Using Event and Performance Logs in Dependability Evaluation

J. Sosnowski, M. Król
Institute of Computer Science, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
J.Sosnowski@ii.pw.edu.pl

ABSTRACT

The paper describes the problem of evaluating dependability of computer systems with on-line monitoring mechanisms. The main features of possible measurements (related to system operation) and the scope of collected data are shortly outlined. 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. The presented considerations are illustrated with practical monitoring results. They relate to long term observations of many workstations and servers.

Key Words: dependability evaluation, on-line monitoring, event logs, performance logs, failure detection

1. Introduction

The increasing complexity of hardware and software creates more attention to dependability, maintainability and performance issues. In particular various measurement-based studies have proliferated. They relate to the problem of detecting or predicting critical situations. Most published results are focused on narrow and fragmented problems encountered in the systems considered by the authors (e.g. [3-6,8-10,25,27] and references). In contemporary computer systems various monitoring mechanisms are included. The most popular relate to event and performance monitoring [1,2,7,12,19,23]. Such monitoring can generate enormous quantity of various data, which may be a valuable source of information on system operation. This problem needs further investigation 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 simplify problem identification and applying appropriate reactions.

In the literature most papers are targeted at finding some characteristic pattern in log files related to well defined critical problems (e.g. leading to system crash) [8,18,25], detecting attacks [26], or monitoring various performance parameters to predict network or processor bottlenecks [2,5,6,9,17,20]. Specialised statistical or data mining models are being developed for specific problems, hardly applicable to other systems.

We have faced various dependability and maintainability problems in computer systems used within the university for didactic and research purposes. They are available to many students and researchers. 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. We have also some experience with systems handling many customers with fluctuating usage profiles. Long-term observations of these systems allowed us to improve monitoring techniques and dependability. We have developed and installed some special software modules, collected a lot of data and performed various analyses.

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 features of normal operation 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 (section 2) and performance data (section 3).

2. Event Logging

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 exceeds 10000 [22]. 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:

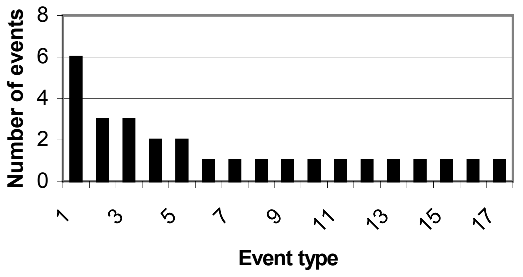
1) *Event counting* – the distribution (e.g. in decreasing order) of the number of registered events;
2) *Time between events* - time distribution between events of the same or different types;
3) *Event occurrence distribution* – statistics of the number of the selected event type in relevance to months, 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 (fig.1) of appearance, entropy, etc.

While analysing the registered events we should identify system start up and shutdown. When the 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 Event distribution before restarts



Looking for the sources of restarts we can analyze 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).

The developed system *LogMon* provides various data exploration results. 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 restart: Tab. 1 shows restart statistics for 4 laboratories (L1-L2) each comprising 17 workstations. 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 identified restarts (unknown) 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.

Tab. 1 Distribution of restart sources

	L1	L2	L3	L4
updates	20.2%	16.5%	22.6%	24.3%
applic.	19.2%	29.7%	12.0%	29.6%
hardware	1.2%	0.5%	2.0%	0.4%
ambig	1.4%	5.6%	5.0%	5.4%
unknown	59.4%	53.4%	63.4%	45.7%

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).

Fig. 2 Distribution of non activity periods

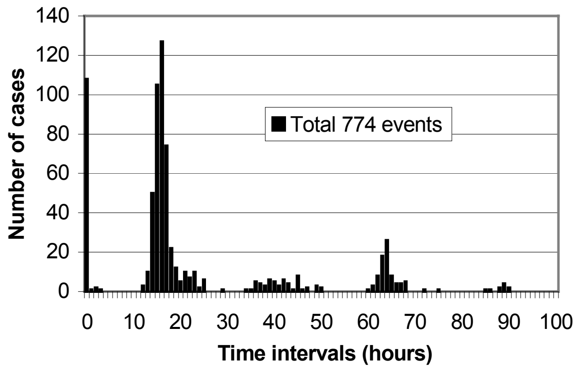


Fig. 2 shows distribution of time periods between event A and B, which correspond to closing and starting the system. The x-axis of the plot has the granularity of single hours. The first bar (108 cases) relates to short time intervals (less than 1 hour) and it corresponds to short operation breaks related 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 period. 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 (power down - switching off) and 3234:UPS (power up - switching on) generated by UPS power supply. For one of the servers the distribution of registered power outage duration *d* was as follows: 22%, 56%, 8.5%, 6.8% and 4.7% for $d<1s$, $1s\leq d<2s$, $2s\leq d<3s$, $3s\leq d<4s$ and $d\leq 12s$, respectively. The registered power outages were tolerated in systems with UPS. Computers without UPS crashed and needed restarting. Some crashes appeared simultaneously in several computers (common power failure).

