POZNAN UNIVERSITY OF TECHNOLOGY

Faculty of Computing

Doctoral Dissertation

Security and System Events Monitoring in Distributed Systems Environment

Łukasz Kufel

Supervisor

Professor Jan Węglarz

Poznań, Poland

2017

To my beloved wife, Lidia.

Abstract

A distributed system is a set of network connected devices that communicate between each other to perform requested actions. From end user perspective, the distributed system is one logical entity. To provide high availability, compliance with security standards and reliable performance of distributed systems, an organization requires management tools, such as a monitoring solution. Before choosing the solution to monitor distributed systems environment, the IT departments need to understand what infrastructure components are critical to the organization's success. The second aspect to consider is, how quickly the restoration actions can be taken or at least the relevant support teams be notified when IT failures occur. Once those principles are known, the organization may begin identifying a monitoring solution.

This study presents monitoring fundamentals, variety of monitoring approaches that can be used to collect events data from the systems, review of paid and open source monitoring tools currently available on the market, and details of deploying selected monitoring solutions in multiple infrastructure environments. Moreover, I define key factors to be considered when selecting a monitoring solution and discuss a design and implementation of novel hybrid approach, an order-based monitoring (OBM).

Finally, I conduct an experiment to measure network latency impact on overall monitoring process performance in multiple datacenters scenario. Then, a concept of local monitoring Distributor is introduced to optimize that impact.

Contents

Acknowledgements

First, I would like to thank my supervisor Professor Jan Węglarz for accepting me as his PhD student. My research area is rather a narrow field of science and finding a supervisor was not an easy task. During my PhD studies and while preparing publications, I knew I could always rely on my supervisor and would receive prompt advice when questions were arising.

I would like to thank my family and friends for their support and care during the last few years when working on publications. I especially thank my beloved wife Lidia for her mental support and Mieczysław Torchała for his invaluable tips and help.

Chapter 1

Introduction

1.1 Motivation

My research adventure with distributed systems monitoring started in 2006 when a company asked me to search for a tool that can monitor performance metrics of Tomcat application servers. A quick Google search suggested verifying ManageEngine Applications Manager solution which I did. After initial setup, I engaged myself in exploring it further and further to the level that few months later it became one of the most important tools in the organization's IT departments. My passion to monitor distributed systems, applications, and build monitoring solutions at scale had just begun.

Today, organizations performance and reputation depend much more on IT systems than in the past. From internal communication systems, such as email, employees' news portal and instant messaging services, to back office, HR, payroll applications and front-end systems in e-commerce organizations; all of them depend on infrastructure built upon IT systems. Those systems are crucial to the organizations success, especially in current times with offices and employees dispersed around the countries and continents.

To sustain today's IT systems high availability and performance reliability, each organization needs to identify, deploy and maintain a monitoring solution that will continuously analyze systems availability, performance and compliance with security standards. Moreover, the selected solution requires multiple notification mechanisms to alert, inform and engage relevant support teams in case a failure in monitored systems has been detected. Lastly, the organization will benefit from the monitoring solution advantages when it offers various interfaces and protocols to seamlessly integrate with existing IT ecosystem [46].

1.2 Scope and Goals of the Thesis

There are multiple monitoring solutions and approaches currently available on the market [20, 21]. All the solutions are placed into two main categories, i.e. security information and event management (SIEM) and infrastructure monitoring tools. Each of them operate using one or more of four available monitoring approaches, such as agent-based, agentless, hybrid and most recently introduced data streams. Selecting and deploying the solution that meets organization's requirements can be complex, as it might be expensive, or the solution won't be able to monitor systems health at designated polling interval, or the tool simply won't allow monitoring in-house or custom built business critical applications and services. On the other hand, choosing the right monitoring solution will minimize organizations systems' downtime and improve their reputation as security threats and IT failures can be detected and mitigated earlier or even prevented from occurring.

In this thesis, I focus on the process of selecting, designing and deploying a monitoring solution in a variety of distributed environments, as well as on identifying possibilities to monitor anything on demand, and finally examining network latency impact on overall monitoring process. To achieve this, I formulated the following research goals:

- Review monitoring approaches and tools for security and system events monitoring,
- Identify and discuss key factors when selecting a solution for distributed systems monitoring,
- Deploy and test some of the monitoring tools in distributed systems environments, in one datacenter, in multiple geographically dispersed datacenters and in the cloud,
- Design, implement, and test a monitoring approach based on advanced order placement idea,
- Conduct an experiment and analysis with network latency impact on overall monitoring process performance,
- Propose a concept that can reduce network latency impact on remotely monitored systems.

1.3 Distributed Systems and Monitoring

Monitoring of distributed systems involves the collection, analysis, filtering, and presentation of gathered data. It all starts with choosing the monitoring solution tool, deploying it in the distributed systems environment, configuring the tool and adding all the systems in scope to be monitored. As a result, the organization's IT departments can actively verify current status of their complete IT infrastructure, report of systems usage, and get notified when failures are detected.

My primary research focus was on the first aspect of the entire monitoring process – the collection part. Analysis of collected data such as data mining, clustering [9, 26, 37, 41, 43], then techniques of filtering the alarm events (aka event correlation) [33, 39, 42, 44, 45] and visualization of gathered data [7, 30] were out of scope of my study.

1.4 Related Papers

Many researchers have studied distributed systems monitoring and usually focused on single domain or application specific problems. The most popular area of scientists' research in distributed systems monitoring is related to network performance, availability of network services, network management [6, 8, 10, 11, 22, 27, 29, 38] as well as analyzing real-time systems status from important sensors [3, 31, 34]. The network is a key component in operations of distributed systems, and any failure, disruption or delay may negatively affect availability and performance of organization's critical IT services.

Further areas of distributed systems monitoring research include verifications of key infrastructure components such as processors, disk and memory usage [4, 36] and status of the software that is being hosted on the distributed systems [5, 13, 32, 48]. Researchers have also discussed implementations of monitoring solutions for grid systems [2, 23, 40, 47] and more recently surveyed monitoring characteristics of distributed systems running in cloud environments [1, 12, 16, 25, 35].

This thesis contributes towards monitoring problems affecting multiple domains, such as networks, systems infrastructure metrics, security events and monitoring in the

cloud environments. The research results presented in this thesis are related to my articles that were published in peer-reviewed journals:

- Kufel L., Tools for distributed systems monitoring, Foundations of Computing and Decision Sciences, vol. 41, no. 4, pp. 237-260, 2016 [21],
- Kufel L., Network latency in systems event monitoring for multiple locations, Scientific Programming, vol. 2015, article ID 371620, 2015 [19],
- Kufel L., Security event monitoring in a distributed systems environment, IEEE Security & Privacy, vol. 11, no. 1, pp. 36-43, 2013 [20].

1.5 Thesis Outline

Chapter 2 introduces monitoring fundamentals, such as IT environment layers, security and system events, monitoring threshold profiles, and security and infrastructure areas that can be monitored. In this chapter, I also define systems criticality, present sample monitoring polling intervals and data retention policy based on systems criticality.

In Chapter 3, I discuss popular monitoring approaches as well as recently introduced hybrid and data streams approaches.

Chapter 4 contains comprehensive reviews of security information and event management (SIEM) tools, and infrastructure and application performance monitoring tools.

Chapter 5 is devoted to the core objectives of this thesis, such as process of selecting and deploying a monitoring solution in one datacenter, in multiple geographically dispersed locations and in the clouds. It also introduces novel hybrid monitoring approach based on advanced order placement idea – order-based monitoring (OBM). Results of network latency experiment, concept of local Distributor, and common monitoring issues close this chapter.

Finally, Chapter 6 concludes the thesis and provides suggestions for future work.

Chapter 2

Monitoring Fundamentals

2.1 Introduction

There are monitoring solutions and diagnostic tools used in distributed systems environments to verify availability and performance of IT systems. The diagnostic tools are needed when for example in-depth investigations are required to understand a root cause of a failure. In those cases, dedicated tools are used, like software debuggers, extensive network packets collectors and professional operating system diagnostics applications [62]. Once the cause of the failure is identified and understood, the diagnostic tools are set on a standby and disused.

In opposite to ad-hoc diagnostic tools, the monitoring solution is a continuous process of data gathering, storing, analyzing, displaying and notifying when certain conditions are met. This process runs on predefined time intervals and collects the data to visualize, and to build trends of current and historic states of business services, applications and infrastructure systems.

In this chapter I discuss monitoring fundamentals concepts, monitoring events structures, alerts' threshold levels, and other key basics required to setup a monitoring solution in a distributed systems environment.

2.2 Layers

Monitoring solution's design consists of three major layers. Each of the layers presented in Figure 1 plays an important role in the monitoring solution's performance and effectiveness and needs to be carefully examined before deploying the solution. A proof of concept (POC) deployment with test systems is usually a great opportunity to understand the network flow and bandwidth usage, as well as the value of monitoring

solution being reviewed, especially from data presentation and timely failure detection perspective.

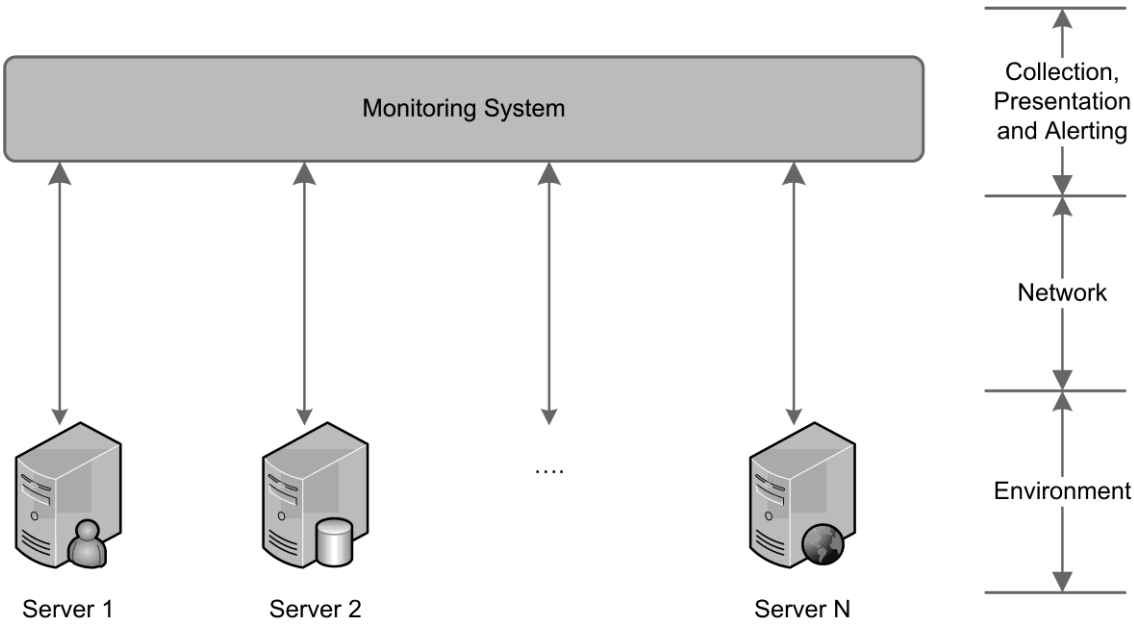


Figure 1. Three major monitoring layers representing a sample design of monitoring solution.

