Strategy for Early Recognition and Proactive Handling of Disruptions Regarding the Service of Computer Centres and IT Infrastructures Based on Statistical Methods

Martin Zinner, Kim Feldhoff, Wolfgang E. Nagel
Center for Information Services and High Performance Computing (ZIH)
Technische Universität Dresden
Dresden, Germany
E-mail: {martin.zinner1, kim.feldhoff, wolfgang.nagel}@tu-dresden.de

***Abstract*—Ensuring smooth operations of data centres is key to a company's success. This endeavour is becoming more and more difficult. Given the continuously increasing amount of data, real-time demands and continuous system availability requirements, the Information Technology (IT) infrastructure is increasingly becoming more and more complex and unmanageable. Humans are not able to manually identify in time the breakdowns of the IT systems, let alone predict or avoid them. Hence, there is a need for an overall strategy regarding the early recognition or rather, the avoidance of outages. In the focus of our attention are technologies in the area of artificial intelligence, data analytics, anomaly detection, logging, and parsing, etc. We define an overall strategy in this regard, such that by automating and/or reducing routine activities the support team can concentrate its activity on setting up leading edge technologies rather than relying on the individual skills of some team members in fixing or preventing the failures. To conclude, our strategy supports a paradigm shift from more or less subjectively designed individualistic conceptions in handling of disruptions regarding the service of computer centres and IT infrastructures towards objectively established optimal solutions.**

***Index Terms*—Computer centre; Data Analytics; Statistical methods; Anomaly detection; Artificial Intelligence; Trend analysis; Failure diagnosis; Failure prevention; Monitoring system; Event logs.**

I. Introduction

Given the continuously increasing amount of data, cloud enabling of applications, virtualisation and software as a service, the IT infrastructure is increasingly becoming more and more complex, such that the Information Technology (IT) departments lose the overview of their systems [1] [2]. Moreover, the aging legacy systems, which are critical to day-to-day operations, but are based on outdated technologies, additionally contribute to the confusions within the IT departments. Hence, humans are not able to manually identify in time the failures (breakdowns) of the IT systems, not to mention their ability to identify the cause of the failure and remedy the outage within the required time frame as specified by the support agreements. An IT system includes all hardware, software, and peripheral equipment, operated by a limited group of IT specialists. Hence, the IT system can be an application, a *Virtual Machine* (VM), a server, the whole computer centre or IT infrastructure.

Moreover, cascading of failures should be avoided, since they can cause a partial or a complete breakdown of the IT systems, leading to production outages. Hence, by reacting in time to failures, often far below the legal required response time imposed by contracts, possible negative side effects and subsequent failures can be avoided or their number can be considerably reduced. The optimal functioning of the computer centres can be measured by the *Quality of Service* (QoS) they deliver [3]. The QoS describes the overall performance of a service as seen by the users of the service or measured by appropriate metrics. A metric is a quantitative measure used to characterise, evaluate, and compare performance, whereas measurement is the process by which numbers or symbols are assigned to attributes of entities [4].

An experienced car driver knows that while the engine is running and the oil pressure fails, after a short period of time, the pressure control system stops the compressor and consequently, – in order to avoid damage – shuts down the engine. A much better strategy is to avoid low oil pressure by monitoring the values of the sensors and by taking appropriate proactive actions if these values reach some predefined thresholds.

Regarding our study, two major perspectives on the health of a specified IT system have been identified:

a) Objective perspective - the view which the surrounding system has on the specified IT system,
b) Subjective perspective - the view which the IT system has regarding his own health.

For example, the event log files, which an application generates, are part of the subjective perspective of the application regarding its own health. On the other hand, metrics like the daily standard deviation of the *Central Processing Unit* (CPU) load and *Random Access Memory* (RAM) usage regarding the application, is part of the objective perspective, the operating system has on the application. The subjective perspective, an application has regarding his own health, may be inaccurate, since for example, the log files may be incomplete or even erroneous. On these grounds, it is essential to set up a proper objective perspective. Within this article, we suppose that the objective perspective is reliable and accurate. This is not true per se, since the operating system, which gathers the metrics, may also be buggy, but a discussion in this regard is outside the scope of this article. Although, the objective perspective is accurate, the default metrics delivered by the usual applications like the operating systems do not deliver enough information regarding the health of the IT systems. The ultimate challenge is to set up an appropriate strategy regarding the estimation of the health of the IT systems based

 ISBN: 978-1-61208-997-3

on the available basic information delivered by the operating systems, log files, including the system log files, sensors, etc.

A. Motivation

1) Rapidly increasing data amount: The total amount of data created, captured, and consumed globally is forecast to increase rapidly, reaching more than 180 Zettabytes in 2025, as opposed to 64.2 Zettabytes in 2020 and 15.5 Zettabytes in 2015 [5]. Worldwide by 2022, over 50 billion *Internet of Things* (IoT) devices including sensors and actuators are predicted to be installed in machines, vehicles, buildings, and environments and/or used by humans.

2) Real-time demands: Real-time information processing has become a significant requirement for the optimal functioning of the manufacturing plants [6]. Demand is also huge for the real-time utilisation of data streams [7]. The operations of a real-time system are subject to *time constraints* (deadlines), i.e., if specified timing requirements are not met, the corresponding operation is degraded and/or the QoS suffer and it can lead even to system failure [8]. There is a general tendency that real-time requirements are becoming crucial requisites.

Travellers require current flight schedules on their portable devices to be able to select and book flights; in order to avoid overbooking, the flight plans and the filled seats must be kept reasonably current. Similarly, people expect instant access to their business-critical data in order to make informed decisions. Moreover, they may require up-to-date aggregated data or even ad-hoc requests. This instant access to critical information may be crucial for the competitiveness of the company [9].

3) 24/7/365 (round-the-clock) system availability: Continuous availability requirements for production-support IT systems, for example in the semiconductor industry, are the general rule. Many companies consider their tolerance for factory non-scheduled downtimes, due to their revenue losses, plain and simple to be zero. Manufacturing systems tend towards fully automated production systems. Moreover, in the semiconductor industry, in order to avoid chaotic situations, even the outage of some central IT systems, like the *Manufacturing Execution System* (MES) leads to a graceful shutdown of the production.

B. Aim

Our intention is to define an implementable strategy to achieve *Early Recognition and Proactive Handling of Disruptions* (ERPHD) regarding the service of computer centres and IT infrastructures. Furthermore, the objective of our strategy is to keep operational expenses low, such that by reducing routine activities, the IT staff can focus mainly on innovation. We need a strategy of maximising outcome through leading edge technologies while minimising maintenance and support effort.

In conclusion, our strategy is enabling a new overall perspective on handling the disruptions regarding the service of computer centres and IT infrastructures, such that it contributes to a paradigm shift from more or less subjectively designed individualistic conceptions in handling of disruptions towards objectively established optimal solutions.

C. Paper organisation

The remainder of the paper is structured as follows: Section II gives an overview regarding existing work related to the described problem. An informal description of our strategy is presented in Section III. The presentation of the main results and discussions based upon these results constitute the content of Section IV, whereas Section V summarises our contributions and draws perspectives for future work.

II. Related work

The focus of this section is primarily on the existing industry-specific solutions. We are not aware of similar open source implementations.

A. HPE InfoSight

Hewlett Packard Enterprise (HPE) helps customers address potential infrastructure issues with HPE *InfoSight*, a cloud-based *Automated Infrastructure Operations* (AIOps) platform, that applies InfoSight *Artificial Intelligence* (AI) and advanced machine learning to go beyond simple troubleshooting. HPE InfoSight puts the focus on prevention. It uses predictive analytics to predict, prevent, and auto-resolve problems from storage to VMs. HPE InfoSight prevents customers from ever seeing a known issue through advanced pattern-matching algorithms. HPE InfoSight eliminates most of the pain of managing HPE infrastructure alone by automatically predicting and being able to resolve most of the issues automatically. For example, InfoSight might identify performance issues between the VMs and the storage system. This allows for increased application availability and reduced costs, while avoiding the typical headaches of manually digging into log files. In order to achieve the above benefits, HPE must collect telemetry data from the systems [10] [11].

B. Ironstream

Ironstream® integrates machine data from traditional legacy IBM systems into leading IT analytics platforms to work seamlessly with Splunk®, ServiceNow®, Micro Focus®, Microsoft®SCOM, Elastic, Apache Kafka® [12] [13]. Hence, organisations can monitor and manage their IT systems from a single management console by integrating event and system performance data from these traditional IBM systems into their platforms.

C. Splunk

Splunk offers a central platform for analysing machine generated data by developing advanced analytics [14] [15]. It can gather data from various log files including also applications event logs. Splunk can collect data from various location and combines it into centralised indexes and aggregates the log files from many sources to make them centrally searchable. It can drill down to the period when the problem occurred in order to be able to determine its cause. Appropriate alerts can

 ISBN: 978-1-61208-997-3

be generated to avoid similar problems in the future. Moreover, important patterns and data analytics can be derived. A Search Processing Language (SPL) [16]–[18] has been developed in order to filter, summarise, and visualise large amount of data.

D. SAP HANA Troubleshooting and Performance Analysis Tool

In order to support its in-memory database HANA, the software company SAP introduce onto the market a troubleshooting and performance analysis tool [19]. This tool fulfils the expected requirements, like performance and high resource utilisation, has a section of common symptoms and troubleshooting, root causes and solutions, etc. for the general part. For example the section root causes and solutions has a part regarding thread monitoring: "What and how many threads are running, what are they working on, and are any of these threads blocked?" or "Are any operations running for a significantly long time and consuming a lot of resources? If so, when will they be finished?".

Moreover, the tool has a part corresponding to database issues, like SQL statement analysis, query plan analysis, advanced analysis, etc. For example, the advance analysis comprises analysing column searches, analysing table joins, etc. The troubleshooting and performance analysis strategy of SAP regarding HANA is very complex and fulfils the requirement and expectations of the users. Unfortunately, it is built by considering the implementation and particularities of a proprietary in-memory database. Of course, some concepts as the importance of the thread monitoring can be considered.

E. Toward Resilience in HPC

High-Performance Computing (HPC) is a technology that harnesses the power of supercomputers or computer clusters to solve complex problems requiring massive computation. In addition to parallel processing, HPC jobs also require fast disks and high-speed memory. Therefore, HPC systems include computing and data-intensive servers with powerful CPUs that can be vertically scaled. HPC systems can also scale horizontally by way of clusters. These clusters consist of networked computers, including scheduler, compute, and storage capabilities [20]. At the University of Technology Dresden/Germany, "jam–e jam", a prototype to analyse and predict system behaviour based on statistical analysis has been developed. It detects node-level failures on HPC systems, as early as possible, in order to employ appropriate protective measures in useful time. The main source of monitoring data are the system log entries – data using the syslog protocol – due to their availability and information richness [21].

In conclusion: the main focus of the existing industry-specific solutions is primarily system performance data on proprietary hardware like HPE InfoSight and Ironstream, and software like SAP HANA. Additionally, Splunk focuses on gathering and interpreting event log data based on a proprietary language (SPL), which increases the learning curve. In contrast, our strategy comprises an overall solution, which is vendor-independent, it does not need inside knowledge regarding the implementation and functioning of the applications and it is based on leading edge but harmonised technologies. It can be conceived as a central monitoring and troubleshooting environment. Additionally, it proposes a uniform event log parsing and analysis strategy, which does not assume inside knowledge of the structure. We are not aware of any commercially available tool which uses this approach.

III. Strategy Description

In a nutshell, the strategy regarding the disruptions of the services of computer centres and IT infrastructures might be:

a) Monitoring the health of the IT systems,
b) Predicting possible malfunctions, interruptions, and downtimes,
c) Proactive actions in order to avoid possible degradation of the QoS delivered by the IT systems,
d) Immediate alerts on failures,
e) Appropriate actions in order to avoid subsequent degradation of the QoS delivered by the IT systems,
f) Corrective actions in order to facilitate the resumption of the failed activity, automatic if possible, else manual actions, including bypass solutions (workarounds),
g) Appropriate actions to avoid similar cases in the future.

A. Event logs

Generally speaking, an event log is an automatically produced, usually, but not necessary time-stamped documentation of events relevant to a particular IT system. We analyse the advantages and disadvantages of the event logs within the ERPHD strategy and make proposals in order to be able to use them advantageously.

1) Importance of the event logs: Event logs are semi-structured text which are generated by logging statements in software code [22]. Unfortunately, event logs may be inaccurate, they may not unequivocally identify the cause of the anomaly and the subsequent steps that have to be taken in order to remedy the failure. According to at least two studies, around 60% of failures due to software faults do not leave any trace in event logs, and 70% of the logging patterns aim to detect errors via a checking code placed at the end of a block of instructions [22].

The event logs are written in general by application developers in order to facilitate the debugging and error finding in their programs. The purpose of the event logs, especially of those written by developers, is to support the development process and not to ease troubleshooting once the product goes productive. Ideally, the information delivered by the event logs should be enough to unequivocally identify the cause of the anomaly, and hence to enable the support team to take appropriate corrective actions. Unfortunately, in real-world systems this is not always the case. Hence, the findings in the entries delivered by the event logs should be augmented by additional sources of information.

The event log files are generally human readable, they are either plain format text files, in XML or JSON format, etc.,

 ISBN: 978-1-61208-997-3

and enable an efficient and simple one way of communication of the applications with the outside world and they probably will still widely be used in the future. The biggest advantage of the event log files represents also the major difficulty when trying to parse them, namely their flexibility and designing freedom. The effort to harmonise the information parsed from the event log files should not be underestimated.

2) Parsing strategies: To parse signifies to break something into its parts, parsing comprises a syntactic analysis of a string, such that it is separated into more easily processable components by identifying tokens and looking for recognisable patterns. It converts formatted text into a data structure. When parsing the message, the footprint of the actual text message – i.e., the identification mark of the message without considering irrelevant numbers – should be stored additionally. This is advantageous, since this way, the text of the message can be uniquely identified independent of the included numbers. Hence, to each message, an additional *Unique Text Identifier* (UTID) is associated as its text part. Additionally, some keywords like "error", "warning", etc., and time stamps can/should be tracked. Thus, an *Unique Identifier* (UID) can be defined by using the UTID and some additional attributes. This way, the complexity of the event logs has been reduced to a tuple of identifiers, including UTID, UID, and additional keywords. These tuples can be historised/aggregated and used for evaluations. Obviously, a mapping of the UID to the original message should be set up, such that the stored information can be human readable and interpretable.

Generally, multilingual event log files should not pose difficulties, if the event log entry of a particular statement of the monitored application does not switch the language aleatorily. We are only interested in the footprint of the respective message and are not analysing its content. Moreover, keywords or key phrases such as "error", "warning", etc., which are evaluated, can be mapped to the corresponding English idiom [23] [24].

In conclusion: event logs could/should be extensively used within our strategy, but since they are highly unreliable, they should be used with precaution in the classical sense. Moreover, the history of their behaviour should be carefully analysed and appropriate decision should be taken based on deviation from the expected appearance. There is no set of rules regarding the form and content of the event logs, which makes the analysis of the logs even more difficult.

B. Failures

Generally, a *failure* is an issue with the IT system that prevents it from functioning properly and it is an observed property of the run-time behaviour of the system [25]. It occurs when the "delivered service no longer complies with the specifications, the latter being an agreed description of the system's expected function and/or service" [26]–[28]. The IT system may end abnormally if a failure occurs [29]. We analyse the types of failures which can occur within an IT system and examine the high level strategies to address them.

1) General considerations: The handling of incorrect results, which do not produce abnormal termination is outside the scope of this work. For example, if the mathematical formula is incorrectly implemented and delivers wrong results, but the programs terminates in the usual way then this is a debugging issue for the development or for the testing team, and it is considered a defect or a bug.

A *crash* is a serious software failure, such that the software process terminates unexpectedly. Crashes can be reproducible (e.g., triggering an unhandled exception) or non-reproducible (e.g., accessing invalid memory addresses). We do not make any difference between a failure and a crash which is reproducible. It seems that the most difficult problem besides preventing outages is troubleshooting, i.e., finding the cause of the crash. Troubleshooting is the process of identifying and resolving a problem, error or fault within a software or computer system, whereas debugging requires finding the cause of a problem related to software code and fixing it. A fault is an incorrect part of an IT system, which leads to unintended behaviour and in the end to its failure, whereas an error describes any issue that arises unexpectedly that cause a computer to not function properly [30].

Finding the cause for a non-reproducible crash may be a very cumbersome problem, which is hard to debug or predict. For example, dump analysis can help. In some other cases, the cause is a consequence of traceable malfunctions, for example memory leaks can cause the crash of a program.

An *outage* is a discontinuity in the provision of the service of an IT system or group of IT systems including the computer centre or the IT infrastructure. Analogously, a *disruption* is an unplanned event that causes the general system or major application to be inoperable for an unacceptable length of time (e.g., minor or extended power outage, extended unavailable network/equipment or facility damage/destruction) [31].

2) Detecting the cause of the crash: Using our strategy, finding the cause of a crash becomes a routine activity, and one does not have to rely on the "inside knowledge" of some highly qualified IT personnel. Moreover, the time to restrict the cause of the crash can be reduced to minutes instead of hours.

Our aim is to avoid failures, but this intent is not realistic, bearing in mind the complexity of the present-day computer centres. Hence, methodologies should be set up, such that a very fast reaction to overcome the drawback of failures is generally possible:

a) Sound methods should unequivocally determine that a failure has happened, i.e., the crash is instantly recognised as such,
b) Within short time, reliable techniques should localise and subsequently identify the cause of the failure,
c) Apply the predefined workaround (e.g., by deleting some wrong data, etc.),
d) Restart the application, if necessary,
e) Setup a solution, such that similar failures can be avoided in the future.

In conclusion: The collaboration of a couple of specialists in

 ISBN: 978-1-61208-997-3

order to narrow the cause of outages is not any more necessary. We are seeking a *paradigm change*: from art (troubleshooting) to very well founded rules.

C. Leading edge technologies

We provide a very general overview of the leading edge technologies which can be applied within our strategy. A detailed consideration would go beyond the size of this article.

1) Artificial Intelligence: There are many definitions of *Artificial Intelligence* (AI) depending of the goals one is trying to achieve with an AI system. Amazon defines AI as "the field of computer science dedicated to solving cognitive problems commonly associated with human intelligence, such as learning, problem solving, and pattern recognition" [32].

2) Data Analytics: Data Analytics is the science of analysing data in order to draw conclusions based upon the analysed data. There are couple of basic types, as descriptive, diagnostic, predictive and prescriptive. Their names also suggest their functionality, e.g. predictive analytics forecast possible events, the prescriptive analytics propose an action plan [33].

3) Trend Analysis: Trend analysis attempts to predict future events, it uses historical data in order to forecast future directions [34]. It is based on the idea that what happened in the past gives hints regarding the future. There is no guarantee that the forecast will be correct.

4) Machine learning: Machine learning (ML) is a branch of AI which focuses on the use of data and algorithms to imitate the way that human learn [35]. For example, Amazon builds a lot of its business on machine learning systems. Machine learning is so important to Amazon, they stated, "Without ML, Amazon.com couldn't grow its business, improve its customer experience and selection, and optimise its logistic speed and quality" [32]. The roots of machine learning dates back to Arthur L. Samuel 1959, see a revised version of his research paper [36]. Nowadays, due to the technological advances in storage and processing power, we are witnesses of the revival and further development of innovative machine learning technologies [35].

5) Anomaly Detection: Anomaly detection refers to finding data instances that do not fit the established patterns. *Outlier detection* is similarly defined. Detecting outliers or anomalies in data has been studied in the statistics community since the end of the 19th century [37] [38]. Outliers are not necessary "bad" data, they are extreme data points within data. On the other hand anomalies are outside what is defined as "good" data. Hence, if the model is accurate, both anomaly detection and outlier detection yield the same results, if this is not the case, the model has to be adapted accordingly.

In conclusion: The fundamentals of the leading edge technologies in order to set up our failure prevention strategy have been known in the statistics community for decades. The importance and renaissance of these technologies is due to the progress in the storage and computer processing power area, such that a much higher amount of data can nowadays be evaluated at lower costs.

D. Historisation

Historisation is the process of keeping data available over time. It offers additional aggregation possibilities and hence extended analysis possibilities. It is indispensable to consider historical information in order to compare past and present and therefore to be able to make predictions. Moreover, we need appropriate strategies to detect the alterations due to version upgrades and if possible avoid false alarms. Nowadays, the monitoring possibilities of the technical parameters of networks and VMs are very good. The primary goal of these monitoring systems is to offer current information regarding those metrics in order to support the production requirements. But we need historisation of the data of the VMs to be able to perform trend analysis.

In conclusion: Historisation is absolutely essential in order to be aware of previous correct states, usually up to one year, definitely as long as aggregated data is evaluated for trend analysis, etc.

E. Strategies for failure recognition and troubleshooting

There is a need to recognise instantly that an application has stopped functioning in the expected way. There should be a method in place, such that crashed application can be unequivocally distinguished from the non-responding ones. The latter can resume their normal activity after the cause of non-responsiveness has been eliminated, for example high CPU load or short network outage. Moreover, large SQL-queries (e.g. full table scans) can block some database application as long as the query is running. For example, an unusual number of threads can be an indication that the application has crashed.

Troubleshooting an application crash may be sometimes very difficult and time consuming. Using statistical methods, the number of possible suspicions can be meaningfully reduced. Moreover, these methods can give hints for possible malfunctions, such that the identification of the cause of the failure should be more or less straightforward. All available historised information including event log files, input/output information, consumed resources, etc., can/should be evaluated. This is advisable since the event log files are unreliable and for example outliers are not necessarily a sign of malfunctions; the challenge is to deliver reliable results using unreliable methods. Thankfully, there are reliable techniques, such that a raised suspicion can be confirmed or not. For example, for a database application, an unexpected or a corrupt dataset value can cause a failure if there is no appropriate catch mechanism in place, which detects the unexpected value at the time of its first use. The error propagates, and finally, when the exception is catched, there is no hint in the event logs regarding the real cause of the failure. Hence, multiple diagnosis methods can make the difference. For example, machine learning strategies can be used to identify unusual data sets in data entry.

In conclusion: Failure diagnosis can be a cumbersome issue, since the event log files do not always contain accurate hints regarding the cause of the failure. Multiple methods, like the evaluation of the event log files, statistics based on the metrics

 ISBN: 978-1-61208-997-3

of the resource utilisation, etc., should be simultaneously used in order to improve the accuracy of the failure suspicions and exclude proper functioning IT systems.

F. Strategies for failure avoidance

Generally speaking, the strategies presented in Subsection III-E regarding failure diagnosis can be used for failure prevention. In fact, the technology of identifying non-suspicious IT-systems and the strategy for failure prevention are very similar. Strategies such as the rigour in the development process, verification and validation activities, automated testing using comprehensive test cases, etc., would go beyond the scope of this study [39]. These strategies should be deployed before the roll-out process of the application.

Setting up appropriate metrics and historisation of their values facilitates trend analysis, predictive analytics, and outlier detection. As already mentioned, outliers are not necessarily faulty data, but they can give hints regarding further out of order functioning. Furthermore, a reliable notification infrastructure should be set up in order to be able to react in a timely manner if suspicions occur. For example, by setting up the metric "memory consumption" on application level and by applying trend analysis, potentially large memory consumption, i.e., memory leaks can be detected. This way, by taking appropriate actions, the deterioration of the QoS delivered by the application can be prevented.

In conclusion: by applying statistical methods, the future behaviour of an IT system can be anticipated. The major challenge is to set up the appropriate metrics – i.e. quantitative measures used to characterise, evaluate, and predict anomalies – to best model trend analysis, anomaly detection, failure diagnosis and prevention, etc.

G. Monitoring

Monitoring is the process of gathering metrics regarding the activity of the IT systems. We give a general overview of the overall monitoring strategy with subsequent detailed explanations.

1) General considerations: Commercially available monitoring systems are expected to gather data necessary to be able to decide whether an application is working correctly or not. Unfortunately, while this may be usually the case, the default vendor-delivered metrics are not fully suitable for solving the problem as above. Additional effort is necessary for:

a) Resource monitoring on the application side,
b) Resource monitoring on the server side,
c) Event log monitoring,
d) Input/output monitoring,
e) Direct health validation including threads surveillance,
f) Historisation of the collected information,
g) Using leading edge technologies in order to establish trends and pattern recognition,
h) Establishing strategies for error recognition and efficient troubleshooting,
i) Appropriate strategy in order to avoid the occurrence of crashes.

2) Strategy: The monitoring and evaluation strategy should be set up such that it can be carried out with a medium educated stuff. We are looking now in detail to each of the above items.

a) In order to be able to identify the resources used by the application, commercial or self-made solutions can be used. These information should be historised, such that comparison with successful completion can be done. Every deviation which is behind some thresholds could indicate an anomaly that can lead to failures.
b) Resource monitoring on the server side is similar to resource monitoring on the application side, the additional benefit is that if some resources on the server side surpass for an extended time some threshold, the server may become inoperable. For example, if the CPU is at 100% utilisation, the application may become unresponsive, although it is functioning correctly. If the application has its own VM then the distinction as above is obsolete.
c) There is a need to parse and structure the event logs as indicated above. Furthermore, the event log files along with the newly created structure should be historised, such that statistical evaluations can be performed on the newly created structures.
d) Furthermore, of crucial interest is input/output monitoring, although it is almost always neglected. Unfortunately, unforeseen changes in the input stream can cause unwanted behaviour of the application. For example, for database application an unexpected input value can cause unexpected application reactions, which is almost impossible to predict, since no one counted with this situation. This is also a good counterexample, where possible crash prediction is improbable. To remedy issues as above, an appropriate filter on the input data stream can be set up, but the problem in principle remains.
e) Under the heading *direct health validation* we understand the strategy of gathering direct information from the application itself regarding his health status. For example, a database application can automatically be queried by standard SQL statements within self-made tools. For example, metrics like the response time of the application can be tracked and historised. If the response time degrades under some predefined thresholds then certainly the QoS of the application will degrade accordingly. Furthermore, the behaviour of the threads, possibly including their resource use, can be further supervised. For example, if an application alternates between 4 and 8 threads and the *Operating System* (OS) detects only 2 threads, then the application has crashed, although the OS does not classify it as terminated.
f) Generally speaking, all of the collected information regarding the health of an IT system should be historised in order to allow comparison with earlier behaviour. Commonly, historisation for example using a *Data Warehouse* (DWH) is not a primordial concern of the vendors of monitoring systems, their primary focus is on the

 ISBN: 978-1-61208-997-3

operative part.

g) The leading edge technologies to establish trends, outliers, and pattern recognition have been summarised in Subsection III-C. The presented technologies are complementary, such that multiple technologies can be used in order to achieve our goals.
h) Establishing strategies for error recognition and efficient troubleshooting is one of the most ambitious phases of our strategy. Since not all crashes are recorded in the event logs, establishing whether an application has crashed or not, is not always a straightforward task. Moreover, application may be irresponsive having some residual threads detectable by the operating system, compare Subsection II-D. Well suited technologies are available in the area of anomaly detection in order to identify patterns, which are out of order and give hints regarding application crash, see Subsection III-E for more details.
i) In order to anticipate malfunctions, trend analysis and pattern recognition can be promising approaches, see Subsection III-F for more details.

In conclusion: Monitoring is a complex activity, which involves taking into account all available information regarding the IT systems to be considered. But monitoring alone is not sufficient, the collected data has to be historised in order to be able to compare successful historical operational sequences with the current ones. Additionally, methodologies have to be set up on the historical data based on statistical methods in order to automatically detect failures, their causes, and forecasts malfunctions of the IT systems.

IV. Outline of the results

The main achievement of this work is the cognition that an overall strategy termed ERPHD regarding instant recognition and preventions of failures of the IT systems of a computer centre and/or the IT infrastructure using statistical methods is realisable. This strategy also includes the evaluation of the event log files using similar approaches as for machine or system performance data. Thus, a central point of surveillance can be set up. We give an overview of the pros and cons of the ERPHD strategy and close the section with some general remarks.

1) Advantages of ERPHD strategy: the strategy for early recognition and proactive handling of disruptions provides the technology, such that:

a) Involving almost the same effort, all event logs can be taken into account due to harmonised parsing strategy,
b) All existing applications can be monitored regarding early recognition and proactive handling of disruptions due to the general properties of our strategy,
c) The routine support activity decreases, hence the actual support team can be kept smaller due to the failure prevention strategy,
d) A central monitoring system can be set up, versus many monitoring tools due to our uniform strategy,
e) The learning curve of the support team is manageable due to a central monitoring system,
f) There is an automated evaluation of the event log files, long manual search is obsolete due to the harmonised parsing strategy for all event logs,
g) It is based on reliable leading edge technology,
h) Enables real-time computer centres and IT infrastructure due to our failure prevention strategy,
i) Supports straightforward design strategies due to clear, easy understandable architectural and implementation principles,
j) Avoids or reduces “hot working phases” at night for the IT personnel due the failure prevention capabilities,
k) Ensures very good scalability due to uniform design strategies,
l) Supports easy maintenance due to transparent and straightforward software development process, and last but not least,
m) Supports early detection of erroneous data sets due the advanced statistical methods.

2) Difficulties of ERPHD strategy:

a) There is no open-source or commercially available out-of-the-box product,
b) Difficult architectural set-up, i.e., new algorithms have to be designed and implemented,
c) Longer development times due to a new architectural design strategy,
d) IT staff has to be additionally trained due to unconventional architectural and maintenance strategies,
e) Heterogeneous development teams including mathematicians and data scientist should be built upon, i.e., the algorithmic part of the development may be sophisticated,
f) Increased development costs due to the unconventional development strategies, and last but not least,
g) Strong management commitment to overcome the difficulties due to the anticipated challenges.

3) Final considerations: Assuring and/or improving the QoS of a data centre is a very complex endeavour, in which the human component also plays a very important role. There should be rules set up, such that the actions or decisions taken by the service team should not be based upon the individual skills of its members, but on generally accepted guidelines. In this respect, the experience and know-how of an individual employee should not be lost if he leaves the company.

In conclusion: If the computer centre or the IT infrastructure exceeds a certain size and importance, the advantages of the ERPHD strategy may prevail over its disadvantages. Definitely, if the computer centre supports a round-the-clock production, if there are real-time requirements in place, or high requirements regarding the QoS then the advantages of the ERPHD strategy outweigh the implementation costs.

V. Conclusion and future research perspective

A. Conclusion

There is an increased request for real-time application capability in the industry and research amid rapidly increasing data amount. Furthermore, there is a need for round-the-clock IT

systems availability due to 24/7/365 production requirements. If these requirements cannot be met then the outage of the IT systems leads to production outages with catastrophic consequences for the business. In addition, the complexity of the present day computer centres and IT infrastructures makes it very difficult to manually assure their reliable operability. Hence, to overcome the difficulties as above, there is a need for an overall failure and outage prevention strategy as well as very fast failure detection and diagnosis methodology. Our article is a contribution in this direction.

The existing industrial implementations focus primarily on vendor dependent machine and system performance data, using inside knowledge of the respective IT systems. In contrast, our strategy relies on well-established leading edge statistical methods and takes into account all data generated by the IT systems, including the event logs. The main advantage of our strategy, termed ERPHD is the possibility to centrally monitor the computer centre and IT infrastructure using vendor independent technologies. Moreover, all IT systems including all applications can be monitored using the same technology, hence once the technology has been set up, the effort to extend the monitoring by additional IT systems should pose no problems. This way, once the technology is in place, the QoS of all applications can be substantially improved, raising the QoS of non-production relevant applications with minimal additional effort. Our contribution is a step away from an artisanal approach in handling disruptions towards objectively established optimal solutions. The real challenge is to set up the metrics to best model the IT system.

B. Future research perspectives

A substantial question one can ask himself is the meaningfulness of the objective (e.g. system performance data) versus subjective (e.g. event log files) approach. Additionally, the relevancy of the statistical approach applied on event logs versus the classical approach, where the event logs are content-wise analysed, is of utmost interest. Identifying the weak points of statistical approaches including the limits of applicability, may be of advantage. Last, but not least, the question arises whether there are alternative approaches, which do not rely on proprietary information regarding the IT systems.

REFERENCES

[1] C. Schinko, "Künstliche Intelligenz im Rechenzentrum: So vermeiden Sie IT-Störungen proaktiv [in English: Artificial intelligence in the data center: How to proactively prevent IT disruptions]," *CANCOM.info*, 2018, retrieved: September 2022. [Online]. Available: https://www.cancom.info/2018/11/kuenstliche-intelligenz-rechenzentrum-it-stoerungen-vermeiden/

[2] E. E. Ogheneovo, "On the Relationship between Software Complexity and Maintenance Costs," *Journal of Computer and Communications*, vol. 2, pp. 1–16, 2014, retrieved: September 2022. [Online]. Available: https://doi.org/10.1016/j.procir.2020.05.012

[3] M. Zinner *et al.*, "Techniques and Methodologies for Measuring and Increasing the Quality of Services: a Case Study Based on Data Centers," *International Journal On Advances in Intelligent Systems, volume 13, numbers 1 and 2, 2020*, vol. 13, no. 1 & 2, pp. 19–35, 2020, retrieved: September 2022. [Online]. Available: http://www.thinkmind.org/articles/intsys_v13_n12_2020_2.pdf

[4] N. Fenton and J. Bieman, *Software metrics: a rigorous and practical approach*. CRC press, 2014.

[5] Statista, "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025," 2021, retrieved: September 2022. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

[6] R. Sousa, R. Miranda, A. Moreira, C. Alves, N. Lori, and J. Machado, "Software tools for conducting real-time information processing and visualization in industry: An up-to-date review," *Applied Sciences*, vol. 11, no. 11, p. 4800, 2021, retrieved: September 2022. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/4800

[7] K. Yasumoto, H. Yamaguchi, and H. Shigeno, "Survey of real-time processing technologies of iot data streams," *Journal of Information Processing*, vol. 24, no. 2, pp. 195–202, 2016, retrieved: September 2022. [Online]. Available: https://doi.org/10.2197/ipsjjip.24.195

[8] I. Sommerville, "Software engineering 9th edition," *ISBN-10*, vol. 137035152, p. 18, 2011.

[9] Zinner *et al.*, "Real-time information systems and methodology based on continuous homomorphic processing in linear information spaces," 2015, retrieved: September 2022. [Online]. Available: https://patentimages.storage.googleapis.com/ed/fa/37/6069417bdcc3eb/US20170032016A1.pdf

[10] R. Sheldon, "How HPE InfoSight AI proactively spots, solves infrastructure issues," *SearchStorage*, 2019, retrieved: September 2022. [Online]. Available: https://www.techtarget.com/searchstorage/tip/How-HPE-InfoSight-AI-proactively-spots-solves-infrastructure-issues

[11] "HPE InfoSight for Servers User Guide," *Hewlett Packard Enterprise*, 2021, retrieved: September 2022. [Online]. Available: https://www.hpe.com/psnow/doc/a00061446en_us

[12] "Unlocking Real-time Mainframe Operational Intelligence," *Syncsort Ironstream*, 2015, retrieved: September 2022. [Online]. Available: https://www-50.ibm.com/partnerworld/gsd/showimage.do?id=41083

[13] *Precisely Ironstream*, 2022, retrieved: September 2022. [Online]. Available: https://www.precisely.com/product/precisely-ironstream/ironstream

[14] D. Carasso, "Exploring Splunk Search Processing Language (SPL) Primer and Cookbook," *CITO Research New York, NY*, 2012, retrieved: September 2022. [Online]. Available: https://www.splunk.com/pdfs/exploring-splunk.pdf

[15] J. Miller, "Mastering Splunk Optimize your machine-generated data effectively by developing advanced analytics with Splunk," *PACKT Publishing*, 2012, retrieved: September 2022. [Online]. Available: https://www.splunk.com/pdfs/exploring-splunk.pdf

[16] S. Luedtke, "Power Of Splunk SPL," *splunk*, 2016, retrieved: September 2022. [Online]. Available: https://conf.splunk.com/files/2016/slides/power-of-spl.pdf

[17] "Quick Reference Guide," *splunk*, retrieved: September 2022. [Online]. Available: https://www.splunk.com/pdfs/solution-guides/splunk-quick-reference-guide.pdf

[18] K. Subramanian, "Practical splunk search processing language: A guide for mastering spl commands for maximum efficiency and outcome." Springer, 2020, p. 349.

[19] SAP HANA, "Troubleshooting and Performance Analysis Guide," *SAP Help Portal*, 2018, retrieved: September 2022. [Online]. Available: https://help.sap.com/docs/SAP_HANA_PLATFORM/bed8c14f9f024763b0777aa72b5436f6/7d28bc8c4e54413caf2716731494da88.html?version=2.0.03

[20] IBM, "What is high-performance computing (HPC)?" *IBM Homepage*, 2022, retrieved: September 2022. [Online]. Available: https://www.ibm.com/topics/hpc

[21] S. Ghiasvand, "Toward resilience in high performance computing: A prototype to analyze and predict system behavior," Ph.D. dissertation, Dresden University of Technology, Germany, 2020, retrieved: September 2022. [Online]. Available: https://tud.qucosa.de/api/qucosa%3A72457/attachment/ATT-0/

[22] S. He *et al.*, "A survey on automated log analysis for reliability engineering," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–37, 2021, retrieved: September 2022. [Online]. Available: https://arxiv.org/pdf/2009.07237.pdf

[23] M. R. Ghorab, J. Leveling, D. Zhou, G. J. Jones, and V. Wade, "Identifying common user behaviour in multilingual search logs," in *Workshop of the cross-language evaluation forum for European languages*. Springer, 2009, pp. 518–525, retrieved: September 2022.

 ISBN: 978-1-61208-997-3

[Online]. Available: https://doras.dcu.ie/16037/1/Identifying_Common_User_Behaviour_in.pdf

[24] G. M. D. Nunzio, J. Leveling, and T. Mandl, "Multilingual log analysis: Logclef," in *European Conference on Information Retrieval*. Springer, 2011, pp. 675–678, retrieved: September 2022. [Online]. Available: https://doras.dcu.ie/16438/1/Multilingual_Log_Analysis_LogCLEF.pdf

[25] L. Hatton, "Characterising the diagnosis of software failure," *IEEE Software*, vol. 18, no. 4, pp. 34–39, 2001, retrieved: September 2022. [Online]. Available: https://www.leshatton.org/Documents/Diag_IS799.pdf

[26] J.-C. Laprie, "Dependability: Basic concepts and terminology," in *Dependability: Basic Concepts and Terminology*. Springer, 1992, pp. 3–245.

[27] D. V. L. Bartlett, "The failure phenomenon: a critique," *International Journal of Performability Engineering*, vol. 6, no. 2, p. 181, 2010, retrieved: September 2022. [Online]. Available: http://www.ijpe-online.com/EN/article/downloadArticleFile.do?attachType=PDF&id=3362

[28] B. Randell and M. Koutny, "Failures: Their definition, modelling and analysis," in *International Colloquium on Theoretical Aspects of Computing*. Springer, 2007, pp. 260–274.

[29] S. Dalal and R. S. Chhillar, "Case studies of most common and severe types of software system failure," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 8, 2012, retrieved: September 2022. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1073.5008&rep=rep1&type=pdf

[30] "E-Definitions," *ComputerHope*, 2021, retrieved: September 2022. [Online]. Available: https://www.computerhope.com/jargon/e/error.htm

[31] "CNSSI 4009-2015 [Superseded] from NIST SP 800-34 Rev. 1 - Adapted," 2015, retrieved: September 2022. [Online]. Available: https://csrc.nist.gov/glossary/term/disruption

[32] B. Marr, "The Key Definitions Of Artificial Intelligence (AI) That Explain Its Importance," *Enterprise Tech*, 2018, retrieved: September 2022. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/02/14/the-key-definitions-of-artificial-intelligence-ai-that-explain-its-importance/?sh=7a87d4734f5d

[33] J. Frankenfield, "Data Analytics," *Investopedia*, 2022, retrieved: September 2022. [Online]. Available: https://www.investopedia.com/terms/d/data-analytics.asp

[34] A. Hayes, "Trend Analysis," *Investopedia*, 2021, retrieved: September 2022. [Online]. Available: https://www.investopedia.com/terms/t/trendanalysis.asp

[35] "Machine Learning," *IBM Cloud Education*, 2020, retrieved: September 2022. [Online]. Available: https://www.ibm.com/cloud/learn/machine-learning

[36] F. Gabel, "Artificial Intelligence for Games: Seminar," , 2019, retrieved: September 2022. [Online]. Available: https://hci.iwr.uni-heidelberg.de/system/files/private/downloads/636026949/report_frank_gabel.pdf

[37] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[38] ——, "Outlier detection: A survey," *ACM Computing Surveys*, vol. 14, p. 15, 2007.

[39] ScienceDirect, "Fault Prevention," *Elsevier*, 2022, retrieved: September 2022. [Online]. Available: https://www.sciencedirect.com/topics/computer-science/fault-prevention

 ISBN: 978-1-61208-997-3