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.

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 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 [10,11,14-16,24]. 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 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 [14]. 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).

3. Performance Monitoring

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) [7,17]. 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.

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.

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 times accesses to files opened successfully were denied (improper access authorisation, etc.).

There are many counters characterising network traffic or TCP/IP protocol activity. Here are 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). *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.

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. cyberattack 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 reconfigurations).

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 installation of a new version of the operating system and after 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 [26]. The authors give statistical properties of various performance variables related to different objects for 11 known cyber attacks. Analysing these results we have checked the observability properties of these attacks.

Tab. 2 The impact of attacks on performance

Property	Objects	Variables
M+	3-16 (7.7)	10-362 (105)
M-	3-11 95.9)	17-182 (60)
DUL	1-4 (2.5)	1-33 (10.3)
DUR	3-9 (5.7)	9-52 (27.8)
DMM	3-9 (5.7)	24-52 (43.3)

Tab. 2 shows minimal, maximal (average) numbers of monitored performance objects and related variables (counters) which showed 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. Such sensitivity analysis allows the designer to minimise the number of monitored variables and assure good detection accuracy.

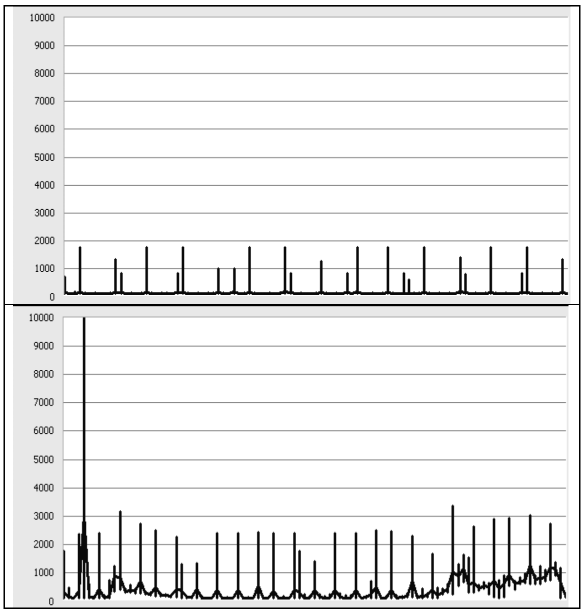


Fig. 3 Control operations on files

For many other problems we have to trace different properties. It is worth noting that the behaviour of performance variables can be different and needs deeper analysis. Quite often we observe some spikes in time plots of variables. This is illustrated in fig. 3 which shows the counter related to control operations on files (system object parameter) for the system without load (background system activities – upper plot) and during disc defragmenation (lower plot) with the average values 85 and 379, respectively. The correct behaviour comprises a high value spike at the beginning and many periodic spikes.