The three major layers of monitoring solution's design are as follows:

- *Collection, Presentation and Alerting.* This layer represents the key area of monitoring solution, where the monitoring software is installed and all the gathered data of monitored metrics are stored. This is also the place where monitoring process is being initiated for agentless approach, where new systems are being added and configured, where alerts are deployed and their triggering conditions are set. Furthermore, this section is the central point for end user of the monitoring solution where data such as systems availability and capacity need to be visualized, presented on the dashboards or simply verified and analyzed. In distributed systems environment, this layer can also be distributed to multiple console nodes located in multiple datacenters or regions. However, there always will be one central point linked to all those consoles.

- *Network.* This layer shows the data flow diagram and is focusing on the network paths configuration and amount of data being transmitted from a source (monitored systems) to a destination (monitoring system). This

section is important for local area networks (LAN) and crucial for wide area networks (WAN) when deploying a monitoring solution for distributed systems in multiple geographic locations. The key parameter in this layer is the network latency. While it is negligible in LAN networks, it impacts the monitoring process in WAN environments, especially in data collection part. Sample results how network latency can impact monitoring solution in geographically distributed environments were demonstrated in [19].

- *Environment.* This layer presents the actual distributed systems infrastructure that will be examined by selected monitoring solution. The systems include network devices, computer hardware, operating systems, applications, databases and business services. In agent-based and data streams monitoring approach, this is the place where dedicated software agents are being deployed and usually configured; some monitoring solutions offer centralized configuration panel from where the configuration settings are pushed to the agents. Moreover, this layer combines all the systems in the organization, despite their datacenters geographic locations.

The concept of monitoring layers is an example of how monitoring solution can be architected. The key layers mentioned in this section show principles of interoperability between the monitoring system software and tangible systems that are going to be monitored.

2.3 Events

One of the reasons why organizations deploy monitoring solution is to gather data from IT systems and analyze their security and system events. Similarly to the definition presented by Terenziani et al. [39] and Tierney et al. [40] I define the event as a time-stamped information about state of a service, an application, or an operating system and its relevant metrics such as availability, CPU and disk usage, network connections, and security auditing details to comply with industry standards. Every event needs to be

described by meaningful details that include time and date when the event was generated, which host it occurred on, and event specific characteristics. A sample security event on Windows-based operating system is demonstrated in Figure 2 while Figure 3 shows sample system events on Unix-based operating system.

Event Type: Success Audit
Event Source: Security
Event Category: System Event
Event ID: 517
Date: 10/10/2016
Time: 21:38:37
User: NT AUTHORITY\SYSTEM
Computer: M5WIN
Description:
The audit log was cleared
Primary User Name: SYSTEM
Primary Domain: NT AUTHORITY
Primary Logon ID: (0x0,0x3E7)
Client User Name: Luk
Client Domain: M5WIN
Client Logon ID: (0x0,0x1E455)

Figure 2. Sample security event on Windows-based operating system.

Jul 17 18:19:41 nagios kernel: Warning: Intel CPU model - this hardware has not undergone testing by Red Hat and might not be certified. Please consult https://hardware.redhat.com for certified hardware.
Jul 17 18:19:41 nagios kernel: Initializing cgroup subsys cpuset
Jul 17 18:19:41 nagios kernel: Initializing cgroup subsys cpu
Jul 17 18:19:41 nagios kernel: Linux version 2.6.32-642.3.1.el6.x86_64 (mockbuild@worker1.bsys.centos.org) (gcc version 4.4.7 20120313 (Red Hat 4.4.7-17) (GCC)) #1 SMP Tue Jul 12 18:30:56 UTC 2016
Jul 17 18:19:41 nagios kernel: Command line: ro root=/dev/mapper/vg_nagios-lv_root rd_LVM_LV=vg_nagios/lv_swap rd_NO_LUKS KEYBOARDTYPE=pc KEYTABLE=uk LANG=en_US.UTF-8 rd_NO_MD rd_LVM_LV=vg_nagios/lv_root SYSFONT=latarcyrheb-sun16 crashkernel=auto rd_NO_DM rhgb quiet
Jul 17 18:19:41 nagios kernel: KERNEL supported cpus:
Jul 17 18:19:41 nagios kernel: Intel GenuineIntel
Jul 17 18:19:41 nagios kernel: AMD AuthenticAMD
Jul 17 18:21:13 nagios nagios: Warning: Return code of 255 for check of service 'CPU Stats' on host 'linux_promet' was out of bounds.
Jul 17 18:21:19 nagios ndo2db: Error: Connection to MySQL database has been lost!

Figure 3. Sample system events on Unix-based operating system.

Depending on operating system platform, each event will be classified into one of five types on Windows platforms [60] or into one of eight severity levels on Unix platforms [56, 57]. Table 1 presents all the types and severities available on both platforms, as well as indication if given event requires any action by support team.

Table 1. Event types and severities depending on operating system platform.

Windows Event Type	Unix Event Severity	Action Required?
Error	0 - Emergency 1 - Alert 2 - Critical 3 - Error	Yes, immediately
Failure Audit Warning	4 - Warning 5 - Notice	Yes, in near future
Success Audit Information	6 - Informational 7 - Debug	No

The sample system events and their types on Windows platforms are as follows:

- *Error.* Error event is registered when an application is unable to establish a connection with a remote host, or a system cannot start the service, or the device did not respond in each time interval (timed out), or a system update (patch) failed during the installation process.
- *Warning.* This type of events is generated when TCP/IP has reached the security limit imposed on the number of concurrent TCP connect attempts, or when an error was detected on a device during a paging operation, or when a system clock is unsynchronized, or when the system process was unable to power off the computer, or one of the computer disks is at or near its capacity and user may need to delete some files.
- *Information.* Operating system will create information events to document successful driver load operation, or to provide a periodic update about current system's uptime, or when the event log service was started (first event when the computer starts) and when it was stopped (last event when

the computer shuts down), or when one of the system services has entered the running state.

Apart from system and application events, Windows has capabilities to log events concerning compliance with security standards. Those events are being classified as:

- *Failure Audit.* This type of events is created when the user has entered an incorrect user name and/or password, or when the user was trying to access a resource to which its permission was not granted, or when the user was attempting to log on a computer using disabled or an expired account.
- *Success Audit.* Success audit events are similar to system's information events and record successful activities such as the user has logged on a computer or an application, or the user has successfully logged out or disconnected, or when the audit log was cleared, or the user has reconnected to a disconnected terminal server session.

On Unix-based platforms there are eight severity levels to classify the events. *Emergency* and *debug* severities are exclusive levels and should only be used as per their original definition and word meaning. The other levels are designated for applications and their usage as well as purpose is entirely dependent on applications' developers. As per RFC 5424 [56] following hierarchy and description of severity levels is defined:

- *0 - Emergency: system is unstable.* This level should never be used by applications. It manifests scenarios such as 'kernel panic' event or loss of power.
- *1 - Alert: action must be taken immediately.* Sample event would be loss of the primary network connection.
- *2 - Critical: critical conditions.* This severity level can be used when critical business service or application stops processing business transactions and requests.
- *3 - Error: error conditions.* For example, the business application has reached the defined number of failed transactions or errors per minute.

- *4 - Warning: warning conditions.* This severity level can be set when any of the filesystems has exceeded the 85% usage condition or number of concurrent connections has reached 150 out of 200 available.
- *5 - Notice: normal but significant condition.* Events with this severity level would record for example situations when an application is running at its peak load but still in a stable state, i.e. it can accept further requests as there are resources available.
- *6 - Informational: informational messages.* This level of severity is similar to information type of Windows events and represents successful operation like service has started, application has ended, or an operating system's software update has been downloaded and is ready for deployment.
- *7 - Debug: debug-level messages.* This severity level traces for example all actions within application's runtime and logs them for further analysis. The debug level is not recommended for applications running in the production environments due to the performance impact it may cause when writing large amount of details to the log files.

2.4 Thresholds

After monitoring solution gathers data for monitored metrics, the data are being analyzed if any alert needs to be triggered. That process uses alert's configuration known as threshold profile or simply a threshold. Each threshold has dedicated settings and usually consists of three level conditions (see Table 2), such as Error (indicated by red color), Warning (indicated by orange or amber color) and OK (indicated by green color). Furthermore, after data are analyzed with predefined threshold conditions, an alert can be triggered and notification like an email to support team may be sent out. A notification can be configured on all threshold levels or on none. When there is no notification setup, the alert will only be displayed in monitoring solution dashboard(s).

The error threshold condition represents for example a status of an application, a system or a system's resource that is unavailable, unstable or has reached its designed

limits. Typically, an action by support teams is required to restore affected application or system to its normal condition.

Table 2. Sample thresholds' configurations of error, warning and OK situations based on monitored metric.

Metric Name	Error Threshold Condition	Warning Threshold Condition	OK Threshold Condition
CPU Usage	More than 95% for 5 minutes	Between 80% and 95% for 5 minutes	Less than 80%
Memory Usage	More than 90% for 10 minutes	Between 75% and 90% for 10 minutes	Less than 75%
Disk Usage	More than 98% for 30 minutes	Between 95% and 98% for 30 minutes	Less than 95%
Transaction's Request Response Time	Average response time more than 200 ms per minute	Average response time between 150 ms and 200 ms per minute	Average response time less than 150 ms per minute
System's Response Time (a result of ping command)	Average response time more than 100 ms per 10 packets	Average response time between 70 ms and 100 ms per 10 packets	Average response time less than 70 ms per 10 packets

The warning threshold level indicates situations that require increased vigilance as a failure or an incident may occur if nothing changes. For example, an application or system may become unstable or unresponsive when affecting factors do not change or when support team does not take any interventional action in a near future. In real IT environments, this could be a scenario when disk or filesystem space is gradually being filled up by web application logs on a very busy web server. Without taking preventive actions, the disk or filesystem will be filled up causing the application to stop taking new requests and eventually becoming unavailable.

