's notion of density-reachability allows for clusters with pairs of points that may be quite distant.

K-means requires a feature vector representation of traces rather than a distance matrix, so the time-series distance metrics are not appropriate inputs to this algorithm. However, for Stat14 and MaxMed, the results are similar to the hclust results presented in this section. By contrast, dbscan produces ANMI values that are the same or worse than those produced by hclust and k-means, even after tuning the ϵ parameter. This is not surprising, since dbscan can perform poorly for clusters with widely varying densities. Our reference clustering does exhibit this property: On the one hand, the baseline traces are all similar; on the other hand, the peak power consumption of workloads like calib and stream varies much more with the resource allocation.

In conclusion, the ANMI data shows that MaxMed is the more promising of our feature-based signatures, and DTW is the more promising distance metric for our timeseries-based signatures. These two representations yield comparable results when used to cluster by workload. MaxMed has the advantage of being compact, easy to compute, and more flexible (since it is compatible with arbitrary distance metrics).

VI. Classification Results

In Section V, we presented two findings that motivate using classification to identify power traces. First, it is reasonable to attempt to categorize traces by workload; and second, statistical summaries such as the MaxMed feature vector are reasonable ways to represent a power trace and to determine the distance between two power traces. We use the LCRF data set ($n = 225$) as input to a classifier that we train to identify the workload of a given power trace. Because random forest is relatively robust to overfitting [44], we use both formulations of the feature vector power signature introduced in Section III: MaxMed and Stat14.

Parameters to random forest are *ntree* and *mtry* (see Section IV-D); having sampled several possible combinations, we find that *ntree* $= 2000$ and *mtry* $= 1$ for MaxMed (*mtry* $= 7$ for Stat14) yield favorable results. Figure 6 shows the accuracy (measured using leave-one-out cross-validation) of random forest with these parameters for MaxMed, Stat14, and an additional feature vector, Stat3, which we will construct and evaluate in this section. For each dataset, we are able to classify traces with over 85% accuracy. The cross-platform accuracy is comparable to the within-platform accuracy for RF, but lower than that for LC. Finally, we see random forest's robustness to overfitting in the improved performance of the Stat14 feature vector.

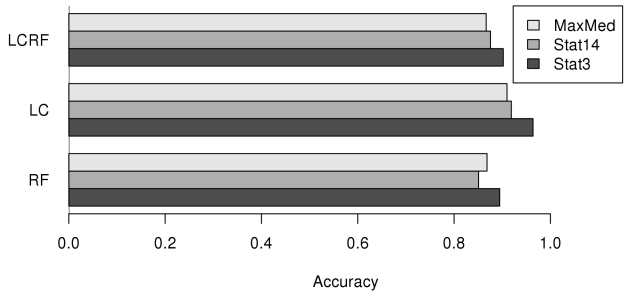

Fig. 6. Random forest leave-one-out accuracy; using parameters ntree $=$ 2000, mtry $=$ 1 for MaxMed and Stat3, and mtry $=$ 7 for Stat14.

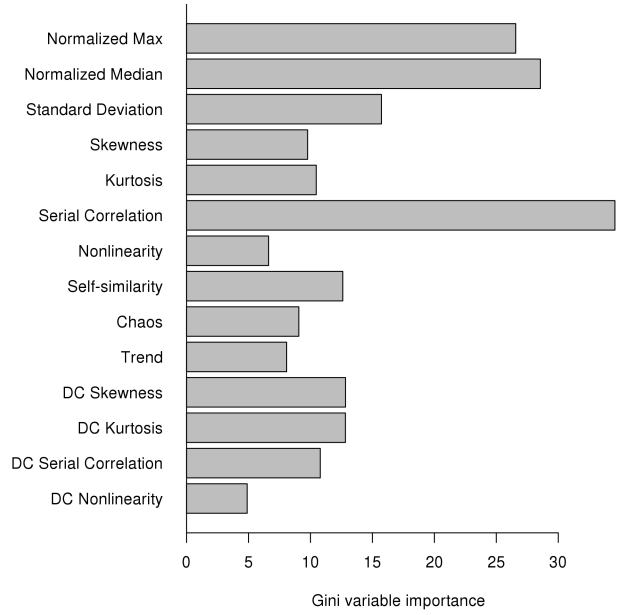

Fig. 7. Gini variable importance for random forest using Stat14 feature vector, with parameters ntree $=$ 2000 and mtry $=$ 7.

Random forest also provides estimates of the relative importance of each variable in the model. One such estimate is the decrease in Gini index due to each predictor variable, which measures how well that predictor partitions the data by workload. As we train and test a random forest by leave-one-out cross-validation, we accumulate the average Gini importance over the 225 training sets. Figure 7 illustrates this measure for each element of the Stat14 feature vector.

The MaxMed components are among the most important, but the graph suggests that Serial Correlation is even more important. This observation motivates the augmentation of the MaxMed summary vector to form a new feature vector: Stat3. Figure 6 shows the Accuracy of the random forest classifier trained on Stat3 data.

Random forest's importance measures also provide a principled way to search the feature space for clustering the

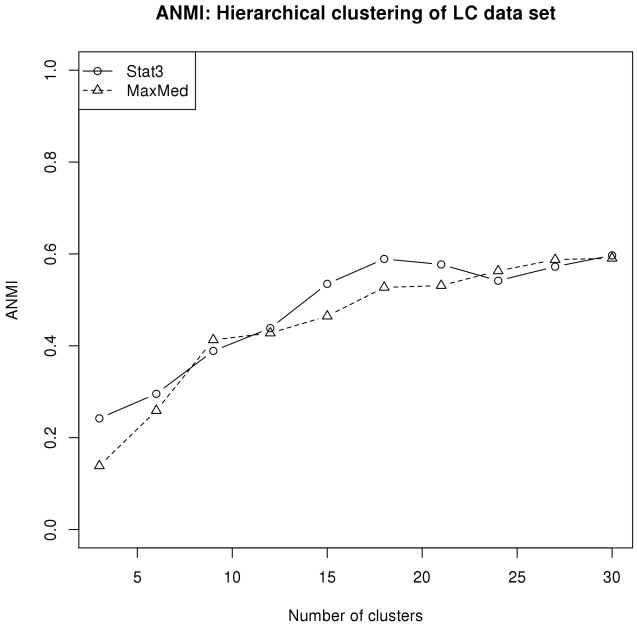

Fig. 8. ANMI plot based on hierarchical clustering of LC data ($n = 111$). Stat3 and MaxMed feature vectors are comparable.

traces. Therefore, we return to clustering and compare the new Stat3 feature vector to the MaxMed vector that we found was successful; Figure 8 presents this comparison. Based on this ANMI data, it is not obvious that Stat3 is superior to MaxMed in this context, possibly because random forest is more robust to overfitting than our clustering algorithms.

VII. Conclusion

This work is the first to quantitatively show that the power consumption behavior of HPC workloads can be accurately captured by very simple feature-based signatures. We show that these concise signatures distinguish workloads from each other just as well as vastly more compute- and space-intensive time-series techniques.

The simplicity of these feature vectors is also surprising; many of the features that Wang et al. [27] successfully used to cluster time series were not helpful for our data. Within the narrower realm of power-trace characterization in computing, our power traces still exhibit little periodicity: at one extreme, our individual HPC workloads lack the time-of-day or seasonal variation found in datacenter traces. At another extreme, they also differ from fine-grained processor-level measurements, such as those provided by RAPL. At finer granularity, traces show a high degree of periodicity and phase behavior associated with loops, which temporal aliasing smooths out at our much coarser 1 Hz granularity. While some of our workloads exhibit phases at this granularity, those phases are not necessarily periodic. The two-pass *nsort* is a good example: the first and second passes can be clearly distinguished in the power traces, but each phase occurs only once. Overall, the lack of periodic behavior simplifies the signatures.

Our goals for future work are to use this foundation to develop workload-aware energy-efficient scheduling and

resource allocation policies while minimizing the burden on users. We are also interested in expanding our dataset to encompass more workloads, particularly accelerator-based and heterogeneous workloads, and hardware platforms. Finally, our results could be refined by answering the following questions:

- To what extent is it possible to classify a power trace in real time? How long does an application need to run before its power signatures can be reliably classified?
- How sensitive are these results to measurement accuracy and sampling rate? At what sampling rate does periodic behavior emerge for these workloads?
- We built and analyzed power traces from single-node workloads. Is it possible to build power signatures for multi-node workloads?
- Since power measurements are often aggregated over a cabinet, is it possible to disaggregate these measurements and identify their constituent workloads?

ACKNOWLEDGMENTS

This work is supported by the United States Department of Defense and used resources of the Extreme Scale Systems Center located at the Oak Ridge National Laboratory. The submitted manuscript has been authored by a contractor of the U.S. Government under Contract No. DE-AC05- 00OR22725. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes.

REFERENCES

[1] S. Hemmert, "Green HPC: From nice to necessity," *Computing in Science and Engineering*, vol. 12, no. 6, pp. 8–10, 2010.

[2] P. Alonso *et al.*, "Tools for power-energy modelling and analysis of parallel scientific applications," in *ICPP*, 2012.

[3] M. Gamell *et al.*, "Exploring power behaviors and trade-offs of in-situ data analytics," in *SC*, 2013.

[4] Z. Gong and X. Gu, "PAC: Pattern-driven application consolidation for efficient cloud computing," in *MASCOTS*, 2010.

[5] S. Song *et al.*, "Energy profiling and analysis of the HPC Challenge benchmarks," *Int. Journal High Performance Computing Applications*, vol. 23, no. 3, pp. 265–276, 2009.

[6] J. Laros *et al.*, "Topics on measuring real power usage on high performance computing platforms," in *CLUSTER*, 2009.

[7] B. Subramaniam *et al.*, "Trends in energy-efficient computing: A perspective from the Green500," in *IGCC*, 2013.

[8] S. Kamil *et al.*, "Power efficiency in high performance computing," in *HPPAC*, 2008.

[9] C.-H. Hsu and S. W. Poole, "Power signature analysis of the SPECpower_ssj2008 benchmark," in *ISPASS*, 2011.

[10] C.-H. Hsu *et al.*, "Application power signature analysis," in *HPPAC*, 2014.

[11] P. C. Kocher *et al.*, "Differential power analysis," in *CRYPTO*, 1999.

[12] S. S. Clark, B. Ransford, and K. Fu, "Potentia est scientia: Security and privacy implications of energy-proportional computing," in *HotSec*, 2012.

[13] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *IEEE Computer*, vol. 40, no. 12, Dec. 2007.

[14] H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in *MobiSys*, 2008, pp. 239–252.

[15] G. L. T. Chetsa, "A blind analysis of HPC systems for energy saving purposes," in *PASA*, 2013.

[16] C. Isci *et al.*, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *MICRO*, 2006.

[17] C. Isci and M. Martonosi, "Phase characterization for power: Evaluating control-flow-based and event-counter-based techniques," in *HPCA*, 2006.

[18] C.-H. Chang *et al.*, "Sampling-based phase classification and prediction for multi-threaded program execution on multi-core architectures," in *ICPP*, 2013.

[19] M. A. Laurenzano *et al.*, "Reducing energy usage with memory and computation-aware dynamic frequency scaling," in *Euro-Par*, 2011.

[20] A. Sembrant *et al.*, "Efficient software-based online phase classification," in *IISWC*, 2011.

[21] ——, "Phase behavior in serial and parallel applications," in *IISWC*, 2012.

[22] A. Avritzer *et al.*, "Monitoring for security intrusion using performance signatures," in *ICPE*, 2010.

[23] T. R. Scogland *et al.*, "A power-measurement methodology for large-scale, high-performance computing," in *ICPE*, 2014.

[24] D. Wang *et al.*, "ACE: Abstracting, characterizing and exploiting datacenter power demands," in *SIGMETRICS*, 2013.

[25] N. R. Herbst *et al.*, "Self-adaptive workload classification and forecasting for proactive resource provisioning," in *ICPE*, 2013.

[26] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 26, no. 1, pp. 43–49, 1978.

[27] X. Wang *et al.*, "Characteristic-based clustering for time series data," *Data Mining and Knowledge Discovery*, vol. 13, no. 3, pp. 335–364, 2006.

[28] J. Haslett and A. E. Raftery, "Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource," *Applied Statistics*, vol. 38, no. 1, pp. 1–50, 1989.

[29] H. Kantz, "A robust method to estimate the maximal Lyapunov exponent of a time series," *Physics Letters A*, vol. 185, no. 1, pp. 77–87, 1994.

[30] C. Nyberg *et al.*, "Nsort," Ordinal Technology Corp. [Online]. Available: http://www.ordinal.com/

[31] "Great Internet Mersenne prime search." [Online]. Available: http://www.mersenne.org

[32] "LINPACK." [Online]. Available: http://www.netlib.org/linpack/

[33] S. Rivoire *et al.*, "A comparison of high-level full-system power models," in *HotPower*, 2008.

[34] J. D. McCalpin, "STREAM: Sustainable memory bandwidth in high performance computers." [Online]. Available: http://www.cs.virginia.edu/stream/

[35] "The Graph500 list." [Online]. Available: http://www.graph500.org/

[36] J. Lothian *et al.*, "SystemBurn: Principles of design and operation release 3.1," Oak Ridge National Laboratory, Tech. Rep. ORNL/TM-2013/234, 2013.

[37] C.-H. Hsu and S. W. Poole, "Power measurement for high performance computing: State of the art," in *Proc. Int. Workshop Power Measurement and Profiling (PMP)*, 2011.

[38] D. Defays, "An efficient algorithm for a complete link method," *The Computer Journal*, vol. 20, no. 4, pp. 364–366, 1977.

[39] M. Ester *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD*, 1996, pp. 226–231.

[40] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[41] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846–850, 1971.

[42] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[43] N. X. Vinh *et al.*, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *Journal of Machine Learning Research*, pp. 2837–2854, 2010.

[44] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.