Interpretable Analysis of Production GPU Clusters Monitoring Data via Association Rule Mining

Baolin Li*, Siddharth Samsi†, Vijay Gadepally†, Devesh Tiwari*
* Northeastern University, † MIT Lincoln Laboratory

***Abstract*—Modern high-performance computing (HPC) and cloud computing systems are integrating powerful GPUs to accelerate increasingly demanding deep learning workloads. To improve cluster efficiency and better understand user behavior and job characteristics, system operators will collect operational data for trace analysis. However, previous efforts on these system logs have lacked the interpretability aspect, and there is no systematic approach that can be widely applied to different datacenter traces and return interpretable results. In this work, we propose a workflow to discover hidden association relationships between collected features of system jobs. The outcome of our analysis approach yields useful association rules that can be directly interpreted into operational insights. Using this approach, we have conducted case studies using the traces of three large-scale multi-tenant GPU clusters running production machine learning workloads. We have focused on the observations of GPU underutilization and job failures, revealing the possible reasons for these job behaviors and suggesting solutions to mitigate them. Our case studies have demonstrated the feasibility of our interpretable analysis workflow, which can be widely used by more HPC and cloud computing system operators.**

I. Introduction

The rapid development of machine learning (ML) algorithms has resulted in an increasing number of ML-based applications in various areas such as computer vision [1], natural language processing [2], and personalized recommendation [3]. As ML models continuously grow larger [4], [5] and have more and more users, datacenter and cloud platforms provide high-performance GPUs to accelerate these workloads. However, due to advances in novel ML models and new GPU architectures, we need to continuously update our understanding of the job characteristics and user behaviors when operating a GPU cluster. For example, a few years ago convolutional neural network (CNN) models such as ResNet [6] dominated the ML workloads, while the increasingly popular recommendation models and language models have shown distinctive memory and compute characteristics from ResNet [7], [8]. *Improving such understanding requires open-source datasets and portable methodological tools for analysis – Unfortunately, our HPC community lacks this or has limited resources.* This paper attempts to address this issue.

To keep up with the advances in ML algorithms and accelerator hardware, cloud computing, and HPC system administrators deploy datacenter monitoring and management tools to collect system operational information such as scheduler logs and node resource measurements. However, often these datasets are not made public for deeper community-driven analysis. This work aims to address this challenge.

Second, performing large-scale analysis on the collected data is required to enable system operators to have a better understanding of perspectives such as the resource consumption of the ML workloads, the types of workloads submitted by the user, and the resource usage patterns and how efficiently they are used. Although recent works have shown meaningful insights derived from data analysis [9], [10], [11], [12], these analyses are based on the system operator's experience, and such experience varies over time and across different sites [13], [14], [15], [16]. Unfortunately, we lack a common methodology and framework that can be easily applied across datasets to perform analysis. Furthermore, prior works are mostly focused on black-box-based prediction models, instead of automatically extracting the relationships between certain features and the prediction label. This work aims to address this challenge, too.

In this work, we propose a systematic, widely applicable analysis workflow to generate interpretable results from a cluster trace. To make the results directly readable by system operators and translatable to operation guidance, we adopt working principles from **association rule mining**, a popular data mining technique that has been widely used in areas of market transaction analysis and bioinformatics [17], [18]. This technique generates association rules that directly reflect the strength of relationships between certain grouped attributes. We apply our designed approach to study the MIT SuperCloud [10], a GPU cluster that we operate, and two representative GPU cluster traces of Alibaba PAI [9] and Microsoft Philly [11].

Contribution Summary. First, we introduce an interpretable analysis framework to perform systematic association rule mining and analysis for GPU-based systems – previous research in the GPU-cluster system operation lacks this perspective. We have performed data engineering that is specific to datacenter operation, and designed rule pruning techniques to hide uninteresting, redundant rules and highlight insightful rules for system operators. Second, we perform a detailed analysis of three real-world GPU clusters that accelerate various types of ML workloads. Due to the extensibility of our approach, we are able to uncover hidden patterns in all three traces. We show operational insights including why certain jobs underutilize the GPU cores, the common characteristics of jobs that have low GPU utilization, and general reasons and characteristics of job failure. We have open-sourced our production-grade traces at `https://dcc.mit.edu/` and our interpretable analysis framework at `https://doi.org/10.5281/zenodo.10680695` for the academic community to use.

II. Background

In this section, we briefly introduce the GPU-accelerated HPC system traces studied in this work: PAI [9], SuperCloud [10], and Philly [11] traces. We list the overview of each trace in Table I.

PAI is an Alibaba platform for ML-as-a-Service (MLaaS), a cloud service mode similar to Platform-as-a-Service (PaaS), but focusing on supporting ML workloads. PAI supports all popular Deep Learning (DL) frameworks, and the workloads cover a wide range of applications including computer vision (CV), natural language processing (NLP), recommendation (RecSys), reinforcement learning (RL), and graph neural networks (GNNs). We organize PAI trace by tasks, where one user can submit multiple tasks of different roles simultaneously (i.e., a parameter server task and a worker task). A distributed training worker task will allocate multiple GPUs. For simplicity and consistency with other traces, we use the term “job” to refer to a PAI task. PAI is a heterogeneous cluster, the user can choose from NVIDIA P100, V100, and T4 GPUs. If none is specified, the system will assign a miscellaneous low-end GPU type to the job.

SuperCloud is a GPU cluster studied in this paper. It is a homogeneous cluster, each node has two Xeon Gold 6248 CPUs and two NVIDIA V100 GPUs (32GB memory). To enable the AI workflow, SuperCloud provides a full suite of tools including Tensorflow [19], Pytorch [20], Horovod [21] and many other dependencies. The NVIDIA Collective Communications Library (NCCL) is configured for GPUDirect and RDMA over ethernet to support distributed training. For each job, we collect the CPU, memory, and I/O usage from Slurm [22] with an interval of 10 seconds. We collect the GPU utilization (or SM utilization for Streaming Multiprocessor), GPU memory (bandwidth) utilization, memory used (GB), and GPU power from NVIDIA System Management Interface (`nvidia-smi`) as time-series data every 100ms. The MIT SuperCloud trace is currently open-sourced as a dataset [23] at `https://dcc.mit.edu/`.

Philly is a cluster shared across groups in Microsoft. The trace contains logs from 14 virtual clusters (VCs), on two different types of GPUs – one with 12GB memory and the other with 24GB memory (the device name is unknown). Philly uses a distributed monitoring system [24] to record measurements of CPU utilization, GPU utilization, and server memory used with a granularity of 1 minute. The majority of the applications are convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Philly is a very representative trace and has been used in various studies [25], [26], [27], [28], [29], hence we include Philly Trace in this work.

TABLE I: Overview of studied traces in this work.

Name	Operator	Time	Jobs	Users	GPUs	Application
PAI [9]	Alibaba	2 months	850k	1242	>6k	Cloud MLaaS
SuperCloud [10]	MIT	8 months	98k	310	450	AI Research
Philly [11]	Microsoft	75 days	100k	319	2.5k	ML Training

III. Building an Interpretable Analysis Framework

A. Motivation

The analysis of large-scale HPC traces is generally divided into two major categories: predictive analysis and descriptive analysis. Predictive tasks aim to build classification or regression models to predict a target feature. In this work, we are more interested in descriptive analysis, where the objective is to discover hidden relationships in the trace data, generating straightforward takeaways on HPC system design and operation.

Previous efforts in trace analysis have lacked the Interpretability aspect. For example, one can train machine learning models such as Support Vector Machines (SVM) or Artificial Neural Networks (DNN) to predict job exit status and hardware failures [30], [31]. However, SVM uses kernels to map original data into higher dimensional space before performing a hyperplane linear separation, and DNNs include multiple hidden layers of linear perceptrons and nonlinear activations – it is difficult to interpret the relationship between the input features and output targets. Even for decision trees that are known to have better interpretability than other machine learning models, the analysis typically requires a feature transformation stage to make the features abstract [32]. Thus, the split decision cannot be directly translated into operation insights.

Previous works have used clustering analysis to understand the underlying nature of common behaviors of HPC system jobs [33], [34], [35]. While dividing points closer to each other into groups is intuitively compelling, it lacks reasoning about the grouping, and slight differences in initialization can lead to qualitatively different results [36]. Therefore, such analysis cannot generate readable information with a confidence guarantee. Unsupervised learning methods such as autoencoder generates an abstract-level representation of the data, such representation does not reveal the cause and characteristics of certain phenomenons such as resource underutilization or contention.