The OK threshold (also known as a green status) is the most expected level as it confirms the monitored metrics are within expected boundaries and application's as well as system's performance is stable. In situations that some repair actions were taken, as the monitored metric has triggered an alert, having the metric changing its status to OK threshold level would ensure that those actions were relevant. Continuing the disk or filesystem situation described in warning threshold paragraph, implementing a log

rotation mechanism or deleting old log files for the web application would clear the alert and mark the monitored system's capacity metric as in OK status.

2.5 Areas of Monitoring

In distributed systems environments, there are many areas from where metrics can be collected. Starting from security events like users logging to systems and applications, through availability verifications, systems capacity till overall performance, the monitoring solution should be capable of gathering data from all of them.

2.5.1 Security Events Monitoring

Today, organizations reputation depends much more on secured IT systems than in the past. As more data are being gathered and processed through various systems, the more advanced security and protection systems need to be applied and then constantly monitored. The market has even forced multiple software development companies to create security events oriented monitoring solutions known as security information and event management (SIEM) tools [20].

The main purpose of those tools is to collect events from firewalls, network devices, operating systems' security logs and business applications' specific logs. In organizations using public facing web sites it is now crucial to collect web servers' access logs that include remote IP addresses, browser version details and URL patterns used (aka user data). Gathering security events from multiple sources and analyzing them in a single console is one of the quickest methods to investigate distributed denial of service (DDoS) attacks. Those attacks have recently become very popular and very impactful to organizations businesses and availability of their e-commerce services.

When the monitoring solution is not offering security events analysis out of the box, it should at least provide customization capabilities such as execution of custom scripts, APIs, and interfaces to integrate with other tools. There are multiple third party plugins available to monitor security events as well as open source solutions like AlienVault's OSSIM.

2.5.2 Availability Monitoring

Another key area the monitoring solution covers is systems, applications and services availability metrics. Those metrics not only represent the result of *ping* command, but also validate if users can access the website by for example content verification checks, services can exchange data by testing accessibility of TCP or UDP ports and if required processes are running, or the system user accounts are active and have valid permissions.

The availability metric is calculated as a percentage of duration for which the system or an application was running and accessible to the period for which the metric is being measured. A sample visualization of system host availability is presented in Figure 4.

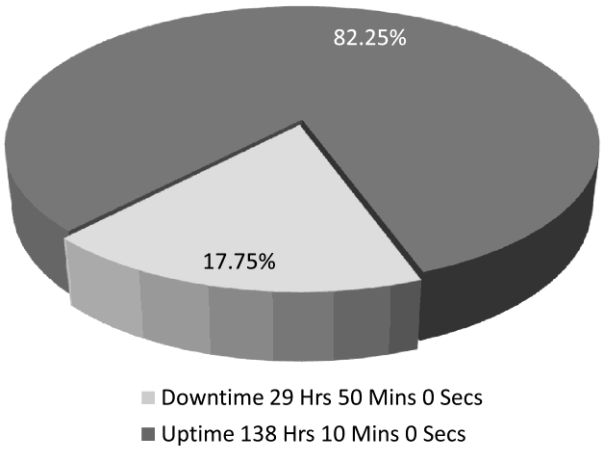


Figure 4. A visualization of availability metric based on system host availability measured for 7 days.

Many business services are now being designed to be available on a 24 x 7 basis. Although it is possible to plan and architect highly available systems, it is very difficult to achieve 100% overall availability on an annual report. Table 3 shows durations of unavailability (also known as a "downtime") and their impact on a monthly and an annual availability reports.

Table 3. Downtime durations and their impact on availability reports.

Downtime per month	Downtime per year	Availability %
72 hours	36.5 days	90% "one nine"
7.20 hours	3.65 days	99% "two nines"
43.8 minutes	8.76 hours	99.9% "three nines"
4.38 minutes	52.56 minutes	99.99% "four nines"
25.9 seconds	5.26 minutes	99.999% "five nines"

2.5.3 Capacity and Performance Monitoring

The last important metrics being gathered by monitoring solution are details about systems capacity and performance. The systems in distributed and cloud environments are using more computing resources year over the year (see example of Wikipedia platform presented in Figure 5). Monitoring solution can help measuring the current state of resources such as the number of physical and virtual servers, switches, routers, firewalls etc. as well as their usage levels. The traditional capacity metrics include CPU usage, memory usage, disk usage and bandwidth usage for network devices.

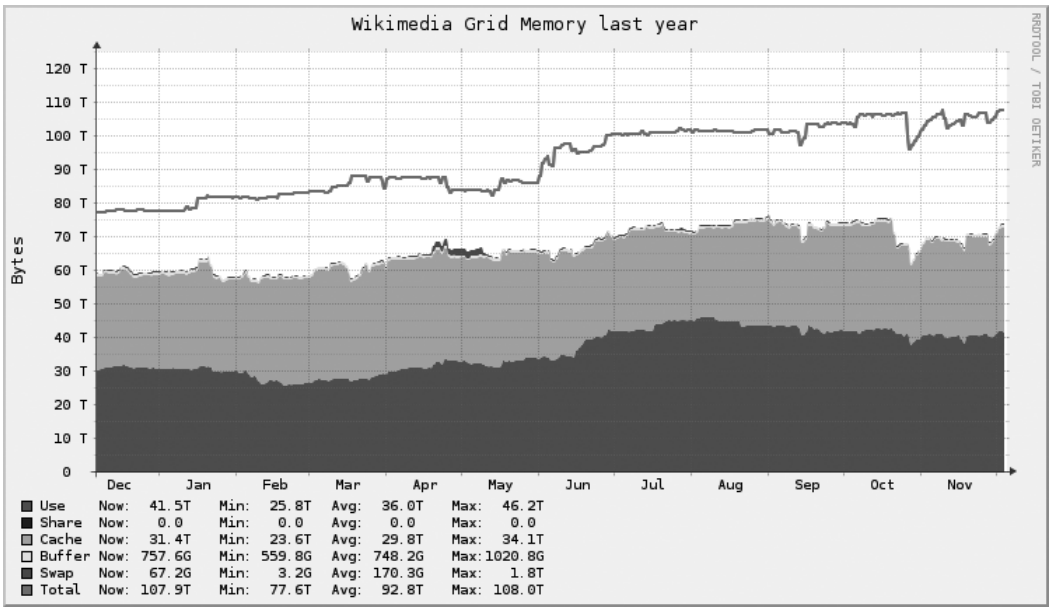


Figure 5. A visualization of capacity metric based on Wikipedia – grid memory usage over last year (Dec 2015 till Dec 2016).

Having accurate figures and usage levels of organization's IT infrastructure would support business decisions in for example budget allocation discussions for next years of IT operations. This would also help identifying systems that are under- or overutilized and allow giving suggestions to improve architecture designs of relevant systems and applications. As a result, this would give better and smarter usage of IT infrastructure as well as when required it would let the organization quicker adapt to the changing conditions on the market.

Moreover, the performance metrics collected by monitoring solution are crucial in e-commerce organizations, where customers are expecting relatively fast response times of the organization's websites and same behavior is expected despite seasonal sales and marketing campaigns [28].

2.6 Systems Criticality

In distributed systems environments, not all the systems are equal in terms of their importance to the organization. Despite the infrastructure's environment categorization based on software development cycle that is development, test, user acceptance testing (UAT), production, and disaster recovery [54], in each of those categories systems can be classified in accordance to their business criticality that follows:

- *Mission Critical or Five Stars*. This is the highest available categorization based on business importance of the system. Systems with this classification are usually designed to operate on a 24 x 7 basis with minimal or zero downtime for scheduled maintenances. In production environments, mission critical systems are hosting for example web front end applications in e-commerce organizations or applications that control robots on the production line in a car factory. In development environments, mission critical systems are running platforms with software code repository or regular software builds compilations. Unavailability of systems classified as mission critical would impose severe financial impact to the organization or affect employee's productivity and cause delays in delivering final product or software feature.

- *Critical or Important or Four Stars*. This category is given to the systems that need to be available during regular organization's business days. The typical scenario of expected continuous availability is Monday till Friday, 24 hours per day for five days (a 24 x 5 basis). The scheduled maintenances and planned upgrades for systems classified as critical are being performed over the weekend days. In production environments, critical systems are hosting for example internal email platform, employees' internal communication and news portal, internal telephony platform, backoffice or administration systems or human resources (HR) systems. In organization with software development cycle, critical systems can host software testing and user acceptance testing (UAT) environments as well as the software development platforms.

- *Standard or Three Stars*. The least business impactful systems will be classified as standard. Their commonly expected availability is Monday till Friday within core hours such as 8AM till 6PM. The planned maintenances on those systems can be scheduled every week day outside of the core hours and any time throughout the weekend days. Furthermore, this is the default category of system's classification. Moving the system to the higher category level is primarily dependent on expected availability duration and / or its business impact to the organization.

Implementing systems criticality levels will assist in deploying a monitoring solution and setting up relevant alerting mechanisms. The mission critical and critical systems will certainly require more frequent collection intervals than the standard systems. Moreover, the alerting for critical systems needs also be adjusted to be more interactive with support teams, for example a text message on the mobile phone or an automated phone call rather than just a simple alert over the email. Those steps are mandatory to reduce the duration of systems unexpected behavior and unavailability or in other words to minimize their mean time to restore (MTTR) [55] when a failure occurs.

2.7 Polling Intervals and Data Retention

During the configuration phase of monitoring solution setup process, one of the procedures the organization needs to define is a policy about polling intervals for collecting the data of monitored metrics. The policy should take into consideration multiple factors, such as:

- Business criticality of monitored systems,
- Characteristics of the metrics for which the data are being gathered,
- Industry standard recommendations,
- Retention periods of collected data.

Based on those factors and preliminary assessments, a policy with monitored metrics polling intervals can be defined (sample shown in Table 4) and eventually followed by the department responsible for deployment and maintenance of organization's monitoring solution. The polling intervals will directly influence the performance of monitoring solution, its capacity due to the amount of data being gathered and archived, and the mean time to detect (MTTD) when a failure in distributed systems environments occurs.

Table 4. Sample policy of monitored metrics polling intervals and their retention periods.

Metric	Standard Systems	Critical Systems	Mission Critical Systems
Availability monitoring	Every 10 minutes	Every 5 minutes	Every 1 minute
Capacity monitoring, for example CPU usage	Every 20 minutes	Every 10 minutes	Every 5 minutes
Security and system logs [18]	Every 3 to 24 hours	Every 15 to 60 minutes	Every 5 minutes
Data retention [18]	1 to 2 weeks	1 to 3 months	3 to 12 months

The sample polling intervals policy presented in Table 4 includes standard metrics being monitored by a solution using agent-based, agentless or hybrid monitoring approach. When the organization plans to implement a monitoring solution based on data streams approach, additional metrics should be defined in the policy. Those metrics

will be specific to business transactions and include details such as, how often should the data from data streams forwarder agent be transferred over to central monitoring system (typically between 10 seconds to 60 seconds), number of alert conditions reoccurrences before triggering a valid alert (this is due to the time scale format the data are being collected), and events grouping intervals to prevent from a flood of alerts.

