We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

20

Dependability Evaluation Based on System Monitoring

Janusz Sosnowski and Marcin Król
Institute of Computer Science, Warsaw University of Technology
Poland

1. Introduction

Recently dependability, maintainability and performance are becoming challenging issues for system designers and users. This results from the increasing complexity of hardware and software. In consequence these issues triggered various measurement-based studies in the literature, in particular they relate to the detection or prediction of critical situations. Most published results are focused on restricted and fragmented problems encountered in the systems or applications considered by the authors e.g. (Daniel et al., 2008; Ganapathi & Patterson, 2005: Hoffmann et al., 2007; Li et al., 2006; Makanju et al. 2008). In contemporary computer systems various monitoring mechanisms are provided, usually they relate to event and performance monitoring (John & Eckhout, 2006; Simache & Kaaniche 2001; Stearley, 2004; Zhang 2008; Ye, 2008). Such monitoring can generate a huge quantity of various data. An important issue is the selection and exploration of this data, to characterise the system operation. This is a non-trivial task even for experienced system administrators and analysts. Hence, it needs further investigations in the following aspects: system observation techniques, selecting the most sensitive observation parameters, creating the model of normal and abnormal (dangerous) behaviour of the system to facilitate problem identification and applying appropriate reactions.

In the literature most papers concentrate on finding some characteristic patterns in log files related to well defined critical problems encountered in the considered systems e.g. leading to system crash (Kalyanakrishman et al., 1999; Sahoo et al., 2004; Xu et al., 1999). Various performance parameters have been monitored to predict specified network or processor bottlenecks (Cherkasova et al., 2008; Li et al., 2006; Reinders, 2007; Simache & Kaaniche, 2001), to detect attacks (Ye, 2008) or asses system dependability (Heath et al., 2002; Malek, 2008). Some statistical or data mining models have been developed for specific problems, however they are hardly applicable or irrelevant to other systems (Bertino et al., 1998; Hoffman et al., 2007, Lim et al., 2008). Hence, further studies covering various systems are still required to get better knowledge of monitoring capabilities and limitations.

We have faced many dependability and maintainability problems in computer systems used by students and scientists within the university for didactic and research purposes. Moreover, the load of the systems changes in time or place, hardware and software are updated or tuned, various maintenance and administrative activities occur sporadically, etc.

Hence, some operational problems or configuration inconsistencies arise. Such systems create a good basis for studying monitoring techniques. We have also some experience with other commercial systems handling many customers with fluctuating usage profiles. Long-term observations of these systems allowed us to improve monitoring techniques and dependability. For this purpose we have developed some special software modules, collected a lot of data and performed various analyses.

The paper outlines the main features of possible measurements (related to system operation) and the scope of collected data. On the basis of this survey we formulate problems of selecting and processing the collected data in relevance to dependability issues. We concentrate on software implemented monitoring systems, which provide combined exploration of event logs and performance counters. As opposed to other approaches the developed monitoring systems are interactive and adjusted to appearing problems. Moreover, we deal with a wider scope of observations, so we rely on many data sources simultaneously (e.g. event logs, performance logs, and exceptions). We have combined two approaches: identifying normal operation features and exploring long term trends (neglected in the literature); detecting various abnormalities. In both approaches we take into account correlation with environment and configuration changes. The paper describes this in relevance to two monitoring techniques based on various event logs and performance data. The presented considerations are illustrated with practical monitoring results. They relate to long term observations of many workstations and servers.

In section 2 we give an outline of event logs collected in computer systems. They are illustrated with some statistical data derived from long term observations of computers in didactic laboratories, some comments on data exploration are also included. Section 3 describes performance objects and related performance variables, which are usually monitored. The capabilities and problems with performance monitoring are presented in relevance to results from real systems. Final conclusions are given in section 4.

2. Event logs in computer systems

2.1 Event specifications and statistics

Computer systems are instrumented to provide various logs on their operation. These logs comprise huge amounts of data describing the status of system components, operational changes related to initiation or termination of services, configuration modifications, execution errors, etc. In Windows various events are stored in one of the three log files:

- *security log* comprises events related to system security and auditing processes,
- *system log* is used primarily to store diagnostic messages, abnormal conditions, events generated by system components (e.g. services, drivers),
- *application event log* reports errors that occur during the application execution (e.g. failing to allocate memory, aborting the transfer of a file, etc.).

Each event log record comprises the following fields:

- *event specification* - specifies 5 event types related to event severity level: error, warning, information, success audit, failure audit; this is supplemented with the event category, ID, date and time,
- *event source* – name of the user and the computer that generated the event,
- *description* – event details.

The list of possible events in Windows systems exceeds 10000 (Sosnowsk & Poleszak, 2006). In Unix and Linux systems over 10 sources of events and more priority levels are distinguished (*Syslog*). During normal operation of workstations or servers a large amount of events is registered in the logs. Hence, we have developed a special software system *LogMon* which collects data logs from specified computers within LAN and performs predefined processing to identify critical, abnormal and other situations (e.g. unavailability, warnings). *LogMon* co-operates with standard services (e.g. *Eventlog*) and provides some statistical and data exploration techniques. The performed analysis can be targeted at individual computers or specified computer subsets to find various correlations, etc.
The event files can be filtered preliminary according to specified rules related to event identifier, source, type, system user, computer identifier, date and time (specified intervals by two points in time, specified month, week day, etc.). Complex multi step filtering is also possible, we can combine filtered files in one file, etc. Typical statistics relate to:

1) *Event counting* – the distribution (e.g. in decreasing order) of the number of registered events;
2) *Time between events* - time distribution between events of the same or different types;
3) *Event occurrence distribution* – statistics of the number of the selected event type in relevance to months, weeks, days or hours of the day;
4) *Event frequency profile* – the frequency of a selected event in the considered time period.

The calculated statistics are visualised in graphical forms, including a scatter plot where x axis is the time and y axis represents different event categories, or system components, etc. Such visualisations are useful to interpret the collected data, e.g. identification of significant patterns. The collected events can also be presented according to some ranking features e.g. frequency of appearance, entropy, etc.
The developed tool *LogMon* collects data from logs of many computers via internet. It provides many possibilities of filtering, searching specified event sequences, and visualising results. The log analysis can be targeted at different problems e.g. identifying critical situations, evaluating system availability, activity, system load, power problems. This process needs some knowledge of log specificity and experience with the used tool. We illustrate this in the subsequent section.