In this work, we aim to characterize and provide root cause analysis for interesting observations in large-scale GPU-accelerated HPC systems for AI workloads. We apply association rule mining to discover direct relationships between certain job properties and observations, guiding system design and workload scheduling.

B. Association Rule Analysis

Association analysis is a popular technique used to discover interesting relationships hidden in large data sets. It was originally implemented for market basket transactions and became widely used in other application domains such as bioinformatics, web mining, and scientific data analysis [18], [37], [38]. The objective is to generate meaningful association rules from a data set of transactions – these can be transaction records of customer purchases, and in our case, these are scheduling and execution logs of submitted jobs in an HPC system. The problem of association rule mining can be formally defined as the following.

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of all possible items in the transactional database, where each item in the set represents a unique attribute of a job. For example, these items can be "Job Failure", "Multi-GPU", or "Tensorflow" to indicate that the job has failed, the job is a multi-GPU job, or the job uses the Tensorflow framework, respectively. Any transaction can then be represented by a collection of items from the item set I. Let $D = \{t_1, t_2, ..., t_m\}$ be the set of all transactions in the database, where each transaction t_i contains a subset of items in I. In our analysis specifically, each transaction corresponds to a unique job record in the datacenter job trace.

An association rule is defined as an implication expression of the form $X \Rightarrow Y$, where X and Y are two disjoint sets of items (a set of items is also named *itemset*) that are both subsets of I, in other words, $X \subseteq I, Y \subseteq I, X \cap Y = \emptyset$. The left-hand-side itemset of the rule (in this case X) is the *antecedent* and the right-hand-side itemset (Y) is the *consequent*. The rule implies that when events in X occur, events in Y are also likely to be true, an intuitive example is $\{``Snow"\} \Rightarrow \{``Winter"\}$. To evaluate the quality of a rule, three metrics are typically used, namely *support*, *confidence*, and *lift*.

An itemset X is contained by a transaction t_i if $X \subseteq t_i$. Support of an itemset X is defined as the number of transactions that contain X (also known as support count of X, denoted by $\sigma(X)$) as a fraction of the total number of transactions:

$$supp(X) = \frac{\sigma(X)}{|D|} \tag{1}$$

where $|D|$ is the total number of transactions in database D. Since the antecedent and consequent are mutually exclusive, a transaction containing the rule of $X \Rightarrow Y$ must contain all the items in both itemsets X and Y. The support (range $[0, 1]$) of a rule $X \Rightarrow Y$ can be calculated by the probability of seeing both itemsets together:

$$supp(X \Rightarrow Y) = P(X, Y) = \frac{\sigma(X \cup Y)}{|D|} \tag{2}$$

Here we use $P(\cdot)$ to denote probability. The support reflects the frequency of the antecedent and consequent appearing together in the database, but it does not address the relationship of the data. The confidence metric (range: $[0, 1]$) is used to describe how frequently the items in the consequent appear in transactions that contain the antecedent.

$$conf(X \Rightarrow Y) = P(Y \mid X) = \frac{\sigma(X \cup Y)}{\sigma(X)} \tag{3}$$

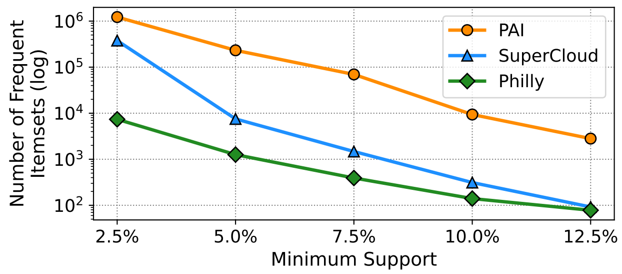

Fig. 1: Number of frequent itemsets at different levels of minimum support. (PAI has more entries and features than others)

Higher confidence indicates a higher likelihood of observing Y upon observation of X, thus the rule $X \Rightarrow Y$ should be a stronger rule. However, the metric is biased towards consequent itemsets that are naturally frequent. For example, a rule $\{``SingleGPU"\} \Rightarrow \{``JobSuccess"\}$ has 0.8 confidence, indicating single-GPU jobs are most likely to execute successfully. But if 80% of the jobs in the trace are finished successfully, this is a misleading rule as the two itemsets can be independent of each other. The lift metric (range: $[0, \infty]$) can be calculated by normalizing the confidence of the rule with the support of the consequent, and it is essentially evaluating the dependency of the antecedent and consequent:

$$lift(X \Rightarrow Y) = \frac{conf(X \Rightarrow Y)}{supp(Y)} = \frac{P(X, Y)}{P(X)P(Y)} \tag{4}$$

The lift measures the support of a rule against the baseline support computed under the statistical independence assumption: when two itemsets X, Y are independent, $P(X, Y) = P(X)P(Y)$, the lift will be 1. A lift value greater than 1 suggests that the antecedent and consequent are dependent, a larger lift indicates stronger dependence.

Evaluating the three metrics provides a comprehensive understanding of the meaningfulness of a rule. For instance, if a rule $X \Rightarrow Y$ has a support of 0.1, confidence of 0.8, and lift of 2, it means that X and Y appear together in 10% of the transactions, Y appears in 80% of the transactions that contains X, and we are twice more likely to see X and Y together than we would have expected assuming the two itemsets are independent. The objective of association analysis in this work is to generate meaningful rules from the data using the support, confidence, and lift measurements.

C. Itemset Generation

We take a two-step approach to generate association rules – first, generate *frequent itemsets* from the database, then generate rules from the frequent itemsets. A *frequent itemset* is defined as an itemset whose support is greater than a minimum support threshold. Determining the minimum support threshold requires domain knowledge about the application case. A threshold that is too high would miss interesting rules that appear at a lower frequency, while a threshold that is too low would generate spurious frequent itemsets that come from randomness, and generate too many rules that are difficult to manage. We set the support threshold to 5% of the total number of jobs in the trace. When there are 100k jobs (Philly trace), a frequent itemset is contained in at least 5k jobs, which is a decently large number of samples to avoid randomness. In Fig. 1, we show that even for the smallest trace Philly, we have generated more than 1.2k frequent itemsets with the current support threshold. For PAI and SuperCloud trace, we have generated approximately 232k and 7.5k frequent itemsets, respectively.

We use the FP-Growth algorithm [39] which is a popular alternative to the traditional Apriori algorithm [40] for extracting frequent itemsets. FP-Growth uses a data structure called FP-tree to deal with performance issues (exponential runtime and memory requirements) presented in the Apriori algorithm when the database is large. It is the state of the practice algorithm for frequent itemset generation [41], [42], [43], [44].

D. Association Rule Pruning

We can generate association rules from extracted frequent itemsets. For example, an itemset $\{X, Y, Z\}$ can generate rule $\{X\} \Rightarrow \{Y, Z\}$, $\{X, Y\} \Rightarrow \{Z\}$, $\{Z, X\} \Rightarrow \{Y\}$, etc. To prevent the number of generated rules from getting out of control, we apply rule filtering based on recent work on the root cause analysis systems deployed in Meta/Facebook datacenters [44]. First, we limit the maximum length of frequent itemsets to 5, which prevents generating rules that are too descriptive and specific to the samples. Then, we specify a minimum lift threshold for rule generation. A lift threshold ensures that the antecedent and consequent of a rule do not have strong independence, otherwise the rule is not useful for decision-making (III-B). We set the lift threshold to be 1.5, meaning the rules we generate are 50% more likely to appear together than expected assuming the rule antecedent and consequent are independent.

We also design rule pruning conditions on top of the lift-based filtering. We are interested in **cause analysis** of certain events and also the **characteristics** of certain observations. For instance, we observed that a significant number of jobs that requested GPU have zero GPU usage. In our analysis, we are interested in finding out what kind of job submission typically has this behavior (i.e., user group, framework), as well as what other characteristics we can extract from this kind of job (i.e., low memory usage, short runtime). To improve the value of generated rules, we designed and applied four pruning conditions to our analysis. We first introduce a concept of keyword in our definition:

Definition 1: A keyword K is an item that we are particularly interested in for the analysis. $K \in I$.

Recall that I is the set of all items in the database. The keyword represents our objective of rule mining. For example, if we want to investigate job failure, we will set the keyword to "job failure" and only look at relevant rules that contain the keyword. *Rules that have the keyword in consequent can be used for cause analysis. As for rules with the keyword in the antecedent, they can be used for characteristic analysis.* For each rule containing the keyword, we check the following conditions for pruning decision:

Condition 1: When there exist two rules $X_i \Rightarrow Y$ and $X_j \Rightarrow Y$, where keyword $K \in Y$ and $X_i \subset X_j$, if $C_{lift} * lift(X_i \Rightarrow Y) \geq lift(X_j \Rightarrow Y)$, where $C_{lift} \geq 1$, then prune rule $X_j \Rightarrow Y$. Else if $C_{supp} * supp(X_j \Rightarrow Y) \geq supp(X_i \Rightarrow Y)$, where $C_{supp} \geq 1$, prune rule $X_i \Rightarrow Y$.

In this cause analysis scenario, when there exist two rules with similar antecedents and the same consequent containing the keyword, discard the rule with a longer antecedent if the shorter rule has a similar or higher lift. Otherwise, discard the shorter rule when the longer rule has a higher lift as well as similar support. The two parameters C_{lift} and C_{supp} are extended from a previous work [44] to tune how easily the conditions can be satisfied depending on the nature of the dataset. C_{lift} is used to regulate the margin of lift difference, and C_{supp} is used to loosen the comparison criteria for support since $supp(X_j \Rightarrow Y) \leq supp(X_i \Rightarrow Y)$ always holds true when $X_i \subset X_j$ (Eq. 2). Consider the following rules R_1 and R_2 for the intuition of this conditional pruning with keyword as "job failure":

```
Rule $R_1$: {user A} ⇒ {job failure}
Rule $R_2$: {user A, job type B} ⇒ {job
failure}
```

If the lift of R_1 is similar or higher than R_2, it means any job type the user runs tends to fail, we do not need to specify job type B. If R_2 has a higher lift and similar support as R_1, specifying job type B is more informative, and it is observed in enough samples, so we can prune R_1.

Condition 2: When there exist two rules $X \Rightarrow Y_i$ and $X \Rightarrow Y_j$, where keyword $K \in X$ and $Y_i \subset Y_j$, if $C_{lift} * lift(X \Rightarrow Y_j) \geq lift(X \Rightarrow Y_i)$ and $C_{supp} * supp(X \Rightarrow Y_j) \geq supp(X \Rightarrow Y_i)$, where $C_{lift} \geq 1, C_{supp} \geq 1$, then prune rule $X \Rightarrow Y_i$. Otherwise, if $C_{lift} * lift(X \Rightarrow Y_j) < lift(X \Rightarrow Y_i)$, prune rule $X \Rightarrow Y_j$.

This condition for characteristics analysis covers the case when two rules share an antecedent containing the keyword, one rule is more specific (or longer) than the other rule. We would prefer the rule with a more specific consequent if its lift and support values are not too far from the shorter rule. We demonstrate this condition with the following example:

```
$R_1$: {job failure} ⇒ {short runtime}
$R_2$: {job failure} ⇒ {short runtime,
cluster C}
```

Rule R_2 reveals more characteristics about jobs that have failed – they also run at cluster C. If these two rules have a similar lift and support, R_2 will be more valuable to the system operator. But if R_1 has a clear lift advantage over R_2, binding failed jobs to cluster C is misleading, thus we should keep the more preservative rule.

Condition 3: When there exist two rules $X \Rightarrow Y_i$ and $X \Rightarrow Y_j$, where keyword $K \in Y_i$, $K \in Y_j$ and $Y_i \subset Y_j$, if $C_{lift} * lift(X \Rightarrow Y_i) \geq lift(X \Rightarrow Y_j)$, where $C_{lift} \geq 1$, then prune rule $X \Rightarrow Y_j$.

This condition is for cause analysis, but one rule has more specific consequent than the other rule. Since we only care about the cause (antecedent), we would prefer a more concise consequent when the two rules have a similar lift. We explain with the following example:

R_1: {`user A`} $\Rightarrow$ {**`job failure`**}
R_2: {`user A`} $\Rightarrow$ {**`job failure`**, `cluster C`}

Both rules suggest that jobs submitted by user A are a source of failure. There is no need to keep rule R_2 if R_1 has a decent lift since R_2 does not provide new information about the cause of failure. Similarly, we would like the characteristic analysis to be concise as well.

Condition 4: When there exist two rules $X_i \Rightarrow Y$ and $X_j \Rightarrow Y$, where keyword $K \in X_i$, $K \in X_j$ and $X_i \subset X_j$, if $C_{lift} * lift(X_i \Rightarrow Y) \geq lift(X_j \Rightarrow Y)$, where $C_{lift} \geq 1$, then prune rule $X_j \Rightarrow Y$.

When the keyword is in the antecedent, we care about other interesting characteristics that are also present in jobs that have observed the keyword. For two rules that have shown the same consequent, if a shorter antecedent generalizes well, we do not need a more specific antecedent.

R_1: {**`job failure`**} $\Rightarrow$ {`short runtime`}
R_2: {**`job failure`**, `cluster C`} $\Rightarrow$ {`short runtime`}

When both rules have a similar lift, we know that jobs that have failed also tend to have short runtime, whether they are in cluster C or not. Therefore, we can prune rule R_2.

In summary, we designed four conditional filters, that can be applied to rules depending on: (1) whether the keyword is in the antecedent or the consequent; and (2) whether the difference is in the antecedent or the consequent. We have set both C_{supp} and C_{lift} to be 1.5 for all three traces as we see that these filters have significantly reduced the number of redundant rules while still maintaining the key information (Sec. IV).

E. Trace Preprocessing

Applying association rule analysis to an original HPC system trace is infeasible due to several reasons. Firstly, the data is collected at different levels, thus different features of a job are scattered across different files. For example, in our SuperCloud trace collection, job information such as user, runtime, and exit status is collected at the scheduler level, while measurements such as CPU and GPU utilization are collected at the node level. The PAI trace and Philly trace are also organized similarly. Our first effort was to merge all the features into a single file since rule generation requires all transaction features to be available in the mining database.

Secondly, each database transaction should be organized as a set of nominal or interval attributes. In our analysis, since each transaction corresponds to a unique job, some attributes such as the job's GPU streaming multiprocessor (SM) utilization and execution time are continuous in nature. For such features, we perform equal frequency binning [45] which assigns the value into one of the multiple bins, each bin having roughly the same number of data points. Choosing the number of bins for discretization comes with trade-offs. If the bin size is too small, the generated rules would have low support. If the bin size is too large, the rules would have low confidence and lift. We find the bin size of a quarter works well, thus each value is represented by one of the following bins:

- Bin1: [min, 25^{th} percentile)
- Bin2: [25^{th} percentile, median)
- Bin3: [median, 75^{th} percentile)
- Bin4: [75^{th} percentile, max]

We also tried equal-width binning, where we divide the full range of the feature into bins of equal intervals. This method does not work well because some features such as runtime have long tails, thus bins at higher values tend to be empty. After the binning, the database gets transformed using one-hot encoding into the FP-Growth algorithm's supported format.

Thirdly, the distribution of attribute values can be highly skewed, which generates uninteresting frequent itemsets. For example, if 90% of jobs have requested a single GPU while 10% of jobs have requested multiple GPUs, most frequent itemsets would include the item "single GPU", and generate rules that suggest most items are associated with "single GPU" – these rules do not reveal much insight. Instead, we should drop these items and only keep "multi-GPU" items in the database. In our preprocessing, we drop the items that are present in more than 80% of the jobs in the trace.

Finally, a categorical attribute can have many possible values, and some of these values may have very low support. To handle this issue, we aggregate the low-support attribute values into groups. For example, the PAI trace has the neural network model of the workload as a feature, and we aggregate the items "resnet", "vgg", and "inception" into a new item called "CV", items "bert", "nmt", and "xlnet" into an item called "NLP". The job user is also one of such categorical features since a lot of users only submit a few jobs. We have sorted the users by the number of job submissions in the trace and grouped the most active users responsible for 25% of the jobs in the trace as "frequent user", and the least active users responsible for 25% of the jobs in the trace as "new user".

Next, we demonstrate the analysis results of the PAI, SuperCloud, and Philly traces.