Chapter 3

Monitoring Approaches

3.1 Introduction

There are two popular approaches used to monitor systems in distributed environments: agent-based and agentless. Most recently two more approaches were introduced: hybrid approach that combines benefits of traditional approaches, and data streams that focuses on business transactions and services monitoring and offers a real-time dashboards with monitored metrics.

In this chapter I present all four approaches, their architecture designs and functional comparison.

3.2 Agent-based Approach

The agent-based monitoring approach requires dedicated software that needs to be installed on monitored systems. It focuses on monitored metrics in more detailed level and in more frequent polling periods. Once the data are collected, they are being sent to monitoring console for further processing and analysis (see Figure 6). As defined by Jennings [15] “an agent is an encapsulated computer system that is situated in some environment and that is capable of flexible, autonomous action in that environment in order to meet its design objectives”.

In this monitoring approach, there are two common types of agents being deployed, a system agent and an application agent. The system agent is designed to monitor performance metrics of dedicated operating system, for example CPU usage on Windows server or filesystem usage on Unix system. The purpose of application agent is to monitor application’s specific metrics, like number of connections to the database or Java heap size.

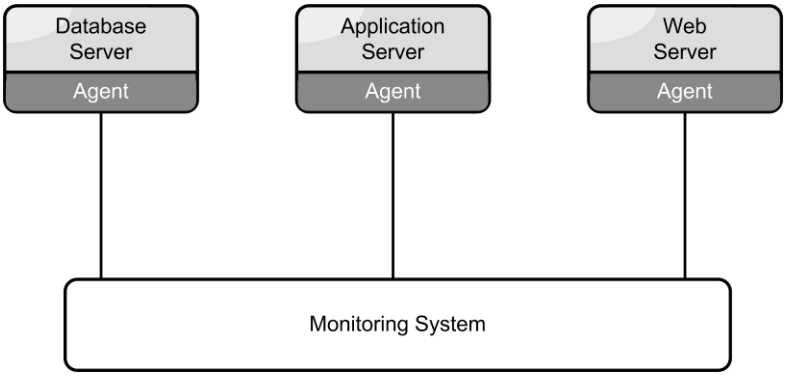


Figure 6. Architecture of agent-based monitoring approach. Dedicated agents are installed on monitored systems.

The agent-based approach is dedicated to monitor critical business systems that require high availability of all the processes and sometimes have unique metrics that need to be monitored. The approach provides in-depth information and helps diagnose the systems in relatively short time. This can significantly reduce the services downtime when outages happen to occur. On the other side, this approach is complex to manage and maintain as it requires additional resources such as disk space and CPU on each of monitored server. Furthermore, distributed systems availability monitoring will not work without additional modules. A lightweight implementation of agent-based approach is also available and sample implementation was examined by Dobre [11].

3.3 Agentless Approach

The agentless approach provides monitoring solution using built-in system protocols such as Simple Network Management Protocol (SNMP) or technologies like Windows Management Instrumentation (WMI) [61]. This means there is no need to install additional software on monitored systems as built-in monitoring features can be easily enabled and configured. Because of the architecture design, this approach provides systems availability monitoring capability without extra modules.

Maintaining the software for agentless solution only requires updating it on monitoring system server. Therefore, implementing an agentless approach in distributed systems environment can be achieved in days or weeks instead of months and at lower cost. Generic architecture diagram for agentless approach is illustrated in Figure 7.

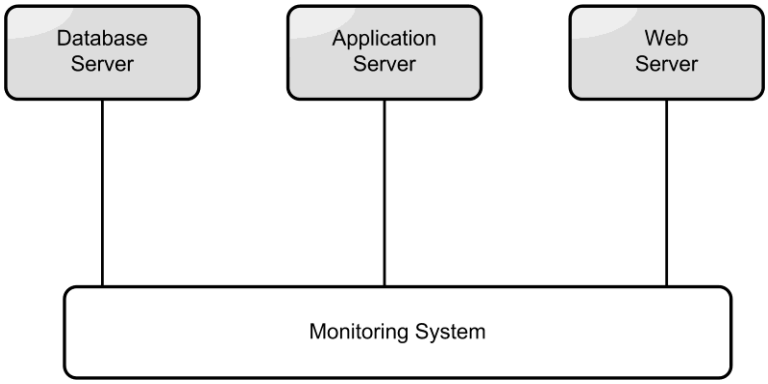


Figure 7. Architecture of agentless monitoring approach. Monitoring system uses built-in monitoring protocols and technologies. No additional software is required.

The agentless approach provides many advantages to IT operation departments. However, it is limited in the granularity and scope of metrics it can monitor. This is mainly due to the limitation of used monitoring protocols and technologies. Fulfilling that gap can be achieved by using platform specific diagnostic tools, like Windows Sysinternals [62].

3.4 Hybrid Approach

The hybrid approach offers flexible way of collecting monitored metrics by combining agent-based and agentless approaches [20]. It can operate as a lightweight agent and use system's schedulers like cron in Unix and Task Scheduler in Windows to execute a third party diagnostic program or a script written in any operating system supported language such as shell scripting language. The script can also use built-in monitoring protocols like SNMP or technologies like WMI similarly to the agentless approach.

The hybrid approach provides full flexibility in deploying and developing monitoring solution, and allows using best methods to gather monitored metrics. Depending on business needs, agents that collect in-depth data can be installed on monitored systems or agentless approach is used when basic system availability and performance metrics are required (see Figure 8 for architecture design).

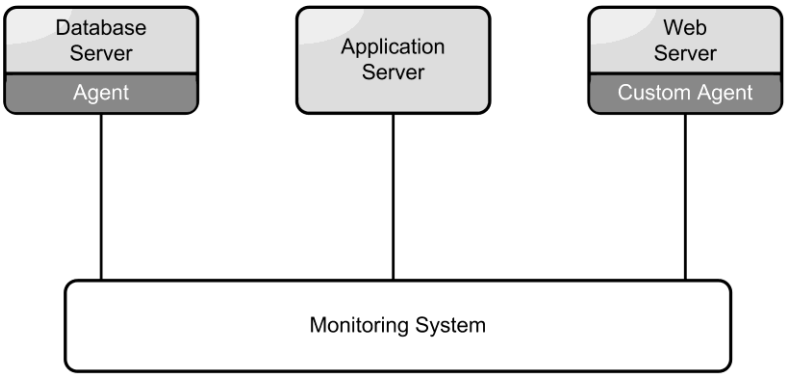


Figure 8. Architecture of hybrid monitoring approach. Each monitored system uses the most suitable monitoring approach to meet business requirements.

In typical monitoring solution deployment with hybrid approach, agents are installed on mission critical systems, while agentless approach is used for standard systems. Even the hybrid approach gives an extensible set of monitoring optimization techniques to IT departments; it may require expert knowledge in maintaining and configuring the entire solution, especially when developing custom scripts or upgrading the monitoring system software to the latest version.

The hybrid approach also allows quicker integration with monitoring solutions currently used in the organization. As it is adjustable by design, it provides multiple API interfaces and wide choice of extensions (aka plugins).

3.5 Data Streams Approach

The data streams approach was recently developed to meet requirements from business transactions and application services monitoring perspective. The key element in this approach is its integration with monitored application's source code through local agent called data forwarder (see Figure 9). The integration allows providing service monitoring including basic metrics such as transaction execution time, number of successful and failed transactions per minute as well as in-depth details from application error code stack when a failure occurs.

The data streams approach is a great offering for application services monitoring where its systems infrastructure is available in cloud environments [1] or in multiple geographic locations. As this approach focuses on end-to-end service monitoring, it can

measure in near real-time average transactions execution time that then allows faster outliers' detection when services encounter a failure. This monitoring scenario can also be implemented using traditional approaches, however it will require setting up individual monitoring instances for each downstream service and showing average transactions execution time might not be easily achievable.

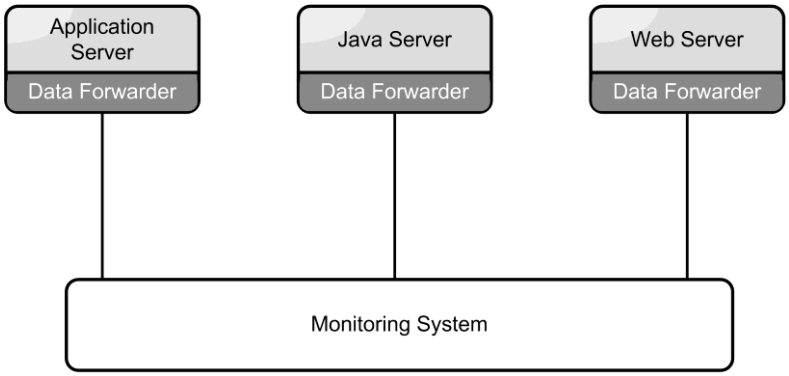


Figure 9. Architecture of data streams monitoring approach. The data forwarders send details of monitored metrics as information streams, mostly in real-time.

The data forwarder is installed on each monitored system and acts as a local agent. It sends monitored metrics as data streams to central monitoring system or generates a web page with all the metrics details that is frequently being scraped (usually over HTTPS protocol) by central or region specific monitoring instance. This is the only approach that allows IT departments monitoring business transactions in near real-time and configuring alerts based on trends and systems behavior. The data streams approach is very similar to agent-based approach; however it gives better interfaces to integrate with organization's business transactions and services.

This approach has become very popular in open source communities and many tools and ready solutions were recently developed free of charge. Another unique feature in this approach is a grouping mechanism. As this the most recent monitoring approach invented, it has advanced algorithms to prevent floods of events when a major failure happens. Many organizations, especially with large, distributed infrastructures and systems running in cloud environments will benefit because of that capability.

3.6 Comparison of Approaches

I examined the monitoring approaches currently available on the market over eight characteristics. Each approach was verified using one of the infrastructure tools evaluated in Chapter 4.

The characteristics I took into consideration are:

- *Platform dependency*. This feature presents flexibility of given monitoring approach. When the approach is platform dependent, it constitutes as additional software needs to be installed on monitored systems. The platforms I reviewed include Windows and Unix operating systems, as well as network and storage devices. When approach is not platform dependent, it allows relatively easy deployment and maintenance of the monitoring solution. However the list of metrics that can be gathered is limited to operation systems built-in monitoring protocols and technologies.