2.2 Illustrative results

While analysing the registered events we should identify the system start up and shutdown. When the Windows system is booted, event 6009 is logged and then it is followed by event 6005, which corresponds to *EventLog* service start-up. The termination of *EventLog* service registers event 6006. Event *6006* should be the last one registered in the system log after shutting down the system (clean shutdown). Nevertheless we observed some unexpected events registered after 6006 (anomalous situation). The event 6008 is recorded when a dirty shutdown ("blue screen") occurs. The description part of this event comprises system time stamp (date and time). It may happen that the system cannot record 6006 or 6008 event, however 6009 and 6005 events are recorded. This complicates identification of system restarts, etc. After the system restart (e.g. in consequence of power supply outage) caused by event 6008 other registered events may give more details.
Within the events, which are correlated with system restarts, we can distinguish 4 groups: system and application updates, errors in applications and system services, hardware errors, unidentified restarts. Update events relate to restarts forced by installing new programs,

system updating or recovery of the previous version (with deletion of the updated ones). Typically they are initiated by: *Automatic Updates, NtService Pack, MsiInstaler, WindowsMedia, Print*. The events specify types of updates, information if it has been successfully accomplished or not, etc. For example for one of the computers the distribution of antivirus data base updates was as follows: for 270 registered events of this type (4570; *McUpdate*) 62 appeared in time period less than 1 day, 34 in the period from 1 to 2 days, etc. The analysis covered the log of 668 days. For some computers this frequency was sporadically disturbed – due to some configuration problems. Updates of different programs were performed successfully in over 80% cases, however for some computers non-successful updates were reported. The deeper analysis proved configuration inconsistency and network problems. Not successful clock synchronisation appeared on average in 10% cases.

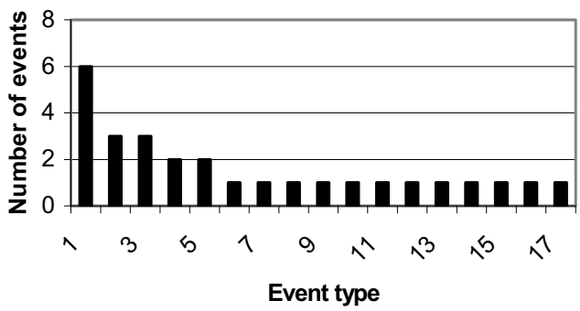


Fig. 1. Distribution of event types before restarts

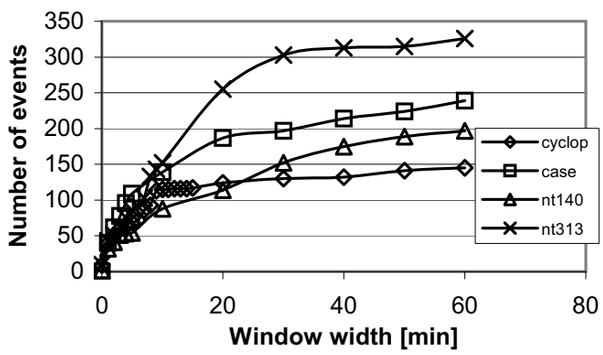


Fig. 2. The number of events within the specified time window

Looking for the sources of restarts we can analyse the distribution of events registered in the specified window before the event sequence related to reboot (6006, 6009, 6005) This is illustrated in fig. 1. The x-axis specifies different events In particular: x=1 relates to event *2013Srv* (the disk is almost full, you may need to delete some files); x=2 - event *54w32time* (the Windows Time Service was not able to find a Domain Controller, a time and date update was not possible); x=13 – event *21automatic program updates*; x=15 – event 26 *application error,* etc. Such graph facilitates to identify the most frequent sources of restarts. Complete distribution

of all registered events in a decreasing order of occurrence is also useful to identify other problems. Typically 90% of registered events related to only 36 different event types (from the total list of over 10000 possible events).

To find sources of some event A it is useful to check events before this event within a specified time window Δ. For this purpose *LogMon* provides the capability of finding such statistics for a specified window Δ. Fig. 2 shows the number of registered events for 4 servers in function of the time window width Δ before restarts. We can observe some kind of saturation for 10<Δ<30 minutes, depending upon the system. Basing on the registered events we can identify restarts. For the considered servers (fig. 2) we have identified 24-60 restarts (on average 5 events per restart in the window).

The developed system *LogMon* provides various data exploration capabilities. In particular it can identify reasons of restarts and dirty restarts. In the case of restarts we have defined some regular expressions describing events most probably related to specified situations e.g. program update restarts. Tab. 1 shows restart statistics for 4 laboratories (L1-L4) each comprising 17 workstations and 3 servers (S1-S3). It gives the percent of restarts caused by program updates, application errors and hardware errors. In some cases the restart source is ambiguous - related to more than one source (mostly a program update and some other source). Quite significant percentage of detected restarts (unknown cause) did not comprise additional events facilitating their identification. They relate to power downs and restarts initiated by the user in response to some messages appearing on the screen, some of them can be identified from the application log. The table comprises the restart frequency (RF) expressed in the number of restarts per day (per single computer). Relatively low values of RF for two servers (S2 and S3) result from the stable profile of their usage.

	L1	L2	L3	L4	S1	S2	S3
RF	0.20	0.21	0.19	0.18	0.19	0.04	0.05
updates	20.2%	16.5%	22.6%	24.3%	32.5%	53.8%	3.9%
applic.	19.2%	29.7%	12.0%	29.6%	10.4%	15.4%	11.8%
hardware	1.2%	0.5%	2.0%	0.4%	9.1%	0%	0%
ambig.	1.4%	5.6%	5.0%	5.4%	6.5%	26.9%	1.3%
unknown	59.4%	53.4%	63.4%	45.7%	48.1%	30.8%	84.2%

Table 1. Restart statistics for workstations and servers

Special attention is needed to dirty restarts. Dirty restarts mostly relate to such events as 6008, 1000, 1001 *save dump*. Pressing RESET button also causes dirty restart. At the system level power down is treated in the same way as fast switching off the computer. In logs with power down closing event 6006 was missing. Analysing events in the window before the dirty restart we observed small number of events. This relates mostly to the situation with no possibility of recording the event due to the restart problem. In a period of 100 days we have identified typically 2-5 dirty restarts per computer. However, for a few computers this was in the range of 20-50 (old computers). In the case of servers practically the dirty restarts were caused by hardware errors. In the case of workstations dirty restarts were caused mostly by application errors, which caused system hang-ups or operational instability (due to developing and testing various pogrom modules). Moreover, many student projects comprised critical errors leading to restarts.

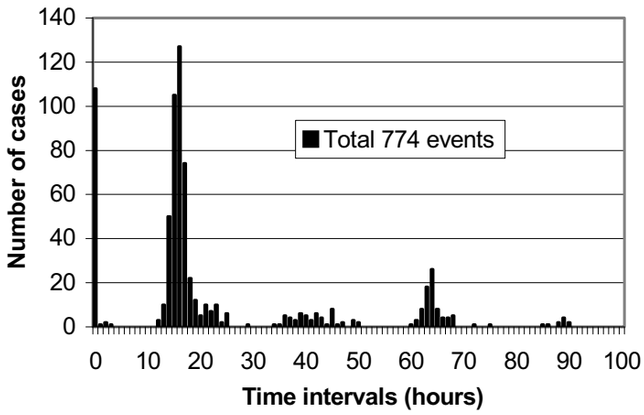


Fig. 3. Distribution of non activity periods in a workstation

Fig. 3 shows distribution of time periods between event A and B, which correspond to closing and starting the system. The x-axis of the plot has the granularity of single hours. The first bar (108 cases) relates to short time intervals (less than 1 hour) and it corresponds to short operation breaks related to restarts. The next group of higher bars relates to the periods of 15-16 hours, corresponding to switching off the computer for the evening and night periods. Subsequent groups of bars relate to longer non activity periods e.g. weekends, holidays, etc. Power supply problems can be directly identified by checking the time between events 3230:UPS (notification of power down and supply delivered from the batteries) and 3234:UPS (power recovery) or 3231:UPS (power switched off in the system) generated by UPS power supply. For an illustration we give some power statistics related to 2 servers SA and SB. In particular we specify the number of power down events per month for 9 subsequent months (October- June):

SA: 1; 1; 3; 1; 2; 2; 0; 0; 1 (total 9 events) and SB: 1; 9; 8; 7; 12; 10; 12; 1; 0 (total 60 events)

For all power down events the servers were supplied from UPS batteries, due to short power outages (power outages were tolerated by UPS). Power outages longer than 17 minutes and 150 sec result in SA and SB server switching off, respectively. The distribution of the duration of power outages was as follows: for server SA - (6 events) < 1 minute, 2 minutes ≤ (2 events) < 7 minutes and 1 event with 17 minutes duration; for server SB - (13 events) < 1s, 1s ≤ (33 events) < 2s, 2s ≤ (5 events) <3 s, 3s ≤ (4 events) < 4s and 5s ≤ (4 events) ≤ 12s. Computers without UPS crashed and needed restarting. Some crashes appeared simultaneously in several computers (common power failure). In 3 cases the power downs signalled by UPSs of SA and SB servers were correlated (caused by the same power network outage), however the attributed timestamps differed by about 20 minutes, due to the lack of the clock synchronisation in both servers (such anomalies need identification).
An important issue is to evaluate the behaviour of various used programs within longer periods and correlate it with system upgrades, reconfiguration, load (number of users, processor stress). For an illustration table 2 shows the frequency distribution of registered program errors per day for two workstation (WS1 and WS2) within the one-year period. WS1 is less reliable due to higher number of used programs.

Computer	Number of errors per day							
	0	1	2	3	4	5	6	6
WS1	68%	10%	6%	3%	3%	2%	o%	1%
WS2	82%	5%	1%	2%	1%	2%	1%	0%

Table 2. Distribution of program errors for workstation WS1 and WS2

Events related directly to hardware faults appear before restarts. The most frequent relate to memory media, network cards, printers and other I/O devices. For example event 26 with description that the system could not write or read data from a specified file, device, etc. Other examples are: faulty block of CD ROM, timeout situation, IP address conflict, failure to load specified drivers, application errors, etc. It is worth noting that many events do not comprise descriptions, on the other hand some descriptions are ambiguous. Many faults can be identified from sequences of events. For this purpose some knowledge database can be systematically developed taking into account the gained experience from the system exploitation and maintenance.

2.3 Exploring data in event logs

Analysing logs is the basis for automatic system management and helpful in assuring high dependability. The registered reports may be related to different formats, the text messages are usually relatively short, contain a free format description of events (using a large size vocabulary), and quite often are ambiguous. Hence, data mining is not trivial and needs many preliminary studies to identify specificity and abnormal behaviour of the monitored systems. In this process some categorisation of text messages into a set of common classes over various system components is required. Moreover, an important issue is to visualise various statistics, temporal dependencies, etc.

Simple data mining can be targeted at discovering frequent and some specific well defined patterns (Lim et al., 2008; Makanju et al., 2008; Peng et al., 2005; Razavi & Kontogiannis, 2008; Vaarandi, 2008). In more advanced analysis of log reports we should take into account not only the individual messages but also their temporal dependencies, which can provide supplementary context information. For example a massage on starting a program update may be followed by some errors due to inconsistency in the system configuration. Having transformed messages in some concise categorised form simplifies further data processing and finding characteristic patterns. Unfortunately, different systems use different log formats, etc. Hence, data collection and analysis has to be tuned to these systems. This is sufficient for individual system monitoring. In practice we are also interested in general properties of many systems, so some specification of similarities has to be defined to identify common characteristics, etc.

The log pre-processing may involve visualisation of event types or categories in relation to their appearance (time stamps). From such plot it is easy to identify some general properties e.g. the fact that event A usually happens after event B, the time distance between such events (it can be deterministic or random). An event may appear with some periodicity (e.g. antivirus updates, system heartbeat) or randomly. Some events may form a loop in a circular pattern or an event chain (e.g. related to a problem progress in predictable way). An event may appear simultaneously with other events. Various temporal relationships can be represented by appropriate graphs (Peng et al., 2007). Looking for temporal dependencies we analyse the distribution of time distance between events or compare the unconditional

probability of the waiting time for some event with a conditional probability in relevence to some other event. Various event patterns may signal system problems or confirm its health (e.g. heartbeats, successful program updates). Their interpretation can be simplified by correlating them with performance properties (section 3).
An important issue in tracing event logs is appropriate system configuration If we are interested in monitoring system activity it is important to activate writing into logs supplementary information on user logins, file openings, processor usage, etc. It is also important to assure completeness of collected data. Some danger arises if logs are read periodically, so overwriting may happen. Hence, it is reasonable to collect this data systematically and storing it in a separate server. Complete information on all computers simplifies finding various correlations.

3. Performance Monitoring

3.1 Performance objects and variables

In most computer systems various data on performance can be collected in appropriate counters (e.g. provided by Windows, Linux) and according to some sampling policy (e.g. in 1-minute periods) (John & Eckhout, 2006; Reinders, 2007). These counters are correlated with performance objects such as processor, physical memory, cache, physical or logical discs, network interfaces, server of service programs (e.g. web services), I/O devices, etc. For each object many counters (variables) are defined characterising its operational state, usage, activities, abnormal behaviour, performance properties, etc. Special counters related to developed applications can also be added. These counters provide data useful for evaluating system dependability, predicting threats to undertake appropriate corrective actions, etc. The list of counters, which can be configured, is very long. For an illustration we describe some representative counters related to Windows systems.
Processor Time is the percentage of elapsed time that the processor spends to execute a non-idle thread. This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. *User Time* and *Privileged Time* relate to the percentage of elapsed time the processor spends in the user and in privileged mode, respectively. *Processor Queue Length* is the number of ready threads in the processor queue. *Processes* is the number of processes at the time of data collection. Similarly are counted threads, events, semaphores, etc. *Context Switches/sec* is the combined rate at which all processors on the computer are switched from one thread to another e.g. when a running thread voluntarily relinquishes the processor (is pre-empted by a higher priority ready thread), or switches between user-and privileged (kernel) mode to use an executive or subsystem service.
Interrupts/sec is the average rate, in incidents per second, at which the processor received and serviced hardware interrupts. It does not include deferred procedure calls (DPCs), which are counted separately. This value is an indirect indicator of the activity of devices that generate interrupts, such as the system clock, the mouse, disk drivers, network interface cards, and other peripheral devices. These devices normally interrupt the processor when they have completed a task or require attention. The system clock typically interrupts the processor every 10 milliseconds, creating a background of interrupt activity. This counter displays the difference between the values observed in the last two samples. *Interrupt Time*

is the time the processor spends receiving and servicing hardware interrupts during sample intervals.
System Up Time is the elapsed time (in seconds) that the computer has been running since it was last started till the current time. *C1 Time* is the percentage of time the processor spends in the C1 low-power idle state (enables the processor to maintain its entire context and quickly return to the running state), similar times are measured for C2 (a lower power and higher exit latency state than C1, it maintains the context of system cache) and C3 (a lower power and higher exit latency state than C2, is unable to maintain the coherency of its caches) states. There are also counters related to transitions to these states.
Available Bytes is the amount of physical memory, in bytes, available to processes running on the computer. It is calculated by adding the amount of space on the *Zeroed, Free,* and *Standby* memory lists. *Free* memory is ready for use; *Zeroed* memory consists of pages of memory filled with zeros to prevent subsequent processes from seeing data used by a previous process; *Standby* memory is memory that has been removed from a process working set (its physical memory) on route to disk, but is still available to be recalled. This counter displays the last observed value.
Free Space is the percentage of total usable space on the selected logical disk drive that was free. *Avg. Disk Bytes/Read* is the average number of bytes transferred from the disk during read operations, similar counter on write operations is available also.
Page Faults/sec is the average number of pages faulted per second (a referenced page in virtual memory is not available in the working area). Hard faults require disk access and soft faults cover faulted pages found elsewhere in physical memory. Most processors can handle large numbers of soft faults without significant consequences. However, hard faults, which require disk access, can cause significant delays. Similarly *Cache Faults/sec* is the rate at which faults occur when a page sought in the file system cache is not found and must be retrieved from elsewhere in memory or disk.
Page Reads/sec is the rate at which the disk was read (the number of read operations, without regard to the number of pages retrieved in each operation) to resolve hard page faults. *Pages Output/sec* is the rate at which pages are written to disk to free up space in physical memory. A high rate of *Pages Output* might indicate a memory shortage. *Pool Paged Failures* is the number of times allocations from paged pool have failed. It indicates that the computer's physical memory or paging file are too small.
File Read Operations/sec is the combined rate of file system read requests to all devices on the computer, including requests to read from the file system cache. This counter displays the difference between the values observed in the last two samples, divided by the duration of the sample interval. *File Control Operations/sec* is the combined rate of file system operations that are neither reads nor writes, such as file system control requests and requests for information about device characteristics or status. *Split IO/Sec* reports the rate at which I/Os to the disk were split into multiple I/Os. It may result from requesting data of a size that is too large to fit into a single I/O or that the disk is fragmented.
Server performance counters give: the number of bytes the server has received (or sent) from the network (indicates the server load); the number of sessions that have been closed due to unexpected error conditions or sessions that have reached the autodisconnect timeout and have been disconnected normally; failed logon attempts to the server (password guessing programs are being used to crack the security); the number of sessions that have been forced to logoff (due to logon time constraints); the number of sessions that have terminated

normally (this allows to find percentage of the sessions time outs or errors). Other counters provide some statistics on file operations such as: the number of times accesses to files opened successfully were denied (improper access authorisation, etc.), the number of failed file opens (attempting to access files not properly protected), the number of files currently opened in the server, the number of searches for files currently active, the number of sessions currently active in the server (indicates current server activity).

There are many counters characterising network traffic or TCP/IP protocol activity. Here are given some examples. *Bytes Received/sec* is the rate at which bytes are received over each network adapter, including framing characters. *Current Bandwidth* is an estimate of the current bandwidth of the network interface in bits per second. *Packets Received Errors* is the number of inbound packets that contained errors preventing delivery to a higher-layer protocol. *Packets Received Discarded* is the number of inbound packets that were discarded even though no errors had been detected (e.g. to free up buffer space). *Packets Received Unknown* is the number of packets received through the interface that were discarded because of an unknown or unsupported protocol. *Output Queue Length* is the length of the output packet queue, if this is longer than two, there are delays and the bottleneck should be found and eliminated. *Connection Failures* is the number of times TCP connections have made a direct transition to the CLOSED state from the SYN-SENT or SYN-RCVD state, and to the LISTEN state from the SYN-RCVD state.

There are also counters related to I/O devices e.g. for printers they count current number of jobs in a print queue, number of references (open handles) to this printer, peak number of references, current or maximal number of spooling jobs in a print queue. Accumulated statistics comprise data since the last restart e.g. number of out of paper errors, not ready errors and job errors in a print queue.

Resuming we can state that the number of possible performance variables is quite big and monitoring all of them is too expensive due to the additional load to the system processors and memory. Hence, an important issue is to select those variables which can provide the most useful information. This depends upon the goals of monitoring, the sensitivity of variables to the monitored properties of the system, the system usage profile, etc. To deal with this problem some preliminary studies of the system behaviour are needed. They facilitate tuning the monitoring tasks to the current needs and system specificity. We outline this problem in the next section.

3.2 Performance monitoring goals

Depending upon the goal of monitoring we have to select and configure appropriate counters within the objects of interest, to evaluate how well they are performing. Too large number of counters results in some additional load to the system and more complex data analysis. Hence, an important issue is to check which counters are most sensitive to the monitored problems. We have performed such studies in relevance to hardware and software failures as well as configuration or maintenance inconstancies, effectiveness of some services, etc. Moreover, the applications can also use counter data to determine how much system resources to consume. For example, to determine how many data to transfer without competing for network bandwidth with other network traffic. The application could adjust its transfer rate as the bandwidth usage from other network traffic increases or decreases. Having specified performance counter thresholds we can generate alert

notifications, query performance data, create event tracing sessions, capture a computer's configuration, and trace the API calls in some of the Win32 system DLLs.
Most authors concentrate on well-defined critical problems e.g. cyberattacks or system availability. We have extended the scope of analysis to checking the normality of system operations e.g. periodicity of backups, program updates, acceptable level of signalled errors (e.g. rejected packets) and to detect abnormalities which may result in future problems, this relates mostly to long term observations and detecting dangerous trends e.g. decreasing of free memory. We correlate performance counters with event logs as well as with changes in configurations, system load, temporal disturbances in the operational environment (system maintenance and updates).
Some performance measures are directly used to balance system loads, etc. To assure this we have to analyse short term and long-term trends, correlate them with working hours, weekends, summer months (seasonal system behaviour), user activity profiles, etc. The considered systems were specific due to frequent configuration changes, many users with different profiles (students and different courses, projects, used programming environment, etc.) or servicing thousands of customers with random activities, influenced by various events (e.g. dynamic changes of the stock market).
Some problems are relatively easy to identify e.g. decreasing free memory in relevance to systematically increasing number of users and their higher engagement in more complex calculation problems, bigger data bases, etc. However, new not known problems are not evident and need deeper data multidimensional exploration. For example a higher rate of application warnings in the log was correlated with an installation of a new version of the operating system and the increased number of users. This related to configuration inconsistencies, which were alleviated later on.
Analysing the performance variables we can look at their instantaneous values, statistical properties, correlation with other variables or events. These statistics may relate to specified time periods. Moreover, we can target the analysis at averaged variable values (within specified periods, etc.) or analyse spikes, their frequencies, time distribution, periodicity, etc. All this depends upon the monitoring goal. For example in detection of cyber attacks we can try to find characteristic statistical deviations caused by the attack as compared with normal workload. Interesting studies have been presented in (Ye, 2008) for Windows systems. The authors give statistical properties of various performance variables related to different objects for 10 known cyber attacks. Analysing these results we have checked the observability properties of these attacks.

Property	Objects	Variables	Attacks	Sensitivity
M+	3-16 (7.7)	10-362 (105)	1-9 (7.7)	77/100
M-	3-11 (5.9)	17-182 (60)	1-10 (5.6)	59/180
DUL	1-4 (2.5)	1-33 (10.3)	1-8 (3.1)	25/80
DUR	3-9 (5.7)	9-52 (27.8)	1-10 (3.9)	58/110
DMM	3-9 (5.7)	24-52 (43.3)	2-9 (5.0)	57/120

Table 3. The impact of attacks on performance parameters

Tab. 3 shows minimal - maximal (average) numbers of monitored performance objects and related variables (counters) which reveal characteristic statistical properties during attacks. They related to an increase (M+) or decrease (M-) in the mean value, unimodal left skewed

(DUL), unimodal right skewed (DUR) and multimodal (DMM) distribution properties as compared with normal workload statistics. The 4-th table column gives the distribution (minimal, maximal and average) of the number of attacks affected by the considered objects (over each object class). The last column specifies the number of nonzero entries in the matrix correlating objects and attacks. Each entry in this matrix gives the number of object variables affecting (by specified statistical property) the appropriate attack. This gives some view on attack sensitivity (the number of all entries in the matrices is given after / character). For DUL, DUR and DMM statistics we observe lower number of affected objects and variables as compared with M+ and M-. For each distribution change we have found that *Process* object affects the most of attacks 7-10 (within all 10 attacks). Moreover, this object involves the biggest number of affected variables per single attack: up to 18, 21 and 23, for DUL, DUR and DMM distribution change, respectively. Some objects show maximal number of affected variables by specific attacks (dominating). Such sensitivity analysis allows the designer to minimise the number of monitored variables and assure good detection accuracy. For other problems we have to trace different properties, this is illustrated in the sequel.

3.3 Illustrative results

To give a better view on performance monitoring we present some illustrative results related to monitoring selected system objects and performance variables. The dynamic properties of these variables depend upon active applications, system load, environment interactions, etc. Hence, their behaviour in time can be very diversified resulting in different shapes and characteristics of related time plots. This creates some challenges for data exploration.

In general we can be interested in short term or long term monitoring results. In the first case we collect many samples which assure high accuracy. The results can be presented graphically with specified average (horizontal line), minimal and maximal values (vertical lines) for each sample. This is illustrated in fig 4a and 4b, which give the number of disc writes per second (y-axis covers the range 0-100 operations/s partitioned in 10 segments) for the system with no active application and for an application displaying a film of about 30 minutes (from a file in HDTV standard - 1280x720 pixels). The samples were collected every second. It is worth noting low activity of disc writes, nevertheless even in no active system there is some background activity related to operational system and Internet tasks. The both plots differ in time and amplitude distribution of spikes. Bigger difference was observed for processor usage (0% vs 24%) and disc read operations/s (no disc reads with 8 short spikes of 30 operations/s vs continuous average activity of 15 operations/s with many additional smal spikes). The number of disc operations is much higher for disc defragmentation (on average 379 control operations per sec). Short-term observations are useful to find application properties, identify their disturbances etc. It is worth noting that the behaviour of performance variables may differ upon applications not only in the average values of analysed parameters (during the application run) but also in time (plot shapes). Quite often we observe some spikes in time plots of variables, their frequency and amplitudes may also characterise the applications (compare. fig. 4).

Long term observations give a view on general trends in the system. We illustrate this with some results in fig. 5-7. Fig. 5 shows some increase (from 20000 to over 100000) of transmitted bytes on the network in 6-month period. This resulted from adding new users.

Fig. 6 presents the number of created connections with a www server, the middle pulse (100-350 connections) corresponds to day hours 8.00-17.00, the negative pulse at the end of the plot corresponds to the switch problem on the next day (time period 8.00-10.00). Fig. 7 illustrates the number of connections related to 13 subsequent days. It is almost equal for all working days (a little bit over 200), much lower for Saturdays (below 100) and close to 0 for Sundays.

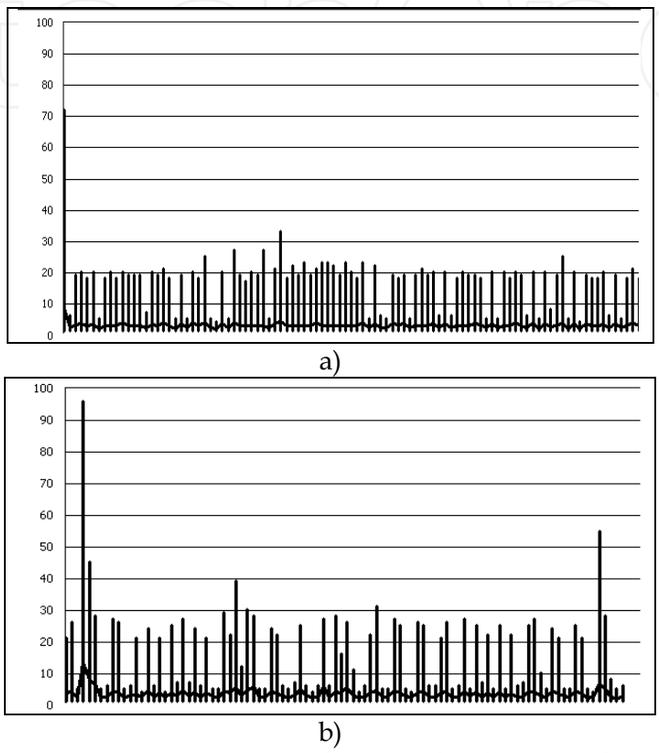


Fig. 4. Disc writes operations: a) no active applications, b) playing a film.

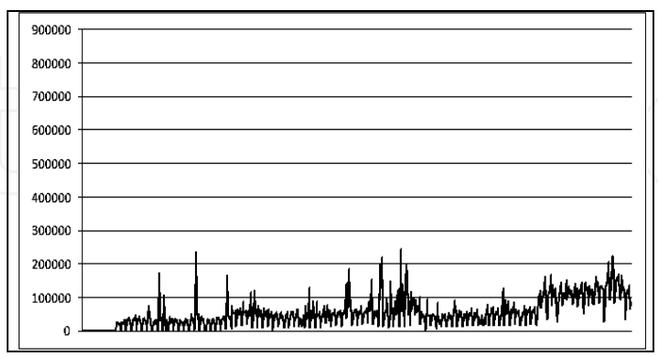


Fig. 5. Sent out bytes per second (half year profile)

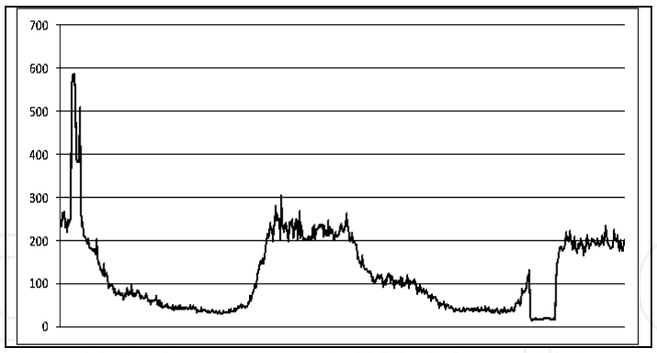


Fig. 6. The number of established connections in TCP (3 day profile)

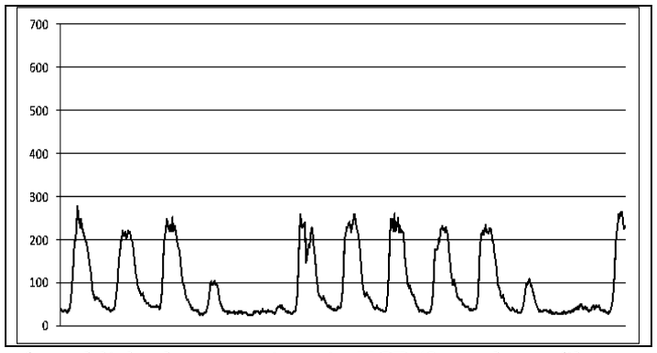


Fig. 7. The number of established connections in TCP (2 week profile)

Fig. 8 shows the number of active processes on Unix server *mion*, which handles Emails of students (about 3500 students within the Faculty of Electronics of our University) in the period 1st June till 31st December. The y-axis covers the range 0-25000 processes (partitioned into 5 equal segments – each 5500). The plot shows increasing trend of active processes. However on 12 September the system reconfiguration and restarting resulted in deleting many zombie and other not useful processes.

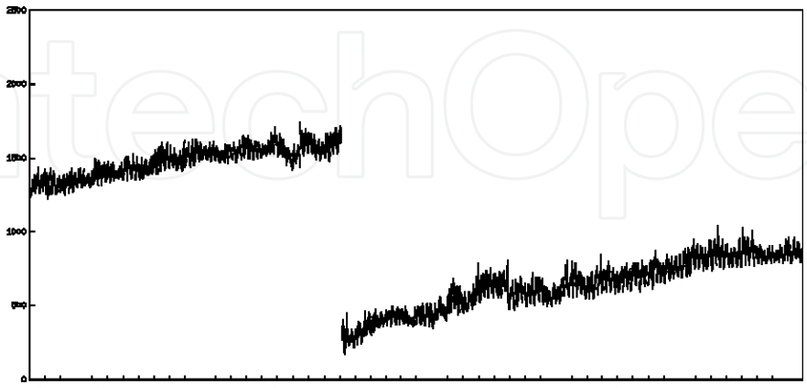


Fig. 8. The number of processes in the communication server (*mion*)

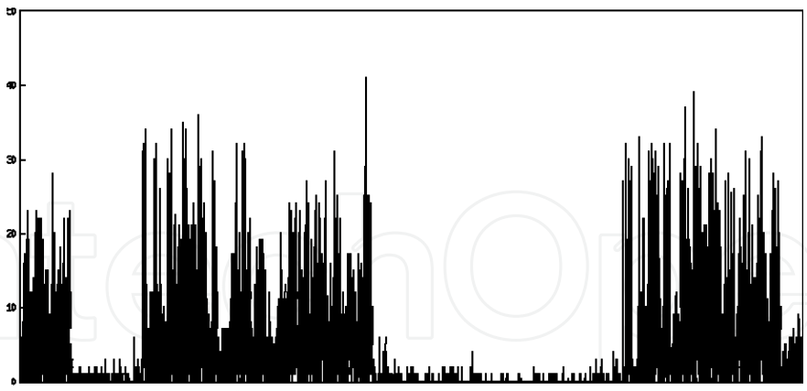


Fig. 9. The number of processes in the data processing server (*ikar*)

Fig. 9 shows the number of logged users to Unix server *ikar*, which handles programming laboratory with 16 workstations. The time scale on the x-axis covers the period of 12 months. The y-axis covers the range 0-50 users portioned in 5 equal segments (each 10 users). The first period of low activity relates to the winter vacation (February) and the second longer one corresponds to summer vacation (3.5 months).

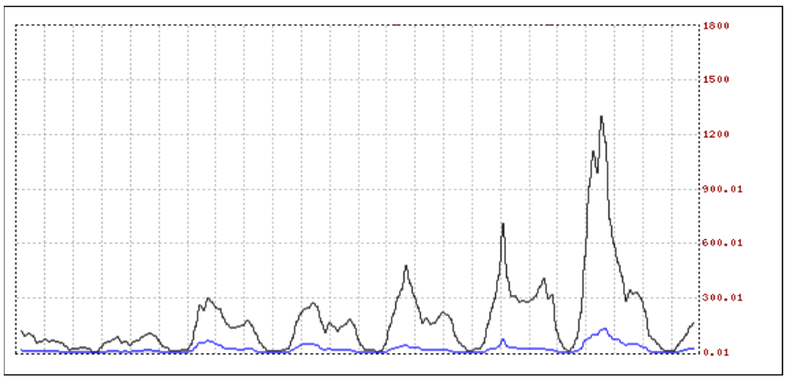


Fig. 10. Distribution of incoming session requests

Interesting observations were made for a farm of 16 servers providing some web services to many thousands of customers. The system uses quite sophisticated load balancing algorithm. Fig. 10 shows one-week plot of the total number of sessions (lower line relates to the number of sessions created within the last minute) handled by the farm within the 14th server. The y-axis covers the range 0-1800 session requests partitioned into 6 segments (300 requests each). The subsequent peaks of the plot relate to 7 days. The highest peak corresponds to Friday, for the weekend low activity is visible. In tab. 4 we give the server CPU usage (UP in percents) corresponding to the highest load on each day. Moreover we give also the outgoing traffic (OT in KB/sec) on the network port of the server. For each server the farm-managing program monitors its available resources (in particular processor and memory) and attributes user requests so as to achieve balanced load of all servers adapted to their functional capabilities and current activity. There are several classes of servers with different processors and architectural features. Hence, we take into account not

only the current values of performance parameters but also architectural capabilities. In particular the same level of processor usage expressed in percents does not reflect the real available processing power, which also depends upon the processor speed, etc. Monitoring the operation of all servers confirmed the effectiveness of the used load-balancing algorithm. Moreover, it allows finding critical deviations in farm operation and identifying problems (e.g. to eliminate a faulty server and move the traffic to other servers).

	Mon.	Tue.	Wed.	Thur.	Fri.	Sat.	Sun.
OT	700	600	500	550	620	150	30
UP	22%	23%	42%	38%	45%	8%	4%

Table 4. Outgoing traffic and processor usage in 14th server of a processing farm

The presented plots confirm a large diversity in possible shapes related both to normal and erroneous operation, hence their qualification needs advanced techniques, which take into account various correlation factors. In practice it is reasonable to correlate performance variables with event logs as well as environment changes, activities of administrators (maintenance, reconfigurations), software updates, user profiles, etc.

4. Conclusion

The available system logs and possible monitoring of various performance features provide enormous amount of data on system operation. This is a very useful source of information to evaluate and improve system dependability. However, selecting this information is not a trivial problem. It is possible to monitor and collect data on various aspects using pre-programmed counters, etc. Monitoring too many variables may result in system performance loss and high memory load with collected logs. So some optimisation is required here, in particular we can select the most sensitive variables related to various dependability issues. The next problem is interpretation of the collected data. This requires gaining some experience from long-term observations and correlating them with opinions of users and administrators. This simplifies creating procedures for automatic data exploration targeted at dependability issues. Hence, it is reasonable to enhance the available system mechanisms and software modules with an integrated database and advanced visualisation, statistical and data mining procedures (provided in the presented systems).

Further research is targeted at correlating various logs from many computers, identifying typical operational profiles, system loads, finding their changes in time and developing more efficient data exploration techniques to predict as soon as possible requirements of reconfigurations, detect inconstancies or usage anomalies, etc. Having itemised specific patterns we can formulate appropriate actions. The gained experience is useful in defining event reduction rules, correlation rules (identifying events which are symptoms of specific problems), and problem avoidance rules (for some problems several stages of progress can be distinguished, early detection can prevent critical situation). We also plan to enhance the collected data from the field with logs relevant to injected faults (Sosnowski & Gawkowski 2006).

Acknowledgment

This work was supported by Ministry of Science and Higher Education grant 4297B/T02/2007/33. We express our appreciation to J. Machnicki for experiments related to fig. 7-10.

5. References

Bertino, E., Ferrari, E. & Guerrini, G. (1998). An approach to model and query event based temporal data, *Proc. of 5th Int. Workshop on Temporal Representation and Reasoning,* pp. 122-131, IEEE Comp. Society

Cherkasova, L.; Ozonar, K.; Mi, N.; Symons, J. & Smirni, E. (2008). Anomaly? application change? or workload change?, *Proc. of IEEE DSN Symposium*, pp. 452-461, IEEE Comp. Society, ISBN 1-4244-2398-9

Daniel, E.; Lal, R. & Choi, G. (1999). Warnings and errors: A measurement study of a UNIX server, *FastAbstracts* of *IEEE Int. Symp. on Fault-Tolerant Computing*, FTCS-29, www.crhc.uiuc.edu/FTCS-29/fastabs.html.

Ganapathi, A. & Patterson, D.(2005). Crash data collection: A windows case study, *Proc. of IEEE DSN Symposium*, pp.772-184, IEEE Comp. Society , ISBN 0-7695-2282-3

Heath, T; Martin, R.P. & Nguyen, T.D. (2002). Improving cluster availability using workstation validation. *Proc. ACM SIG-METRICS Conf. Measurement and Modelling of Computer Systems*, pp.217–227.

Hoffmann, G. A.; Trivedi, K.S. & Malek, M. (2007). A best practice guide to resource forecasting for computing systems, *IEEE Transactions on Reliability*, vol.56, no. 4, pp.615-628, ISNN 0018-9529

John, L.K. & Eckhout, L, (editors) (2006). *Performance evaluation and benchmarking*, CRC Taylors & Francis, ISBN10-0-8493-3622-8

Kalyanakrishman, M.; Kalbarczyk Z. & Iyer, R.K. (1999). Failure data analysis of a LAN of Windows NT based computers, *Proc. of 18th IEEE Symposium on Reliable Distributed Systems*, pp.178-188, IEEE Comp. Society

Li, M.; Wang, S. & Zhao, W. (2006). A real-time and reliable approach to detecting traffic variations at abnormally high and low rates, In *ATC 2006, LNCS 4158*, 2006, L.T. Yang et al., (Ed.) pp.541-550, Springier Verlag, ISBN 3-540-69294-0, New York.

Lim. Ch.; Singh, N. & Yainik, S. (2008). A log mining approach to failure analysis of enterprise telephony systems, *Proc. of IEEE DSN Symposium*, pp. 388-403, IEEE Comp. Society, ISBN 1-4244-2398-9

Makanju A., Brooks, S.; Zincir-Heywood, A.N.& Milin, E.E.. (2008). LogView: visulizing event log clusters, *Proc. of Annual Conf. on Privacy, Security and Trust*, pp. 99-108, ISBN: 978-0-7695-3390-2

Malek, M. (2008). Online dependability assessment through runtime monitoring and prediction, *Proc. of EDCC –7.*, pp.181, IEEE Comp. Society, ISBN 978-0-7695-3138-0

Mansouri-Somani, M. & Sloman, H. (1996). A configurable event service for distributed systems, *Proc. of IEEE 3rd Int. Conf. on Configurable Distributed Systems*, pp. 210-217, IEEE Comp. Society, ISBN 0-8186-7395-8

Peng, W.; Peng, Ch.; Li, T. & Wang, H. (2007). Event summarization for system management, *Proc. of ACM KDD'07*, pp. 1028-1032.

Peng, W.; Li, T. & Ma, S. (2005). Mining logs files for computing system management, *SIGKDD Explorations*, vol. 7, issue 1, pp. 44-51.

Razavi, A. & Kontogiannis, K. (2008). Pattern and policy driven log analysis for software monitoring, *Proc. of IEEE Int. Computer Software and Applcations Conference*, pp.108-111, IEEE Comp. Society, ISNN 0730-3157

Reinders, J. (2007). *VTune performance analyser essential*, Intel Press, ISBN0-9743649-5-9

Sahoo, R.K.; Sivasubramanian, A.; Squillante, M. & Zhang, Y. (2004). Failure data analysis of a large-scale heterogeneous server environment, *Proc. of IEEE DSN Symposium*, pp.283-285, IEEE Comp. Society, ISBN 0-7695-2052-9

Simache, C. & Kaâniche, M. (2002). Event Log Based Dependability Analysis of Windows NT and 2K Systems, *Proc. of IEEE Pacific Rim Int. Symposium on Dependable Computing*, pp.311-315, IEEE Comp. Society, ISBN 0-7695-1852-4

Simache, C. & Kaâniche, M. (2001). Measurement-based availability of Unix systems in a distributed environment, *Proc. of 12th International Symposium on Software Reliability Engineering (ISSRE'01),* pp.346-355, IEEE Comp. Society

Sosnowski, J. & Gawkowski, P. (2006). Enhancing fault injection test bench, *Proc. of DepCos_RELCOMEX Conference*, pp.76-83, IEEE Comp. Society, ISBN 0-7695-2565-2

Sosnowski, J. & Poleszak, M. (2006). On-line monitoring of computer systems, *Proc. of IEEE DELTA 2006 Workshop*. pp. 327-331, IEEE Comp. Society, ISBN 0-7695-2500-8, (complementary presentation in LATW 2006 and ICIT 2009)

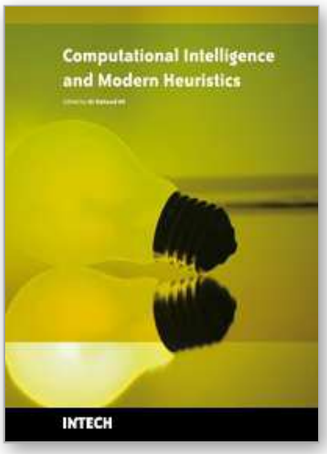
Stearley, J. (2004). Towards informatic analysis of Syslogs, *Proc. of IEEE Int. Conf. on Cluster Computing*, pp. 309-318, IEEE Comp. Society, ISBN 00-803-8430X

Vaarandi, R. (2008). Mining event logs with SLCT and LogHound. *Proceedings of the 2008 IEEE/IFIP Network Operations and Management Symposium*, pp. 1071-1074.

Xu, J.; Kallbarczyk, Z. & Iyer, R.K. (1999). Networked Windows NT system field failure data analysis. *Technical Report CRHC 9808*, University of Illinois at Urbana- Champaign,

Ye, N. (2008). *Secure Computer and Network Systems*, John Wiley& Sons, Ltd. ISBN 978-0-470-02324-2, Chichester

Zhang Y. & Sivasubramanian, A. (2008). Failure prediction IBM BlueGene?L event logs, *Proc. of Int. Symp. on Parallel and Distributed Processing*, pp.1-5, IEEE Comp. Society, ISBN 978-1-4244-2030-8



Computational Intelligence and Modern Heuristics

Edited by Al-Dahoud Ali

ISBN 978-953-7619-28-2
Hard cover, 348 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

The chapters of this book are collected mainly from the best selected papers that have been published in the 4th International conference on Information Technology ICIT 2009, that has been held in Al-Zaytoonah University, Jordan in the period 3-5/6/2009. The other chapters have been collected as related works to the topics of the book.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen