An Empirical Study of Major Page Faults for Failure Diagnosis in Cluster Systems

Edward Chuah[1*], Arshad Jhumka[2] and Sai Narasimhamurthy[3]

[1*]The University of Aberdeen, Aberdeen, AB24 3FX, UK.
[2]The University of Warwick, Coventry, CV4 7AL, UK.
[3]Seagate Technology, Portsmouth, PO9 1SA, UK.

*Corresponding author(s). E-mail(s): thuan.chuah@abdn.ac.uk; Contributing authors: H.A.Jhumka@warwick.ac.uk; sai.narasimhamurthy@seagate.com;

Abstract

High-Performance Computing (HPC) systems conduct extensive logging of resource usage data and system logs, and parsing this data is an often advocated basis for failure diagnosis. Major page faults are known to be one of the most common cause of performance problems in large cluster systems. We conduct an empirical study of major page faults on two large cluster systems. We set up three regression algorithms including the LASSO, Ridge and Elastic Net regression techniques. To the best of our knowledge, there is no work that studied different regression models to diagnose major page faults in a large cluster system. In this paper, we (a) propose an approach for diagnosing major page faults, and (b) evaluate the LASSO, Ridge and Elastic Net regression algorithms on real resource use data and system logs. As part of our contributions, we (a) compare the accuracy of the three regression algorithms, (b) identify the resource use counters which are correlated to major page faults and the system events which are correlated to page fault events, and (c) provide insights into major page faults and page fault events. Our work highlights empirical observations that could facilitate better handling of node failures in cluster systems.

Keywords: Large cluster systems, Major page faults, Regression analysis, Resource use data, System logs

1 Introduction

Correlation analysis has been widely applied to diagnose node failures in HPC systems [1, 2], with its strengths in dealing with many system events with linear and non-linear patterns, which can result in significant improvements in diagnostics accuracy. A very recent study empirically evaluated a set of correlation techniques to diagnose node failures in large cluster systems, including Pearson correlation, Spearman-Rank correlation and Partial correlation, using system logs obtained on two widely deployed cluster systems [3]. From their evaluation, they observed that the Partial correlation method outperformed the Pearson and Spearman-Rank correlation algorithms, identifying memory data updates and Error Correcting Code memory errors as indirect causes of node crashes. They highlighted that memory data updates were associated with the Lustre filesystem I/O activities, and corrupted memory indexes were associated with segmentation faults that led to node crashes on many dates.

Several important large-scale failure analysis studies have shown that failures in exascale computing systems are caused by different applications, system software and hardware [4–12]. M. Snir *et al.* [4] presented a detailed report on resilience in exascale computing systems. They defined the problem of providing resilience for exascale computers, described the sources of hardware and software errors, and discussed software that can prevent, detect and recover from those errors. S. Mitra *et al.* [6] performed in-depth analyses of the usage patterns of applications and libraries on the Conte community cluster. They introduced novel analysis techniques that identified hidden trends and diagnosed job failures in cluster systems. C. Di Martino *et al.* [5] provided a detailed analysis of failures and discussed the impact of those failures on the Blue Waters supercomputer. They identified the causes of single node failures, analyzed system-wide outages, and assessed the error resiliency of the memory, CPU, network and filesystem. S. Gupta *et al.* [7] presented a detailed study that analyzed the failure characteristics of several large-scale supercomputers. They identified 23 different types of failures that include hardware failures such as node hangs and machine check exceptions, and software failures such as Lustre filesystem server and operating system kernel panic. R. Kumar *et al.* [9] provided a detailed analysis of job and node failures on two large cluster systems. They observed that node and job failures are correlated with the system resource usage, and they proposed a model that predicted job failures. Z. Liu *et al.* [10] performed a detailed correlation analysis of hardware event logs and task failure logs on the MIRA supercomputer and analyzed job runs, I/O behaviour and failure characteristics. E. Rojas *et al.* [11] presented a detailed reliability analysis of the Titan supercomputer using five years worth of failure and workload data. They classified failures, modelled the failure rate and assessed the relationship between failures and workloads to characterize Titan's reliability. K.B. Ferreira *et al.* [12] analyzed memory failure data and presented a detailed analysis of memory failures on the Astra large cluster system. Differently to these works [4, 5, 7, 9–12], we study the impact of major page faults on node failures. E. Rojas *et al.* [8] presented the results of analyzing five years worth of failure and system workload data from the Titan supercomputer. They developed a methodology that consists of five phases: (a) workload data formatting, (b) failure data filtering, (c) failure analysis, (d) interplay analysis and (e) visualization. Different to the work by E. Rojas *et al.* [8], we developed an approach for diagnosing major page faults. Our approach consists of three phases: (a) data preprocessing, (b) event correlation and (c) statistical validation. It has been widely reported that major page faults are a common cause of performance problems in cluster systems [13–15]. Major page faults incur significant overheads on the cluster system's performance, which reduces its ability to effectively utilize its resources and complete executing running jobs on time.

In the operating system, page fault handling is a mechanism that handles data or code required by the CPU but they are not in the main memory. It is a crucial mechanism that increases the amount of memory available to applications. However, it also puts pressure on the storage system. As such, it is as important to capture relationships between major page faults and node failures, as it is to capture relationships between multiple system components. We defined a node failure as a node crash or an operating system hangup. The state-of-the-art failure diagnostics frameworks have implemented workflows that use Pearson correlation, Spearman-Rank correlation and/or partial correlation to correlate system errors to node failures [1–3, 16, 17]. Although these works have shown that Pearson correlation, Spearman-Rank correlation and partial correlation techniques can identify the cause of node failures, these correlation algorithms have some limitations that we address in this paper. Firstly, the Pearson and Spearman-Rank correlation methods can only identify the relationship of two variables and the partial correlation method can only identify the relationship of two variables while controlling for the effect of a third variable. Secondly, the correlation approach can produce many pairs of correlated variables that must be thoroughly looked at before it is possible to make a diagnosis, which is not desirable as it is a time consuming process that incurs a significant delay in diagnosing a node failure. For example, the diagnostics framework presented in [3] may generate the following diagnosis: "CPU core 1 and CPU core 2 are waiting for I/O operations to complete which led to a node crash". A HPC server typically contains 16 to 68 cores per CPU socket and multiple I/O operations may be executing in parallel. Thus, it is important to understand the role of the page fault handling mechanism in relation to multiple CPUs waiting on I/O operations to complete and a compute node soft lockup.

As such, we leverage the power of regression algorithms, which are supervised statistical learning techniques that learn the regression coefficients of multiple independent variables to a target or dependent variable. We compare the LASSO, Ridge and Elastic Net regression algorithms, which are three established and widely used supervised statistical learning techniques. We apply the LASSO, Ridge and Elastic Net regression algorithms on the resource usage data and system logs to determine the regression algorithm's applicability to identify the resource use counters which are correlated to major page faults and the system events which are correlated to page fault events.

In this paper, we conduct an empirical analysis of major page faults in large cluster systems and present several new findings. The correlation of multiple CPU and Lustre filesystem resource use counters to major page faults, and the correlation of multiple Lustre error messages and multiple Linux system error messages to page fault events are new and have not heretofore been reported in our earlier papers [3, 16, 17]. We validate our diagnostics approach on two large cluster systems operated by the Texas Advanced Computing Center at The University of Texas at Austin. The benefits of applying regression models to diagnose major page faults are given as follows: (a) when multiple CPU and Lustre filesystem resource use counters are strongly positive correlated to the major page fault counter, and (b) multiple Lustre error messages and multiple Linux error messages are strongly correlated to a page fault event, it shows that the page fault handling mechanism is associated with the generation of Lustre and Linux error messages. As a result, those correlations can be used to monitor the state of the operating system's page fault handling mechanism. Our main contributions are given as follows:

- We propose an approach to diagnose major page faults using real world resource usage data and system logs.
- We compare multiple regression algorithms to ascertain their accuracy in diagnosing major page faults on two large cluster systems.
- We apply a statistical validation step to ensure accurate diagnosis of major page faults.

Our initial assumption is that when multiple resource use counters or multiple system events are used to train the regression algorithms, the accuracy for diagnosing a major page fault or a page fault event will improve. We compared the LASSO, Ridge and Elastic Net regression models on the two cluster systems and observed that (a) multiple regression models replicated the observed values in the resource usage data and system logs with the highest accuracy on different dates, and (b) there is no difference between the accuracy of the LASSO and Ridge regression models that were trained only on the strongly correlated resource use counters and the accuracy of the LASSO and Ridge regression models that were trained on all the resource use counters, so training the regression algorithms on the correlated resource use counters produces regression models that identify major page faults with the highest accuracy. Furthermore, we observed that (a) the LASSO and Ridge regression models replicated the observed values in the system logs with the highest accuracy on different dates, and (b) the LASSO regression algorithm that was trained on all the system events obtained the highest accuracy compared to the LASSO regression algorithm that was trained only on the correlated system events, so training the LASSO regression algorithm on all the system events produces a regression model that identifies page fault events with the highest accuracy.

This paper is organized as follows. We analyze the related work in Section 2 and present the system and fault models in Section 3. The sequence of events that occur when a major page fault occurs and the details of our diagnostics approach are described in Section 4. We evaluate our diagnostics approach on two large cluster systems in Section 5, discuss the limitations of our work in Section 6 and conclude with a summary and future work in Section 7.

2 Related Work

In this section, we review the related works with respect to the following criteria: (a) failure diagnostics frameworks that were evaluated on large cluster systems, (b) failure diagnostics frameworks that used different and multiple types of system logs, and (c) frameworks that diagnosed failures in large cluster systems.

A. Oliner *et al.* [1] proposed a novel method called *Structure-of-Influence Graphs* (SIGs) for diagnosing problems in large, complex computer systems. Constructing a SIG consists of four steps. In step 1, two models are developed. The first model uses message timing information produced by each system component. The second model uses the message terms contained in the system messages. In step 2, the Kullback-Leibler divergence is used to compute anomaly signals. In step 3, correlations between anomaly signals are obtained with the Pearson correlation algorithm. In step 4, a SIG is constructed for a subset of n system components, which addressed the time complexity problem in constructing a SIG from all the system components.

Z. Zheng *et al.* [2] presented a novel 3-dimensional root-cause diagnostics approach for diagnosing cluster system failures. Their approach consists of four steps: (a) preprocessing the data, (b) integrating information across multiple types of system logs, (c) identifying the failure layer and (d) identifying the location and time of the cause of a given failure. In step 1, outliers in three different types of logs are removed using wavelet transformation. Then, redundant log-events are removed using temporal-spatial filtering. The Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is used to group jobs that share similar characteristics. In step 2, enviromental features which are closely related to the fatal log-events are extracted. In step 3, a job similarity score is computed and used to distinguish hardware, application or system software failures. In step 4, the location and time of the cause of the fatal event is identified using probabilistic causal pruning and dynamic time window generation techniques.

X. Fu *et al.* [18] proposed an event correlation mining framework called *LogMaster*. It consists of innovative algorithms for filtering the event logs, mining correlations of events from the event logs and constructing event correlation graphs (ECGs). Subsequently, X. Fu *et al.* [19] extended LogMaster and provided deeper and more accurate diagnosis of failures. Their approach mined correlation of events and used ECGs to analyze the relationships between fatal and non-fatal events.

E. Chuah *et al.* [16] presented a novel anomaly-correlation approach called *ANCOR* that linked resource usage anomalies with node failures. ANCOR consists of four modules: (a) resource use anomaly extractor, (b) system message types extractor, (c) time-bin correlator and (c) log-events sequence constructor. In the anomaly extractor module, three feature extraction methods were used to detect anomalous workload patterns. In the system message types extractor, message types are extracted from the system logs. In the time-bin correlator module, the Pearson correlation algorithm is used to identify the strongly positive correlated message types. In the log-events sequence constructor module, sequences of the correlated log-events are extracted given a failure event. Differently to ANCOR, the CRUMEL approach [17] applies Pearson correlation and Spearman-Rank correlation on the resource use data and Rationalized message logs. A Rationalized message log is a new type of log that incorporates a logical structure and job identification [20]. CRUMEL was evaluated on a large cluster system and it was showed that Spearman-Rank correlation identified more correlated system error events on more dates. Differently to CRUMEL, IFADE [3] uses partial correlation to identify previously unknown causes of node failures in cluster systems. The IFADE approach consists of three phases: (a) preprocessing the data, (b) extracting the features and (c) identifying partial correlation of those features. In the data preprocessing phase, message types and resource use counters are extracted from the system logs and resource use data, respectively. In the feature extraction phase, four different types of feature extraction methods are used to identify the important resource use counters and message types. In the partial correlation phase, the partial correlation method is used to identify a pair of message types or a pair of resource use counters which are strongly positive correlated after controlling for the presence of a third message type or resource use counter. If the correlation coefficient for two message types or two resource use counters is greater than or equal to 0.8, those message types or resource use counters are strongly positive correlated.

To the best of our knowledge, there is no work that applied regression models in failure diagnosis to diagnose major page faults in large cluster systems. A summary of the functions of the reviewed failure diagnostics frameworks is given in Table 1. A. Oliner *et al.* [1] integrated the Kullback-Leibler divergence and Pearson correlation algorithm to identify anomaly signals and diagnose the cause of system failures. Z. Zheng *et al.* [2] integrated temporal-spatial filtering and the DBSCAN algorithm to identify the location and time of system failures. X. Fu *et al.* [18, 19] integrated event correlation mining and correlation graphs to diagnose failures. E. Chuah *et al.* [3, 16, 17] used the Pearson correlation algorithm [16], the Pearson and Spearman-Rank correlation algorithms [17] and partial correlation [3] to diagnose node failures. Differently to [1–3, 16–19], we (a) developed a diagnostics approach that enables comparing different regression models, (b) evaluate the ability of the regression model to identify correlations of resource use counters or correlations of system events, and (c) identify the nodes which are associated with page fault events and compute node soft lockups.

Table 1 Functions of the Failure Diagnostics Frameworks

Study	Technique	Dataset	Focus
SIGs [1]	KL-divergence, Pearson correlation	HPC logs, vehicle sensor data	Failure diagnosis
Z. Zheng *et al.* [2]	Temporal-spatial, filtering, DBSCAN	BlueGene/L logs, Performance data	Failure diagnosis
IFADE [3]	Feature extraction, Partial correlation	Rationalized logs, Syslogs, Resource use data	Failure diagnosis
ANCOR [16]	Feature extraction, Pearson correlation	Resource use data, Rationalized logs	Failure diagnosis
LogMaster [18]	Events correlation, ECGs	HPC logs, Hadoop logs, BlueGene/L logs	Events mining
X. Fu *et al.* [19]	Events correlation, ECGs	HPC logs, Hadoop logs, BlueGene/L logs	Failure diagnosis
CRUMEL [17]	Pearson correlation, Spearman-Rank correlation	Resource use data, Rationalized logs	Failure diagnosis
This paper	**LASSO, Ridge & Elastic Net**	**Resource use data, system logs**	Failure diagnosis

3 System and Fault Models

In this section, we present the system model to which our diagnostics approach is applicable in Section 3.1. Then, we define the fault model in Section 3.2.

3.1 System Model

Our diagnostics approach is based on a generic model for cluster systems [16]. The model is illustrated in Fig. 1. It consists of a set of X number of nodes $N_1 ... N_X$, a set of Y number of jobs $J_1 ... J_Y$, a set of Z number of data collection time-bins $D_1 ... D_Z$, a set of system components including a filesystem FS, and a job scheduler JS. The job scheduler JS assigns jobs to nodes. Each node N_i, $1 \leq i \leq X$, job J_j, $1 \leq j \leq Y$, job scheduler JS and filesystem FS may write resource usage data to containers $RUD_1 ... RUD_n$, and write system logs to containers $SL_1 ... SL_m$. Data in the filesystem FS may be retrieved by each node N_i and job J_j.
Data collection: A typical system data collection setup is depicted in Fig. 2. It consists of three components: (a) system monitor, (b) data store, (c) analysis console. The system monitor aggregates various statistics from the nodes and jobs, and the data is transmitted to a data store. The data store receives the

resource use data and system logs from the system monitor. The resource use data and system logs are retrieved by the analysis console and the data is analyzed within the context of an activity, such as detecting page faults.

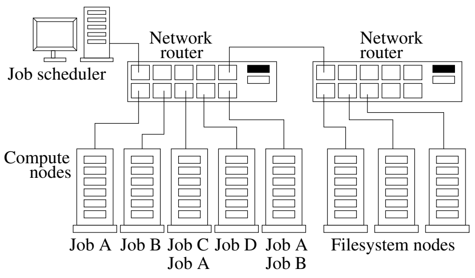

Fig. 1 Cluster system model

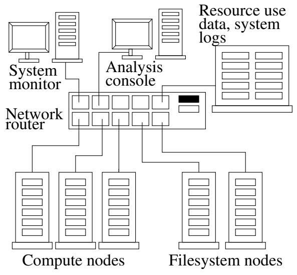

Fig. 2 Data collection

3.2 Fault Model

A fault model is an engineering model that represents something that can go wrong in the development or operation of a particular piece of equipment. A fault model consists of three terms [21]: (a) fault, (b) error, and (c) failure. A fault is represented as the hypothesized cause of an error. An error is produced when a service deviates from its correct state. When an error results in a loss of the service, a failure has occurred. A typical error propagation process is depicted in Fig. 3.

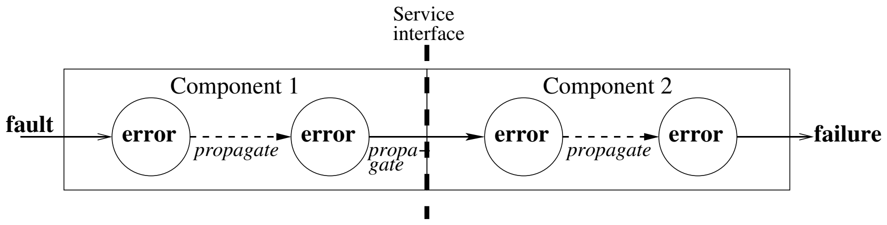

Fig. 3 An illustration of an error propagating through two system components.

When a fault occurs in one system component, the resulting error can propagate beyond that system component and affect multiple system components. For example, a corrupted inode (i.e., a fault) may result in an error in accessing a directory on the filesystem, and subsequently results in an error in the page fault handling mechanism if the required data or code is contained in that file directory. These errors can occur at different times and on multiple nodes. As such, there is a need to consider various discrete fault models in a HPC environment.

The system errors associated with a node failure are interleaved between thousands of other types of system events. Furthermore, the process for extracting the sequence of system errors is mostly manual and ad-hoc. There are also potentially different types of fault models associated with various system components, which make the task of determining the cause of a node failure more difficult.

4 Diagnosing Major Page Faults

A page fault is divided into two types: (a) minor page fault, and (b) major page fault. A minor page fault occurs when the data is present in the main memory, but the process has not obtained a logical mapping to the page. In contrast to a minor page fault, which does not require copying the data from the harddisk to main memory, a major page fault occurs when the data is not present in the main memory and the data must be copied from the harddisk to the main memory. A typical sequence of events when a major page fault occurs is depicted in Fig. 4 [22].

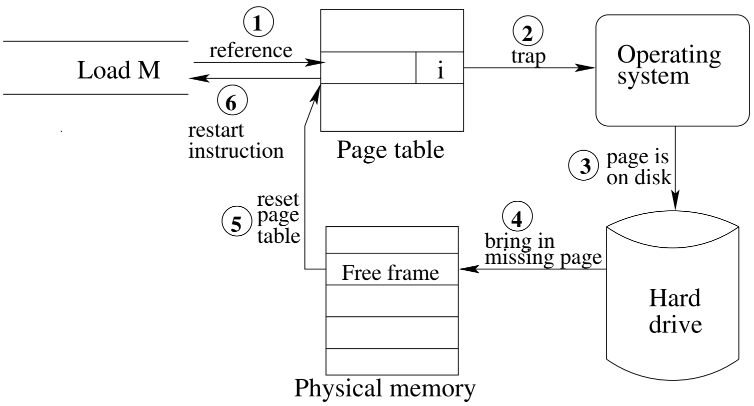

Fig. 4 Major page fault sequence of events.

