# Ranking Anomalous High Performance Computing Sensor Data Using Unsupervised Clustering

Adam Morrow[1]
College of Physical and Mathematical Sciences
Brigham Young University
Provo, UT 84604
Email: adampmorrow@byu.edu

Elisabeth Baseman[2]
USRC[3]
Los Alamos National Laboratory
Los Alamos, NM 87544
Email: lissa@lanl.gov

Sean Blanchard
USRC[3]
Los Alamos National Laboratory
Los Alamos, NM 87544
Email: seanb@lanl.gov

***Abstract*—Environmental sensors monitor supercomputing facility health, generating massive data in the largest facilities. Current state-of-the-art is for human operators to evaluate environmental data by hand. This approach will not be viable on Exascale machines, nor is it ideal on current systems. We evaluate effectiveness of the DBSCAN algorithm for identifying anomalies in supercomputing sensor data. We filter large portions of data showing normal behavior from anomalies, and then rank anomalous points by distance to the nearest normal cluster. We compare DBSCAN to $k$-means and Gaussian kernel density estimation, finding that DBSCAN effectively clusters sensor data from a Cray supercomputing facility. DBSCAN also successfully clusters synthetic injected data, avoiding the false positives generated by $k$-means and Gaussian kernel density estimation.**

## I. Introduction

In recent years, many researchers have focused on leveraging high performance computing (HPC) to improve machine learning capabilities [1]–[4]. However, little emphasis has been placed on the ability of machine learning to enhance HPC. Running and optimizing an HPC facility is a complicated and nuanced task that involves careful decision-making, frequently by a human operator. Interestingly, machine learning techniques have not yet been used to improve the management of these large facilities. In this work, we investigate the use of machine learning for anomaly detection in order to better monitor the health of supercomputing facilities.

Currently, state-of-the-art monitoring capabilities for supercomputing facilities involve a human analyst looking through monitoring data by hand, and alerting on human-defined and sometimes arbitrary or ill-informed thresholds. These human analysts are required to sift through a plethora of data, and are overwhelmed. We propose to use machine learning to streamline the analysts' tasks, saving time and effort, as well as improving performance. Machine learning techniques for anomaly detection can learn anomaly scoring on their own, eliminating the need for human-designed thresholds. In addition, these techniques can efficiently filter out uninteresting data, presenting only important information to the analyst.

One such data source is the streaming data from environmental sensors, recording values such as voltage, power consumption, temperature, air speed, etc. from a wide variety of hardware components. Unusual values in any of these measurements could indicate serious global and/or local system problems. We investigate the use of machine learning for anomaly detection to simplify the task of finding anomalous environmental sensor measurements in HPC settings.

This paper proceeds as follows: Section II reviews related work on anomaly detection and machine learning for HPC support. Section III describes the environmental data we use in our study. Section IV introduces our approach to anomaly detection. Section V details our experimental setup, and Section VI presents our results. Finally, Section VII concludes.

## II. Related Work

Significant previous work exists on the use of machine learning clustering algorithms for anomaly detection on spatial and non-spatial data. However, there is little previous work on anomaly detection specifically in the HPC domain.

Clustering for anomaly detection assumes that anomalous data points occur far from their closest neighbors in the dataset, and that normal behavior occurs in areas of high density [5]. These assumptions imply that clustering for anomaly detection requires a similarity metric on the dataset. We briefly review the use of clustering for anomaly detection.

Rajasegarar *et al.* use distributed fixed-width clustering for anomaly detection in the context of large sensor networks [6]. For their application, sensors have compute nodes leading to an elegant network topology approach. However, in our case, sensors are strictly measurement devices.

In the intrusion detection domain, Münz *et al.* use k-means clustering to distinguish normal behavior from expert-defined unusual behaviors [7]. However, their approach requires a training set, meaning that the approach will only ever be able to identify the kinds of anomalies it has already seen. Similarly, Portnoy *et al.* approach the intrusion detection problem using single-linkage clustering with a human-provided cluster width [8]. The assumption is that data points generated by anomalous behavior will be assigned to the smallest clusters.