- *Availability monitoring.* This characteristic demonstrates duration of operational state of monitored resources. It requires the monitoring instance to be located outside of monitored systems and general availability metric can be provided by running simple command such as *ping*. In some scenarios, operational state of application or service can only be confirmed by checking if dedicated port is open, specified processes or services are running or the service returns known response. For example, web services are usually verified if known text is being displayed on the web page, or the HTTP status is returned as "200" or the SOAP request returns "OK" message.

- *Capacity monitoring.* This feature provides metrics that characterize performance and usage levels of monitored systems and applications, as well as number of servers, network devices and equipment's physical footprint in the datacenter. The standard set of performance metrics includes CPU usage, availability of free disk space and memory usage. Many organizations use capacity metrics to estimate future IT growths, expansions and next year's budget costs. Depending on monitoring approach, this is accomplished on general or in-depth level. The other advantage this characteristic gives is actual consumption of IT resources, which can help identifying under- and overutilized

systems and applications. Underutilized systems should be migrated to other platforms to save physical space in the datacenter, while overutilized would require additional research and perhaps redesign of current architecture.

- *Alerting and notifications.* Alerting plays a critical role in detecting a fault and notifying relevant support teams. The shorter time it takes to discover the error or anomaly, the quicker service recovery is possible and a failure is less impactful to the organization. The alerting module uses baselines or static thresholds to report on systems statuses such as availability, if a metric is within expected result, in warning or in critical state. Various platforms and metrics can have different alerting criteria and levels. Notification module usually represents a method of making aware and notifying support teams. Although, sending an email is the most popular notification technique, all the monitoring approaches offer many more ways to inform about the alert condition. This includes sending an SNMP trap, executing a custom script or remote command, custom integration with organization's service desk tool over dedicated API, or lately sending SMS text message, or even making an automated phone call.
- *Monitored data granularity.* A monitoring approach with greater level of monitoring details is recommended when it is being deployed for mission critical systems and applications. More details allow quicker root cause analysis when support teams are notified about a failure. On the other side, larger number of monitored metrics details will directly affect the size of data that are being collected, transmitted over the network, and stored. When deploying a monitoring solution using given monitoring approach, the organization needs to decide how often the detailed metrics have to be gathered, how long the collected data should be retained before being archived and finally after what time the data should be deleted. Those decisions must be taken before any monitoring approach and monitoring solution are chosen.

- *Monitored data gathering mode.* There are three modes how the data can be gathered from monitored systems.

  The first mode, Push, is used based on time-driven scenarios, such as send collected data every five minutes or event-driven situations like send data when disk usage exceeds warning threshold. In this mode data are usually sent by agent running on monitored system to central monitoring system. Push mode is used in agent-based and data streams approaches.

  Request and Response is the second available mode for transmitting the data. In this mode, an instance of monitoring system sends a request to monitored system and awaits a response. When the monitored system gathers all the information to fulfill the request, the response is being sent back to the monitoring system. This scenario is typical for agentless approach that uses built-in monitoring protocols and technologies.

  The last mode is Pull. In this mode, local agent or system job scheduler (cron in Unix, Task Scheduler in Windows) collects data of all the required metrics and puts the results in a known place such as shared folder or into the message queuing bus. Then, the instance of monitoring system periodically checks the known place and gathers the results. This mode is typical for hybrid approach where lightweight agent or custom scripts are required by the organization to meet particular monitoring requirements.

- *Additional software required on monitored systems.* The additional software characteristic shows platform dependency of given monitoring approach. Furthermore, in large environments this step would significantly affect the deployment time and process of initial configuration of monitoring solution. In some situations, it would even be impossible to install additional software due to organization's internal policies or simply because the chosen monitoring solution does not support unique operating system. The additional software also means that dedicated software is required for each type of monitored resources. For example Microsoft SQL database agent would be unable to monitor Oracle or Mongo databases, Unix operating system agent cannot be installed on Windows system and so on. Monitoring of network devices may require dedicated appliance.

- *Solution type.* This feature presents overall impact on organization's IT resources usage including systems capacity such as computing power usage, disk space, and network's bandwidth usage as well as maintenance efforts from support teams' perspective. It is recommended to each organization to conduct a proof of concept (POC) before deciding and deploying the entire monitoring solution based on chosen monitoring approach.
- *Deployment and maintenance.* The last characteristic shows the level of complexity in deploying and maintaining the monitoring approach. This includes day-to-day operations, adding new systems to be monitored, expanding the existing monitoring solution to support new datacenters, regions and countries. Moreover, within the maintenance of monitoring solution support teams would need to establish data backup policies and schedules as well as plan future monitoring systems software patches and upgrades.

The results of my review are presented in Table 5. Neither approach provides a comprehensive solution. Even the hybrid approach that combines agent-based and agentless approaches will be unable to provide near real-time monitored metrics as data streams approach is capable of. The organization would need to first define the services that are critical from a business perspective, what is the infrastructure they are running on, how much downtime is acceptable before engaging a support team and finally, how long should the data be kept for. Based on this research, appropriate monitoring approach or approaches can be chosen and tested on a small set of systems as a proof of concept.

The POC results will give better indications of required business alignments to the chosen monitoring approach and a monitoring solution. The last features a monitoring solution should offer are software extensions (aka plugins) and integration capabilities with custom scripts or APIs. Many organizations may have in-house built applications that are crucial to their business. Having a monitoring solution with an open integration protocol will definitely play an important role in monitoring those in-house developed applications.

Table 5. Comparison of monitoring approaches.

	Agent-based approach	**Agentless approach**	**Hybrid approach**	**Data streams approach**
Platform dependency	Yes	No	Yes	Yes
Availability monitoring	No	Yes	Yes	Yes
Capacity monitoring	Yes	Yes	Yes	No
Alerting and notifications	Yes	Yes	Yes	Yes
Monitored data granularity	In-depth, full	General, limited	In-depth, full	In-depth, full
Monitored data gathering mode	Push	Request and Response	Push, Request and Response, Pull	Push
Additional software required on monitored systems	Yes	No	Yes - for in-depth data No - for standard data	Yes
Solution type	Heavy, lightweight	Lightweight	Lightweight	Lightweight
Deployment and maintenance	Difficult	Easy	Intermediate	Intermediate

Chapter 4

Monitoring Tools

4.1 Introduction

There are multiple commercial and open source monitoring tools available on the market. The tools are divided into two main categories. Tools that monitor security information events constitute first category [20]. The second category is infrastructure tools that monitor systems availability, capacity and performance [21].

4.2 Security Information and Event Management (SIEM) Tools

Security information and event management tools are becoming an essential part of organization's portfolio of IT solutions. Their main purpose is to collect security event logs, analyze gathered data and report on any vulnerability. Table 6 presents the tools I reviewed based on evaluation [17] performed by research company Gartner.

Table 6. Security information and event management tools for distributed systems.

Tool	License	Monitoring Approach	Cloud Support	Target Market Size	Unique Feature(s)
AlienVault USM	Open source, proprietary	Hybrid (agent-based and agentless)	Yes	Small and medium	Availability of open source version, open community for security threats
BlackStratus	Proprietary	Hybrid (agent-based and agentless)	Yes	Small and medium	Availability of Software as a Service (SaaS) model

EMC (RSA)	Proprietary, freeware	Agent-based	Yes	Medium, large and enterprise	Availability of freeware version for real-time network packet captures
EventTracker	Proprietary	Hybrid (agent-based and agentless)	Yes	Small and medium	Availability of SaaS model
Fortinet (AccelOps)	Proprietary	Hybrid (agent-based and agentless)	Yes	Small and medium	Applications availability and performance monitoring, availability of SaaS model
HP ArcSight	Proprietary	Hybrid (agent-based and agentless)	Yes	Medium, large and enterprise	Multiple integration plugins, availability of SaaS model
IBM QRadar SIP	Proprietary	Hybrid (agent-based and agentless)	Yes	Medium, large and enterprise	Availability of SaaS model and Infrastructure as a Service (IaaS) model
Intel Security ESM	Proprietary	Hybrid (agent-based and agentless)	Yes	Medium, large and enterprise	Integration with other Intel Security technologies
LogRhythm	Proprietary	Hybrid (agent-based and agentless)	Yes	Medium, large and enterprise	Contextual and unstructured search engine
ManageEngine Log360	Proprietary	Agentless	Yes	Small and medium	Focus on agentless monitoring approach, comprehensive Active Directory auditing
Micro Focus (NetIQ)	Proprietary, freeware	Hybrid (agent-based and agentless)	Yes	Medium, large and enterprise	Support of mainframe platforms, freeware version of Sentinel Log Manager
SolarWinds LEM	Proprietary	Agent-based	No	Small and medium	Easy deployment, good addition to other SolarWinds products
Splunk SIP	Proprietary	Agent-based	Yes	Medium, large and enterprise	Robust, real-time search engine, availability of SaaS model, multiple deployment options

Trustwave	Proprietary	Hybrid (agent-based and agentless)	Yes	Medium, large and enterprise	Multiple deployment options including vendor co-managed installations

During my review I verified following criteria:

- *License*. I checked each solution for availability of open source version.
- *Monitoring approach*. This feature characterizes how security event logs are being collected by the tool. When there are more approaches, the solution can better align to organization's needs; for example, agentless approach would avoid installation of additional software on monitored systems.
- *Cloud support*. Today more organizations run their applications in cloud environments. All the tools I reviewed have this feature now apart from the solution provided by SolarWinds.
- *Target market size*. The tools target two markets: small- and medium-sized (less than 500 systems to collect events from) and medium-, large- and enterprise-sized (more than 500 systems).
- *Unique feature(s)*. Distinctive features of reviewed solutions were described in this column.

Additional comments from my review are presented in further parts of this section.

4.2.1 AlienVault Unified Security Management (USM)

AlienVault offers complete suite of security management tools that includes log management and collection, asset discovery and inventory, vulnerability assessment and remediation, intrusion detection, file integrity monitoring and behavioral monitoring. The solution provides centralized configuration and management of all of its components. It also provides native products to incorporate security monitoring within Amazon Web Services environment. AlienVault hosts Open Threat Exchange community where everyone can discuss and share security threats information.

Additionally, AlienVault was the only vendor in my review that offers an open source version of its solution (OSSIM - open source security information management).

That version has reduced number of features such as no threat intelligence data, no correlation directives, basic reports and single server deployment architecture.

AlienVault USM is available as virtual and hardware appliances. The company offers a 30-day free trial version to test security monitoring, simple security event management and reporting, and fast deployment as a virtual appliance.