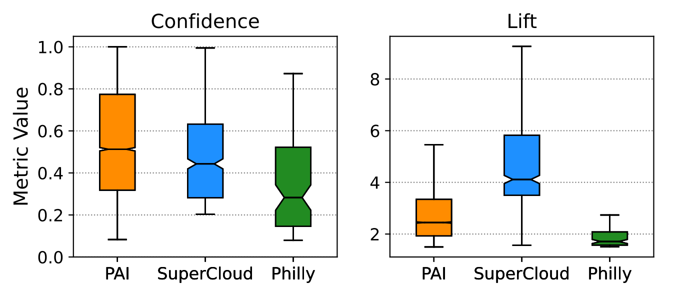

Fig. 2: Box plot of confidence and lift of rules across traces.

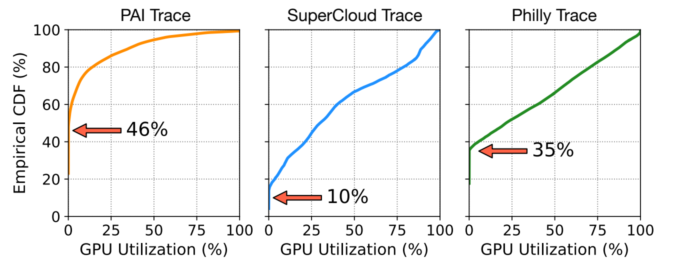

Fig. 4: GPU SMs are often not utilized by a significant number of jobs.

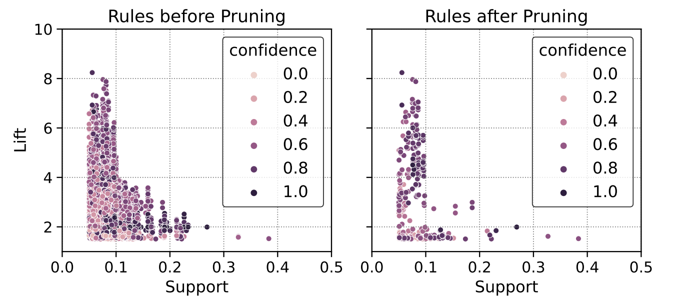

Fig. 3: Rules visualized by their support, confidence, and lift values in PAI trace. The number of rules has been greatly reduced after rule pruning.

TABLE II: GPU underutilization rules from PAI trace.

	Antecedent	**Consequent**	**Supp.**	**Conf.**	**Lift**
C1	GPU Request = Bin1	SM Util = 0%	0.13	0.94	1.88
C2	Memory Used = Bin1	SM Util = 0%	0.23	0.92	1.85
C3	Freq Group, GPU Type = None	SM Util = 0%	0.13	0.82	1.65
C4	CPU Util = Bin1, Runtime = Bin1	SM Util = 0%	0.05	0.77	1.54
C5	CPU Request = Std	Freq User, SM Util = 0%	0.11	0.61	2.73
A1	Freq User, SM Util = 0%	Mem Request = Std, GPU Type = None, Tensorflow	0.21	0.96	1.94
A2	CPU Request = Std, SM Util = 0%	Freq User, GPU Type = None, Tensorflow	0.11	0.78	2.96
A3	GPU Request = Bin1, SM Util = 0%	Freq User, CPU Request = Std, Mem Request = Std	0.07	0.61	4.07

IV. Application of the Framework: Case Studies

A. Overview

We apply our methodology for trace analysis to three different traces described earlier (Sec. II) and discuss interesting observations. *For each observation and analysis, we set the keyword to the item that represents this observation, perform association rule mining as described in Sec. III, and examine the rules that have the keyword in the consequent (cause analysis) and rules that have the keyword in the antecedent (characteristic analysis).*

Why is the analysis presented for different systems/traces individually? We have noticed an inherent difference between the traces – shown in Fig. 2, an example study of GPU underutilization rules. In this figure, the confidence and lift values of extracted rules are highly varied across traces. Consequently, it is not appropriate to compare similar rules from different traces quantitatively and present analysis together. *Nevertheless, this highlights how a portable methodology like ours is more useful to find "system-specific" insights instead of finding "generic" insights.*

Pruning the extracted rules is a key step when analyzing the traces, especially for the PAI trace. Due to its high number of jobs and features, we are able to extract tens of thousands of rules from it. As shown in Fig. 3, when studying GPU underutilization rules, we visualize each extracted rule as a point based on the rule's support and lift. After applying the rule pruning mechanisms (Sec. III-D), we have substantially reduced the number of rules, especially the ones with lower lift values, making the results readable and manageable by humans to assist with decision-making.

In the following results, we perform rule analysis centered around the two most common concerns in the traces: jobs that underutilize GPU cores and jobs that failed. Besides the common ones, we also analyze topics specific to the trace and share some insights.

B. Low GPU-Utilization Jobs

In the PAI and Philly trace, we found that a significant number of jobs have an average GPU SM utilization of near zero. In our SuperCloud trace, we have also observed this behavior in some jobs. We show the cumulative distribution function (CDF) of the GPU SM utilization of all jobs in Fig. 4: there are 46%, 10% and 35% of jobs that barely use the GPU processor in the PAI, SuperCloud and Philly trace, respectively. Note that all jobs in the trace have requested at least one GPU. Thus, this is an alarming observation as increasing hardware utilization is a key objective of HPC system operation [46], [47].

PAI Trace. Table II shows some association rules related to the jobs that show 0% GPU SM utilization in the PAI trace. *We first take a methodological detour of how the data is organized.* Note that we use bin numbers to indicate the range of non-categorical attributes, such as the number of GPUs requested and memory utilization. However, for the number of CPU cores requested, we find that approximately 50% of the jobs have requested 600 cores, which is also the median of all jobs. We infer that this could be the default or standard CPU request count, thus we create a bin "Std" to represent this interval (standard request count). This is also true for the amount of memory requested, so we create a bin for the standard memory request as well. The rules are labeled with "C" if the keyword is in the consequent, and "A" if the keyword is in the antecedent.

The rules C1 to C5 show several signs of potential 0% GPU core utilization. C1 and C2 show that a low GPU request number and a low memory usage are good indicators of no GPU core usage. The frequent group attribute in C3 states that a job from a frequent job group that does not specifically request GPU type is likely to not use the GPU. The job group is specific to the PAI trace, where jobs are assigned a group label depending on several parameters including library invocations, input arguments, data sources, and sinks. For example, when different users perform BERT pre-training using a provided WikiText dataset, their jobs may belong to the same group. We classify the most frequently used job groups that correspond to 25% of job submissions as the frequent group. C4 suggests that when a job has both low CPU usage and short runtime, its GPUs tend to be idle. This rule reveals one possible reason for GPU being underused: the user is trying to test something out quickly in debug mode, then kills the job. C5 does not have very high confidence, but its high lift value suggests that when a job has the standard CPU core request, it is much more likely to come from a frequent user and have 0% SM utilization.

Likewise, we can interpret the rules A1 to A3 for characteristic analysis. The common characteristics of a job that has requested but not used GPU include frequent user, unspecified GPU type at request, and Tensorflow framework. It is interesting knowing that these jobs often use the Tensorflow framework since GPU underutilization should be framework agnostic. This implies that Tensorflow is used as a template framework, thus users are more likely to use it to explore the functionalities of the MLaaS platform. A key characteristic of these low GPU usage jobs is low request customization by the user – they submit the job request with standard CPU and memory request, unspecified GPU type, and the template framework. These rules all have one more item besides the keyword in the antecedent, this is because we did not find strong rules that are applicable to all 0% GPU utilization jobs. We have to know at least one more attribute of the job to implicate its other characteristics.

Takeaways: The PAI cluster shows associations between certain job request patterns and GPU underutilization, a prediction model can be used to identify jobs that tend to underutilize GPU cores at the job submission stage before it gets scheduled.

SuperCloud Trace. Table III lists the rules extracted from SuperCloud trace. In the cause analysis rules C1 to C4, the low CPU utilization and short runtime are good indicators of GPU underutilization, which is seen in Table II as well. In addition, in our trace collection, we capture the GPU power consumption and GPU memory utilization (this is memory bandwidth, not memory used) that are not present in the other two traces. We find that low power consumption and low GPU memory utilization are also good indicators for no usage of GPU SM. Rule C3 shows that new users are associated with low GPU SM and memory utilization, while the rules in the PAI trace do not show any association with new users.

TABLE III: GPU underutilization rules from SuperCloud trace.

	Antecedent	Consequent	Supp.	Conf.	Lift
C1	GMem Util = Bin1, GMem Util Var = Bin1	SM Util = 0%	0.11	0.94	6.28
C2	CPU Util = Bin1, GMem Used = Bin1, GPU Power = Bin1	SM Util = 0%	0.06	0.81	5.37
C3	GPU Power = Bin1, New User	SM Util = 0%, GMem Util = Bin1	0.05	0.60	3.99
C4	GPU Power = Bin1, Runtime = Bin1	SM Util = 0%, GMem Util = Bin1	0.05	0.60	3.99
A1	SM Util = 0%, SM Util Var = Bin1	GMem Util = Bin1, GMem Util Var = Bin1, GMem Used = Bin1	0.08	1.00	10.59
A2	SM Util = 0%	GMem Util = Bin1, GPU Power = Bin1	0.13	0.88	4.30

TABLE IV: GPU underutilization rules from Philly trace.

	Antecedent	Consequent	Supp.	Conf.	Lift
C1	Min SM Util = 0%, Runtime = Bin1	SM Util = 0%	0.09	0.87	2.74
C2	CPU Util = Bin1	SM Util = 0%	0.13	0.60	1.87
A1	SM Util = 0%, GPU 24GB Mem	Min SM Util = 0%, CPU Util = Bin1	0.08	0.69	3.85

Recall that in SuperCloud trace we capture the GPU metrics at an interval of 100ms, we also generated the variation of each metric (Sec. II). The rules A1 and A2 in Table III have different antecedents: A1 is the characteristics for jobs that have the SM utilization constantly low all the time, while A2 includes jobs whose average SM utilization is near 0%, but the GPU may have been used in short intervals throughout its job lifetime. The characteristics of low GPU memory usage are not present in rule A2, meaning a job could keep a GPU memory occupied but does not use the compute cores. This is common for model inference when the user sends inference requests occasionally.

Philly Trace. Table IV show the rules we discover in Philly trace. Since SM utilization is collected as a 1-minute average time series, we also use the minimum and maximum SM utilization within the job period as features in addition to average SM utilization. Rule C1 shows that if the job has 0% SM utilization within any 1-minute time span, and has short runtime, the overall SM utilization is likely to be 0%. Similar to the other traces, low CPU utilization and short runtime appear in the antecedent. Rule A1 is specific to jobs scheduled at nodes with 24GB memory GPUs.

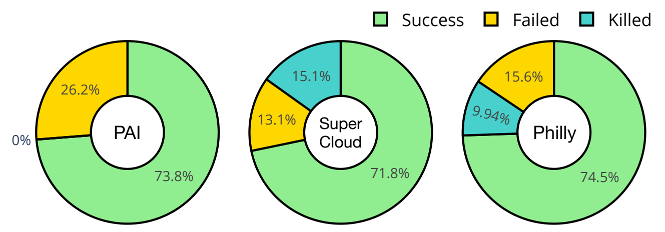

Fig. 5: Job exit status in the traces.

Takeaways: We suggest building a lower-tier system for allocation of debugging and exploratory jobs using cheaper and lower-performance GPUs, so that the higher-performance GPUs such as NVIDIA A100 and H100 are dedicated to more demanding jobs. This analysis also provides evidence to support the integration of GPU-sharing tools such as NVIDIA Multi-Process Service and Multi-Instance GPU.

In summary, we have shown that our analysis methodology can be applied to study GPU underutilization behavior in all three traces. Low CPU utilization and short runtime appear in all three trace cause analyses, which reveals the information that these jobs are likely spawned as debug runs – the run may work out quickly and the user terminates the job, or the run may require more modification to the code, so the CPU would also have low utilization. The PAI and SuperCloud trace also have other good indicators of GPU underutilization, such as resource request, job group, and GPU memory usage.

C. *Failed GPU Jobs*

Job failure is also an important monitoring metric for HPC system operation. We have observed that a significant number of jobs have failed or terminated unexpectedly in the three systems in Fig. 5. In SuperCloud and Philly trace, the killed label is for jobs that are manually terminated by the user, and the failed label is for unexpected failure. PAI trace has the highest job failure rate, but in its successfully terminated jobs, there is no label for user-killed jobs. In all three traces, the failed job accounts for a considerable portion ($>$ 13%), and this motivated us to study the association rules related to these failed jobs.

PAI Trace. In Table V, we show several strong rules related to job failure. First, we do rule mining for cause analysis (C1-C6) and then, characteristics analysis (A1-A2) – in both cases, we discuss how system operators can infer easier rules for better or proactive operations.

In the cause analysis (rule C1 to C6), we observe that job failures can be predicted by job information at submission, i.e., request CPU/GPU resources, user id, and job group. C1 reveals that when a user requests fewer CPU cores than one usually would for a frequent group job, the job is most likely to fail. In C2, the used GPU memory is still 0GB when the job fails, meaning the failure happens before the model and parameters get loaded into the GPU. This is possibly due to a library import error. Surprisingly, in rule C3, when a frequent user submits a job belonging to a common job group, 91% of the jobs have failed. This is primarily due to one user submitting a large number of jobs in the PAI trace. This information is helpful for root cause identification, as system operators can focus on the high failure rate of users and provide corresponding support.

TABLE V: Job failure rules from PAI trace.

	Antecedent	Consequent	Supp.	Conf.	Lift
C1	CPU Request = Bin1, Freq Group	GPU Type = None, Failed	0.11	0.95	4.41
C2	Mem Used = Bin1, GMem Used = 0GB, Freq Group	SM Util = 0%, Failed	0.08	0.95	4.32
C3	Freq User, Freq Group	Failed	0.10	0.91	3.46
C4	GMem Used = 0GB, GPU Request = Bin2	Failed	0.08	0.91	3.47
C5	SM Util = 0%, GPU Request = Bin2	Failed	0.1	0.71	2.69
C6	Memory Used = Bin1	Failed	0.17	0.67	2.54
A1	Freq Group, Failed	CPU Request = Bin1, Mem Used = Bin1, Mem Request = Std	0.10	0.81	7.32
A2	Failed	GPU Type = None, Tensorflow, Mem Request = Std, SM util = 0%	0.17	0.63	1.93

In Table V, both C4 and C5 show similar symptoms: a user requests a decent number of GPUs (25 $\leq$ number of GPUs $<$ 100 in PAI trace), but does not properly use the GPU cores and memory. Deep learning job schedulers in HPC systems would typically optimize for distributed jobs that request a high number of GPUs because these jobs require gang-scheduling [9], [11], [12]. The system operator should suggest that the users test their jobs with less number of GPUs before deploying at a large scale. C6 is a straightforward rule as 67% of jobs that have low memory usage have also failed.

From the characteristic analysis rules A1 and A2 in Table V, we see that failed jobs share some similar properties with the GPU underutilization jobs (Table II) in PAI trace: standard memory request, unspecified GPU type, and Tensorflow framework. In A2, 0% GPU SM utilization is even in the consequent of a job failure rule, meaning failed jobs are likely to have GPU underutilization.

Takeaways: For PAI trace, our methodology helps us discover multiple simple rules to understand the characteristics of failed jobs. The presence of multiple strong rules indicates that a simple rule-based or tree-based classifier will suffice for prediction of job failures.

SuperCloud Trace. For the SuperCloud trace, we find a limited number of rules in Table VI. Although the GPU memory and CPU utilization cannot be used as good predictors of job failure due to low confidence, the lift value shows that jobs with low GPU memory and CPU utilization are nearly twice more likely to fail. When the keyword is in the antecedent, A1 strengthens the association between failure and GPU memory utilization. A2 shows that about 40%

TABLE VI: Job failure rules from SuperCloud trace.

	Antecedent	Consequent	Supp.	Conf.	Lift
C1	GMem Util = Bin1	Failed	0.06	0.25	1.93
C2	CPU Util = Bin1	Failed	0.06	0.25	1.90
A1	GPU Power = Bin1, Failed	GMem Util = Bin1	0.05	0.91	3.64
A2	Failed	Runtime = Bin4	0.05	0.41	1.66

TABLE VII: Job failure rules from Philly trace.

	Antecedent	Consequent	Supp.	Conf.	Lift
C1	Multi-GPU	Failed	0.05	0.40	2.55
C2	New User	Failed	0.08	0.38	2.46
A1	Min SM Util = 0%, Failed	Num Attempts > 1	0.06	0.56	3.79
A2	Min SM Util = 0%, Failed	Runtime = Bin4	0.05	0.55	2.20

of failed jobs have relatively long runtime (from 8 hours to a few weeks) in the SuperCloud cluster. Since machine learning workloads have static execution between mini-batch iterations [48], these workloads are unlikely to execute for 8 hours and fail in one particular epoch. Therefore, these errors are likely caused by node failures or exceeding allocated time limits.

Philly Trace. For Philly trace, the results in Table VII show that multi-GPU jobs and jobs submitted by new users are approximately 2.5 times more likely to fail compared to the general failure rate. We expect multi-GPU jobs to fail more often because, in distributed training, the failure of one worker would cause the whole job to fail [19]. We do not observe this association in the PAI trace because more than 99% of the jobs are multi-GPU jobs. The association is also not mined in our SuperCloud trace because 97% of the jobs are single-GPU jobs, hence multi-GPU jobs do not have enough support. As a comparison, 14% of jobs in Philly trace use multiple GPUs. Rule C2 has shown opposite observations to the PAI trace. In Philly trace, new users are more likely to submit failed jobs, whereas in PAI trace it is frequent users. We recognize that even though both PAI and Philly traces are collected on production datacenters for machine learning, they still have significant differences in perspectives including (1) Philly is collected 3 years before PAI; (2) Philly cluster is only for training, PAI is an MLaaS cloud platform for both training and inference; (3) Philly workloads are CNNs and RNNs, while PAI includes not only these workloads but also NLP, RL, and GNNs. Nevertheless, the common observation is that the failures are associated with users instead of nodes or clusters. This also aligns well with an internal study from Microsoft: user/programming errors lead to a lot of failures [11].

Takeaways: To accurately predict failure for systems like SuperCloud and Philly, more complex models such as neural networks will be needed. For highly distributed jobs, the system can set up a small number of nodes dedicated to screening before sending the actual GPU request number to the scheduler for gang scheduling.

TABLE VIII: Interesting trace-specific rules.

	Antecedent	Consequent	Supp.	Conf.	Lift
PAI1	GPU Type = T4	Queue = Bin1	0.18	0.85	3.70
PAI2	GPU Type = None T4	Queue = Bin4	0.06	0.52	1.82
PAI3	Model = RecSys	GPU Type = T4, Multiple Tasks	0.29	0.88	2.98
PAI4	CPU Util = Bin0, SM Util = Bin4	Model = NLP	0.07	0.99	1.71
CIR1	New User	Job Killed	0.05	0.26	1.75
PHI1	Multi-GPU	Runtime = Bin4	0.07	0.50	2.01

The rule A1 in Table VII is specific to the Philly trace, as the Philly cluster automatically attempts to restart the job when there is an error. The rule also shows that failed jobs do not always get another attempt. The rule A2 shows a similar observation as the SuperCloud trace – a significant number of failed jobs have exceptionally long runtime, and not all job failures occur soon after launch.

In summary, we are able to extract some high-confidence rules from the PAI trace to forecast a failure using job submission information. We find that associations exist between job failures and GPU underutilization, addressing one issue will alleviate another. For SuperCloud and Philly, the rules are not high confidence, but they show insights about relationships between certain job attributes to failure. Some failed jobs run longer than most jobs before the error occurs, which requires more attention as more compute cycles get wasted compared to jobs that fail early.

D. Misc. Interesting Rules

In addition to common concerns about GPU underutilization and job failure, we also list a few other interesting rules with attributes unique to the trace in Table VIII.

In PAI system, the user can choose to specify a GPU type from the power-efficient NVIDIA T4, or the performant P100 and V100 GPUs. Due to their low support, we label P100 and V100 as non-T4 GPUs. The rules PAI1 and PAI2 in Table VIII show that the queue wait times of T4 and non-T4 GPUs are opposite. This is interesting because the ratio of T4 to none-T4 GPUs in PAI is 1:3.5. These rules provide insights on how to balance between GPU types when building a heterogeneous cluster.

The PAI trace has provided the ML model types as labels for some jobs. We have filtered out the jobs whose model type label is NaN and applied the analysis on the processed dataset to discover workload-related rules. We have found rules PAI3 and PAI4 that are specific to certain deep learning models in Table VIII. From these rules, we can infer that (1) recommender system models tend to use the T4 GPU which is designed for inference, and each job spawns multiple recommender tasks in parallel; (2) when a job has relatively low CPU compute demand but high GPU core utilization, it is most likely a language model job. System designers can integrate such model-specific information to further optimize the MLaaS infrastructure, for example, assigning recommender jobs to efficient T4 GPUs and NLP jobs to high-performance V100 GPUs.

A significant number of jobs are killed by the user in SuperCloud and Philly systems (Fig. 5). In Table VIII, rule CIR1 implicates that in the SuperCloud system, new users are 75% more likely to kill their jobs. This behavior is different from new users in Philly system, who tend to have job failures instead of manual terminations. Another interesting rule about multi-GPU jobs in Philly is that they tend to run for a very long time (rule PHI1). A job scheduler should consider the potential long execution time of multi-GPU jobs, especially for policies like shortest-jobs-first.

V. Discussions

Simplicity in framework configuration. Utilizing our association rule framework circumvents the necessity for extensive parameter tuning inherent in model-based machine learning methodologies. Distinctively, users are required to determine only the suitable parameters for support and lift, which function more as filters rather than complex hyperparameters demanding meticulous adjustments. Unlike the intricate interactions observed with parameters such as SGD (Stochastic Gradient Descent) learning rates, the adjustment of support and lift parameters is straightforward: to reduce the abundance of rules, one simply increases the thresholds, and conversely, to augment the rule count, decreases them. This procedural simplicity contrasts sharply with the tuning of ML model training hyperparameters like learning rates. Further emphasizing the framework's ease of use, our empirical studies across three distinct datacenter traces consistently applied identical support and lift thresholds, demonstrating the framework's adaptability and user-friendly nature.

Insights and findings from case studies. The implementation of our association rule mining framework has unveiled novel insights into data center operations, challenging and extending previous understandings. For instance, contrary to the absence of correlation in earlier studies [9], our analysis identifies a significant link between the request for a minimal number of GPUs and GPU underutilization (refer to Table II). Additionally, against conventional expectations that frequent users would have a lower incidence of job failures due to more stable and mature job submissions, our findings indicate a pronounced association between high-frequency user groups and increased job failures (see Table V). Furthermore, our framework is capable of deducing the application domain from patterns of CPU-GPU utilization, such as identifying NLP applications (as demonstrated in Table VIII). Such connections were not made in the original analysis of the PAI trace [9].

Beyond uncovering unexpected and insightful correlations, our framework significantly contributes to the field by providing quantitative evidence for each association. While domain experts might intuitively recognize certain patterns based on experience, the relative significance of these association rules has remained elusive. Our framework fills this gap by facilitating a comparative analysis of the strength of various rules. Moreover, it democratizes the process of insight generation, enabling individuals, regardless of their prior domain knowledge, to rapidly derive meaningful conclusions. Importantly, our approach lays the groundwork for a comprehensive analysis framework, generating all high-quality rules in a single execution. This simplifies the initial analysis and also, establishes a foundation for more detailed exploration tailored to specific interests or perspectives.

VI. Related Work

Association rules. Besides its original application area of market basket analysis [49], association rule mining has been widely adopted in big data analysis including studying COVID-19 [50], [38], [51], [52], [53], communication network management [54], [55], [56], and web mining [37], [57], [58]. However, even though a large amount of operational data is generated every day in HPC systems, there are only limited works that have performed rule analysis, and they only focus on certain server events [44], [59]. Our work is the first one to demonstrate rule mining's applicability to general HPC system analysis using real job traces from GPU-accelerated clusters.

Recent advances in association rule mining are focusing on privacy preservation [60], [61], [62], [63], analyzing streaming data [64], [65], [66], [67], and distributed mining in cloud computing [68], [69], [70], [71]. Our analysis of the job traces does not have constraints such as privacy or time window, but if needed, since our pruning techniques are applied after the rules are generated, we can integrate the other works into the workflow.

HPC trace analysis. The PAI [9] and Philly [11] work have conducted a comprehensive study on their traces, but none of the rules discovered in our work are present in the original paper. The Helios trace [12] is also a DL job trace from a GPU-accelerated system, but due to its lack of GPU measurement information in the public repository, we do not include Helios in this study.

For general systems, regardless of hardware acceleration, a number of prior works have performed characterizations on supercomputer operation [72], [73], [74], [75], [76], [33], cloud orchestration [77], [13], [14], [78], [15], [16], user behavior [73], [79], [80], and system failure [30], [81], [31], [82]. These analysis methodologies are either experiential (i.e., they are based on past operational experience, and only focus on particular features of the data, which cannot be extended to other traces) or uninterpretable. As a comparison, our analysis is systematically done across different clusters and can be directly interpreted by system operators.

VII. Conclusion

This work addresses the interpretability issue in data analysis by introducing and designing a workflow of association rule analysis. We evaluate our analysis workflow on three GPU-accelerated production traces of ML workloads and show that the proposed methodology can extract useful information about system operations to mitigate issues such as GPU underutilization and job failure. Our insights are summarized in rule tables, and our analysis framework has been open-sourced along with the SuperCloud trace we collected.

Acknowledgments

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001, and United States Air Force Research Laboratory Cooperative Agreement Number FA8750-19-2-1000. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering, or the United States Air Force. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[3] S. Hsia, U. Gupta, M. Wilkening, C.-J. Wu, G.-Y. Wei, and D. Brooks, "Cross-stack workload characterization of deep recommendation systems," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2020, pp. 157–168.

[4] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley, and Y. He, "Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale," in *International Conference on Machine Learning*. PMLR, 2022, pp. 18 332–18 346.

[5] S. Samsi, D. Zhao, J. McDonald, B. Li, A. Michaleas, M. Jones, W. Bergeron, J. Kepner, D. Tiwari, and V. Gadepally, "From words to watts: Benchmarking the energy costs of large language model inference," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2023, pp. 1–9.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.

[8] U. Gupta, C.-J. Wu, X. Wang, M. Naumov, B. Reagen, D. Brooks, B. Cottel, K. Hazelwood, M. Hempstead, B. Jia *et al.*, "The architectural implications of facebook's dnn-based personalized recommendation," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 488–501.

[9] "MLaaS in the wild: Workload analysis and scheduling in Large-Scale heterogeneous GPU clusters," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/weng

[10] B. Li, R. Arora, S. Samsi, T. Patel, W. Arcand, D. Bestor, C. Byun, R. B. Roy, B. Bergeron, J. Holodnak *et al.*, "Ai-enabling workloads on large-scale gpu-accelerated system: characterization, opportunities, and implications," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 1224–1237.

[11] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, "Analysis of large-scale multi-tenant {GPU} clusters for {DNN} training workloads," in *2019 {USENIX} Annual Technical Conference ({USENIX}{ATC} 19)*, 2019, pp. 947–960.

[12] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[13] S. Luo, H. Xu, C. Lu, K. Ye, G. Xu, L. Zhang, Y. Ding, J. He, and C. Xu, "Characterizing microservice dependency and performance: Alibaba trace analysis," in *Proceedings of the ACM Symposium on Cloud Computing*, 2021, pp. 412–426.

[14] M. Wang, C. Meng, G. Long, C. Wu, J. Yang, W. Lin, and Y. Jia, "Characterizing deep learning training workloads on alibaba-pai," in *2019 IEEE international symposium on workload characterization (IISWC)*. IEEE, 2019, pp. 189–202.

[15] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces," in *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*. IEEE, 2019, pp. 1–10.

[16] Y. Cheng, Z. Chai, and A. Anwar, "Characterizing co-located datacenter workloads: An alibaba case study," in *Proceedings of the 9th Asia-Pacific Workshop on Systems*, 2018, pp. 1–3.

[17] Y. Kurnia, Y. Isharianto, Y. C. Giap, A. Hermawan *et al.*, "Study of application of data mining market basket analysis for knowing sales pattern (association of items) at the o! fish restaurant using apriori algorithm," in *Journal of Physics: Conference Series*, vol. 1175, no. 1. IOP Publishing, 2019, p. 012047.

[18] G. Ceddia, L. N. Martino, A. Parodi, P. Secchi, S. Campaner, and M. Masseroli, "Association rule mining to identify transcription factor interactions in genomic regions," *Bioinformatics*, vol. 36, no. 4, pp. 1007–1013, 2020.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[20] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[21] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[22] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*. Springer, 2003, pp. 44–60.

[23] S. Samsi, M. L. Weiss, D. Bestor, B. Li, M. Jones, A. Reuther, D. Edelman, W. Arcand, C. Byun, J. Holodnack *et al.*, "The mit supercloud dataset," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2021, pp. 1–8.

[24] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[25] J. Gu, M. Chowdhury, K. G. Shin, Y. Zhu, M. Jeon, J. Qian, H. Liu, and C. Guo, "Tiresias: A {GPU} cluster manager for distributed deep learning," in *16th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 19)*, 2019, pp. 485–500.

[26] K. Mahajan, A. Balasubramanian, A. Singhvi, S. Venkataraman, A. Akella, A. Phanishayee, and S. Chawla, "Themis: Fair and efficient {GPU} cluster scheduling," in *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, 2020, pp. 289–304.

[27] S. Chaudhary, R. Ramjee, M. Sivathanu, N. Kwatra, and S. Viswanatha, "Balancing efficiency and fairness in heterogeneous gpu clusters for deep learning," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[28] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed {DNN} training in heterogeneous gpu/cpu clusters," in *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, 2020, pp. 463–479.

[29] A. Qiao, S. K. Choe, S. J. Subramanya, W. Neiswanger, Q. Ho, H. Zhang, G. R. Ganger, and E. P. Xing, "Pollux: Co-adaptive cluster scheduling for goodput-optimized deep learning," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021.

[30] D. Das, M. Schiewe, E. Brighton, M. Fuller, T. Cerny, M. Bures, K. Frajtak, D. Shin, and P. Tisnovsky, "Failure prediction by utilizing log analysis: A systematic mapping study," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, 2020, pp. 188–195.

[31] B. Nie, J. Xue, S. Gupta, T. Patel, C. Engelmann, E. Smirni, and D. Tiwari, "Machine learning models for gpu error prediction in a large scale hpc system," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2018, pp. 95–106.

[32] L. Wang, Q. Weng, W. Wang, C. Chen, and B. Li, "Metis: Learning to schedule long-running applications in shared container clusters at scale," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–17.

[33] E. Costa, T. Patel, B. Schwaller, J. M. Brandt, and D. Tiwari, "Systematically inferring i/o performance variability by examining repetitive job behavior," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.

[34] M. Isakov, E. Del Rosario, S. Madireddy, P. Balaprakash, P. Carns, R. B. Ross, and M. A. Kinsy, "Hpc i/o throughput bottleneck analysis with explainable local models," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–13.

[35] P. J. Pavan, J. L. Bez, M. S. Serpa, F. Z. Boito, and P. O. Navaux, "An unsupervised learning approach for i/o behavior characterization," in *2019 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. IEEE, 2019, pp. 33–40.

[36] J. Kleinberg, "An impossibility theorem for clustering," *Advances in neural information processing systems*, pp. 463–470, 2003.

[37] Y. Djenouri, H. Drias, Z. Habbas, and H. Mosteghanemi, "Bees swarm optimization for web association rule mining," in *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, vol. 3. IEEE, 2012, pp. 142–146.

[38] M. Shahin, W. Inoubli, S. A. Shah, S. B. Yahia, and D. Draheim, "Distributed scalable association rule mining over covid-19 data," in *International Conference on Future Data and Security Engineering*. Springer, 2021, pp. 39–52.

[39] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data mining and knowledge discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[40] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215. Citeseer, 1994, pp. 487–499.

[41] Y. Wu and J. Zhang, "Building the electronic evidence analysis model based on association rule mining and fp-growth algorithm," *Soft Computing*, vol. 24, no. 11, pp. 7925–7936, 2020.

[42] H.-Y. Chang, J.-C. Lin, M.-L. Cheng, and S.-C. Huang, "A novel incremental data mining algorithm based on fp-growth for big data," in *2016 International Conference on Networking and Network Applications (NaNA)*. IEEE, 2016, pp. 375–378.

[43] D. Dong, Z. Ye, Y. Cao, S. Xie, F. Wang, and W. Ming, "An improved association rule mining algorithm based on ant lion optimizer algorithm and fp-growth," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1. IEEE, 2019, pp. 458–463.