[1]This work was performed during an internship at the Ultrascale Systems Research Center at Los Alamos National Laboratory in Los Alamos, NM.

[2]Contact author

[3]This work was performed at the Ultrascale Systems Research Center (USRC) at Los Alamos National Laboratory, supported by the U.S. Department of Energy contract DE-AC52-06NA25396. The publication has been assigned the LANL identifier LA-UR-16-26069.

However, this does not account for the fact that even normal behavior may not be isotropically distributed, resulting in clusters that do not match the normal/anomalous classifications.

Work on clustering for anomaly detection, and indeed other forms of anomaly detection, specifically in the domain of HPC remains sparse. Several studies by Fu *et al.* and Xu *et al.* use principal component analysis to look for unusual behavior patterns in HPC monitoring data as well as system logs, with some success [9]–[13]. Other approaches, by Florez *et al.* and Peiris *et al.* are based on signatures and human-defined thresholds [14], [15]. Dueño *et al.* investigate the use of clustering for HPC data, however lack enough data to fully evaluate their approach [16]. We aim to fill this gap.

## III. Available Data

On Cray systems, the System Environment Data Collections (SEDC) utility gathers environmental data in real time from the machine [17]. The SEDC utility collects this data from sensors, referred to by scan IDs, located on hardware components at the cabinet and blade level. Locations can include memory, fans, processors, and power supplies. These sensors give information on a variety of environmental conditions including power consumption, voltage, CPU temperature, and air speed.

Specifically, we consider voltage and temperature readings collected from a Cray supercomputing facility featured in Usenix's Computer Failure Data Repository [18]. The data is reported in time snapshots where each sensor reports a scan ID number, a timestamp, and the voltage or temperature. After accounting for temperature and voltage measured simultaneously, three snapshots are available. In each of these snapshots, temperature and voltage are recorded in one second intervals. Prior to clustering, we remove any corrupt data, such as broken or not reporting sensors, or meaningless readings, because these cases are already easy for an analyst to detect.

## IV. Approach

Our anomaly detection approach consists of two phases: (1) unsupervised learning, and (2) ranking of anomalous points. We use one of three unsupervised learning algorithms, detailed below. Then, we rank the data points using algorithm-specific anomaly scoring strategies in order to obtain anomalies.

***K*-means:** $K$-means clustering is a common method used to partition $n$ observations into $k$ clusters, where each cluster represents a single mean vector [19]. $k$ points are chosen to be centroids for the initial clusters. Next, each point in the dataset is assigned to the cluster associated with the nearest centroid. After all data points have been assigned to a cluster, the $k$ center points are updated to be the most central point with respect to each cluster. The entire process is repeated to convergence — i.e., until all $k$ centroid points no longer change with each iteration. Mathematically, $k$-means minimizes a squared error function. A common problem with the $k$-means algorithm is that without previous knowledge of the data it is difficult to select the correct $k$. To obtain anomaly scores and rank anomalies via $k$-means, we calculate the distance from each point to the mean vector of its assigned cluster. This is the point's anomaly score, with large distances being designated as more anomalous.

**Kernel Density Estimation:** Kernel density estimation is an unsupervised method for estimating the probability density of data [20]. The idea when using a Gaussian kernel is to create a histogram of the data, but instead of binning the data and drawing rectangles around each bin, we draw Gaussian distributions centered on each bin. This smoothes the histogram and provides a continuous estimate of the data density at each point in the feature space. For anomaly detection, the usual method is to estimate the density of each data point, and call the least dense points anomalies.

**Density-Based Spatial Clustering:** Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering method which filters out clusters from noise (anomalous) data [21]. DBSCAN requires two parameters: $\epsilon$, which defines a neighborhood around each data point, and `MinPts`, the minimum cluster size. The algorithm selects an arbitrary point from the data set, and determines whether the point has already been assigned to a cluster. If it is already associated with a cluster, this point's $\epsilon$-neighborhood is added to the same cluster. Otherwise, if the size of the data point's $\epsilon$-neighborhood satisfies `MinPts`, DBSCAN forms a new cluster including the data point, its $\epsilon$-neighborhood, and the $\epsilon$-neighborhoods of its neighbors. If the data point's $\epsilon$-neighborhood does not satisfy `MinPts`, the data point is marked as noise/anomalous. This process is repeated until all data points have been visited. DBSCAN classifies each point in the dataset as either "core", "border", or "noise". Points contained in clusters with $\epsilon$-neighborhoods satisfying `MinPts` are designated core points, while those assigned to clusters but in areas of lower density are border points. Points that are not assigned to any clusters are noise points. These noise points are our anomalies.

## V. Experimental Setup

We use temperature and voltage readings from the System Environment Data Collections (SEDC) utility from a Cray supercomputing facility, provided in Usenix's Computer Failure Data Repository [18]. The data includes three snapshots in time, between three and five seconds of each other for both voltage and temperature. On these intervals we run $k$-means, Gaussian kernel density estimation, and DBSCAN. Because we expect to see one normal mode of behavior, we set the number of clusters in $k$-means to be one. For DBSCAN, we set `MinPts` to 4, as recommended by Ester *et al.* [21], and $\epsilon$ to be 4 °C or 4 mV.

For $k$-means and DBSCAN, instead of using traditional Euclidean distance, we implement a custom distance metric specific to the high performance computing sensor domain. To compute distance, we consider only the absolute values of differences in sensor values, i.e., a sensor reading 24 mV is 4 units away from a sensor reading 20 mV, and similarly for temperature sensors, regardless of the sensor scan ID. This allows DBSCAN and $k$-means to more intelligently compute similarities between data points.

| Temperature Sensor Kernel Density Estimation | | Voltage Sensor Kernel Density Estimation | |
|---|---|---|---|
| Value (°C) | Score | Value (mV) | Score |
| 13 | -5.57 | 1357 | -6.03 |
| 13 | -5.57 | 1360 | -6.02 |
| 41 | -5.24 | 1383 | -5.99 |
| 29 | -4.95 | 1363 | -5.02 |
| 40 | -4.41 | 1380 | -4.40 |
| Temperature Sensor $k$-Means | | Voltage Sensor $k$-Means | |
| Value (°C) | Score | Value | Score |
| 12 | 22.66 | 1357 | 18.33 |
| 13 | 21.66 | 1360 | 15.33 |
| 41 | 6.34 | 1363 | 12.33 |
| 29 | 5.66 | 1365 | 10.33 |
| 40 | 5.34 | 1365 | 10.33 |
| Temperature Sensor DBSCAN | | Voltage Sensor DBSCAN | |
| Value (°C) | Score | Value (mV) | Score |
| 12 | 17.0 | 1357 | 3.0 |
| 13 | 6.0 | | |

TABLE I: Top 5 most anomalous values found for each detection algorithm, for both temperature and voltage. For Gaussian kernel density estimation a lower score indicates a point is more anomalous, while for $k$-means and DBSCAN a higher score suggests a more anomalous point. Note that DBSCAN finds only two and one anomalous points for temperature and voltage respectively.

**Preprocessing:** We selected sensors that ran at corresponding times and had sufficient recorded data such that detectable patterns were present in order for the algorithms to distinguish between and classify normal and anomalous data. Because we are interested in finding anomalous data, within this subsection of the data, any missing values still present were replaced with the relevant feature average value. This was done so that essentially any missing data became negligible and could not be reported as an anomalous.

**Evaluation:** We report both quantitative and qualitative results. We quantitatively test the performance of our anomaly detection algorithms by injecting synthetic but realistic anomalies into our available data set. We inject five unusually high and low values in areas of low density, as well as injecting several points in areas of high density that should not be detected. We report the top five anomalies as found by each of the three anomaly detection algorithms on the raw data, as well as precision at 5 for each of the algorithms run on the data with our injected anomalies. Our qualitative evaluation comes from an expert HPC systems analyst.