4.2.2 BlackStratus

BlackStratus offers three solutions: LOGStorm, SIEMStorm and CYBERShark. LOGStorm is powerful and flexible log management tool that collects, stores and reports on security events data. It is capable of correlating events in real-time and also has incident response management system. LOGStorm is available as virtual and physical appliances.

SIEMStorm provides management of collected security events that includes analytics, vulnerability correlation, threat intelligence integration, and reporting modules for PCI, HIPAA and SOX standards. It is available as software and as a virtual appliance.

CYBERShark offers SIEM capabilities in Software as a Service (SaaS) model and was designed for small organizations. The solution includes multi tenancy support, real-time attack visualization, vulnerability correlation and advanced reporting.

4.2.3 EMC (RSA)

EMC provides enterprise level solution that software's name has recently been updated to RSA NetWitness Suite. The solution focuses on real-time monitoring of security events through logs, network full-packet capture, NetFlow protocol and various endpoints. The platform is available as physical or virtual appliances. For small organizations, the vendor provides all-in-one hybrid appliances.

RSA NetWitness supports large and geographically distributed datacenters. It also has built-in incident response management module that allows support teams manage security threats. Recently, the vendor added new detection method based on behavior analytics, improved log retention policies and now fully supports AWS cloud through CouldTrail logs.

To start using this solution, EMC offers a free version of NetWitness Investigator tool that can capture and analyze network traffic from OSI 2-7 layers in a real-time. The free version has restrictions on number and size of storage collections and customer support.

4.2.4 EventTracker

EventTracker allows monitoring network traffic for suspicious activity via logs collections, event correlation, behavior analysis, dashboards and event source knowledge packages. It is available as a software download or in software as a service model.

The solution is relatively easy to deploy and maintain. Large number of included reports allows many organizations to stay on top of regulatory compliance standards. EventTracker offers whitelisting process for applications as well as for internal IP addresses, IP reputation integrations and alerting, and supports JSON data-interchange format.

EventTracker natively supports AWS and Azure cloud platforms and is also available in a 21-day free trial version.

4.2.5 Fortinet (AccelOps)

Fortinet has recently acquired AccelOps and now offers security event monitoring solution as FortiSIEM. The solution gives unified user interface to monitor and analyze network traffic in a real-time, as well as monitor applications availability and performance.

FortiSIEM gives also automated module for self-learning Configuration Management Database (CMDB) and event consolidation mechanism to prevent flood of events detected in near time. Additionally, vendor has created XML-based event parsing language that allows the solution to quickly analyze the logs and relatively easily adjust its configuration to meet organization’s requirements.

FortiSIEM is available as virtual appliance and in service as a software model. To test all the features, the solution is available in a 30-day trial program.

4.2.6 HP ArcSight

Hewlett Packard provides its security solution in three products: ArcSight Data Platform (ADP) for log collection, data management and reporting; ArcSight Enterprise Security Management (ESM) to monitor security events at large, distributed environments; and all-in-one appliance ArcSight Express for medium-sized organizations. The solution is available as software, physical and virtual appliances.

HP ArcSight offers user behavior analytics, real-time event correlation and flexible architecture which allow adjusting the solution to organization's needs and future growths. HP ArcSight also includes large number of third party plugins that are essential in integrations with organization's existing systems.

Additionally, the company provides two trial versions, first to check log collection and storage software (ArcSight Logger is available for free for one year), and second its SIEM platform in software as a service model (ArcSight SIEM available for free for 30 days).

4.2.7 IBM QRadar Security Intelligence Platform (SIP)

IBM offers comprehensive security platform that includes QRadar Log Manager, Data Node, SIEM, Risk Manager, Vulnerability Manager, VFlow Collector, and Incident Forensics. The platform delivers real-time correlation mechanism to identify high risk threats, attacks and security breaches as well as proactive analysis of existing risks and incident response capabilities.

The solution can be deployed using virtual and physical appliances, in infrastructure as a service (IaaS) and software as a service (SaaS) models. IBM also provides threat intelligence forum with quarterly issued report that contains best security practices how to protect organizations' critical assets.

Additionally, IBM QRadar solution is available as all-in-one package that would allow relatively easy and fast deployment in medium-sized organizations.

4.2.8 Intel Security Enterprise Security Manager (ESM)

The security information event monitoring solution provided by Intel Security constitutes of McAfee Enterprise Security Manager (ESM), Event Receiver, and the Enterprise Log Manager. The solution is available as software, virtual and physical appliances.

Intel Security ESM provides real-time capabilities of identifying and responding to security threats. The Event Receiver collects thousands of events per second and uses highly indexed database to quickly retrieve data for analysis. The Enterprise Log Manager stores and optimizes the collection of event logs and keeps audit trail of all activities.

4.2.9 LogRhythm

LogRhythm provides entire suite of security monitoring tools for medium-, large- and enterprise-sized organizations. The solution includes security intelligence platform, SIEM, security analytics, log management, network monitoring and forensics, and endpoint monitoring. It is available as software, virtual and physical appliances as well as all-in-one package for medium-sized organizations.

The LogRhythm SIEM tool was built to quickly identify security threats and vulnerabilities. Its analytics engine uses machine learning and other techniques to detect suspicious behaviors. The product also provides unstructured and contextual search capabilities as it has recently integrated Elasticsearch engine. The tool's user interface is very interactive and customizable.

Additionally, LogRhythm created risk-based-priority algorithm that helps security operation support teams setting the highest focus on most critical events. Based on embedded case and incident scenarios, the solution can automatically respond to identified threat and minimize human intervention.

4.2.10 ManageEngine Log360

The solution provided by ManageEngine combines two products EventLog Analyzer and ADAudit Plus into one platform. The solution is relatively easy to deploy including scenarios with geographically distributed systems. EventLog Analyzer provides simple log management, event correlation and over 1,000 of predefined reports including PCI DSS, HIPAA and SOX. ADAudit Plus focuses on collecting events and auditing Active Directory in a real-time.

Log360 offers in-depth file information monitoring on systems such as Windows file servers, EMC file servers and NetApp filers.

Both tools use agentless approach to gather security event logs and are available as software download and virtual appliance. The vendor offers a 30-day trial program to test all the features.

4.2.11 Micro Focus (NetIQ)

Micro Focus acquired NetIQ in 2014 and offers its security information monitoring platform as Sentinel Enterprise. The solution is based on Change Guardian that monitors hosts and audits shared files as well as Configuration Manager that controls organization's security compliance with industry standards. Additional modules such as threat intelligence, exploit detections and log management are also available.

The platform is relatively easy to deploy and maintain, even in large environments. It supports medium-, large- and enterprise-sized organizations, and was the only solution in my review that supports mainframe systems.

The solution is available as software download and as a virtual appliance. The log management tool (Sentinel Log Manager) is available in a 60-day trial version to test all the features. After this period the tool can still be used free of charge however with limited capabilities and event collection will be limited to 25 events per second (EPS).

4.2.12 SolarWinds Log & Event Manager (LEM)

The solution provided by SolarWinds consists of LEM Manager (log storage and management) and LEM Console (data display and search engine). It is a simple product comparing to other SIEM solutions and additional features would require purchase of other SolarWinds products. This solution was the only one in my review that currently doesn't support security event monitoring in cloud environments.

SolarWinds LEM allows in memory events correlation, provides file integrity monitoring as well as monitoring of USB devices. Its active response mechanism can block suspicious IP address, stop service or disable user account. Built-in reporting repository includes PCI DSS, HIPAA, SOX and many other security compliance reports.

SolarWinds LEM is available as a virtual appliance and is relatively easy to deploy in small- and medium-sized environments. The company offers also a 30-day trial version with all features enabled.

4.2.13 Splunk Security Intelligence Platform (SIP)

The platform provided by Splunk is composed of core product Splunk Enterprise and security addition Splunk Enterprise Security. The Splunk Enterprise collects and stores event logs and offers great search engine with its own query language. The logs are being forwarded (Splunk Forwarder agent) to centralized, indexed database (Splunk Indexers) and are available to end user through web interface (Splunk Search Head).

The Splunk Enterprise Security component provides set of security dashboards and event correlation rules. It also gives access to multiple compliance reports, real-time visualizations and alerting mechanisms.

The solution can be deployed in organization's infrastructure, in public or private clouds, and in a hybrid infrastructure. It is also available in software a service model. To validate all available features, the vendor offers multiple trial versions such as Splunk Enterprise for 60 days, Splunk Cloud for 15 days, and Splunk Enterprise Security in Cloud for 7 days.

4.2.14 Trustwave

Trustwave targets its security platform offering to medium-, large- and enterprise-sized organizations. Their solution consists of SIEM Enterprise and Log Management Enterprise (LME). SIEM Enterprise offers powerful, advanced correlation engine, and intuitive, web browser-based user interface. The tool has 2,600 available reports including 600 compliance-focused, which can be scheduled or run ad-hoc. Log Management Enterprise is responsible for collecting and storing security event logs.

The solution is available as virtual and physical appliances, and Trustwave co-managed SIEM with 24 x 7 staffed experts. Additionally, LME is available as an AWS advanced metering infrastructure.

4.3 Infrastructure Tools

The main goal of infrastructure monitoring tools is to provide availability, capacity and performance metrics of monitored systems and applications. The data for monitored metrics are collected using agent-based approach, agentless approach, hybrid approach and nowadays more frequently with data streams approach. Table 7 shows comparison of tools currently available on the market.

Table 7. Infrastructure monitoring tools for distributed systems.