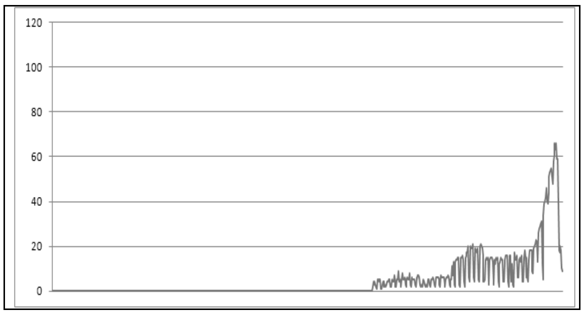


Fig. 4 Processor usage trend

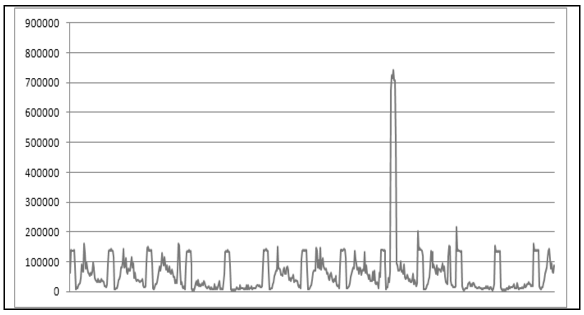


Fig. 5 The number of received bytes/sec

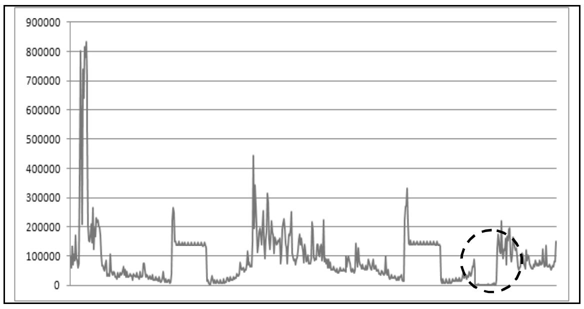


Fig. 6 The number of transmitted bytes /sec

Fig. 4 shows systematic increase of processor usage (in percents) within 26 months, this increase is visible in the last 16 months. It resulted from additional program installations available to students. Having acquainted with these programs they systematically increased the load, so after 16 months system upgrade has been done, to prevent problems, which started to appear several months earlier. Fig. 5 shows the number of received bytes within the network interface during 12 days, on Sundays and Saturdays lower traffic level is visible, on Thursday a high peak appeared. This significant increase has been detected by the monitor and warned the administrators. This resulted from an erroneous burst of emails. Fig. 6

shows transmitted bytes to the network, the circled area of the plot relates to no traffic within 2 hours of a switch crash.

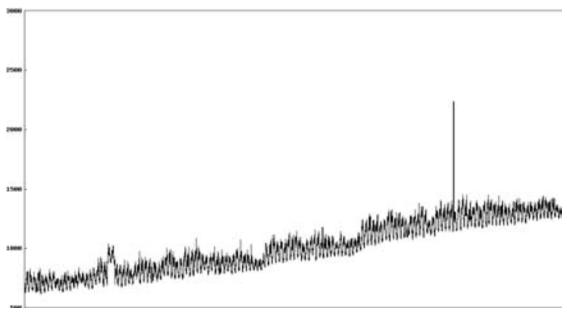


Fig. 7 Number of processes in data processing server

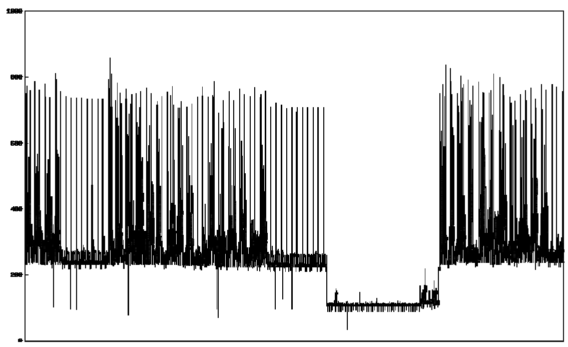


Fig. 8 Number of processes in communication server

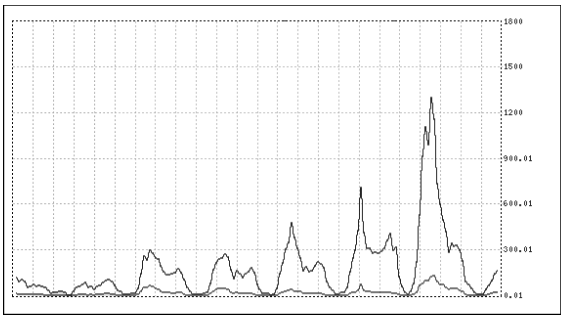


Fig. 9 Distribution of incoming session requests

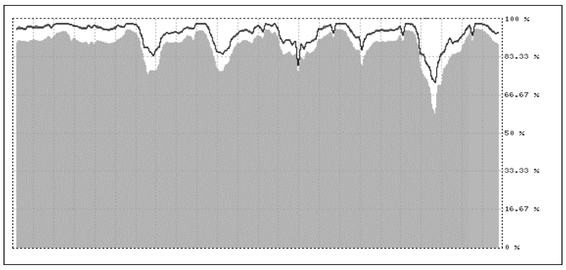


Fig. 10 Server resources and quality function

Fig. 7 shows the plot of the number of active processes in one of the servers within 5 months. A high spike at the end of May corresponds to student activities related to final semester reports. The plot reflects steady increase of the workload (after one year system upgrade was recommended). Fig. 8 shows the number of processes in another server within one year with low activity related to summer months.

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.9 shows one week plot of the total number of sessions (lower line relates to sessions created in the last minute) handled by the farm. Fig. 10 presents the available server resources (upper line and server quality function shaded plot). The quality function is used by the system to distribute loads between 16 servers in the farm according to the session requirements (number of customers) and available resources (processor and memory usage). Close relation between quality function and the available resources confirms good prediction of load and correct system behaviour. This difference is monitored by the system to identify problems (e.g. to eliminate a faulty server and move the traffic to other servers). 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.

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 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 problems. 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 [21].

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.

References:

1. Bertino, E.,Ferrari, E., Guerrini, G., An approach to model and query event based temporal data, Proc. of 5th Int. Workshop on Temporal Representation and Reasoning. IEEE Comp. Soc., 1998, pp. 122-131.
2. Cherkasova L., et al., Anomaly? application change? or workload change?, Proc. of IEEE DSN Symposium, 2008, pp. 452-461.
3. Daniel,E., Lal, R., Choi, G., Warnings and errors: A measurement study of a UNIX server, IEEE Int. Symp. on Fault-Tolerant Computing, FTCS-29, 1999, www.crhc.uiuc.edu/FTCS-29/fastabs.html.
4. Ganapathi, A., Patterson, D., Crash data collection: A windows case study, Proc. of IEEE DSN Symposium, 2005, pp.772-184.
5. Heath, T., .Martin, R.P., Nguyen, T.D., Improving cluster availability using workstation validation. Proc. ACM SIG-METRICS Conf. Measurement and Modelling of Computer Systems, 2002, pp.217–227.
6. Hoffmann, G. A., .Trivedi, K.S., Malek, M., A best practice guide to resource forecasting for computing systems, IEEE Trans. on Reliability, vol.56, no. 4, 2007, pp.615-628.
7. John, L.K., Eeckhout, L., (editors), Performance evaluation and benchmarking, CRC Taylors &Francis, 2006.
8. Kalyanakrishman, M., Kalbarczyk, Z., Iyer, R.K., Failure data analysis of a LAN of Windows NT based computers, 18th IEEE Symposium on Reliable Distributed Systems, 1999, pp.178-188.
9. Li, M., Wang, S., Zhao, W., A real-time and reliable approach to detecting traffic variations at abnormally high and low rates, In L.T. Yang et al., (edits) ATC 2006, LNCS 4158, 2006, pp.541-550.
10. Lim. Ch., Singh, N., Yainik, S.., A log mining approach to failure analysis of enterprise telephony systems, Proc. of IEEE DSN Symposium, 2008, pp. 388-403.
11. Makanju A., et al.., LogView: visulizing event log clusters, Proc. of Annula Conf. on Privacy, Security and Trust, 2008, pp. 99-108.
12. Malek, M., Online dependability assessment through runtime monitoring and prediction, Proc. of EDCC – 7, IEEE Comp. Soc., 2008, pp.181.
13. Mansouri-Somani, M., Sloman, H., A configurable event service for distributed systems, Proc. of IEEE 3rd Int. Conf. on Configurable Distributed Systems, 1996, pp. 210-217.
14. Peng, W., Peng, Ch., Li, T., Wang, H., Event summarization for system management, Proc. of ACM KDD'07, 2007, pp. 1028-1032.
15. Peng, W., Li, T., Ma, S., Mining logs files for computing system management, SIGKDD Explorations, vol. 7, issue 1, 2005, pp. 44-51.
16. Razavi A., K. Kontogiannis, Pattern and policy drven log analysis for softeawe monitoring, Proc. of IEEE Int. Computer Software and Applcations Conference, 2008, pp.108-111.
17. Reinders, J., VTune performance analyser essential, Intel Press, 2007.
18. Sahoo, R.K., Sivasubramanian, A., Squillante, M., Zhang, Y., Failure data analysis of a large-scale heterogeneous server environment, Proc. of IEEE DSN Symposium, 2004, pp.283-285.
19. Simache, C., Kaâniche, M.:, *Event Log Based Dependability Analysis of Windows NT and 2K Systems,* IEEE Pacific Rim International Symposium on Dependable Computing, 2002, pp.311-315.
20. Simache, C., Kaâniche, M.:,Measurement-based availability of Unix systems in a distributed environment, 12th International Symposium on Software Reliability Engineering (ISSRE'01), IEEE Comp. Soc., 2001, pp.346-355.
21. Sosnowski, J., Gawkowski, P., Enhancing fault injection test bench, Proc. of DepCos_RELCOMEX Conference, IEEE Comp. Soc., 2006, pp.76-83.
22. Sosnowski, J., Poleszak, M.:, On-line monitoring of computer systems, Proc. of IEEE DELTA 2006 Workshop., 2006, pp. 327-331.
23. Stearley, J.:, Towards informatic analysis of Syslogs, Proc. of IEEE Int. Conf. on Cluster Computing, 2004, pp. 309-318.
24. Vaarandi, R.:, Mining event logs with SLCT and LogHound. Proceedings of the 2008 IEEE/IFIP Network Operations and Management Symposium, 2008, pp. 1071-1074.
25. Xu, J., .Kallbarczyk, Z., Iyer, R.K.:,. Networked Windows NT system field failure data analysis. Technical Report CRHC 9808 University of Illinois at Urbana- Champaign, 1999.
26. Ye, N., Secure Computer and Network Systems, John Wiley& Sons, Ltd. , 2008.
27. Zhang Y., Sivasubramanian, Failure prediction IBM BlueGene?L event logs, Proc. of IEEE Int. Symp. on Parallel and Distributed Processing, 2008, pp.1-5.