## VI. Results

**Quantitative:** Table I shows the top five results per anomaly detection algorithm from experiments using raw SEDC temperature and voltage data. The score column indicates the "anomalousness" of the value reported by the respective detection algorithm. For $k$-means and DBSCAN, a higher anomaly score indicates greater anomalousness, while for Gaussian kernel density estimation a more negative score indicates greater anomalousness. Note that DBSCAN only reports two and one anomalies for temperature and voltage respectively, because DBSCAN only reports the top five noise points. In this case, there were only two and one noise points respectively in the data. This suggests that DBSCAN, while its top ranked anomalies may be the same as the other two algorithms, significantly decreases the probability of false positives.

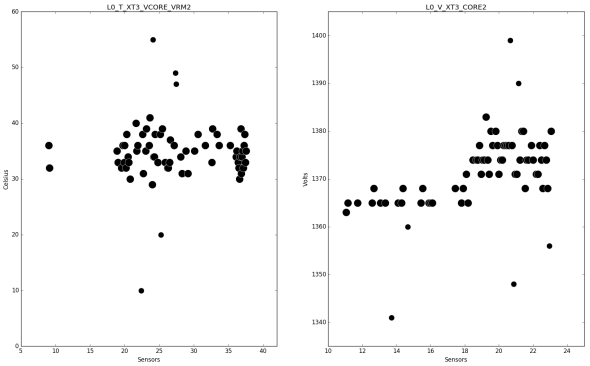
| Data | Algorithm | Precision at 5 |
|---|---|---|
| Temp | DBSCAN | 1.0 |
| Temp | KDE | 1.0 |
| Temp | $k$-means | 1.0 |
| Volts | DBSCAN | 1.0 |
| Volts | KDE | 0.8 |
| Volts | $k$-means | 0.8 |

TABLE II: Precision at 5 results for all three anomaly detection algorithms. Note that DBSCAN is the only algorithm with perfect precision.

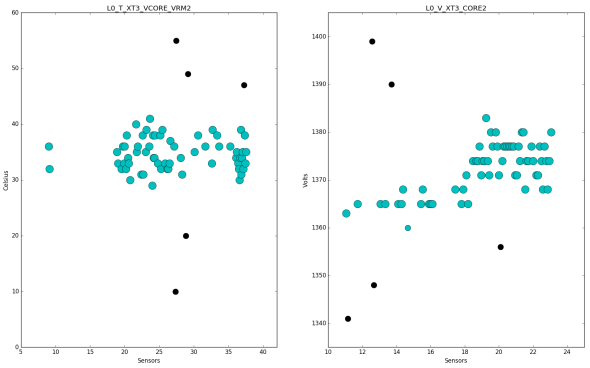
We also report precision at 5 for $k$-means, Gaussian kernel density estimation, and DBSCAN run on raw data with injected anomalies in Table II. We find that while precision is very high for all three algorithms, DBSCAN was the only algorithm with perfect precision for both temperature and voltage values. We believe this is due to the custom distance metric we use in DBSCAN, which allows DBSCAN to consider both density and a domain-specific similarity metric, while $k$-means considers only the metric and Gaussian kernel density estimation considers only density. Figure 1 shows the results of running DBSCAN on the data with injections. The results of $k$-means and Gaussian kernel density estimation on data including injected anomalies miss some of the injected anomalies, instead choosing points that DBSCAN classifies as normal behavior. In this way, DBSCAN reduces the number of false positives, simplifying the work of human analysts.

**Qualitative:** Sensor data on HPC systems is key in monitoring the overall machine health. Anomalous values in temperature or voltage may indicate any number of problems varying from benign improper sensor calibration to potentially harmful environmental conditions outside the design thresholds of the hardware. Quickly identifying harmful conditions can result in significant cost savings. In future Exascale class machines maximum power consumption will be tightly controlled, and careful monitoring of temperature and voltage will be key to overall stability of large computing systems.

DBSCAN successfully discovers anomalies in real HPC environmental sensor data without presenting false positives or missing important anomalies. In the cases without synthetic injection it identifies clear problems on two temperature sensors as well as a possibly improperly calibrated voltage sensor. All three of these cases are important enough to require human intervention to troubleshoot the sensor's unusual behavior. In the cases with synthetic injection, DBSCAN properly identifies all of the injected anomalous sensor readings without reporting normal data as anomalous. Correct identification of anomalous sensor readings is key to delivering alerts to analysts who have limited time to view such data and who will rapidly ignore tools that deliver excessive false positives.




(a) Preprocessed raw temperature and voltage data with five synthetic injected anomalous points.




(b) DBSCAN run on preprocessed raw temperature and voltage data with five synthetic injected anomalous points. Points characterized as anomalous by DBSCAN are marked in black while points characterized as normal are marked in blue.

Fig. 1: Results of DBSCAN run on preprocessed temperature and voltage data with synthetic injected anomalies. DBSCAN successfully recovers all five injected anomalies.

## VII. Conclusions

We proposed the use of DBSCAN, an unsupervised clustering algorithm, for anomaly detection in HPC sensor data. Our method includes filtering out uninteresting sensor data, clustering normal behavior and separating it from anomalous points, and ranking those anomalous points based on a custom similarity metric from the nearest normal clusters. We find that our method effectively finds injected anomalous points along with correctly classifying normal injected points. In addition, we find that using DBSCAN can also serve as a method for detecting false-positives. While $k$-means and kernel density estimation require a specific number of points that must be labeled as anomalous, DBSCAN will only rank points that are truly anomalous based on their location density. In the future we plan to extend our comparisons to include a streaming approach for analyzing the data, along with integrating our comparisons of data across all SEDC measurements.

## References


[1] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 104–111.

[2] D. Gillick, A. Faria, and J. DeNero, "Mapreduce: Distributed computing for machine learning," *Berkley, Dec*, vol. 18, 2006.

[3] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[4] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, "Distributed graphlab: a framework for machine learning and data mining in the cloud," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.

[5] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[6] S. Rajasegarar, C. Leckie, M. Palaniswami, and J. C. Bezdek, "Distributed anomaly detection in wireless sensor networks," in *2006 10th IEEE Singapore International Conference on Communication Systems*. IEEE, 2006, pp. 1–5.

[7] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *GI/ITG Workshop MMBnet*, 2007.

[8] L. Portnoy, E. Eskin, and S. Stolfo, "Intrusion detection with unlabeled data using clustering," in *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*. Citeseer, 2001.

[9] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 117–132.

[10] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online system problem detection by mining patterns of console logs," in *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 588–597.

[11] W. Xu, L. Huang, and M. I. Jordan, "Experience mining google's production console logs." in *SLAML*, 2010.

[12] Q. Guan and S. Fu, "Adaptive anomaly identification by exploring metric subspace in cloud computing infrastructures," in *Reliable Distributed Systems (SRDS), 2013 IEEE 32nd International Symposium on*. IEEE, 2013, pp. 205–214.

[13] H. S. Pannu, J. Liu, and S. Fu, "Aad: Adaptive anomaly detection system for cloud computing infrastructures," in *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*. IEEE, 2012, pp. 396–397.

[14] G. Florez-Larrahondo, Z. Liu, Y. S. Dandass, S. M. Bridges, and R. Vaughn, "Integrating intelligent anomaly detection agents into distributed monitoring systems," *Journal of Information Assurance and Security*, vol. 1, no. 1, pp. 59–77, 2006.

[15] M. Peiris, J. H. Hill, J. Thelin, S. Bykov, G. Kliot, and C. Konig, "Pad: Performance anomaly detection in multi-server distributed systems," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014, pp. 769–776.

[16] S. J. Q. Dueño, E. A. Sáez, Y. M. C. Bonaparte, and H. Kettani, "Detecting anomalies for high performance computing resilience," in *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, vol. 4. IEEE, 2010, pp. 5–7.

[17] C. Systems, "System environment data collections (sedc) guide," http://docs.cray.com/books/S-2491-7204/S-2491-7204.pdf, accessed: 2016-08-01.

[18] B. Schroeder and G. A. Gibson, "The computer failure data repository," https://www.usenix.org/cfdr, accessed: 2016-07-20.

[19] D. J. MacKay, *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[20] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise."