Improving Log-Based Field Failure Data Analysis of Multi-Node Computing Systems

Antonio Pecchia*, Domenico Cotroneo*, Zbigniew Kalbarczyk†, Ravishankar K. Iyer†
*Dipartimento di Informatica e Sistemistica, Università degli Studi di Napoli Federico II,
Via Claudio 21, 80125, Naples, Italy
†Center for Reliable and High-Performance Computing, University of Illinois at Urbana-Champaign,
1308 W. Main Street, Urbana, IL 61801, USA
{antonio.pecchia, cotroneo}@unina.it, {kalbarcz, rkiyer}@illinois.edu

***Abstract*—Log-based Field Failure Data Analysis (FFDA) is a widely-adopted methodology to assess dependability properties of an operational system. A key step in FFDA is filtering out entries that are not useful and redundant error entries from the log. The latter is challenging: a fault, once triggered, can generate multiple errors that propagate within the system. Grouping the error entries related to the same fault manifestation is crucial to obtain realistic measurements.**

This paper deals with the issues of the tuple heuristic, used to group the error entries in the log, in multi-node computing systems. We demonstrate that the tuple heuristic can group entries incorrectly; thus, an improved heuristic that adopts statistical indicators is proposed. We assess the impact of inaccurate grouping on dependability measurements by comparing the results obtained with both the heuristics. The analysis encompasses the log of the Mercury cluster at the National Center for Supercomputing Applications.

***Keywords*-Field Failure Data Analysis; supercomputer; tuple heuristic; collision; dependability measurements.**

I. Introduction

Log-based Field Failure Data Analysis (FFDA) is a widely-adopted methodology to assess dependability properties of an operational system starting from the raw log produced by different computing entities, such as operating systems, network subsystems, and applications. Analysis of the failure data provides valuable information on actual error/failure behavior and allows identification of system bottlenecks, quantification of dependability measures, and verification of assumptions made in system models. Given a large volume of data collected in real systems, the key step of a measurement-based study consists of removing non-useful data and more importantly, coalescing redundant failure data by grouping entries in the log that are related to the same fault manifestation. This task is challenging because a fault, once triggered, can generate multiple errors that propagate within the system and, consequently, multiple notifications in the log [6], [16]. Identifying propagation phenomena is crucial to obtain realistic measurements.

A widely used strategy to group the entries in the log is the **tuple heuristic** [6]. The intuition underlying this approach is that two entries in the log, if related to the same fault activation, are likely to occur near in time. Consequently, if their time distance is below a predetermined threshold, i.e., the *coalescence window*, they are placed in the same group (called *tuple*). This approach has been originally developed and validated in the context of centralized and small-sized systems, such as Tandem [5]. In these studies, the coalescence window, determined via a sensitivity analysis, is selected to group log entries into tuples. The tuples provide a reasonable approximation of the number of failures that occur during the time the log has been collected.

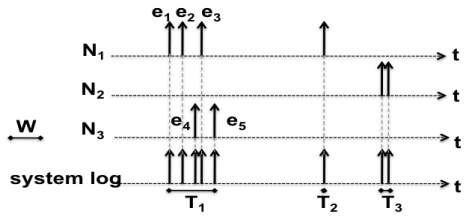


Figure 1: Example of misgrouping of events in the log.

However, an incorrectly selected coalescence window (either too narrow or too wide) for generating tuples can lead to inaccurate results. The identification of a proper coalescence window becomes particularly difficult when analyzing the log of systems composed by hundreds or thousands of nodes. For example, let N_1 and N_3 be two distinct nodes logging the events e_1, e_2, e_3, and e_4, e_5, respectively, as depicted in Fig. 1. It can be noted that the entries interleave in the system log and, for the coalescence window `W`, they are grouped in the same tuple T_1. We may observe two cases, i.e., the entries are triggered by (i) one fault, whose effects propagate between the nodes, (ii) two independent faults, which just occur *coincidentally*, i.e., they are triggered *near* the same time. We will refer to the latter case as *collision*[1] and provides an example of incorrect grouping. Collisions are challenging in case of multi-node systems, as the chance of independent faults occurring coincidentally is not negligible. Furthermore, it is difficult in

[1]The paper focuses on inter-node collisions. As shown by [6], the tuple heuristic is a good strategy to group the entries produced by a single node.

978-1-4244-9233-6/11/$26.00 ©2011 IEEE

practice to differentiate between collisions and propagated errors solely on a temporal basis. Analyzing the entries produced by individual nodes or belonging to a specific error category, e.g., memory, network, etc., *separately*, can mitigate the problem of incorrect grouping. However, this approach provides a more detailed view of the system and makes it hard to characterize the dependencies among the nodes. For this reason, rather than resorting to fine-grain analysis, which, may not completely solve the problem [16], a possible solution is to improve the grouping strategy by analyzing the collision phenomena.

In this paper we develop a heuristic to improve the way entries are grouped by combining (i) the temporal information in the log, and (ii) statistical techniques aiming at quantifying the likelihood of entries produced by different nodes to occur near in time. This information is used to differentiate collisions from propagation phenomena. Furthermore, we compare analysis results obtained with both the tuple and the proposed heuristic to assess the distortion caused by collisions on dependability measurements produced by FFDA and to investigate the causes of the collision phenomena. The data log used in this study is produced by the Mercury cluster at the National Center for Supercomputing Applications (NCSA)[2]. The key findings of our study are summarized in the following:

- Collisions lead to the *overestimation* of the measurements. For example, the difference between the Mean Time Between Failures (MTBF) obtained with the two heuristics, applied to the same dataset, is around 11%.
- The impact of collisions is significant even when characterizing individual categories of error; however, the impact is different among the categories. For example, collisions alter the measure of the MTBF for *input-output* failures by more 9.5%, but the distortion is negligible for *network* or *processor* failures.
- With reference to the available data, the occurrence of collisions does not seem to be related to the temporal distribution of the failures or the length of the tuples.
- Collisions are caused by some specific nodes, which *bias* the content of the log by generating a considerable number of entries. It is very likely that the entries produced by other nodes accidentally interleave the entries produced by the biasing ones.

The rest of the paper is organized as follows. Section II presents related work in the area of FFDA. Section III discusses the data log and some key characteristics of the collected entries. Section IV provides a detailed perspective on the collision phenomena and introduces the basic concepts of the proposed heuristic, extensively discussed in Section V. The failure analysis is conducted in Section VI, while the causes of the collisions are investigated in Section VII. Section VIII concludes the work.

[2]`www.ncsa.uiuc.edu` - University of Illinois at Urbana-Champaign

II. Related Work

Logs are conceived as human-readable text files for developers and administrators. They represent one of the few mechanisms that make visible the system's behavior [16]. For this reason, logs have been widely used by many FFDA studies in the context of different application domains during the last three decades. A non-exhaustive list includes, for example, operating systems [21], [11], control systems and mobile devices [12], [3], supercomputers [16], [13], and large-scale applications [17], [20]. These studies contributed to achieve a significant understanding of the failure modes of these systems, and studies such as [15], made it possible to improve their successive releases. Despite the existence of software packages that integrate state-of-the-art techniques to collect, manipulate, and model the data log, e.g., [22], [24], FFDA also requires ad-hoc strategies and algorithms to identify failure-related entries in the log and to coalesce the entries related to the same problem. That these tasks have been critical to the objective of achieving accurate measurements has been recognized since early work in this area, such as [10], [2].

Authors in [6] discuss the validity of the *tuple heuristic* that coalesces the entries in the log occurring close in time. Authors investigate the collision phenomena and discuss the process of selecting a proper timing window to group the entries. Early FFDA studies have been conducted in the context of centralized and small-scale systems. For example, [6] encompasses the log of a Tandem system consisting of four processors; authors in [23] point out that the failure distributions of different machines are correlated by analyzing the log of a VAXcluster system composed by 7 machines. Subsequently, the tuple heuristic has been adopted to analyze the log of large-scale, networked systems. The methodological improvement of the heuristic is represented by the concept of *spatial coalescence*; errors can propagate among the nodes of the system, and notifications related to the same fault manifestation might be spatially distributed as a result. Authors in [13] analyze the log collected from a BlueGene/L system by combining temporal and spatial filtering. They develop failure predictions methods and show that the methods are effective for predicting, for example, around 80% of memory and network failures. The logs collected from five supercomputing systems are analyzed in [16]. The authors provide an optimization of the filtering algorithm proposed in [13]; however, they recognize that the proposed algorithm might remove independent alerts that, by coincidence, happen near the same time on different nodes.

In this paper we investigate the limitations of time-based coalescence when correctly grouping the entries in the log in case of coincident but independent problems. Several works, e.g., [8], [18], [4], [25], [26], use statistical approaches to identify temporal and/or spatial relationships among the entries in the log. More closely related to our

work, [8] uses statistical techniques to quantify the strength of the relationship among entries in the log, with the aim of recognizing intermittent failures. Authors in [26] use the *lift* indicator in the context of a log-preprocessing technique aiming at preserving error patterns to achieve a more accurate prediction of failures. In this work we use statistical indicators to develop an improved version of the tuple heuristic in order to increase the accuracy of the grouping in case of collisions. The results obtained with the proposed heuristic are compared with the results achievable with the tuple heuristic as used in [13], [16]; similarly to our work, they adopt the heuristic to characterize the failure behavior of supercomputing systems. The comparison allows quantifying the distortion introduced by unaccounted collisions on dependability measurements and analyzing the causes of the collision phenomena. To the best of our knowledge, no previous work has investigated these aspects in a systematic way.

III. Data Log and Preprocessing

The analysis encompasses the log of the Mercury cluster at the NCSA. The log has been collected during a period of about 6 months (from Jan-1-2007 to Jul-2-2007). Mercury is composed by 987 IBM nodes (256 dual 1.3 GHz Intel Itanium2 processors and 731 dual 1.5 GHz Intel Itanium2 processors). Myricom's Myrinet interconnects the nodes. Each node runs a Red Hat 9.0 operating system. Mercury has a three-layer architecture consisting of login, computation, and storage nodes. We indicate these nodes as `tg-loginN`, `tg-cN`, and `tg-sN`, respectively, with `N` denoting an integer number. A dedicated node, named `tg-master`, provides supervision and managing capabilities, such as running the daemon responsible for the IBM General Parallel File System (GPFS).

Entries in the log are collected via the `syslog` daemon. Each entry reports, among the others, a timestamp (resolution of 1 second), the name of the node that generated it, and a *text-free* message providing specific descriptive content. Entries account for a large number of operational conditions resulting from both normal and abnormal activities. The operating system kernel and components, application processes, and daemons generate the events. During the above-mentioned period, about 200 million entries were collected; the size of the log is about 10GB. Not all the entries are actually useful for performing the failure analysis, as many of them just report informational statements. For this reason, the text-free message of each entry is *de-parameterized* (similarly to [14]) in order to identify the relevant content. In other words, variable fields, such as user names, IP or memory addresses, folders, and so on, are replaced with the $*$ wildcard. As a result, 1,124 distinct messages are identified. Next a manual analysis is conducted to identify those messages reporting severe error conditions, e.g., the messages reporting the events that caused the corruption of

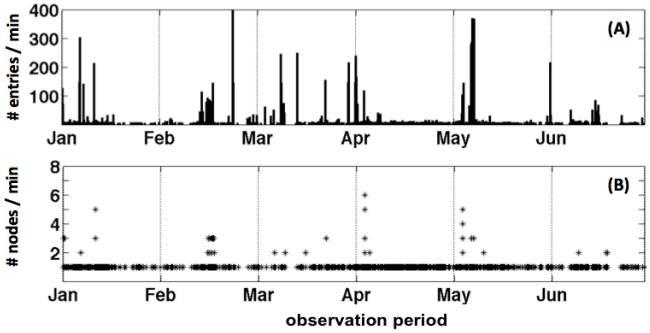


Figure 2: Generation rate of the error entries: (A) # entries per minute, and (B) # nodes per minute

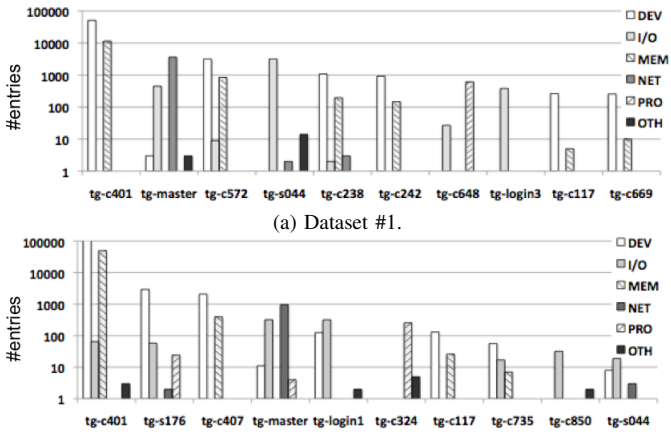
Table I: Error categories and occurrence of the entries.

Category (`acronym`)	Description	dataset #1	dataset #2
Device (`DEV`)	*errors related to peripherals and PCI cards*	57,248	244,301
Memory (`MEM`)	*memory-related non correctable errors*	12,819	49,491
Network (`NET`)	*communication issue raised by a machine*	3,702	973
Input/Output (`I/O`)	*problems, such as SCSI disk errors, damaged sectors*	5,547	1,091
Processor (`PRO`)	*processor exceptions, machine check issues*	1,504	326
Other (`OTH`)	*other errors (not attributed to a specific category)*	34	161
total	*377,197*	80,854	296,343

one or more jobs run in the cluster. To this aim, we use available manuals and, when needed, we communicate with the system management team at the NCSA to get assistance in establishing the meaning of the messages.

A total of 76 unique error messages (out of 1,124) are found, which generated 377,197 entries in the log. *Failure analysis and related measurements focus on these 377,197 entries*. Fig. 2A shows how the generation rate of the error entries, i.e., entries per minute, varies during the period the log has been collected. Fig. 2B reports the number of distinct nodes producing the entries each minute. It can be noted that usually only a single node is responsible for the entries in the log; however, in some cases, we observe up to 6 different nodes generating the entries within the time window of 1 minute. In these cases, it is possible to experience collisions. The overall dataset is split into 2 subsets of 3 months. In the following sections, we refer to them as dataset #1 and dataset #2. The same set of analyses has been repeated for both the datasets and, when needed, obtained results are compared. We will show that the two datasets have different features in terms of total number of entries, and distribution of the entries among different error categories. For this reason, repeating the analyses over the datasets helped at achieving a better understanding of the collision phenomena.

Error messages are classified into 6 categories, as described in the two leftmost columns of Table I along with the acronyms used in the rest of the paper. The two rightmost



(a) Dataset #1.

(b) Dataset #2.

Figure 3: Breakup of the entries by node and category.

columns of Table I report the breakup of the entries in the log by category and dataset. Fig. 3 reports, for each dataset, the breakup of the entries by node and category. Only the 10 nodes with largest number of generated entries are reported in each plot. Nodes belonging to the same architectural layer of the cluster exhibit a similar failure behavior. For example, computation nodes are mostly prone to DEV and MEM errors. Storage nodes exhibit a significant number of I/O errors. The tg-master node exhibits many NET errors, mainly due to communication issues with other nodes. As noted by other studies in the area, e.g., [9], this confirms the existence of a correlation between a node and its workload.

IV. Evidence of the collision phenomena

The tuple heuristic is applied to each dataset to provide evidence of the collision phenomena in case of multi-node systems. To this aim, a sensitivity analysis is conducted to assess a suitable value for the coalescence window W. Fig. 4 shows the tuple count as a function of the size of the coalescence window[3]. According to [6], a reasonable choice for the coalescence window is the value after the "knee" of the curve, where the tuple count sharply flattens. A coalescence window of 240s is thus suitable for both the datasets. For the selected window we observe 476 and 1,206 tuples, for dataset #1 and #2, respectively. These values represent an approximation of the actual number of failures that have occurred in the system. A similar coalescence window has also been observed in other works in the area of log analysis of supercomputing systems, e.g., [13]. We use this value of W to conduct a preliminary analysis, which is reported in this section. A detailed study is presented in Section VI-A.

As discussed in Section I, even when the coalescence window is carefully chosen, the chance of experiencing

[3]For reasons of readability, Fig. 4 does not report the tuple count for W=1s. In this case we observe 11,792 and 31,503 tuples for each dataset, respectively.

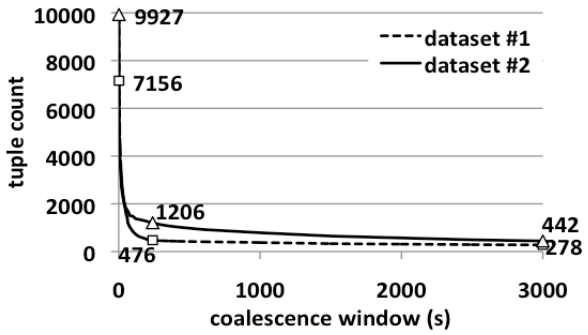


Figure 4: Tuple count as a function of the size of the coalescence window.

distinct faults triggered coincidentally is not negligible in case of multi-node systems. It this case, log entries reporting distinct problems can be grouped together erroneously. The problem is illustrated via examples encountered in the analyzed datasets. The intuition for improving the grouping strategy is also discussed.

Example 1. When applying the tuple heuristic to the dataset #1, it can be observed that the following two entries are placed in the same tuple:

```
(A) 1174245458  tg-master   NET
  stream_eof  connection to * is bad remote service may
  be down  message may be corrupt or connection may have
  been dropped remotely. Node state to down
(B) 1174245678  tg-c324     PRO
  +BEGIN HARDWARE ERROR STATE AT CMC
```

The temporal distance between (A) and (B) is 220s, i.e., smaller than the chosen window of 240s. As discussed in the introduction, this might be the result of a propagation phenomenon or just an accidental collision. It should be noted that there is no discernable relationship between the two entries: they belong to different error categories and are logged by different nodes. Nevertheless, due to the lack of *ground truth*, i.e., the knowledge of the actual failure behavior, it is not possible to determine directly which is the correct option. As a matter of fact, the event log is the only available data source to achieve insights regarding the failure behavior of the system: no additional information is available to supplement the content of existing logs. For this reason, we hypothesize that if an *association* exists between two entries, i.e., the entries share a common fault origin, it is likely that a *significant* (in statistical terms) number of tuples will contain both the entries.

Following this assumption we estimated that entry (A), i.e., the NET event raised by tg-master, appears in 46 tuples. Entry (B), i.e., the PRO event raised by tg-c324, appears in 30 tuple, but just one tuple contains both the entries. This finding enables us to state that it is reasonable to assume that the tuple likely represents a collision. Other examples, reported for the sake of illustration, are obtained by applying the tuple heuristic to the dataset #2.

Example 2. A MEM and DEV error, reported by tg-c407 and tg-c401, respectively, are grouped together because

their time difference is $18s$, again, shorter than the reference coalescence window of $240s$. However, by counting the number of individual and joint occurrences of the entries, it can be observed that (A) and (B) occur 63 and 72 times, respectively: only 2 tuples contain both the entries.

```
(A) 1177454190  tg-c407     MEM
  + Mem Error Detail:  Physical Address: *  Address
  Mask: * Node: * Card: *  Module: *  Bank: *
  Device: *  Row: * Column: *
(B) 1177454208  tg-c401     DEV
  +BEGIN HARDWARE ERROR STATE AT CPE
```

Example 3. Two DEV errors, raised by tg-s176 and tg-c407, respectively, are grouped in the same tuple because their time difference is $125s$, thus shorter than $240s$. We assess individual and joint counts. (A) and (B) occur in 680 and 91 tuples, respectively; however, only 6 tuples contain both the events. It is reasonable to assume that these entries were triggered coincidentally rather than because of a propagating error. This example also shows that it is possible to experience collisions between entries in the log belonging to the same error category.

```
(A) 1181258107  tg-s176     DEV
  + PCI Component Error Detail:Component Info: Vendor Id
  =*  Device Id =*  Class Code =* Seg/Bus/Dev/Func =*
(B) 1181258232  tg-c407     DEV
  +BEGIN HARDWARE ERROR STATE AT CPE
```

V. Proposed solution

The examples analyzed in the previous section suggest that two entries, generated by different nodes of the system, should be placed in the same tuple if both following conditions hold: (i) the two entries are close in time, i.e., their time difference is shorter than the coalescence window W, and (ii) the *degree of association* of the entries is greater than a threshold, which allows us to assume that they are the result of a real propagation phenomenon occurring in the system. In the following, we discuss how to evaluate the degree of association between two entries and how this information can be combined with the tuple heuristic to improve the quality of event grouping.

A. Evaluating the degree of association of the entries

The grouping achieved with the tuple heuristic for a reasonable value of the coalescence window is used as a *basis* to evaluate the degree of association between two entries. For the sake of brevity, given an entry e_i of the log, we will denote with $t(e_i)$, $h(e_i)$, and $m(e_i)$, the timestamp, the originating node and the message of the entry.

The **lift** [1] indicator, used in the context of data mining analysis, provides a suitable metric to assess the degree of association. More specifically, given a database of *transactions*, each of them consisting of a sequence of *items*, the lift measures *how many times two item-sets occur together more often than expected as if they were statistically independent*. This concept can be easily mapped in the context of the problem we are trying to solve. Each tuple is assumed to be a transaction and the entries they contain to be items. Let e_i and e_j be two entries in the log, with $h(e_i) \neq h(e_j)$. As discussed in Section IV, we can calculate the quantities N_i, N_j, $N_{i,j}$, i.e., the number of tuples containing (i) $m(e_i)$ generated by $h(e_i)$, (ii) $m(e_j)$ generated by $h(e_j)$, (iii) both the entries. Given N the total number of tuples, we assess the probabilities $P(e_i) = \frac{N_i}{N}$, $P(e_j) = \frac{N_j}{N}$ and $P(e_i, e_j) = \frac{N_{i,j}}{N}$. Lift is estimated as $l = \frac{P(e_1,e_2)}{P(e_1)\cdot P(e_2)}$. As discussed in Section II, this indicator has been already used in the area of FFDA, e.g., [8], [26].

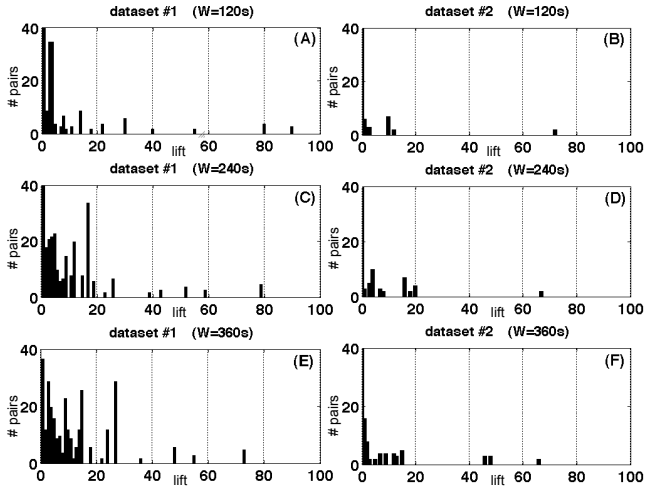


Figure 5: Lift: sensitivity analysis.

Lift can assume any value greater than 0. The greater the lift, the higher the probability for the entries of being related. A sensitivity analysis has been performed to understand how the values of the lift vary with respect to the available datasets. In particular, we estimate the lift for each pair of subsequent events in the log produced by different nodes. Section V-B will show that, since the log is parsed sequentially when applying the heuristic, these values of the lift are used to group the entries. Fig. 5 shows how many pairs (y-axis) exhibit a specific value of the lift (x-axis) in the interval [0, 100]. The analysis has been repeated for the values of the coalescence window in the interval [120, 360]s, i.e., a set of reasonable values for W as shown in Fig. 4. Results show that the values of the lift are rather insensitive to the underlying grouping. Due to space limitations, we only report plots for values of W={120, 240, 360}s.

The plots reported in Fig. 5 show that most of the pairs exhibit low values of the lift, i.e., the values in the left part of each plot. For instance, the lift is around 0 for all the pairs of entries provided as examples in Section IV. Furthermore, it has been observed that the lift might be relatively high even if two entries do not seem to be actually related. For example, regarding the dataset #1, we calculate the lift to be 23 when N_i is 20, N_j is 1 and the number of joint occurrences is 1. In the dataset #2 the lift is 20 when N_i is 50, N_j is 1 and the number of joint

```
1  input:
2    1.) E = {e_1, e_2, ..., e_N} − entries to be grouped
3    2.) W − coalescence window (Section IV)
4    3.) L − association threshold (Section V.A)
5
6  id = 1, grouped = false;
7  tuple_id ← e_1;
8
9  for i = 2...|E| {
10   grouped = false;
11   X = {set of tuples x : t(e_i) − t(last(x)) < W}
12
13   if( |X|! = 0 ){
14     for j = 1...|X| {
15       l = lift(e_i, last(x_j));
16
17       if(h(e_i) == h(last(x_j))){
18         x_j ← e_i; grouped = true;
19       }
20       if(h(e_i)! = h(last(x_j)) && l ≥ L){
21         x_j ← e_i; grouped = true;
22       }
23     }
24   }
25
26   if(grouped == false){
27     id = id + 1; tuple_id ← e_i;
28   }
29 }
```

Figure 6: Proposed grouping algorithm

occurrences is 1. In these cases, it is reasonable to consider the entries to be unrelated. These findings indicate how to select the **association threshold**, i.e., L, in the following, to discriminate between related and unrelated entries; given a pair of entries, if the value of the lift is greater than L, we assume that the entries are related. As discussed, the left part of a plot reporting the distribution of the values of the lift consists of pairs of unrelated entries. The association threshold represents a *border value* between this part of the plot (where most of the pairs converge), and the one containing the pairs exhibiting increasing values of the lift. According to the distribution reported in Fig. 5, L is assumed to be 30 when analyzing the system log (Section VI-A).

B. Proposed grouping algorithm

We integrate the use of the lift indicator into the basic tuple heuristic. Fig. 6 presents the proposed algorithm in a C-like pseudocode. In addition to the notation introduced in Section V-A, "←" indicates the introduction of an entry in a tuple, and $last(t)$, with t denoting a tuple, returns the last entry of the tuple t. Inputs to the algorithm (lines 2-4 in Fig. 6) consist of: (i) the set of the entries to be grouped, (ii) the coalescence window W estimated via the curve "knee" rule, (iii) the association threshold L determined via the sensitivity analysis conducted as described in Section V-A. The grouping of the entries established with the basic tuple heuristic is available during the execution of the proposed algorithm; as discussed, it is used to estimate the lift between a pair of entries in the log (line 15).

The set of the entries to be grouped, E, is parsed sequentially as in the basic heuristic (line 9). Each time an entry e_i is processed, we identify the set of tuples, X, created until that instant of the execution, which satisfy the temporal criteria based on the use of the coalescence window (line 11). Each tuple in X (line 14) is analyzed and e_i is added to the first tuple in X that satisfies one of the criteria shown in line 17 and 20. More specifically, if (i) the same node generates the entries, the e_i is assigned to the tuple by relying *solely* on the timing information (recall that, as stated in Section I, we assume that the tuple heuristic is enough in case of a single node), (ii) two distinct nodes generate the entries we compare the lift with L. If no criteria is matched for any tuple in X, a new tuple is created (line 26).

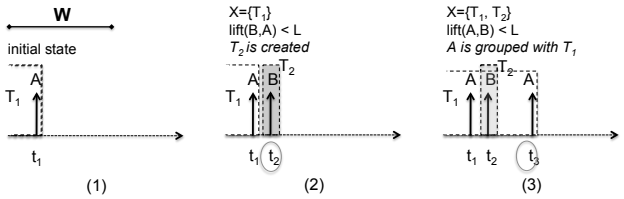


Figure 7: Example of grouping of the entries

An example in Fig. 7 is used to illustrate the need for creating a set X, rather than just trying to add an entry to the last created tuple. Let A and B be entries in the log generated by two different nodes. A and B have a low degree of association, i.e., lower than L, but are close in time (their time difference is lower than that of the coalescence window W reported in the figure). Fig. 7 (1) represents the initial state of the algorithm. When the event B, occurring at t_2, is processed (Fig. 7 (2)), since the lift with A is low, the new tuple T_2 is created. Let A occur again at t_3 (Fig. 7 (3)). The set X (which includes all the tuples to which A could be potentially added solely relying on W) contains T_1 and T_2. Since A and B are not related, and both the events A are generated by the same node, A is assigned to T_1. This approach makes it possible to differentiate multiple interleaving failure dynamics. It should be noted that, with respect to the tuple heuristic, the enhanced algorithm allows creating a new tuple even if the temporal distance between two subsequent entries in the log is shorter than W.

VI. Measurements and Failure Analysis

The tuple and the improved heuristics are used to coalesce the error entries reported in the log. For the sake of brevity, the two heuristics are denoted as T and $T+$, respectively. We demonstrate that even if a proper coalescence window, as described in Section IV, is carefully chosen, unaccounted collisions affect the number of obtained tuples, potentially resulting in the distortion of dependability measurements.

We adopt the *Mean Time Between Failures (MTBF)* and *reliability* as reference metrics. These metrics have been widely used in FFDA studies, e.g., [9], [7], [19], to characterize the behavior of an operational system. MTBF and reliability, estimated by considering both the overall system log and individual error categories, provided further insights into the collision phenomenon. It is worth noting that the

Table II: Sensitivity analysis of the tuple count and the MTBF obtained with T and $T+$ with respect to W.

W (L=30)	T count	MTBF (h)	$T+$ count	c	MTBF (h)	Δ_C %	Δ_M %
dataset #1							
120	725	2.80	773	*48*	2.63	6.62	6.63
150	613	3.32	663	*50*	3.07	8.16	8.17
180	556	3.66	609	*53*	3.34	9.53	9.55
210	497	4.09	551	*54*	3.69	10.87	10.89
240	476	4.27	531	*55*	3.83	11.55	11.58
270	462	4.40	518	*56*	3.93	12.12	12.15
300	453	4.49	509	*56*	3.99	12.36	12.39
330	444	4.58	501	*57*	4.06	12.84	12.87
360	440	4.62	497	*57*	4.09	12.95	12.98
dataset #2							
120	1,374	1.52	1,396	*22*	1.49	1.60	1.60
150	1,338	1.56	1,365	*27*	1.53	2.02	2.02
180	1,297	1.61	1,328	*31*	1.57	2.39	2.39
210	1,266	1.65	1,303	*37*	1.60	2.92	2.92
240	1,206	1.73	1,244	*38*	1.68	3.15	3.15
270	1,161	1.80	1,200	*39*	1.74	3.36	3.36
300	1,124	1.85	1,163	*39*	1.79	3.47	3.47
330	1,098	1.90	1,138	*40*	1.83	3.64	3.65
360	1,068	1.95	1,108	*40*	1.88	3.75	3.75

primary objective of these measurements is to quantify the extent of the distortion introduced by collisions. In other words, we do not discuss the system dependability bottlenecks, impact on overall system usability, or user-perceived availability.

A. Analysis of the system log

We assess how the grouping of the entries and related dependability figures vary when using $T+$. To this end, analysis results produced by both T and $T+$ are compared when a proper coalescence window W is selected. As discussed in Section IV, and according to the plots reporting the tuple count (Fig. 4), 240s is a reasonable choice for W. Nevertheless, in order to drive more general considerations, besides 240s, we repeat the analysis for a subset of values of W in the interval [120,360]s (with a step of 30s). Again, as depicted in Fig. 4, it can be noted that these values of W are in the *stable* part of the curve reporting the tuple count.

Results are summarized in Table II. Column 1 shows the values of the coalescence window. According to Fig. 5, L, i.e, the association threshold, is assumed to be 30 for all the values of W in the considered interval of coalescence windows. Recall that the value of L is relatively insensitive to the underlying grouping of the entries. Column 5 reports the number of collisions c, i.e., the difference between the tuple counts obtained with both the heuristics (reported in columns 4 and 2, respectively). It can be observed that the number of collisions increases as W increases. In other words, the longer W the higher the chance, when using T, to include in the same tuple entries related to independent faults triggered near the same time.

The distortion of the produced measurements is assessed as follows. Let v_T and v_{T+} be the values of a specific measure, e.g., the tuple count or the MTBF, obtained with T and $T+$, respectively. We estimate the percentage difference of obtained values, i.e., Δ_v, as $\frac{|v_T - v_{T+}|}{min(v_T, v_{T+})} \cdot 100\%$. Δ_v quantifies the distortion introduced on the value of the measure due to unaccounted collisions. As shown in column 5 of Table II, the number of collisions is almost in the same order for both the datasets. Nevertheless, the distortion of produced measurements is different, since the final number of tuples is around 530 and 1,200, for each dataset, respectively. For instance, in the case of the reference coalescence window of 240s (highlighted rows in Table II), the number of collisions is 55 and 38, respectively; however, the distortion introduced on the MTBF, i.e., Δ_M, is around 11.5% and 3.15%, respectively (similar percentages can be observed for the distortion of the tuple count Δ_C). In other words, in the first dataset, for a value of the MTBF of about 4h and 15min, the difference of the measure due to unaccounted collisions is about 30min. Difference is small in the second dataset. It has to be noted that the MTBF is the average value of the time intervals between the starting points of two subsequent tuples. As described in Section V-B, $T+$ introduces additional tuples in case of collisions. This results in a greater number of tuples with respect to T within the same observation period of 3 months, thus, the obtained MTBF is shorter with respect to T.

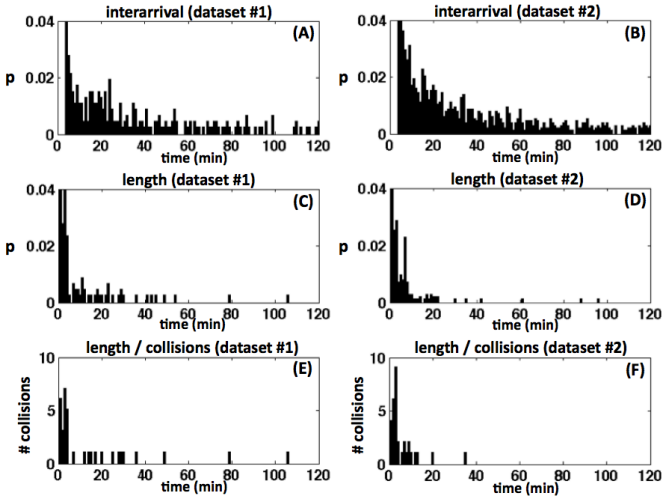
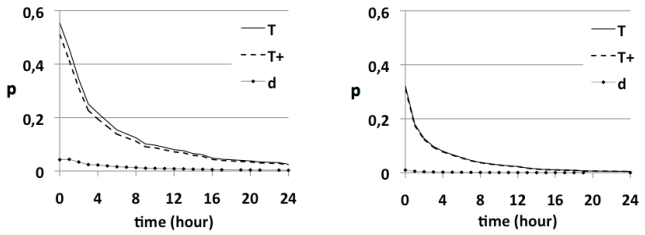


Figure 8: Analysis of the system log: distribution of the interarrival times (A,B) and length (C,D) of the tuples obtained with T. Number of collisions with respect to the length of the tuples (E,F).

The groupings obtained with T and $T+$ differ in terms of length and interarrival times of the tuples. This difference alters reliability, i.e., the probability that the system provides continuity of correct service over a certain amount of time. Fig.8A and 8B report the distribution of the interarrival times of the tuples, obtained with T in case of the reference coalescence window W=240s, for each dataset, respectively. The x-axis reports the duration of the interarrival times (min), and the y-axis the probability of experiencing interarrivals



(a) Dataset #1. (b) Dataset #2.

Figure 9: System reliability obtained with T and $T+$.

of that specific duration. The two datasets exhibit similar distributions. For example, the probability of the interarrivals to be $\geq 10min$ is 0.86 and 0.76 for the dataset #1 and #2, respectively. It should be noted that, when using T, it is not possible to obtain interarrivals shorter than the coalescence window. In other words, it means that the first four bins of the plots 8A and 8B, i.e., 0, 1, 2, and $3min$, do not have any samples. $T+$ overcomes this limitation: it allows introducing a tuple even if the time difference of the entries in the log is shorter than W. Fig. 8C, 8D, 8E, and 8F will be discussed later in the paper

The different distribution of the interarrivals obtained with T and $T+$ alters reliability measurements. This can be observed in Fig. 9. More in details, Fig. 9a and 9b provide the reliability plots obtained with the two heuristics for each dataset, respectively. Furthermore, it is shown the difference between them (d series). The x-axis reports time in hours. The y-axis reports the probability that system does not exhibit any failure, i.e., of any type, such as, DEV, I/O, etc., and from any node, during that interval of time. It can be observed that the probability that the system does not experience any failure, after an operational time of 24 hours, is very low. As for the MTBF, distortion is more significant in the dataset #1. For example, after 2 hours of operations, reliability in dataset #1 is around 0.457 and 0.413, when applying T and $T+$, respectively. The distortion Δ_r, is around 10%. Again, distortion is smaller in dataset #2; the higher number of tuples experienced in this dataset keeps down distortion even if the number of collisions is almost the same as the first dataset.

B. Analysis of the individual error categories

The content of both datasets is analyzed by considering each category of error individually. This type of analysis is useful for pointing out the most failure-prone subsystems and, in the context of our work, makes it possible to achieve further insights regarding the collision phenomenon. In the following, we present the main findings of the analysis. Obtained results show that even when we resort to fine-grain analysis, dependability measurements can be distorted by collisions between failures belonging to the same category.

Results are summarized in Table III. Because of space limitations, we report the tuple count and the MTBF, achieved with both the heuristics and for each dataset, for

Table III: Analysis of the tuple count and the MTBF obtained with T and $T+$ for each category of error.

CTG W-L	T count	MTBF (h)	$T+$ count	c	MTBF (h)	Δ_C %	Δ_M %
DEV	312	6.50	333	*21*	6.09	6.73	6.75
240-30	958	2.11	976	*18*	2.07	1.88	1.88
I/O	93	21.43	102	*9*	19.52	9.68	9.78
240-40	60	32.04	64	*4*	30.00	6.66	6.78
MEM	68	29.12	70	*2*	28.28	2.94	2.99
360-20	87	17.95	90	*3*	17.35	3.45	3.49
NET	66	31.13	68	*2*	30.20	3.03	3.08
240-20	87	23.53	87	*0*	23.53	0	0
PRO	71	27.90	71	*0*	27.90	0	0
120-20	62	32.32	62	*0*	32.32	0	0
OTH	13	163.97	13	*0*	163.97	0	0
360-30	66	29.10	67	*1*	28.66	1.52	1.54

a single value of the coalescence window. In particular, for each category, performing the sensitivity analyses described in the previous sections has identified a suitable combination (W-L); this combination is used to coalesce the entries of that category in both the datasets. Column 1 reports these values. Each category has its own pair (W,L). Given the row related to a category of error (Table III), the upper row reports results obtained for the dataset #1; for the lower one, the results obtained are for the dataset #2.

The number of collisions varies with respect to the specific category of error (Table III, column 5). The DEV and I/O ones exhibit a relevant number of collisions. On the other hand, this phenomenon is almost negligible for NET or PRO errors. Given a specific category, the number of collisions is similar for each dataset, e.g., 21 and 18 for DEV, 9 and 4 for I/O, and so on; however, the distortion introduced on the value of the produced measure, i.e., Δ_C and Δ_M, is significantly different. Again, this is related to the overall number of tuples, which is usually greater in the dataset #2. For example, we observed 21 and 18 collisions for the DEV category. Nevertheless, the distortion of the MTBF is about 6.75% and 1.88%, for each dataset, respectively, thus negligible in dataset #2. On the contrary, collisions significantly distort measurements achieved for both the dataset in case of the I/O category.

Fig. 10A, 10B, 10C, and10D report the distributions of the interarrival times of the tuples obtained when applying T to each category of error. Due to space limitations, for the same category of error we aggregate in the same plot the samples coming from both the datasets. Again, it can be noted that interarrivals shorter than the coalescence window are not allowed with T; this, in turn, alters reliability measurements. For example, Fig. 11a reports the reliability plots obtained for the DEV category, in the dataset #1, and obtained with T and $T+$, respectively. Reliability after 1 hour is about 0.56 and 0.53 according to T and $T+$, respectively; as a result Δ_r is around 5.6%. Similar considerations hold for the I/O category (Fig. 11b). In this case, the distortion due to unaccounted collisions is around 9.5%.

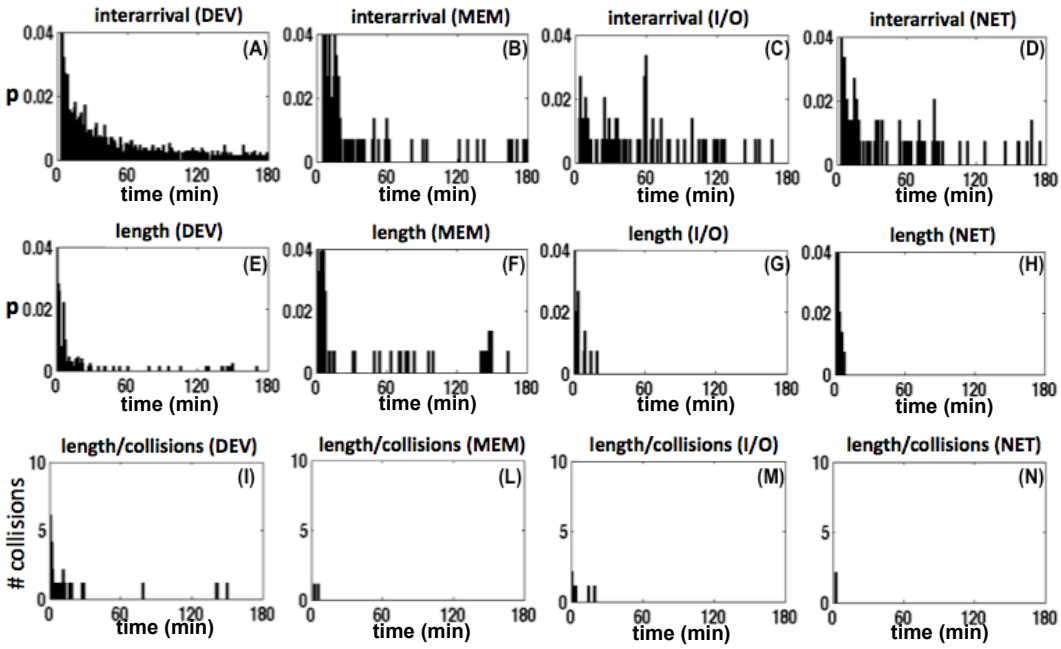
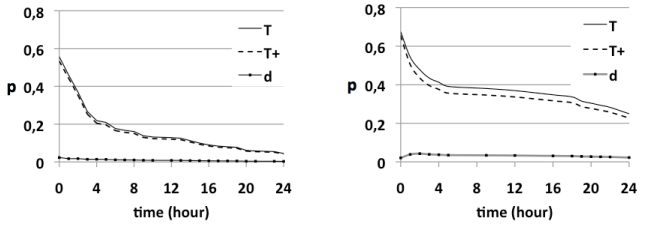


Figure 10: Analysis of the individual error categories: distribution of the interarrival times (A,B,C,D) and length (E,F,G,H) of the tuples obtained with T. Number of collisions with respect to the length of the tuples (I,L,M,N).



(a) DEV - dataset #1. (b) I/O - dataset #1.

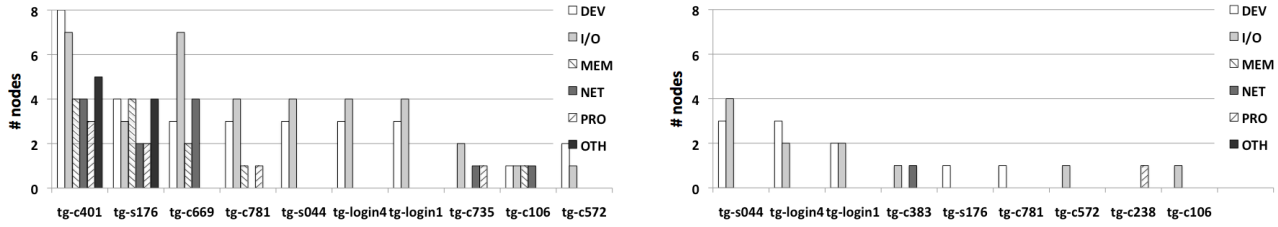
Figure 11: Reliability of DEV and I/O categories obtained with T and $T+$.

C. Key findings: discussion

The analysis of the log of the Mercury cluster reveals that *collisions lead to the overestimation of the measurements obtained by means of FFDA techniques*. The overestimation occurs both for the system log (Section VI-A) and for individual error categories (Section VI-B). The distortion introduced on dependability figures of the system is not negligible: for example, the MTBF is altered by 11.5% with the basic tuple heuristic. Furthermore, reducing the granularity of the analysis, i.e., by considering individual categories of error *separately*, leads to a smaller distortion, e.g., 9.78% in the worst case (I/O failures, dataset #1); however, it does not seem to improve the quality of obtained results. The improvement of the measure is not the only benefit achieved with the proposed heuristic. It has to be noted that better accuracy in differentiating distinct failures occurring near the same time results in more precise diagnosis and identification of failing components.

Analyzed data do not highlight the existence of a particular relationship between collisions and characteristics of the produced tuples, such as the distribution of the interarrivals or their length. For example, similar interarrival distributions result in a different vulnerability to the collision phenomenon. As reported in Fig. 10A and 10B, DEV and MEM exhibit a similar trend, e.g., in both the cases the percentage of interarrivals $\leq 60min$ is around 61% and 62%, respectively; however the number of collisions is almost negligible for the MEM category. A similar finding has been observed for the I/O and NET categories (Fig. 10C and 10D).

The length of the tuples does not seem to affect the chance of experiencing a collision, despite one might think that the longer the tuple the higher the probability of having a collision. Fig. 8C, 8D and 10E, 10F , 10G, 10H report the distribution of length of the tuples for the system log and individual categories of error, respectively. More specifically, the x-axis reports the length of the tuples in min, and the y-axis the probability of experiencing a tuple of that specific length. Similarly, Fig. 8E, 8F and 10I, 10L, 10M, 10N show the number of collision tuples (y-axis), i.e., the tuples that according to $T+$ contain a collision, as a function of the length. In other words, given all the tuples that exhibit a specific length, the y-axis reports how many of them resulted in collisions. The plots indicate a higher number of collisions for short durations of the tuples; however, this is due to the fact that the probability of experiencing short tuples is higher. In general, collisions can be also observed also for a longer duration of the tuples.



(a) Tuple heuristic (T). (b) Proposed heuristic ($T+$).

Figure 12: Analysis of the tuples containing DEV errors.

VII. Investigating the Causes of the Collision Phenomena

Experimental results reveal that unaccounted collisions alter dependability measurements. Furthermore, as discussed in Section VI-C, specific features of the tuples, such as the temporal distribution or the length, do not seem to influence the occurrence of collisions. In order to investigate the causes of collisions experienced in the data log, we compare the content of the tuples obtained with T and $T+$ when applied to the system log for the reference coalescence window of $240s$. A two-steps analysis has been performed. Given a grouping of the entries (obtained with T or $T+$),

1) we identify all the tuples containing at least one entry belonging to a specific error category and produced by a specific node. For example, as shown in Fig. 14, all the tuples containing at least one entry of the DEV category produced by the node tg-c401 are identified;
2) we count how many other distinct nodes, i.e., different from the one considered in the first step, exhibit at least one entry belonging to any other category of error within the set of tuples identified in the first step. Again, given the example in Fig. 14, among the identified tuples, we observe 2 nodes exhibiting a MEM error and other 2 nodes reporting a DEV and an I/O error, respectively.

The described analysis has been repeated for each category of error. Due to space limitations, only the results obtained for the DEV and NET category are discussed. Similar findings have been experienced for the remaining categories.

A. Analysis of the tuples containg DEV errors

The results obtained for the DEV category are summarized in Fig. 12a and 12b, for T and $T+$, respectively. It has to be noted that the results obtained for each dataset have been cumulated in the same plot. The x-axis shows a set of nodes reporting DEV errors in the data log. The y-axis reports the number of distinct nodes (broken down by error category) exhibiting at least one entry in the same set of tuples containing the DEV entries generated by the node indicated on the x-axis. Fig. 12a shows that, when applying T, it is possible to observe any other category of error occurring near in time with a DEV one. For example, when

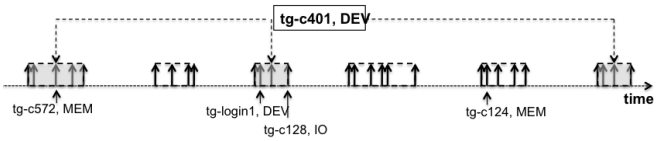
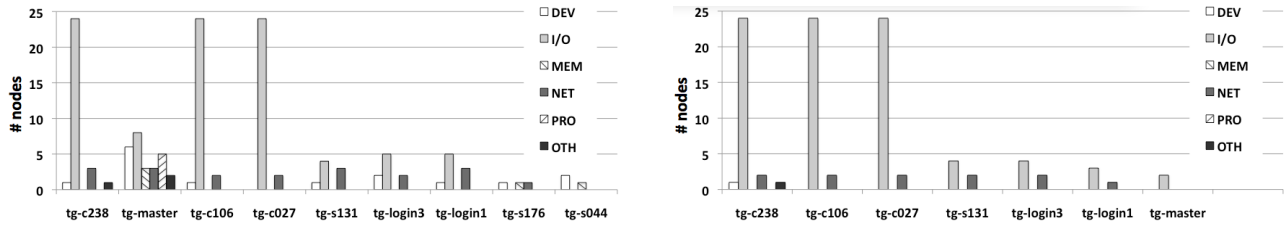


Figure 14: Example of analysis of the tuples.

the node tg-c401 reports a DEV error, other 8 nodes may exhibit a DEV error, 7 an I/O error, 4 a MEM error and so on; especially in the latter case, due to the nature of the errors, it is reasonable to assume that no relationship exists between these problems grouped in the same tuple. In other words, it is not feasible that a DEV failure occurring on a node causes a MEM failure on a different node. Similar considerations apply to other categories. According to Fig. 12a, the problem of incorrect grouping is particularly significant in the case of tg-c401, tg-s176, and tg-c669 nodes.

The content of the tuples obtained with T has been manually inspected in the case of incorrect grouping. Here we discuss some examples for the sake of clarity. For instance, when applying T to the dataset #2, we obtain a tuple of about $7min$ and $30s$, reported on May-2-2007, which contains 3,420 entries raised by tg-c401. Near the same time, 7 entries indicating a PRO failure occurring on tg-c324 are logged coincidentally. These entries interleave the ones raised by tg-c401; thus, the two independent failures cannot be differentiated by T. The finding is similar in the case of tg-s176. For example, this node exhibits a DEV failure on Apr-01-2007, which results in 5 entries in the log. After $22s$ since the last entry reported by tg-s176, the node tg-c117 reports a MEM failure, which led to a further 9 entries in the log. The 14 entries are grouped in the same tuple by T because their time difference is shorter than the coalescence window of $240s$. These examples demonstrate that collisions are tricky to handle and that there is no specific relationship with the length of the tuples. The only analogy among tg-c401, tg-s176, and tg-c669 is that they are among the nodes with the largest number of generated entries in the case of the DEV category. This can be easily observed by recalling Fig. 3. This makes it very likely for the entries raised by different nodes of the system to interleave the entries produced by tg-c401, tg-s176, and tg-c669.



(a) Tuple heuristic (T). (b) Proposed heuristic ($T+$).

Figure 13: Analysis of the tuples containing NET errors.

The analysis of the content of the tuples reporting DEV errors has been repeated for the grouping obtained when applying $T+$. Results are summarized in Fig. 12b. In this case, the nodes with the largest number of generated entries do not bias the content of the tuples. We provide examples of tuples reporting entries logged by different nodes; according to $T+$, these tuples are likely to be a symptom of a real propagation phenomenon other than an accidental collision. In the dataset #2 can be observed a sequence of DEV and I/O errors raised by the storage node tg-s044 on May-25-2007. After $47s$ since the last entry raised by this node, tg-c383 reports a GPFS, i.e., the distributed file system, error. These events are very likely to be related and are kept in the same tuple by $T+$. As a matter of fact, considering the architecture of the cluster, the data is stored from computation to storage nodes: thus, an error occurring in a storage node can affect a computation node. In some cases, even if the entries produced by different nodes are grouped together by $T+$, it is not possible to observe clear evidence of the existence of a relationship among them. For instance, on Jan-10-2007 the node tg-c572 exhibits a DEV failure whose duration was around $12min$. Within the same slot of time the node tg-c850 reports an I/O failure. $T+$ groups the entries together because, in the dataset #1, both these events occur in one tuple, and, in addition, exactly in the same one. In this case, due to the lack of sufficient statistical information, $T+$ preserves the grouping achieved with T.

B. Analysis of the tuples containing NET errors

The analysis has been repeated for the tuples reporting errors in the NET category. Results are reported in Fig. 13a and 13b, for T and $T+$, respectively. The interpretation of the plots is the same as that described in Section VII-A. The main difference between obtained groupings, except for some minor issues, is the absence of the tg-master node, as can be observed in Fig. 13b. The tg-master node produces the largest number of entries in case of NET errors; this can be easily appreciated in Fig. 3. More specifically, it is very likely to produce the message "stream_eof connection to * is bad, remote service may be down,message may be corrupt, or connection may have been dropped remotely. Setting node state to down" *continuously* in the available data set. This message can accidentally interleave the error notifications reported by different nodes. It must be noted that, in this message, the * wildcard represents the name of a node. For this reason, we manually investigate the content of the tuples reporting both this message and at least one notification coming from a different node, in order to figure out in how many cases the tg-master was complaining of the node reporting an error in the same tuple. In 16 out of 19 cases the nodes are different: the entries (erroneously grouped in the same tuple by T) are reasonably symptoms of independent problems, thus, they should not be grouped together. It can be observed that this issue is resolved by applying $T+$.

Except for the tg-master node, Fig. 13a and 13b present a similar perspective. In the following, we discuss the content of the tuple involving the nodes tg-c238, tg-c106, and tg-c027. In particular, on Apr-29-2007, each of these three nodes reported a NET problem, notified by the message "conection down", within a time window of $15s$. The three notifications are followed after $85s$ by 174 I/O entries raised by 24 distinct nodes and for a total duration of about $5min$. Both T and $T+$ group the described events in the same tuple. In the former case, the grouping occurred as a result of the timing information in the log. In the latter case, it was observed that the three NET errors occur in one tuple and, again, exactly in the same tuple. Similar considerations hold for the I/O entries. As discussed for the DEV category, when $T+$ cannot achieve enough statistical evidence, it preserves the grouping provided by T.

VIII. Conclusion

This paper investigated the issues concerning the coalescence task of the log collected in multi-node systems. By means of the data produced by the Mercury cluster at the NCSA, we demonstrated that the tuple heuristic fails to correctly group the entries related to independent faults triggered near in time on different nodes. As a result, final dependability measurements can be distorted. An improved grouping heuristic, relying on the use of the lift indicator, has been developed to discriminate collisions from actual propagation phenomena. In the future, we aim to apply the proposed strategy to the log collected in the context of other supercomputing systems. The analysis will provide new insights on collisions and will possibly suggest further improvements to the proposed solution.

ACKNOWLEDGMENT

This work has been partially supported by the projects "CRITICAL Software Technology for an Evolutionary Partnership" (CRITICAL-STEP, http://www.criticalstep.eu, a Marie Curie Industry-Academia Partnerships and Pathways (IAPP) number 230672), "Overcoming Security Market Obstacles for SME's Involvement in the Technological Supply chain" (OSMOSIS, http://www.osmosisecurity.eu, GA-242416) both in the context of the EU Seventh Framework Programme (FP7), by NSF grants CNS-05-51665 (Trusted Illiac) and CNS-10-18503 CISE, the Department of Energy under Award Number DE-OE0000097, and IBM Corporation as part of OCR (Open Collaboration Research).

REFERENCES

[1] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. pages 255–264. ACM Press, 1997.

[2] M. Buckley, D. Siewiorek, I. Center, and Y. Heights. A Comparative Analysis of Event Tupling Schemes. In *Proc. of Annual Symp. on Fault-Tolerant Computing*, pages 294–303, 1996.

[3] M. Cinque, D. Cotroneo, and S. Russo. Collecting and analyzing failure data of bluetooth personal area networks. In *Proc. Intl. Conf. on Dependable Systems and Networks*, Philadelphia, Pennsylvania, USA, June 2006.

[4] S. Fu and C.-Z. Xu. Quantifying temporal and spatial correlation of failure events for proactive management. In *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 175 –184, 2007.

[5] J. Gray. A census of tandem system availability. In *IEEE Transactions on Reliability*, pages 40–9, 1990.

[6] J. P. Hansen and D. P. Siewiorek. Models for time coalescence in event logs. In *Proc. of Intl. Symp. on Fault-Tolerant Computing*, pages 221–227, 1992.

[7] T. Heath, R. P. Martin, and T. D. Nguyen. Improving cluster availability using workstation validation. In *In Proc. of ACM SIGMETRICS*, 2002.

[8] R. Iyer, L. Young, and P. Iyer. Automatic recognition of intermittent failures: An experimental study of field data. *IEEE Transactions on Computers*, 39:525–537, 1990.

[9] R. K. Iyer, D. J. Rossetti, and M. C. Hsueh. Measurement and modeling of computer reliability as affected by system activity. *ACM Trans. Comput. Syst.*, 4:214–237, August 1986.

[10] R. K. Iyer, L. T. Young, and V. Sridhar. Recognition of error symptoms in large systems. In *Proceedings of 1986 ACM Fall joint computer conference*, ACM '86, pages 797–806, Los Alamitos, CA, USA, 1986. IEEE Computer Society Press.

[11] M. Kalyanakrishnam, Z. Kalbarczyk, and R. K. Iyer. Failure data analysis of a LAN of windows NT based computers. In *Proc. of Symp. on Reliable Distributed Systems*, 1999.

[12] J.-C. Laplace and M. Brun. Critical Software for Nuclear Reactors: 11 Years of Field Experience Analysis. In *Proc. of the 9th Intl. Symp. on Software Reliability Engineering*, 1999.

[13] Y. Liang, Y. Zhang, A. Sivasubramaniam, M. Jette, and R. K. Sahoo. Bluegene/L failure analysis and prediction models. In *Proc. Intl. Conf. on Dependable Systems and Networks*, Philadelphia, Pennsylvania, USA, 2006.

[14] C. Lim, N. Singh, and S. Yajnik. A Log Mining Approach to Failure Analysis of Enterprise Telephony Systems. In *Proc. Intl. Conf. on Dependable Systems and Networks*, June 2008.

[15] B. Murphy and B. Levidow. Windows 2000 Dependability. MSR-TR-2000-56, Microsoft Research, Microsoft Corporation, Redmond, WA, June 2000.

[16] A. J. Oliner and J. Stearley. What supercomputers say: A study of five system logs. In *Proc. Intl. Conf. on Dependable Systems and Networks*, pages 575–584. IEEE Computer Society, 2007.

[17] D. L. Oppenheimer, A. Ganapathi, and D. A. Patterson. Why do internet services fail, and what can be done about it? In *USENIX Symp. on Internet Technologies and Systems*, 2003.

[18] R. K. Sahoo, A. J. Oliner, I. Rish, M. Gupta, J. E. Moreira, S. Ma, R. Vilalta, and A. Sivasubramaniam. Critical event prediction for proactive management in large-scale computer clusters. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 426–435, New York, NY, USA, 2003. ACM.

[19] R. K. Sahoo, A. Sivasubramaniam, M. S. Squillante, and Y. Zhang. Failure Data Analysis of a Large-Scale Heterogeneous Server Environment. In *Proc. Intl. Conf. on Dependable Systems and Networks*, Florence, Italy, 2004.

[20] B. Schroeder and G. A. Gibson. A large-scale study of failures in high-performance computing systems. In *Proc. Intl. Conf. on Dependable Systems and Networks*, 2006.

[21] C. Simache and M. Kaâniche. Availability assessment of sunOS/solaris unix systems based on syslogd and wtmpx log files: A case study. In *PRDC*, pages 49–56. IEEE Computer Society, 2005.

[22] D. Tang, M. Hecht, J. Miller, and J. Handal. Meadep: A dependability evaluation tool for engineers. *IEEE Transactions on Reliability*, 47(4):443–450, 1998.

[23] D. Tang and R. Iyer. Impact of correlated failures on dependability in a vaxcluster system. In *Proc. of the IFIP Working Conference on Dependable Computing for Critical Applications*, 1991.

[24] A. Thakur and R. K. Iyer. Analyze-NOW - An Environment for Collection and Analysis of Failures in a Networked of Workstations. *IEEE Transactions on Reliability*, pages Vol. 45, no. 4,560–570, 1996.

[25] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan. Detecting Large-Scale System Problems by Mining Console Logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating Systems Principles (SOSP)*, 2009.

[26] Z. Zheng, Z. Lan, B. Park, and A. Geist. System log preprocessing to improve failure prediction. In *International Conference on Dependable Systems and Networks*, 2009.