[44] F. Lin, K. Muzumdar, N. P. Laptev, M.-V. Curelea, S. Lee, and S. Sankar, "Fast dimensional analysis for root cause investigation in a large-scale service environment," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 2, pp. 1–23, 2020.

[45] C. Wu, S.-C. Kao, and K. Okuhara, "Examination and comparison of conflicting data in granulated datasets: Equal width interval vs. equal frequency interval," *Information Sciences*, vol. 239, pp. 154–164, 2013.

[46] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective cluster scheduling for deep learning," in *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, 2018, pp. 595–610.

[47] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-Aware cluster scheduling policies for deep learning workloads," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, Nov. 2020, pp. 481–498. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/narayanan-deepak

[48] A. Gujarati, R. Karimi, S. Alzayat, W. Hao, A. Kaufmann, Y. Vigfusson, and J. Mace, "Serving dnns like clockwork: Performance predictability from the bottom up," in *14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 20)*, 2020, pp. 443–462.

[49] M. Kaur and S. Kang, "Market basket analysis: Identify the changing trends of market data using association rule mining," *Procedia computer science*, vol. 85, pp. 78–85, 2016.

[50] M. Tandan, Y. Acharya, S. Pokharel, and M. Timilsina, "Discovering symptom patterns of covid-19 patients using association rule mining," *Computers in biology and medicine*, vol. 131, p. 104249, 2021.

[51] S. Katragadda, R. Gottumukkala, R. T. Bhupatiraju, A. M. Kamal, V. Raghavan, H. Chu, R. Kolluru, and Z. Ashkar, "Association mining based approach to analyze covid-19 response and case growth in the united states," *Scientific Reports*, vol. 11, no. 1, pp. 1–12, 2021.

[52] R. Gupta, M. Bedi, P. Goyal, S. Wadhera, and V. Verma, "Analysis of covid-19 tracking tool in india: case study of aarogya setu mobile application," *Digital Government: Research and Practice*, vol. 1, no. 4, pp. 1–8, 2020.

[53] S. Chakraborty, S. Biswas, and S. Debnath, "Prediction and analysis on covid-19 using positive and negative association rule mining," in *Proceedings of Research and Applications in Artificial Intelligence*. Springer, 2021, pp. 1–11.

[54] X. Zhang, Y. Bai, P. Feng, W. Wang, S. Liu, W. Jiang, J. Zeng, and R. Wang, "Network alarm flood pattern mining algorithm based on multi-dimensional association," in *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2018, pp. 71–78.

[55] D. D. Arifin, M. A. Bijaksana *et al.*, "Enhancing spam detection on mobile phone short message service (sms) performance using fp-growth and naive bayes classifier," in *2016 IEEE Asia Pacific Conference on Wireless and Mobile (APWiMob)*. IEEE, 2016, pp. 80–84.

[56] Q. Zheng, Y. Li, and J. Cao, "Application of data mining technology in alarm analysis of communication network," *Computer Communications*, vol. 163, pp. 84–90, 2020.

[57] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa, "Effective personalization based on association rule discovery from web usage data," in *Proceedings of the 3rd international workshop on Web information and data management*, 2001, pp. 9–15.

[58] C.-H. Lee, Y.-H. Kim, and P.-K. Rhee, "Web personalization expert with combining collaborative filtering and association rule mining technique," *Expert Systems with Applications*, vol. 21, no. 3, pp. 131–137, 2001.

[59] A. Gainaru, F. Cappello, J. Fullop, S. Trausan-Matu, and W. Kramer, "Adaptive event prediction strategy with dynamic time window for large-scale hpc systems," in *Managing Large-scale Systems via the Analysis of System Logs and the Application of Machine Learning Techniques*, 2011, pp. 1–8.

[60] N. Domadiya and U. P. Rao, "Privacy preserving association rule mining on distributed healthcare data: Covid-19 and breast cancer case study," *SN Computer Science*, vol. 2, no. 6, pp. 1–9, 2021.

[61] C. Huang, R. Lu, and K.-K. R. Choo, "Secure and flexible cloud-assisted association rule mining over horizontally partitioned databases," *Journal of Computer and System Sciences*, vol. 89, pp. 51–63, 2017.

[62] X. Yi, F.-Y. Rao, E. Bertino, and A. Bouguettaya, "Privacy-preserving association rule mining in cloud computing," in *Proceedings of the 10th ACM symposium on information, computer and communications security*, 2015, pp. 439–450.

[63] N. Domadiya and U. P. Rao, "Privacy preserving distributed association rule mining approach on vertically partitioned healthcare data," *Procedia computer science*, vol. 148, pp. 303–312, 2019.

[64] T. Martin, G. Francoeur, and P. Valtchev, "Ciclad: A fast and memory-efficient closed itemset miner for streams," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1810–1818.

[65] T. Martin, P. Valtchev, and L.-R. Roux, "Fgc-stream: A novel joint miner for frequent generators and closed itemsets in data streams," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 419–428.

[66] J. Liu, Z. Ye, X. Yang, X. Wang, L. Shen, and X. Jiang, "Efficient strategies for incremental mining of frequent closed itemsets over data streams," *Expert Systems with Applications*, vol. 191, p. 116220, 2022.

[67] Y. Yamamoto, Y. Tabei, and K. Iwanuma, "Parasol: a hybrid approximation approach for scalable frequent itemset mining in streaming data,"

Journal of Intelligent Information Systems, vol. 55, no. 1, pp. 119–147, 2020.

[68] S. Rathee and A. Kashyap, "Adaptive-miner: an efficient distributed association rule mining algorithm on spark," *Journal of Big Data*, vol. 5, no. 1, pp. 1–17, 2018.

[69] V. S. Ms and K. Shah, "Performance evaluation of distributed association rule mining algorithms," *Procedia Computer Science*, vol. 79, pp. 127–134, 2016.

[70] R. Liu, K. Yang, Y. Sun, T. Quan, and J. Yang, "Spark-based rare association rule mining for big datasets," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 2734–2739.

[71] D. Apiletti, E. Baralis, T. Cerquitelli, S. Chiusano, and L. Grimaudo, "Searum: A cloud-based service for association rule mining," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2013, pp. 1283–1290.

[72] G. P. Rodrigo, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards understanding hpc users and systems: a nersc case study," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 206–221, 2018.

[73] T. Patel, Z. Liu, R. Kettimuthu, P. Rich, W. Allcock, and D. Tiwari, "Job characteristics on large-scale systems: long-term analysis, quantification, and implications," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–17.

[74] A. Pecchia, D. Cotroneo, Z. Kalbarczyk, and R. K. Iyer, "Improving log-based field failure data analysis of multi-node computing systems," in *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*. IEEE, 2011, pp. 97–108.

[75] N. A. Simakov, J. P. White, R. L. DeLeon, S. M. Gallo, M. D. Jones, J. T. Palmer, B. Plessinger, and T. R. Furlani, "A workload analysis of nsf's innovative hpc resources using xdmod," *arXiv preprint arXiv:1801.04306*, 2018.

[76] V. V. Stegailov and G. E. Norman, "Challenges to the supercomputer development in russia: a hpc user perspective," *Program Systems: Theory and Applications*, vol. 5, no. 1, pp. 111–152, 2014.

[77] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the Tenth European Conference on Computer Systems*, 2015, pp. 1–17.

[78] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *Proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 153–167.

[79] S. Schlagkamp, R. F. da Silva, J. Renker, and G. Rinkenauer, "Analyzing users in parallel computing: A user-oriented study," in *2016 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE, 2016, pp. 395–402.

[80] N. Wolter, M. O. McCracken, A. Snavely, L. Hochstein, T. Nakamura, and V. Basili, "What's working in hpc: Investigating hpc user behavior and productivity," *CTWatch Quarterly*, vol. 2, no. 4A, pp. 9–17, 2006.

[81] S. Gupta, T. Patel, C. Engelmann, and D. Tiwari, "Failures in large scale systems: long-term measurement, analysis, and implications," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2017, pp. 1–12.

[82] D. Tiwari, S. Gupta, J. Rogers, D. Maxwell, P. Rech, S. Vazhkudai, D. Oliveira, D. Londo, N. DeBardeleben, P. Navaux *et al.*, "Understanding gpu errors on large-scale hpc systems and the implications for system design and operation," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2015, pp. 331–342.