When a program tries to access data that is in its address space, but the data is not located in the main memory, a major page fault occurs ($\textcircled{1}$). When a major page fault is detected, the computer hardware signals to the operating system (O/S) kernel, saves the program counter on the stack and saves the current instruction state in the CPU registers ($\textcircled{2}$). After this information is saved, the O/S locates the virtual address that caused the page fault. Once the O/S locates the virtual address, it checks to see if the address is valid and that access to this address is not protected. If the virtual address is valid, the O/S checks to determine if a page frame in the physical memory is free. If there are no free page frames, a page replacement algorithm is executed to remove page frames. Then, the page is scheduled for transfer to the hard drive and the page transfer is performed by a separate process. Once the page frame in the physical memory is cleaned, the O/S locates the disk address where the needed page is ($\textcircled{3}$). Then, the O/S schedules a disk operation to bring the missing page in ($\textcircled{4}$). When the page is loaded in the physical memory, a disk interrupt operation is triggered. The page table is updated with the location of the page and the frame is marked as being in a normal state ($\textcircled{5}$). The registers and other information is reloaded, the program counter is reset, and program execution continues ($\textcircled{6}$). This process incurs a high number of CPU cycles, which can cause a heavy penalty on the cluster system's performance.

Thus, our objective is to identify which resource use counters are "strongly correlated" to a major page fault, and which system messages are "strongly correlated" to a page fault event. By strongly correlated, we mean the resource use counters or system events which are assigned the largest positive regression coefficients by the regression model. In this paper, we aim to identify the resource use counters and system events for diagnosing major page faults. The research problem we address is presented as follows: Given (a) the resource usage data and system logs, (b) lists of resource use counters and lists of message types, and (c) the range of dates:

- Identify the resource use counters which are assigned the largest regression coefficients by the regression model.
- Identify the system events which are assigned the largest regression coefficients by the regression model.
- Identify the nodes which are associated with page fault events and compute node soft lockups.

Thus, the failure diagnostics workflow we propose consists of three phases, as depicted in Fig. 5. The workflow begins by extracting the raw logs and organizing them into data structures in the *Data preprocessing* phase. Once the data structures are extracted, the next phase of *Events correlation* identifies the resource use counters and message types which are strongly correlated to a given resource use counter and system event, respectively. This phase corresponds to "diagnosing" a node failure from the system logs. Then, the regression models are validated in the *Statistical validation* phase. Next, we describe the details of each phase of the diagnostics workflow.

4.1 Data Preprocessing

The goal of data preprocessing is to present the raw resource use data and message logs in a standardized format that is easily processed by data analysis algorithms [23]. In order to achieve this, we need to resolve three issues: (a) the format of a system log on multiple cluster systems are different, (b) the resource use data and system logs are monitored by different types of monitoring tools, and (c) the resource usage data and system logs are captured at different times.

A system log is generated if the print error function in the code is executed. In contrast, a resource use log is generated at fixed time intervals. As such, the time granularity in the system logs and resource use logs are different. The system logs and resource use logs contain different types of fields. Furthermore, the resource use data and system logs are monitored by different monitoring

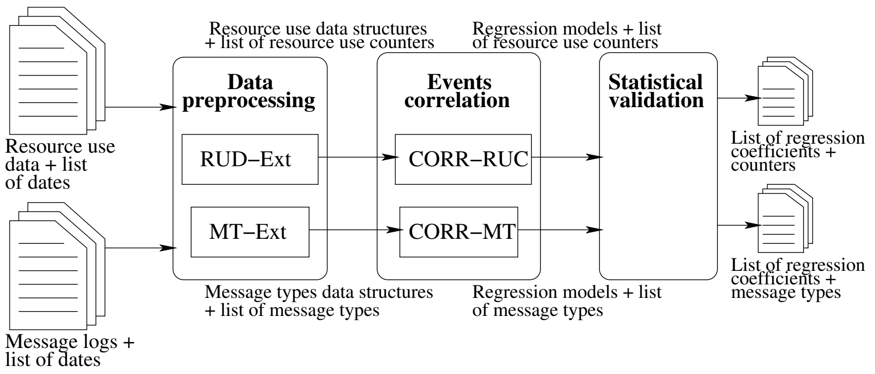
Resource use data + list of dates
Message logs + list of dates
Resource use data structures + list of resource use counters
Regression models + list of resource use counters
Data preprocessing
RUD–Ext
MT–Ext
Events correlation
CORR–RUC
CORR–MT
Statistical validation
Message types data structures + list of message types
Regression models + list of message types
List of regression coefficients + counters
List of regression coefficients + message types

Fig. 5 Our failure diagnostics workflow. It consists of three modules: (a) Data preprocessing, (b) Events correlation and (c) Statistical validation.

tools. Therefore, we preprocess the resource use data and system logs separately. Then, we apply the regression models on the preprocessed resource use data and system logs separately.

TACC_Stats [24] is an open-source resource usage monitor that is deployed on many large cluster systems [25]. TACC_Stats provides online monitoring of system resources. It records all values for all resource use counters at a fixed time interval of 10 minutes. The value for all counters are set to 0 only when a node is reset. An example of a resource use log is shown below:

```
312867 Jul 22 12:20:01 i201-307 vm pgmajfault 619298
pgfault 251833816
```

The resource use log contains eight fields. The first field represents the job-id (`312867`). The second and third fields represent the month (`Jul`) and (date `22`) respectively. The fourth field represents the time stamp (`12:20:01`). The fifth field represents the node-id (`i201-307`). The sixth field represents the virtual memory subsystem (`vm`). The seventh field represents the major page fault counter and its value (`pgmajfault 619298`). The eighth field represents the minor page fault counter and its value (`pgfault 251833816`). The resource use data contains 410 resource use counters which are divided into nine groups. Each group consists of a set of counters that capture resource usage on the network, Lustre filesystem, Lustre network, virtual memory, hard disk, CPU, main memory, Linux processes and NUMA (Non-Uniform Memory Architecture). One day's worth of resource use data contains an average of 19,452,401 resource use logs. Therefore, we generate a resource use data matrix to represent the counters and their values. Next, we describe the process to generate the resource use data matrix: (a) group the resource use logs based on their time stamp, (b) extract to remove all repeated counters and store the unique counter in a list of resource use counters, and (c) for each counter in the list of counters, obtain the value of the counter at the time stamp. We have implemented the process in the RUD-Ext module.

A typical system log contains a time stamp, a node number and an error message. Some system logs such as the Rationalized message logs contain the job number [20], but other system logs may not contain the job number. However, most system logs contain three basic fields. They are the time stamp, node number and error message. Thus, we define a standard format for the system logs and implemented a log-reformator to convert the system logs into the standard formatted system log. The standard formatted system logs contain the following fields:

```
job-id, month, day, hour:minute:second, node-id, program name,
error message
```

The log-entries contained in the system logs are generated by the O/S, filesystem and system software. An example of a system log-entry is shown below:

```
203884 Jul 22 07:30:54 i128-102 kernel do_page_fault 1247
```

The system log contains seven fields. The first field represents the job-id (`203884`). The second field represents the month (`Jul`). The third field represents the date (`22`). The fourth field represents the time stamp (`07:30:54`). The fifth field represents the node-id (`i128-102`). The sixth field represents the program name (`kernel`). The seventh field represents the error message (`do_page_fault 1247`). The error message is composed of a sequence of English words and numbers. In order to extract the *Constants* from the error message, we parse the error message and extract the English words only. One months' worth of system logs contain an average of 3,701,466 log-entries. As was done with the resource use logs, we generate a message types data matrix to represent the Constants and their counts. Next, we describe the process to generate the message types data matrix: (a) group the system logs into time-bins of 10 minutes, (b) extract to remove all repeated constants and store the unique constant termed a *message type* in a list of message types, and (c) for each message type in the list of message types, count the number within the time-bin. We have implemented the process in the MT-Ext module.

4.2 Events Correlation

After the resource use data matrix and message types data matrix are generated, we need to identify (a) the resource use counters which are assigned the largest positive regression coefficients, and (b) the message types which are assigned the largest positive regression coefficients. To achieve this, we use the LASSO, Ridge and Elastic Net regression algorithms to (a) obtain the regression models for multiple resource use counters to a target resource use counter, and (b) obtain the regression models for multiple message types to a target message type.

The multiple linear regression and polynomial regression models are standard techniques for modelling complex relationships with multiple variables [26]. However, they are known to be susceptible to overfitting on the training data and can produce a biased model. In contrast, the LASSO, Ridge and Elastic Net regression algorithms use a regularization function that constrains the coefficient estimates and avoids overfitting the model to the training data. The regularization function penalizes variables that have a large coefficient value and improves the accuracy of the model. A summary of the functions of the LASSO, Ridge and Elastic Net regression models is given in Table 2. In the LASSO regression model [27], a regularization function called $L1$-norm is used to set the coefficients of some of the variables to zero. In the Ridge regression model [28], a regularization function called $L2$-norm is used to shrink all the coefficients towards zero. In the Elastic Net regression model [29], both $L1$-norm and $L2$-norm are used to shrink the coefficients towards zero and set some of the coefficients to zero.

Table 2 Functions of the LASSO, Ridge and Elastic Net regression models

Algorithm	**Penalty Term**	**Function**
LASSO	$L1$-norm	Variable selection, regularization
Ridge	$L2$-norm	Variable selection, regularization
Elastic Net	$L1$-norm and $L2$-norm	Variable selection, regularization

4.2.1 Data scaling

In the LASSO, Ridge and Elastic Net regression algorithms, a complexity penalty is added to the regression coefficients in the cost function. The complexity penalty penalizes a regression model with large regression coefficients and prevents overfitting the model to the training data. The size of the regression coefficient associated with each variable depends on the magnitude of the value in the variable. As a result, the LASSO, Ridge and Elastic Net regression algorithms can produce biased models if the values in the input data are not standardized. For example, if the amount of minor page faults range from 2,000 to 4,000 and the amount of bytes written to the hard disk range from 27,000,000 to 58,000,000 bytes, the regression coefficient for minor page fault of 1 minor page fault change will be a much larger regression coefficient in regard to its change in minor page faults compared to a 1 byte change in the number of bytes written to the disk. If a larger minor page fault coefficient is obtained, the regularized regression algorithm will penalize that coefficient. To solve this problem, we standardize the resource use data matrix and message types data matrix so that all the values are centered around the mean with a unit standard deviation. We input the standardized resource use data matrix and message types data matrix into the LASSO, Ridge and Elastic Net algorithms. Then, we (a) train the LASSO, Ridge and Elastic Net regression algorithms on the resource use data matrix to obtain the fitted regression models, and (b) train the LASSO, Ridge and Elastic Net regression algorithms on the message types data matrix to obtain the fitted regression models.

4.3 Statistical Validation

After the LASSO, Ridge and Elastic Net regression models are trained, we need to assess the accuracy of the regression models. The *coefficient-of-determination* (R^2) is a standard metric for measuring how well the observed values are replicated by a regression model. In contrast to R^2, the Root Mean Squared Error (RMSE) is typically implemented by the cost function in the regression model. It measures the average difference between the predicted values by the regression model and the observed values. The RMSE value ranges from 0 to infinity. However, it is difficult to interpret a large RMSE value. Differently to the RMSE, the R^2 value ranges from 0 to 1. If $R^2 = 0$, it indicates that there is no linear relationship between the independent variables and dependent variable. If $R^2 = 1$, it indicates that there is a strong linear relationship between the independent variables and dependent variable. Thus, we obtain the R^2 values for the LASSO, Ridge and Elastic Net regression models.

Inflation in the R^2 value: The R^2 value is at least weakly increasing when the number of independent variables in the regression model increase [26]. As such, the R^2 value alone cannot be used to determine if the full regression model or the reduced regression model accurately describes the trend in the data. To address this issue, we apply a standard test called the F-Test [30]. The F-Test compares statistical models that are fitted to a dataset and identifies the model that best fits the data. First, we define two hypotheses as (a) the null hypothesis is the sum-of-squares error of the full model regression function ($SSE(F)$) is close to the sum-of-squares error of the reduced model regression function ($SSE(R)$), and (b) the alternate hypothesis is the $SSE(F)$ differs greatly to the $SSE(R)$. Then, we compute the general linear F-statistic [30], $F^* = (\frac{SSE(R)-SSE(F)}{df_R - df_F}) \div \frac{SSE(F)}{df_F}$, where df_R and df_F are the degrees of freedom associated with the reduced model and full model error sum-of-squares respectively. When $F^* \geq 3.95$, we reject the null hypothesis in favour of the alternate hypothesis.

5 Evaluation on Large Cluster Systems

We carry out studies on two representative large cluster systems. System A and System B are composed of 3,936 nodes and 1,888 nodes respectively. Both of these cluster systems provide batch job processing, computation resources for multiple scientific applications, large data storage and high-speed I/O. Many data centers also operate large cluster systems. Their cluster systems also provide user services such as batch job processing, compute resources for multiple applications, data storage and high-speed I/O.

Most cluster systems monitor the resource usage of jobs [24] and generate system logs [20]. To validate our diagnostics approach, we obtained 12 months' worth of resource usage data and system logs on Systems A and B. However, we do not know the dates when node failures occurred. Thus, we randomly select a range of dates on Systems A and B. Table 3 gives a summary of the log-data analyzed.

Table 3 Summary of log-data analyzed on Systems A and B.

System	**Days**	**Resource use data**		**System logs**	
		Size	Quantity	Size	Quantity
System A	26	122.9 GB	626,670,203	9.2 GB	64,569,329
System B	26	43.1 GB	201,285,323	1.04 GB	11,889,241

Node crashes are widely reported as one of the most common problems for the cluster system administrator [31]. A `soft lockup` log-entry is generated in the system logs when the Linux O/S hangs. To identify the dates of a node crash, we implemented a function to search the system logs for log-entries that contain the keywords `soft lockup`. Once a log-entry containing `soft lockup` is found, we extract the date in that log-entry. We identified 4 dates of node crashes on System A and 6 dates of node crashes on System B.

5.1 Phase 1: Identify the Resource Use Counters Correlated to Major Page Faults

To ascertain whether our approach can identify the resource use counters which are strongly correlated to major page faults, first we divide the resource use data into four blocks, where each block contains one week worth of resource use data. When a major page fault occurs, TACC_Stats [24] increases the `pgmajfault` counter. Thus, we set the `pgmajfault` counter as the dependent variable and set all the other counters as independent variables. Then, we trained the LASSO, Ridge and Elastic Net regression algorithms on all the resource use counters, applied 10-fold cross validation and obtained the fitted regression models.

5.1.1 Compare the LASSO, Ridge and Elastic Net regression models on System A

The R^2 values for the LASSO, Ridge and Elastic Net regression models on System A are given in Table 4. From Table 4, we observed that (a) the R^2 value for the LASSO regression model in weeks 1 to 4 is 0.99, (b) the R^2 value for the Ridge regression model in week 1 is 0.92 and the R^2 value for weeks 2 to 4 is 0.99, and (c) the R^2 value for the Elastic Net regression model in weeks 1 to 4 is 0.99.

On System A, in weeks 1 to 4 the LASSO and Elastic Net regression models obtained the highest R^2 value of 0.99, indicating that those regression models replicated the observed values with the highest accuracy. Moreover, both the LASSO and Elastic Net regression models obtained the same R^2 value. Thus, the LASSO regression model can be used as the primary model.

Table 4 R^2 values for the LASSO, Ridge and Elastic Net regression models on System A

	Week 1		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.92	0.99
	Week 2		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99
	Week 3		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99
	Week 4		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99

Next, we obtained the resource use counters that were assigned the largest positive regression coefficients by the LASSO model. A summary of the resource use counters and their regression coefficients for System A is given in Table 5. From Table 5, we observed that (a) on week 1 the `cpu 14 softirq`, `cpu 15 irq`, `share setattr`, `cpu 12 iowait` and `share dirty_pages_hits` counters are correlated to the `pgmajfault` counter with regression coefficients that range from 15 to 204264, (b) on week 2 the `work alloc_node` counter is correlated to the `pgmajfault` counter with a regression coefficient of 17918, (c) on week 3 the `cpu 12 softirq`, `work alloc_node`, `cpu 14 softirq`, `work seek` and `cpu 11 softirq` counters are correlated to the `pgmajfault` counter with regression coefficients that range from 436 to 544657, and (d) on week 4 the `work statfs`, `ps load_15` and `ps load_5` counters are correlated to the `pgmajfault` counter with regression coefficients that range from 63 to 69135. When a hardware signal or a software signal is sent to the CPU that temporally stops a running program, the `cpu irq` or `cpu softirq` counter is incremented, respectively. When a CPU is waiting on the system to complete an outstanding hard disk I/O task, the `cpu iowait` counter is incremented. When the attributes on the Lustre filesystem's share partition are set, the `share setattr` counter is incremented. When cached data in the main memory is written by the host but not yet committed to the hard disk, the `share dirty_pages_hits` counter is incremented. When a node on the work partition of the Lustre filesystem is allocated, the `work alloc_node` counter is incremented. When the Lustre filesystem returns the current read or write location in a file opened on the work partition, the `work seek` counter is incremented. When information about the mounted filesystem's work partition is returned, the `work statfs` counter is incremented. When a request for the 5 minutes and 15 minutes average load on the O/S is sent, the `ps load_5` and `ps load_15` counters are incremented, respectively.

Table 5 Regression coefficients for the resource use counters on System A

			Week 1		
Counter	cpu 14 softirq	cpu 15 irq	share setattr	cpu 12 iowait	share dirty_pages_hits
Coeff	204264	7435	1321	15	15
			Week 2		
Counter	work alloc_node			–	
Coeff	17918			–	
			Week 3		
Counter	cpu 12 softirq	work alloc_node	cpu 14 softirq	work seek	cpu 11 softirq
Coeff	544657	76606	26168	1342	436
			Week 4		
Counter	work statfs	ps load_15	ps load_5	–	
Coeff	69135	635	63	–	

On System A, multiple CPU and Lustre filesystem resource use counters with large regression coefficients were identified in weeks 1, 3 and 4, and one Lustre filesystem resource use counter with a large regression coefficient was identified in week 2, indicating that those counters are strongly correlated to major page faults.

Next, we determine if the resource use counters identified in Table 5 can be used to identify a major page fault. We used the counters in Table 5 to train the LASSO regression algorithm, applied 10-fold cross validation and obtained the reduced regression model. Then, we performed the F-Tests on the full and reduced regression models and obtained the F^* values. A summary of the F-Tests for the LASSO regression model on System A is given in Table 6. From Table 6, we observed that for weeks 1 to 4 the F^* value range from 0 to 0.022. Since $F^* < 3.95$, we fail to reject the null hypotheses.

Next, we obtain the residuals in the reduced LASSO regression model. A residual contains the difference between the value estimated by the regression model and the observed value in the data. The proportion of residuals in the reduced LASSO regression model for weeks 1, 2, 3 and 4 is shown in Fig. 6, Fig. 7, Fig. 8 and Fig. 9. From Fig. 6, we observed that the residual value ranges from 0.0001 to 0.0025. From Fig. 7, we observed that the residual value ranges from 0.0005 to 0.002. From Fig. 8, we observed that the residual value ranges from 0.00002 to 0.0001. From Fig. 9, we observed that the residual value ranges from 0.00001 to 0.00012. When the residual value is close to 0, it shows that the value estimated by the regression model is close to the observed value in the data.

Table 6 F-tests for the LASSO regression model on System A

	SSE	*Degrees of freedom*	F^*	R^2
		Week 1		
Full model	0.00035	1	**0**	0.99
Reduced model	0.00035	404	–	0.99
		Week 2		
Full model	0.00135	1	**0**	0.99
Reduced model	0.00035	408	–	0.99
		Week 3		
Full model	0.000000078	1	**0.00038**	0.99
Reduced model	0.00035	404	–	0.99
		Week 4		
Full model	0.00000011	1	**0.022**	0.99
Reduced model	0.00035	406	–	0.99

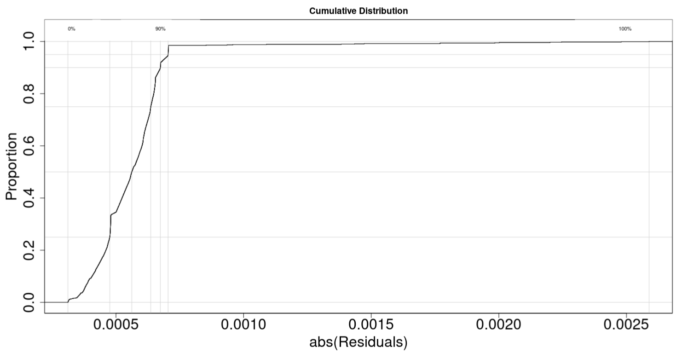

Fig. 6 Proportion of residuals in the reduced LASSO regression model on week 1.

On System A, in weeks 1 to 4 the F^* value range from 0 to 0.022, indicating that there is no difference between the full and reduced regression models. The residual value in the reduced LASSO regression model ranges from 0.00001 to 0.002, indicating that the estimated value is close to the observed value. Furthermore, the R^2 value for the full and reduced regression models is 0.99. When the F^* value is less than 3.95, the R^2 values for the full and reduced regression models are the same, and the residual values in the reduced LASSO regression model is close to 0, the reduced LASSO regression model can be used as the primary model for identifying a major page fault.

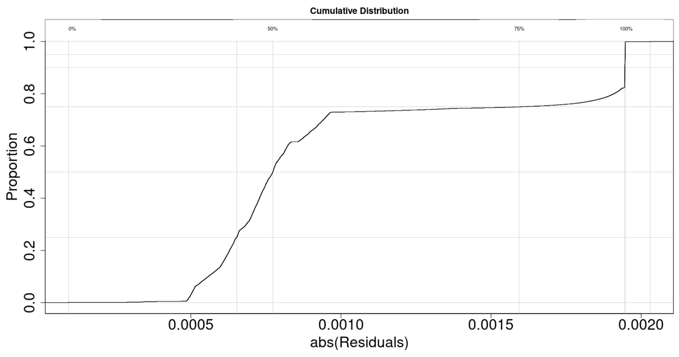

Fig. 7 Proportion of residuals in the reduced LASSO regression model on week 2.

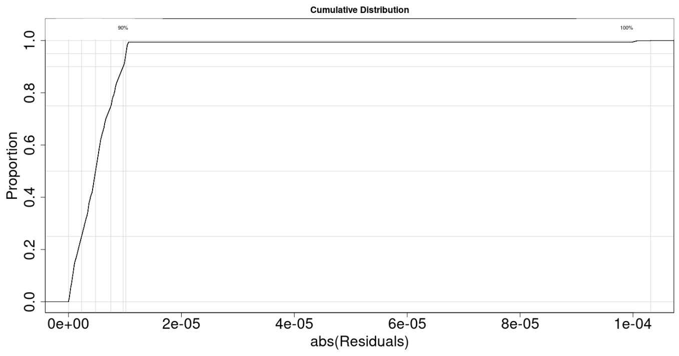

Fig. 8 Proportion of residuals in the reduced LASSO regression model on week 3.

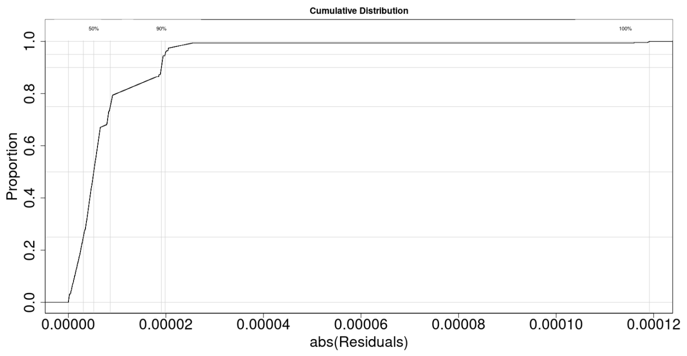

Fig. 9 Proportion of residuals in the reduced LASSO regression model on week 4.

5.1.2 Compare the LASSO, Ridge and Elastic Net regression models on System B

As was done with System A, we obtained the R^2 values for the LASSO, Ridge and Elastic Net regression models on System B. The R^2 values for System B are given in Table 7. From Table 7, we observed that (a) the R^2 values for the

LASSO regression model in weeks 1, 3 and 4 is 0.99 and its R^2 value in week 2 is 0.84, (b) the R^2 values for the Ridge regression model in weeks 1 to 4 is 0.99, and (c) the R^2 values for the Elastic Net regression model in weeks 1, 3 and 4 is 0.99 and its R^2 value in week 2 is 0.98.

Table 7 R^2 values for the LASSO, Ridge and Elastic Net regression models on System B

	Week 1		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99
	Week 2		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.84	0.99	0.98
	Week 3		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99
	Week 4		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99

On System B, in weeks 1 to 4 the Ridge regression model obtained the highest R^2 value of 0.99, indicating that the Ridge regression model replicated the observed values with the highest accuracy.

Next, we obtain the resource use counters that were assigned large positive regression coefficients by the Ridge regression model. A summary of the resource use counters and their regression coefficients for System B is given in Table 8. From Table 8, we observed that in weeks 1 to 4 the regression coefficients for `cpu 2 irq`, `cpu 3 irq`, `cpu 4 irq`, `cpu 5 irq` and `cpu 14 irq` range from 3.1E+11 to 5.2E+11.

On System B, in weeks 1, 2, 3 and 4, multiple CPU counters with large regression coefficients were identified, indicating that those counters are strongly correlated to major page faults.

Next, we determine if the resource use counters identified in Table 8 can be used to identify a major page fault. We used the counters in Table 8 to train the Ridge regression algorithm, applied 10-fold cross validation and obtained the reduced regression model. Then, we performed the F-Tests on the full and reduced regression models and obtained the F^* values. A summary of the F-Tests for the Ridge regression model on System B is given in Table 9. From

Table 8 Regression coefficients of resource use counters on System B

			Week 1		
Counter	cpu 2 irq	cpu 3 irq	cpu 4 irq	cpu 5 irq	cpu 14 irq
Coeff	5.2E+11	4E+11	3.9E+11	3.9E+11	3.8E+11
			Week 2		
Counter	cpu 2 irq	cpu 3 irq	cpu 4 irq	cpu 5 irq	cpu 14 irq
Coeff	3.2E+11	3.2E+11	3.2E+11	3.2E+11	3.1E+11
			Week 3		
Counter	cpu 2 irq	cpu 3 irq	cpu 4 irq	cpu 5 irq	cpu 14 irq
Coeff	3.5E+11	3.5E+11	3.5E+11	3.5E+11	3.4E+11
			Week 4		
Counter	cpu 2 irq	cpu 3 irq	cpu 4 irq	cpu 5 irq	cpu 14 irq
Coeff	3.4E+11	3.4E+11	3.4E+11	3.4E+11	3.4E+11

Table 9, we observed that the F^* value for weeks 1 to 4 range from 0 to 0.0165. Since $F^* < 3.95$ for weeks 1 to 4, we fail to reject the null hypotheses.

Table 9 F-tests for the Ridge models on System B

		Week 1		
	SSE	*Degrees of freedom*	F^*	R^2
Full model	0.00017	1	**0.00014**	0.99
Reduced model	0.00018	404	–	0.99
		Week 2		
	SSE	*Degrees of freedom*	F^*	R^2
Full model	0.0000066	1	**0.00037**	0.99
Reduced model	0.0000076	404	–	0.99
		Week 3		
	SSE	*Degrees of freedom*	F^*	R^2
Full model	0.000012	1	**0.0165**	0.99
Reduced model	0.000092	404	–	0.99
		Week 4		
	SSE	*Degrees of freedom*	F^*	R^2
Full model	0.000079	1	**0.00015**	0.99
Reduced model	0.000084	404	–	0.99

Next, we obtain the residuals in the reduced Ridge regression model. The proportion of residuals for weeks 1, 2, 3 and 4 is shown in Fig. 10, Fig. 11, Fig. 12 and Fig. 13. From Fig. 10, we observed that the residual value ranges from 0.0002 to 0.0008. From Fig. 11, we observed that the residual value ranges from 0.0002 to 0.003. From Fig. 12, we observed that the residual value ranges from 0.0001 to 0.002. From Fig. 13, we observed that the residual value ranges

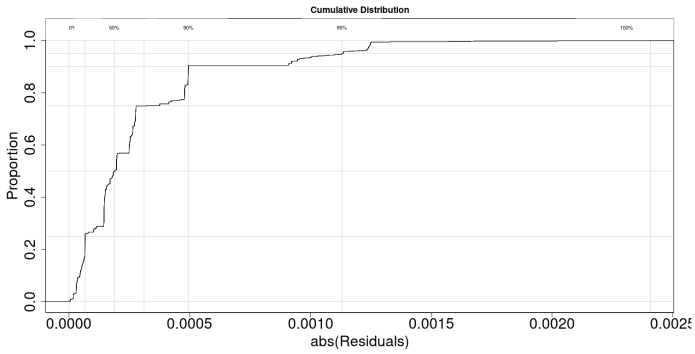

Fig. 10 Proportion of residuals in the reduced Ridge regression model on week 1.

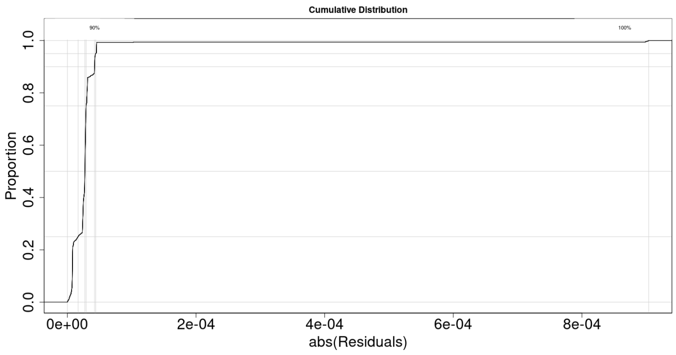

Fig. 11 Proportion of residuals in the reduced Ridge regression model on week 2.

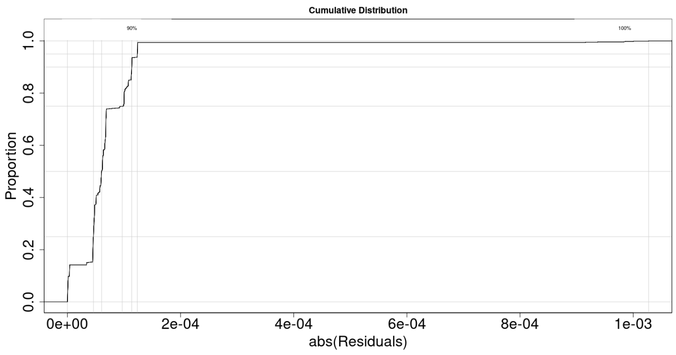

Fig. 12 Proportion of residuals in the reduced Ridge regression model on week 3.

from 0.00001 to 0.00012. When the residual value is close to 0, it shows that the value estimated by the regression model is close to the observed value in the data.

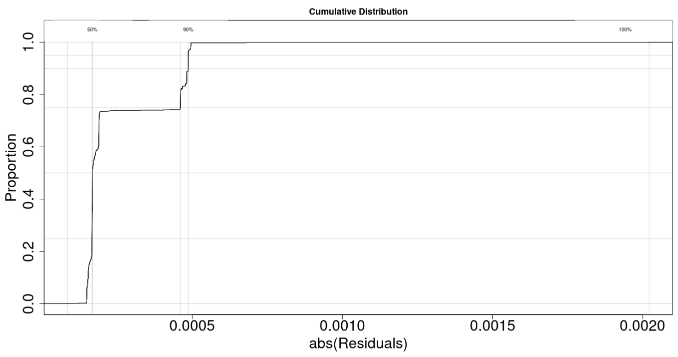

Fig. 13 Proportion of residuals in the reduced Ridge regression model on week 4.

On System B, in weeks 1 to 4 the F^* values range from 0 to 0.0165, indicating that there is no difference between the full and reduced regression models. The residual value in the reduced Ridge regression model ranges from 0.0001 to 0.002, indicating that the estimated value is close to the observed value. Furthermore, the R^2 value for the full and reduced regression models is 0.99. When the F^* value is less than 3.95, the R^2 values for the full and reduced regression models are the same, and the residual values in the reduced Ridge regression model is close to 0, the reduced Ridge regression model can be used as the primary model for identifying a major page fault.

5.1.3 Summary of Findings

On System A, from weeks 1 to 4, an R^2 value of 0.99 was obtained for the reduced LASSO regression model and the residual values range from 0.00001 to 0.002. This result shows that the estimated value from the reduced LASSO regression model is close to the observed value in the data. Thus, the reduced LASSO regression model can be used to identify a major page fault.

On System B, from weeks 1 to 4, an R^2 value of 0.99 was obtained for the reduced Ridge regression model and the residual values range from 0.0001 to 0.002. This result shows that the estimated value from the reduced Ridge regression model is close to the observed value in the data. Thus, the reduced Ridge regression model can be used to identify a major page fault.

From our evaluation of the LASSO, Ridge and Elastic Net regression models, we showed that (a) the reduced LASSO regression model identified major page faults with the highest accuracy on System A, and (b) the reduced Ridge regression model identified major page faults with the highest accuracy on System B. Thus, two different regression models are required for identifying major page faults on two different cluster systems.

5.2 Phase 2: Identify the System Events Correlated to Page Fault Events

The first phase of our analysis is characterized by the identification of the counters that are strongly correlated to major page faults. On System A, we observed that (a) the LASSO regression model replicated the observed values in the resource use data with the highest accuracy, and (b) the CPU and Lustre filesystem counters that were assigned large positive regression coefficients can be used to train the regression model to identify a major page fault. On System B, we observed that (a) the Ridge regression model replicated the observed values in the resource use data with the highest accuracy, and (b) the CPU interrupt counters that were assigned large positive regression coefficients can be used to train the regression model to identify a major page fault. Our next objective is to ascertain whether our approach can identify the system events which are strongly correlated to a page fault event. To attain this, we determine the regression coefficients for multiple system events.

Therefore, we obtain the dates of the page fault events in the system logs. We implemented a function in our diagnostics workflow to search the list of message types containing the keywords "page fault". We scanned the lists of message types for all the 26 days on Systems A and B. On System A, we identified one message type that contained the keywords `do_page_fault` for 4 days. On System B, there is no "page fault" message in the list of message types for all the 26 days. Thus, we focus on the system logs on System A. We set the `do_page_fault` message type as the dependent variable and set all the other message types as independent variables. We trained the LASSO, Ridge and Elastic Net regression algorithms on all the message types, applied 10-fold cross validation and obtained the fitted regression models.

The R^2 values for the LASSO, Ridge and Elastic Net regression models on System A are given in Table 10. From Table 10, we observed that (a) the R^2 value for the LASSO regression model in days 1, 3 and 4 is 0.99 and the R^2 value in day 2 is 0.98, (b) the R^2 value for the Ridge regression model in days 2 and 4 is 0.99, the R^2 value in day 1 is 0.97 and the R^2 in day 3 is 0.92, and (b) the R^2 value for the Elastic Net regression model in days 1 and 4 is 0.99, the R^2 value in day 2 is 0.98 and the R^2 value in day 3 is 0.86.

On System A, while the LASSO regression model obtained the highest R^2 value of 0.99 in days 1, 3 and 4, the Ridge regression model obtained the highest R^2 value of 0.99 in day 2, indicating that the LASSO and Ridge regression models replicated the observed values with the highest accuracy on different dates.

Next, we obtained the message types that were assigned the largest positive regression coefficients by (a) the LASSO regression model in days 1, 3 and 4, and (b) the Ridge regression model in day 2. A summary of the message types and their regression coefficients is given in Table 11.

Table 10 R^2 values for the LASSO, Ridge and Elastic Net regression models on System A

	Day 1		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.97	0.99
	Day 2		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.98	0.99	0.98
	Day 3		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.92	0.86
	Day 4		
	LASSO	**Ridge**	**Elastic Net**
R^2 **value**	0.99	0.99	0.99

From Table 11, we observed that (a) on day 1 the `lustre ll_nopage`, `filemap_nopage`, `cancel RPC` and `request timed out` message types are correlated to the `do_page_fault` message type with regression coefficients that range from 0.003 to 0.99, (b) on day 2 the `recalc_sigpending`, `segfault sus`, `sock_aio_write` and `segfault pgm` message types are correlated to the `do_page_fault` message type with a regression coefficient of 0.007, (c) on day 3 the `alloc_pages` message type is correlated to the `do_page_fault` message type with a regression coefficient of 0.09, and (d) on day 4 the `task blocked`, `down_read`, `import_delay` and `find busiest_group` message types are correlated to the `do_page_fault` message type with regression coefficients that range from 1.2 to 11. When there is no contiguous block of memory in the memory cache or the buffer cache, a `lustre ll_nopage` and a `filemap_nopage` message are generated. When a request sent from a client to the Lustre filesystem server takes longer to reach the server than it is prepared to wait, a `cancel RPC` and a `request timed out` message are generated. When the O/S returns a set of signals that were delivered to a calling thread, a `recalc_sigpending` message is generated. When a message is sent on a network socket, a `sock_aio_write` message is generated. When a program attempts to write or read an invalid memory location, a `segfault` message is generated. When the O/S attempted to allocate a block or more pages in the main memory, an `alloc_pages` message is generated. When a task is blocked by the O/S due to high server I/O load, a `task blocked` message is generated. The task blocked message contained the word `bash`, which represents the Linux bash shell. When a process acquires read-only access to a new mapping in the virtual address space of the calling process, a `down_read` message is generated. When a message sent between two processes is delayed, a `import_delay` message is generated. When an instruction to identify a set of busy CPUs is sent to the job scheduler, a `find busiest_group` message is generated.

Table 11 Regression coefficients for the message types on System A

		Day 1		
Message	lustre ll_nopage	filemap_nopage	cancel RPC	request timed out
Coeff	0.99	0.003	0.003	0.003
		Day 2		
Message	recalc_ sigpending	segfault sus	sock aio _write	segfault pgm
Coeff	0.007	0.007	0.007	0.007
		Day 3		
Message	alloc_pages		–	
Coeff	0.09		–	
		Day 4		
Message	task blocked bash	down_read	import_delay	find busiest_group
Coeff	11	1.5	1.2	1.2

On System A, in days 1 to 4, multiple system events with regression coefficients ranging from 0.003 to 11 were identified, indicating that those system events are correlated to the page fault event.

Next, we determine if the message types identified in Table 11 can be used to identify a page fault event. We (a) used the message types on days 1, 3 and 4 to train the LASSO regression algorithm, and (b) used the message types on day 2 to train the Ridge regression algorithm. We applied 10-fold cross validation and obtained the reduced LASSO and Ridge regression models. Then, we performed the F-Tests on the full and reduced regression models and obtained the F^* values. A summary of the F-Tests for the LASSO and Ridge regression models are given in Tables 12 and 13, respectively. From Table 12, we observed that the F^* value for days 1, 3 and 4 range from 0.00017 to 0.43. Since $F^* < 3.95$ for days 1, 3 and 4, we fail to reject the null hypothesis. Furthermore, we observed that (a) on days 1 and 3 the R^2 value for the full and reduced regression models is 0.99, and (b) on day 4 the R^2 value for the full and reduced regression models is 0.99 and 0.24, respectively. From Table 13, we observed that the F^* value for day 2 is 0.0022. Since $F^* < 3.95$, we fail to reject the null hypothesis. Furthermore, we observed that the R^2 value for the full and reduced regression models is 0.99.

Next, we obtain the residuals in the reduced regression model. The proportion of residuals for days 1, 2 and 3 is shown in Fig. 14, Fig. 15 and Fig. 16. From Fig. 14, we observed that the residual value ranges from 0.0001 to 0.003. From Fig. 15, we observed that the residual value ranges from 0.001 to 0.01. From Fig. 16, we observed that the residual value ranges from 0.001 to 0.008. When the residual value is close to 0, it shows that the value estimated by the regression model is close to the observed value in the data.

Table 12 F-tests for the LASSO regression model on System A

	SSE	*Degrees of freedom*	F^*	R^2
		Day 1		
Full model	0.00003	1	**0.00017**	0.99
Reduced model	0.000035	955	–	0.99
		Day 3		
Full model	0.0114	1	**0.0012**	0.99
Reduced model	0.0147	233	–	0.99
		Day 4		
Full model	0.0027	1	**0.43**	0.99
Reduced model	0.39	319	–	0.24

Table 13 F-test for the Ridge regression model on System A

	SSE	*Degrees of freedom*	F^*	R^2
		Day 2		
Full model	0.003	1	**0.0022**	0.99
Reduced model	0.005	293	–	0.99

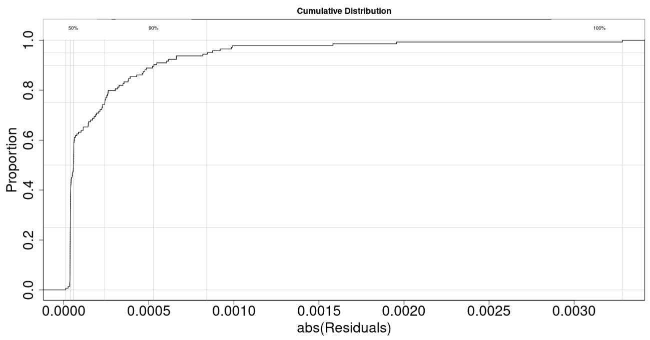

Fig. 14 Proportion of residuals in the reduced LASSO regression model on day 1.

From days 1 to 3, the F^* value range from 0.00017 to 0.0022, indicating that there is no difference between the full and reduced regression models. The residual value in the reduced regression model ranges from 0.0001 to 0.001, indicating that the estimated value is close to the observed value. Furthermore, the R^2 value for the full and reduced regression models is 0.99. When the F^* value is less than 3.95, the R^2 values for the full and reduced regression models are the same, and the residual values in the reduced regression model is close to 0, the reduced regression model can be used as the primary model for identifying a page fault event.

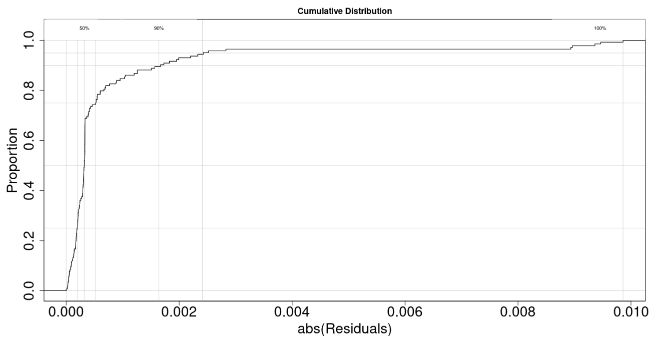

Fig. 15 Proportion of residuals in the reduced Ridge regression model on day 2.

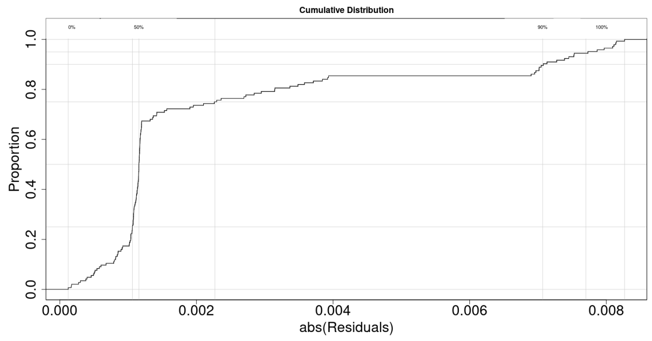

Fig. 16 Proportion of residuals in the reduced LASSO regression model on day 3.

Next, we obtained the residual values for the full and reduced LASSO regression model for day 4. The proportion of residual values for the full and reduced LASSO regression model is shown in Fig. 17 and Fig. 18, respectively. From Fig. 17, we observed that the residual value ranges from 0.001 to 0.03. From Fig. 18, we observed that (a) the residual value ranges from 0.01 to 0.25, and (b) a large proportion of residual values range from 0.1 to 0.25. When the residual value is not close to 0 and there is a large proportion of those residual values, it shows that the value estimated by the regression model is not close to the observed value in the data.

On day 4, the R^2 value for the reduced Ridge regression model is 0.24 and a large proportion of the residual values range from 0.1 to 0.25, indicating that the value estimated by the reduced regression model is not close to the observed value. However, the R^2 value for the full Ridge regression model is 0.99 and the residual values range from 0.001 to 0.03, indicating that the value estimated by the full Ridge regression model is close to the observed value. Thus, the full regression model can be used as the primary model for identifying a page fault event.

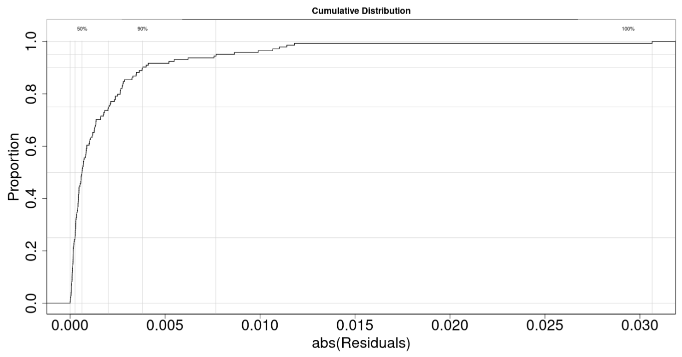

Fig. 17 Proportion of residuals in the full LASSO regression model on day 4.

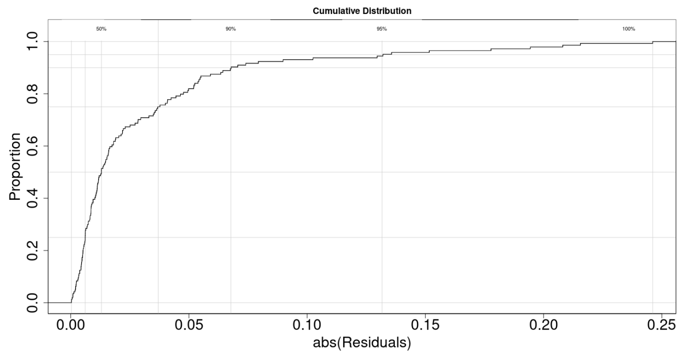

Fig. 18 Proportion of residuals in the reduced LASSO regression model on day 4.

5.2.1 Summary of Findings

On System A, from days 1 to 3, an R^2 value of 0.99 was obtained from the reduced regression model and the residual values range from 0.0001 to 0.001. This result shows that the value estimated by the reduced regression model is close to the observed value in the data. Thus, the reduced regression model can be used to identify a page fault event. On day 4, an R^2 value of 0.99 was obtained from the full LASSO regression model and the residual values range from 0.001 to 0.03, but the R^2 value of 0.24 was obtained from the reduced LASSO regression model and a large proportion of the residual values range from 0.1 to 0.25. This result shows that the value estimated from the full LASSO regression model is close to the observed value in the data.

From our evaluation of the LASSO, Ridge and Elastic Net regression models, we showed that (a) the reduced Ridge regression model identified the page fault events with the highest accuracy on day 2, (b) the reduced LASSO regression model identified the page fault events with the highest accuracy on days 1 and 3, and (c) the full LASSO regression model identified the page fault events with the highest accuracy on day 4. Thus, additional system events can be used to train an accurate LASSO regression model to identify the page fault events.

5.3 Phase 3: Identify the Page Fault Events Correlated to Soft Lockup Events

The second phase of our analysis is characterized by the identification of the system events that are strongly correlated to the page fault event. On System A, we observed that (a) the LASSO and Ridge regression models replicated the observed values with the highest accuracy on different days, and (b) multiple system events can be used to train the regression models to identify a page fault event. Our next objective is to ascertain whether our approach can identify the nodes which are associated with the page fault and soft lockup events. To achieve this, we (a) correlate the `do_page_fault` events to `BUG: soft lockup` events, and (b) identify the nodes which are associated with the `do_page_fault` and `BUG: soft lockup` events.

First, we obtain the Pearson correlation coefficient between the `do_page_fault` and `BUG: soft lockup` events on days 1 to 4. We implemented two functions in our diagnostics workflow: (a) a function that searches for log-entries containing the `do_page_fault` or `BUG: soft lockup` keywords, and (b) a function that calculates the Pearson correlation coefficient between the `do_page_fault` and `BUG: soft lockup` events by time-bins. We obtained the Pearson correlation coefficient r using the formula given as follows [26]: $r = \Sigma_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}) \div \sqrt{\Sigma_{i=1}^{n}(x_i - \bar{x})^2 \Sigma_{i=1}^{n}(y_i - \bar{y})^2}$ where x_i is the value of the `do_page_fault` event at time-bin i, $\bar{x}$ is the mean of the values of the `do_page_fault` event, y_i is the value of the `BUG: soft lockup` event at time-bin i, $\bar{y}$ is the mean of the values of the `BUG: soft lockup` event, and n is the number of time-bins. The Pearson correlation coefficient ranges from -1 to 1, where -1 represents perfect negative correlation, 0 represents no correlation and 1 represents perfect positive correlation. We are interested in page fault events which are strongly positive correlated to soft lockup events. To interpret the strength of the correlation coefficient, we use the following rules of thumb [26]: (a) strong positive correlation 0.8 to 0.99, (b) moderate positive correlation 0.31 to 0.79, and (c) weak positive correlation 0.1 to 0.3. For days 1 to 3, we obtained the Pearson correlation coefficient of 0.22, 0.47 and 0.62. This result shows that the `do_page_fault` and `BUG: soft lockup` events were (a) weakly positive correlated on day 1, and (b) moderately positive correlated on days 2 and 3. For day 4, we obtained the Pearson correlation coefficient of 0.92. This result shows that the `do_page_fault` and `BUG: soft lockup` events were strongly positive correlated.

5.3.1 Identify the nodes associated with page fault and soft lockup events

Next, we obtain the node numbers in the system logs that contain those `do_page_fault` and `BUG: soft lockup` messages on day 4. We implemented a function in our diagnostics workflow to: (a) extract the node number in the page fault and the soft lockup log-entries, and (b) match the node number in the page fault log-entry to the soft lockup log-entry. We identified five compute nodes that contained both page fault and soft lockup messages. The node numbers are given in Table 14.

Table 14 List of compute nodes on System A on Day 4

	Log-entry	
	`do_page_fault`	`soft lockup`
Node number	i114-206, i117-201, i123-112, i136-407, i178-105	i114-206, i117-201, i123-112, i136-407, i178-105

5.3.2 Detailed diagnosis

On System A, in week 4, requests for information about the mounted filesystem and the average load on the Linux O/S were made. These events were correlated to the `do_page_fault` events that occurred during the week. For days 1 to 3, the `do_page_fault` events were not strongly correlated to soft lockup events. This shows that page fault events did not lead to compute node soft lockups on days 1 to 3. On day 4, the Linux O/S blocked the bash shell from executing due to high server I/O load. A process attempted to acquire read-only access to a virtual memory address space. A message sent by a process to another process was delayed, and the job scheduler was instructed to identify the busiest CPUs. These system events were correlated to the `do_page_fault` event, and the `do_page_fault` event was strongly positive correlated to soft lockup events. Thus, this result shows that page fault events led to compute node soft lockups on day 4.

6 Threats to Validity

We have identified the following threats to validity (a) internal validity threat, and (b) external validity threat.

An internal validity threat is concerned with the factors which might influence the results presented in this paper. Those factors include (a) the selection of the dates of the resource use data and system logs, and (b) the validation of the regression models. Regarding the selection of the dates of the system log-data, we may have missed dates of page fault events and compute node soft lockups. To resolve this issue, we implemented search functions in our diagnostics workflow and searched the system logs for all the log-entries that contain the keywords “page fault” or “soft lockup”. Regarding the validation of the regression models, our analysis is based on the standard statistical validation approach. We did not consider other regression algorithms because they are beyond the scope of this paper, nor analyze the source code with the system logs as it would require a substantial amount of resources out of the reach of this paper.

An external validity threat is concerned with applying the conclusions of this study outside its context. Our results are based on two large cluster systems and may not apply to all large cluster systems. However, the cluster systems we used to evaluate our diagnostics workflow are representative for many large cluster systems. Regardless of the generalizability of our conclusions, we showed that (a) multiple CPU and Lustre filesystem counters and multiple system events are correlated to major page faults, and (b) page fault events and compute node soft lockups occurred on multiple nodes.

7 Conclusion and Future Work

An approach based on regression techniques is developed to diagnose major page faults in large cluster systems. We showed that multiple CPU and Lustre filesystem resource use counters are strongly correlated to major page faults, multiple system events are correlated to page fault events, and identified multiple compute nodes that crashed due to a page fault. We applied the coefficient-of-determination and F-tests and ensured accurate diagnosis of major page faults and page fault events.

From this study, we learned that (a) the reduced LASSO and Ridge regression models identified major page faults with the highest accuracy on System A and System B, respectively, (b) the reduced Ridge regression model identified the page fault events with the highest accuracy on day 2 on System A, (c) the reduced LASSO regression model identified the page fault events with the highest accuracy on days 1 and 3 on System A, and (d) the full LASSO regression model identified the page fault events with the highest accuracy on day 4 on System A. As such, while multiple resource use counters and multiple system events can enhance the accuracy of the regression model in diagnosing major page faults and page fault events, incorporating additional system events in the regression algorithm can further enhance its accuracy in diagnosing page fault events. In our future work, we plan to apply our approach to diagnose other types of faults other than page faults in large cluster systems.

Acknowledgements

We would like to thank the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing the resource use data and system logs from their HPC systems. We would also like to thank the anonymous reviewers for their constructive feedback, which helped improve our paper significantly.

Declarations

Ethical Approval

Not applicable

Competing interests

All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Authors' contributions

Edward Chuah prepared the manuscript and conducted the experiments. Arshad Jhumka and Sai Narasimhamurthy reviewed and edited the manuscript.

Funding

No funding was received to assist with the preparation of this manuscript.

Availability of data and materials

The datasets analyzed during this study are available from the corresponding author on request.

References

[1] Oliner, A.J., Kulkarni, A.V., Aiken, A.: Using correlated surprise to infer shared influence. In: Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2010). https://doi.org/10.1109/DSN.2010.5544921

[2] Zheng, Z., Yu, L., Lan, Z., Jones, T.: 3-dimensional root cause diagnosis via co-analysis. In: Proceedings of ACM International Conference on Autonomic Computing (ICAC) (2012). https://doi.org/10.1145/2371536.2371571

[3] Chuah, E., Jhumka, A., Alt, S., Evans, R.T., Suri, N.: Failure diagnosis for cluster systems using partial correlations. In: Proceedings of IEEE International Symposium on Parallel & Distributed Processing with Applications (ISPA) (2021). https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00151

[4] Snir, M., Wisniewski, R.W., Abraham, J.A., Adve, S.V., Bagchi, S., Balaji, P., Belak, J., Bose, P., Cappello, F., Carlson, B., Chien, A.A., Coteus, P., Debardeleben, N.A., Diniz, P.C., Engelmann, C., Erez, M.,

Fazzari, S., Geist, A., Gupta, R., Johnson, F., Krishnamoorthy, S., Leyffer, S., Liberty, D., Mitra, S., Munson, T., Schreiber, R., Stearley, J., Hensbergen, E.V.: Addressing failures in exascale computing. International Journal of High Performance Computing Applications **28**(2) (2014). https://doi.org/10.1177/1094342014522573

[5] Martino, C.D., Baccanico, F., Fullop, J., Kramer, W., Kalbaczyk, Z., Iyer, R.: Lessons learned from the analysis of system failures at petascale: The case of blue waters. In: Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), p. 2014. https://doi.org/10.1109/DSN.2014.62

[6] Mitra, S., Javagal, S., Maji, A.K., Gamblin, T., Moody, A., Harrell, S., Bagchi, S.: A study of failures in community clusters: The case of conte. In: Proceedings of the 2016 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), pp. 189–196 (2016). https://doi.org/10.1109/ISSREW.2016.7

[7] Gupta, S., Patel, T., Engelmann, C., Tiwari, D.: 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 (SC) (2017). https://doi.org/10.1145/3126908.3126937

[8] Rojas, E., Meneses, E., Jones, T., Maxwell, D.: Analyzing a five-year failure record of a leadership-class supercomputer. In: Proceedings of the 31st International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 196–203 (2019). https://doi.org/10.1109/SBAC-PAD.2019.00040. IEEE

[9] Kumar, R., Jha, S., Mahgoub, A., Kalyanam, R., Harrell, S., Song, X.C., Kalbarczyk, Z., Kramer, W., Iyer, R., Bagchi, S.: The mystery of the failing jobs: Insights from operational data from two university-wide computing systems. In: Proceedings of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2020). https://doi.org/10.1109/DSN48063.2020.00034

[10] Liu, Z., Lewis, R., Kettimuthu, R., Harms, K., Carns, P., Rao, N., Foster, I., Papka, M.E.: Characterization and identification of HPC applications at leadership computing facility. In: Proceedings of the 34th ACM International Conference on Supercomputing (ICS). Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3392717.3392774

[11] Rojas, E., Meneses, E., Jones, T., Maxwell, D.: Understanding failures through the lifetime of a top-level supercomputer. Journal of Parallel and Distributed Computing **154**, 27–41 (2021). https://doi.org/10.1016/

j.jpdc.2021.04.001

[12] Ferreira, K.B., Levy, S., Hemmert, J., Pedretti, K.: Understanding memory failures on a petascale Arm system. In: Proceedings of the 31st ACM International Symposium on High-Performance Parallel and Distributed Computing (HPDC), pp. 84–96. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3502181.3531465

[13] Abraham, J.P., Mathew, S.: A novel approach to improve the processor performance with page replacement method. Procedia Computer Science **46** (2015). https://doi.org/10.1016/j.procs.2015.02.054

[14] Tirumalasetty, C., Chou, C.C., Reddy, N., Gratz, P., Abouelwafa, A.: Reducing minor page fault overheads through enhanced page walker. ACM Transactions on Architecture and Code Optimization **19**(4) (2022). https://doi.org/10.1145/3547142

[15] Psistakis, A., Chrysos, N., Chaix, F., Asiminakis, M., Gianioudis, M., Xirouchakis, P., Papaefstathiou, V., Katevenis, M.: Optimized page fault handling during RDMA. IEEE Transactions on Parallel and Distributed Systems **33**(12), 3990–4005 (2022). https://doi.org/10.1109/TPDS.2022.3175666

[16] Chuah, E., Jhumka, A., Narasimharmuthy, S., Hammond, J., Browne, J.C., Barth, B.: Linking resource usage anomalies with system failures from cluster log data. In: Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS) (2013). https://doi.org/10.1109/SRDS.2013.20

[17] Chuah, E., Jhumka, A., Browne, J.C., Gurumdimma, N., Narasimharmuthy, S., Barth, B.: Using message logs and resource use data for cluster failure diagnosis. In: Proceedings of IEEE International Conference on High Performance Computing (HiPC) (2016). https://doi.org/10.1109/HiPC.2016.035

[18] Fu, X., Ren, R., Zhan, J., Zhou, W., Jia, Z., Lu, G.: Logmaster: Mining event correlations in logs of large-scale cluster systems. In: Proceedings of IEEE International Symposium on Reliable Distributed Systems (SRDS), pp. 71–80 (2012). https://doi.org/10.1109/SRDS.2012.40

[19] Fu, X., Ren, R., McKee, S.A., Zhan, J., Sun, N.: Digging deeper into cluster system logs for failure prediction and root cause diagnosis. In: Proceedings of IEEE International Conference on Cluster Computing (CLUSTER) (2014). https://doi.org/10.1109/CLUSTER.2014.6968768

[20] Hammond, J.L., Minyard, T., Browne, J.: End-to-end framework for fault management for open source clusters: Ranger. In: Proceedings of ACM

TeraGrid Conference (2010). https://doi.org/10.1145/1838574.1838583

[21] Avizienis, A., Lapire, J.-C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Transactions on Dependable and Secure Computing **1**(1), 11–33 (2004). https://doi.org/10.1109/TDSC.2004.2

[22] Mano, M.M.: Computer System Architecture. Prentice Hall International Edition, Boston (1993)

[23] Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to Data Mining. Addison-Wesley, Boston (2006)

[24] Evans, R.T., Browne, J.C., Barth, W.L.: Understanding application and system performance through system-wide monitoring. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (2016). https://doi.org/10.1109/IPDPSW.2016.145

[25] Palmer, J.T., Gallo, S.M., Furlani, T.R., Jones, M.D., DeLeon, R.L., White, J.P., Simakov, N., Patra, A.K., Sperhac, J., Yearke, T., Rathsam, R., Innus, M., Cornelius, C.D., Browne, J.C., Barth, W.L., Evans, R.T.: Open XDMoD: A tool for the comprehensive management of high-performance computing resources. Computing in Science & Engineering **17**(4) (2015). https://doi.org/10.1109/MCSE.2015.68

[26] Agresti, A., Franklin, C.: Statistics: The Art and Science of Learning From Data. Prentice Hall International, Boston (2009)

[27] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) **58**(1), 267–288 (1996)

[28] Hoerl, A.E., Kennard, R.W.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics **42**(1), 80–86 (2000)

[29] Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society. Series B (Statistical Methodology) **67**(2), 301–320 (2005)

[30] Walpole, R.E., Myers, R.H., Myers, S.L.: Probability and Statistics for Engineers and Scientists. Prentice Hall International, Boston (1998)

[31] Das, A., Müller, F., Rountree, B.: Systemic assessment of node failures in HPC production platforms. In: Proceedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS) (2021). https://doi.org/10.1109/IPDPS49936.2021.00035