Tool	License	Monitoring Approach	Alerting	Cloud / Custom.	Target Market Size	Unique Feature(s)
AppDynamics	Proprietary	Data streams	Email, SMS, API	Yes / Yes	Large and enterprise	Software as a Service
Datadog	Proprietary	Data streams	Email, SMS, API	Yes / Yes	Small, medium and large	Supports multiple cloud platforms and DevOps teams collaboration
Ganglia	Open source	Agent-based	Optional via Nagios	Yes / Yes	Large and enterprise	Clusters and grid support

Graphite	Open source	Data streams	No	Yes / Yes	Large and enterprise	Can handle 160,000 distinct metrics per minute
HP Operations Manager	Proprietary	Hybrid (agent-based and agentless)	Email, SMS, custom	No / Yes	Enterprise	Integration with other HP products
Hyperic	Open source, proprietary	Hybrid (agent-based and agentless)	Email, SMS	Yes / Yes	Small and medium	Easy deployment and configuration
IBM SmartCloud Monitoring	Proprietary	Hybrid (agent-based and agentless)	Email, SMS	Yes / Yes	Enterprise	Predictive analysis and reports
ManageEngine AppManager	Proprietary	Agentless	Email, SMS, custom	Yes / Yes	Small and medium	Quick and easy deployment, application performance monitoring
Nagios	Open source, proprietary	Hybrid (agent-based and agentless)	Email, SMS, custom	Yes / Yes	Small, medium and large	Multiple plugins, wide community support, community customized versions
New Relic	Proprietary	Data streams	Email, SMS, API	Yes / Yes	Small, medium and large	Software as a Service, synthetic monitoring
Prometheus	Open source	Data streams	Email, SMS, API	Yes / Yes	Small, medium and large	Active community of developers and users
Riemann	Open source	Data streams	Email, SMS, API	Yes / Yes	Small and medium	Wide community support
Sensu	Open source, proprietary	Agent-based	Email, SMS, API	Yes / Yes	Small, medium and large	Configuration files automation via Chef and Puppet
TICK by InfluxData	Open source, proprietary	Data streams	Email, SMS, API	Yes / Yes	Small, medium and large	Software as a Service

Zabbix	Open source	Hybrid (agent-based and agentless)	Email, SMS, custom	Yes / Yes	Large and enterprise	Auto discovery, multiple plugins

Open source *license* tools would usually have wide community support and options to extend to enterprise model like Nagios and Sensu. Proprietary tools typically require vendor consultation when planning deployment for large environments, for example in datacenters with more than 500 systems. *Monitoring approach* characterizes how the data will be collected. When monitoring tool detects a failure, *Alerting* mechanism will be used to notify relevant support teams. A range of alerting methods allows better alignment to organization requirements and easier integration with existing service operations tools. Today many tools offer *Cloud support* feature which allows further monitoring integration with external providers. The tools I examined had options allowing individual *Customization* of the solution, such as execution of custom scripts, formatting of email alerts or development of custom plugins. *Target market size* column specifies the size of the environment the tool can monitor. *Unique feature(s)* of each tool were provided in the last column.

4.3.1 AppDynamics

AppDynamics offers an intuitive solution for applications performance monitoring that can be implemented in three models, in the organization's environment, delivered through Software as a Service (SaaS) model or in hybrid deployment. The data for monitored metrics is collected using data streams approach. Installation is easy and requires minimal configuration. Alerting module in AppDynamics solution uses machine learning algorithms to create dynamic baselines and to detect anomalies in the collected events.

Apart from applications performance monitoring, AppDynamics monitors web applications response time from end-user's browser or mobile phone perspective. This feature helps optimizing page load times and allows detecting errors through content checks when new version of the web site is being released. Additionally, AppDynamics provides a portal with more than 100 extensions that enable the solution to be quickly integrated with popular web application servers, cloud providers, database engines,

other monitoring solutions available in the organization as well as with service desk solutions and collaboration tools used by IT operation teams.

The company offers a 15-day free trial program to test end-to-end performance of applications running in distributed environments, get familiar with user interface and how to configure the tool to collect data from monitored systems and applications.

4.3.2 Datadog

Datadog solution was primarily designed to monitor cloud environments and seamlessly integrate with collaboration applications such as Hipchat, Chatwork or Slack used today by DevOps teams [50]. It works with more than 100 applications and systems that generate metrics which could be monitored. The Datadog agent is supplied on open source license which gives additional flexibility to the organization. The agent deployment can be automated using Chef application.

Datadog gathers application performance data, presents metrics in intuitive dashboards and notifies support teams through various channels when failure is detected. It also aims for live collaboration with interactive dashboards and DevOps collaboration tools. The tool offers access to collected data via documented API which allows additional integrations with other tools and dashboards available in the organization. Datadog would be also able to detect outliers and create an alert event when abnormal situation is identified.

The tool introduces interesting concept of tagging all the monitored resources. This makes the product easier to the end user when querying for metrics data and when building ad-hoc dashboards. The tags are assigned through four ways, inherited from integration (for example Amazon Elastic File System will be tagged by filesystem name), in the configuration file, in the user interface and when using API.

Similarly to AppDynamics, Datadog offers a 14-day free trial program to monitor as many servers as it is requested.

4.3.3 Ganglia

Ganglia was designed at Berkeley academic campus. It main objective was to collect metrics in near real-time for largely distributed systems (up to 20,000 hosts and a requirement to monitor CPU usage every 10 seconds [24]). Ganglia gathers in-depth operating system metrics using modularly designed agent. The agent is based on small plugins, which for example query the "proc" filesystem on Linux platform or operating system Management Information Base (MIB) on OpenBSD to collect CPU metrics. The sFlow agent is used to gather information from network gear such as routers and switches. Due to its design, agent's code is compiled at the installation process and as a result the deployment footprint and overall performance are better optimized comparing to traditional agents.

Ganglia uses XML language for data representation, External Data Representation (XDR) standard [51] for compact data transport and RRDtool for data storage and visualization. Implementation in Unix/Linux environment is robust however on Windows systems it requires (and is limited by) Cygwin libraries.

Ganglia architecture includes three daemons: gmond, gmetad, and gweb. Each daemon is self-contained and will start, and operate without existence of other(s). To make the solution successful, all three daemons need to be installed and running. The gmond daemon is responsible for collecting the data from monitored systems while gmetad gathers the data from gmond agents and stores in RRDtool database solution. Finally, gweb daemon presents the collected data and gives user interface to support teams.

Ganglia was successfully implemented by multiple companies in public and private sectors.

4.3.4 Graphite

Graphite is an open source monitoring solution that stores numeric time-series data and renders graphs based on those data. It is an enterprise ready solution, highly scalable and visualizes monitored metrics gathered by other applications. The solution can handle approximately up to 160,000 distinct metrics per minute. Graphite requires additional plugins to provide alerting mechanism.

Graphite consists of three components, carbon – a daemon that listens for time-series data, whisper – a simple database for storing time-series, and webapp – an application that renders graphs on demand. Depending on organization needs, the database engine can be replaced by alternative software, such as Cyanite, InfluxDB or OpenTSDB.

Graphite does not support RRDtool as a storage solution because of two reasons. The RRDtool requires regular insertion of the data. If there is no data available, the metric will be stored as zero. This may be misleading on visualization of some graphs. The Graphite was designed to support various application metrics that do not always occur regularly. The second reason is capability of inserting multiple data points at the same time. At the time Graphite was invented in 2006, RRDtool was able to insert only single value into database at a time. Considering high scalability and performance, Graphite needed a database which allowed multiple database insertion at a time. Despite RRDtool now supports multiple insertions feature, it is still not supported by Graphite due to the first reason.

Graphite can easily integrate with multiple data collection tools, data forwarders, data storage alternatives, and other monitoring solutions such as Ganglia, Sensu, Icinga, Shinken, and Riemann.

4.3.5 HP Operations Manager

HP Operations Managers provides additional module to HP portfolio of enterprise applications. This solution collects data from physical and virtual infrastructure using agent-based and agentless approaches. It also provides integration mechanism for third-party tools such as network monitors. The built-in discovery module speeds up solution deployment time as it recognizes managed nodes and automatically configures monitoring rules.

The tool provides intuitive user interface and operates as client-server solution. Deployments for large instances of distributed systems would require IT expertise and HP consultation. HP Operations Manager requires additional plugins, so called Smart Plug-Ins (SPIs), to collect detailed information about infrastructure, operating systems and applications.

HP Operations Manager offers proactive performance monitoring which would be very beneficial when planning and budgeting future infrastructure costs. Additionally, the solution gives single operations console that consolidates all detected events – a very useful feature for operation support teams.

The company offers a 60-day free trial program that allows testing all functionalities of this solution. In my review, this product was the only one which didn't support infrastructure monitoring in public clouds, such as Amazon Web Services (AWS).

4.3.6 Hyperic

Hyperic solution is available in two versions, open source vRealize Hyperic Open Source and paid version vRealize Hyperic. The solution was designed to monitor virtual and physical environments using agent-based and agentless approaches. Its key component Hyperic Agent automatically discovers metrics such as memory, CPU usage as well as applications being hosted on that system. Collected details about resources are presented in Hyperic user interface (aka Portal). Hyperic also provides interface for remote services management; it can start and stop application service, perform housekeeping functions on the database, or allows its users to write custom control action scripts.

Hyperic installation and configuration is easy and takes minutes. Alert notifications can be delivered as email, SNMP trap, SMS and integrated with organization's incident management system.

The commercial version of this solution offers enterprise support and has more features for example automated execution of corrective actions when a failure is detected or execution of control actions based on the schedule. Previous versions of Hyperic were named Hyperic HQ (open source), Hyperic HQ-Enterprise (commercial) and vFabric Hyperic (commercial after VMware acquisition).

4.3.7 IBM SmartCloud Monitoring

IBM monitoring solution includes Tivoli Monitoring and Tivoli Monitoring for Virtual Environments. It provides holistic view of monitored resources in the cloud including, availability, capacity, and performance. Even the product's name suggests cloud environment monitoring; this solution can be employed to monitor physical and virtual environments available within the organization.

SmartCloud solution has improved analytic modules, and capacity and reporting tools. This gives better reporting capabilities, including dynamic usage trending and health alerts for all monitored resources. User interface offers multiple role-based views, such as cloud admin, capacity planner, and application's owner. Built-in dashboards provide quick view to entire monitored platform while in-context menu gives an opportunity to quickly drill down through the gathered data.

The solution has also 'what-if' capacity analysis that help modeling future changes in the environment, hence reduce the risk of systems availability and performance bottlenecks. This feature would also optimize infrastructure growth and investment plans. Furthermore, the solution provides capabilities of setting policy-driven actions, for example to enable workload placement for performance and security optimization.

The vendor offers a 90-day free trial version of SmartCloud Monitoring with all functions enabled.

4.3.8 ManageEngine AppManager

Deployment of ManageEngine AppManager is relatively easy and quick. The solution is available in three price categories, free of charge for monitoring five systems and two paid versions depending on size of the infrastructure that the organization wants to monitor. The Enterprise version of this solution is highly scalable with failover capabilities and supports distributed systems monitoring in multiple geographic locations. Both paid versions are also available in a 30-day free trial program.

ManageEngine AppManager is continuously being developed to monitor as many new systems as they become available on the market. Following this principle, the tool monitors new databases such as Cassandra, Couchbase, Redis and MongoDB, a software distribution solution Docker, messaging broker RabbitMQ, storage solution Ceph and big data Hadoop clusters.

As AppManager supports multiple platforms and systems it can easily be deployed in organizations with heterogeneous and geographically distributed environments. The tool provides an interface for execution of custom scripts that can collect any required data. Further integrations with organization's existing systems can be achieved through REST APIs for fetching data from AppManager.

4.3.9 Nagios

Nagios launched in 1999 and is now one of the best-known open source systems for monitoring IT infrastructure. It is available in two versions, free Nagios Core, and commercial Nagios XI. The Core version has limited set of monitoring features; it can send an alert notification by email, SMS or run a custom script. Its web interface is very simple and provides basic reporting capabilities.

Nagios XI provides seamless installation as it only requires one file to be downloaded and one command to be executed. When the installation process finishes, the user can login to web interface and start configuring the solution as well as adding systems to be monitored. This version includes interactive dashboards with hosts overviews, services and network devices. It also provides improved reporting module of performance and capacity planning which helps organizations plan future infrastructure upgrades. Configuration of monitored systems can be easily audited as Nagios XI offers snapshot module that regularly saves application's configuration. Nagios XI is also available in a 60-day free trial program.

To monitor local or remote resources, Nagios Remote Plugin Executor (NRPE) agent needs to be installed. The tool also provides agentless monitoring capabilities with remote command execution through SSH protocol in Unix environments and through Windows Management Instrumentation (WMI) technology in Windows based operating systems.

Nagios has large community support with developers who update existing plugins and build plugins to monitor new systems. They also share best practices with product installation, configuration and maintenance. Additional plugins help especially Nagios Core users to expand monitoring capabilities without major software update.

4.3.10 New Relic

New Relic introduces Software as a Service (SaaS) model for monitoring solution and is available through the cloud and for the systems running in the cloud. This gives many benefits from infrastructure maintenance and cost perspective, however may raise a security concern for some organizations, especially in Europe. Deployment of this solution is very easy and the tool can also monitor on-premise applications.

Since the solution runs in the cloud environment, its main focus is to monitor applications performance from the code, mobile phone and web browser perspective. The metrics are collected through transactions and displayed in intuitive, interactive dashboards. It perfectly integrates with software development cycle as it provides vital insights of web application performance from multiple geographic locations and from multiple devices.

The solution is available in four pricing categories: application performance monitoring, mobile monitoring, browser monitoring and synthetic monitoring. All the options are available in a 14-day free trial program.

4.3.11 Prometheus

An open source Prometheus collects metrics through data streams approach. The gathered data are stored as time-series identified by metric name and key-value pair. Prometheus has modular design and requires Node Exporter agent on each monitored system to collect data of monitored metrics. The metrics are visualized through dashboards that are provided by PromDash module. Alertmanager module is used to configure alert notifications. The data gathering process is performed by scraping monitored system’s metrics website over HTTP protocol. Gathering via data push mode is also supported through an intermediary gateway server.

The solution's configuration files are written in YAML language that is relatively easy to read and understand. However, YAML interpreter is very sensitive and for example extra white space character can change the logic of an alert configuration and make the line to be ignored.

Prometheus has wide community support and is available for download as a complete solution through Docker image distribution. This offers great testing capabilities of the solution and minimizes the time required for initial installation and configuration. Moreover, the community created and shared best monitoring practices when deploying Prometheus, such as metric and label naming, creating dashboards, and configuring alerts.

4.3.12 Riemann

Rieman similarly to Prometheus stores data from monitored metrics as time-series. The solution was written for operation support teams to identify production systems failures and bottlenecks in near real-time, and at scale. The powerful processing language and great architecture design allow the collected metrics to be visualized within milliseconds instead of traditional solutions' times measured in minutes. The metrics are visualized using Graphite application. Alert events can be throttled and grouped into a single notification and delivered as email or as SMS and phone call after integrating with PagerDuty.

Riemann's configuration is written in Clojure programming language which makes the files concise, regular and extendable. That approach helps setting up complex situations and gives extra flexibility while deploying the solution.

Riemann gathers metrics from applications developed in many languages, including C, C++, C#, Erlang, Go, Java, Lua, Node.js, Perl, Python, Ruby, and Scala. Additionally, recently developed handlers and plugins allow this solution to collect metrics from Cassandra and Mysql databases, Chef and Puppet jobs results, other monitoring solutions such as Ganglia, Graphite, Nagios, and syslog-ng.

4.3.13 Sensu

Sensu similarly to Prometheus has modular architecture and offers a solution for public, private and hybrid cloud computing environments. Two versions of this solution are available on the market, open source Sensu Core and paid Sensu Enterprise. The Enterprise version has dedicated console that includes enterprise-level dashboard, role-based access controls, and multiple authentication methods such as LDAP, GitHub, GitLab, and API with tokens. This version has also already built-in third party integration plugins that simply reduce the time required for large deployments and improve solution's performance and security. All users of Sensu Enterprise have access to free annual training from the developers of Sensu as well as to enterprise-class support available on a 24 x 7 basis. Migration from open source to enterprise version requires additional packages download and installation.

Sensu modules' configuration is written in JSON format and uses dedicated TCP port for communication. To monitor local and remote systems, entire Sensu package needs to be installed on each server. Similarly to Nagios, Sensu collects monitored metrics data by executing local scripts. However, the trigger to execute the monitoring script is managed by local Sensu instance rather than central monitoring system like in Nagios. The script results are sent back to Sensu central console using RabbitMQ messaging bus. This transportation technology is also used to update Sensu agents monitoring configurations.

The company offers a 14-day free trial of Sensu Enterprise version.

4.3.14 TICK by InfluxData

InfluxData developed a monitoring solution based on four components: Telegrapf (T), InfluxDB (I), Chronograf (C) and Kapacitor (K). All of them make up a TICK stack. First component, Telegrapf, is responsible for collecting monitored metrics data and stores them as time-series. Second component, InfluxDB, delivers high performance database engine for writing time-series. Third component, Chronograf, visualizes the

data and while Kapacitor component manages the alerting mechanism including ETL processes and anomalies detection.

InfluxData monitoring solution is available as an open source version and as a paid version under InfluxEnterprise name. The latter is also available through AWS platform, which gives extra flexibility to organizations running majority of their systems in AWS cloud. The company offers a 14-day free trial of their enterprise software version.

4.3.15 Zabbix

Zabbix, the enterprise-class monitoring solution for everyone, monitors servers, applications, network devices, databases and virtual infrastructure using agent-based and agentless approaches. It is available as an open source product with addition of enterprise-level support and training program. Zabbix agent runs as a native system process that does not require programming environment like Java or Microsoft .NET. Zabbix also provides hardware monitoring for systems with Intelligent Platform Management Interface (IPMI), for example to gather details about temperature, fan speed, chip voltage, and disk state.

Zabbix installation process is relatively easy however configuration and maintenance may require expert knowledge. On a single node, Zabbix can monitor incredible number of up to 25,000 hosts with 100 metrics per host checked every minute. Zabbix user interface supports 25 languages and was designed to save users time and reduce the downtime of production systems.

The solution can be deployed in three ways, using virtual appliance, installing from package on Red Hat Enterprise Linux (RHEL), CentOS, Debian, Ubuntu and installing from the source code.

Chapter 5

Deploying Monitoring Solution

5.1 Selecting the Tool

As discussed in previous chapters, there are multiple monitoring approaches and monitoring solutions to choose from. Selecting the right approach and the tool is not a trivial task, especially when some financial investment needs to be arranged. In this section and chapter, I highlight key factors the decision maker should consider while choosing a monitoring solution. The recommendations include scenarios of distributed systems running in one local datacenter, systems hosted in datacenters in multiple locations and systems running in the cloud environments.

The most important factor is the *price* of the chosen solution. This includes the software license, the technical support, vendor consultation fees and staff training. Additionally, in large environments with more than 1,000 monitored systems, it is recommended to create a dedicated department that will look after all aspects of the monitoring solution, such as adding new systems to be monitored, documenting internal monitoring procedures and processes, deploying monitoring software updates and maintaining high availability of the entire solution. Majority of the paid solutions offer a trial version limited by the number of days. This gives the opportunity to test solution's functionally before purchasing the finally product. The market has also open source solutions that could be verified in parallel to the commercial products research.

The second factor to consider is *functionality and scalability*. Each organization will have individual monitoring features requirements and the infrastructure size the monitoring solution would need to monitor. The tools available on the market provide relatively comprehensive coverage of features although some are platform and domain specific. The two main categories all the tools are classified into are security and logs management tools (aka SIEM – security information and event management) and infrastructure and application performance monitoring tools. Today many organizations

are deploying new services and systems in the cloud environment and chosen monitoring solution should also support that environment. Furthermore, monitoring a system in the cloud may become difficult when the solution uses only traditional monitoring approaches. This is due to the up and down scaling mechanism which can create new server instance or destroy existing one within few minutes, depending on service performance demands. For traditional monitoring approaches that kind of behavior may create false alerts and engage support team resources unnecessarily, or may require constant overseeing of the available resources to keep the monitoring solution up-to-date. For systems deployed in the cloud and architected as services, it would be better to use a monitoring solution with data streams approach. That approach would provide a great set of metrics for business transactions monitoring and will adequately adjust to the results of up and down scaling mechanism. Apart from supported features, the decision maker should also verify minimum available monitoring polling intervals as tools using agent-based and agentless approaches tend to have that interval as one minute while products with data streams approach can monitor for example every five of 10 seconds. Finally, the monitoring solution needs to be scalable to support growth of the organization's business. The scalability of the monitoring solution will be discussed in further sections of this chapter; however the essential question is how the monitoring solution would work if the existing infrastructure would get doubled or tripled in terms of number of servers and network devices and some of them could be in other geographic regions.

The next key aspect in monitoring tool selection process is *alerting and integration* with existing systems in the organization. The traditional way of alerting IT support teams is by sending an email with alert's details. Except Ganglia and Graphite, this option exists by default in all the tools I examined. On top of email notifications, many solutions offer sending SMS text messages directly to the mobile phones (usually by third party providers) or API integrations with response management vendors such as PagerDuty and OpsGenie as well as group collaboration tools like Hipchat. Response management vendors unify the alerting platform and give full range of emergency notification protocols such as automated phone calls, SMS text messages, notifications through vendor's mobile application and traditional email. All notifications may also include follow up escalation ladder process in case there was no acknowledgment to the

original notification within defined period of time. Typically, when the monitoring solution is being deployed, the organization is already using some IT management tools like service desk, email platform, and user communication portals. It is advised to validate how monitoring solution can work with those systems as well as what are the possibilities to integrate with them. For example, alerts generated by monitoring solution can be directly recorded in the service desk system instead of sending simple email, or status of critical business services can be easily confirmed on the dashboard linked with internal communication portal.

The final factor to consider is *deployment and maintenance*. There are monitoring solutions that can be easily downloaded from the vendor's website and installed within few minutes, for example ManageEngine AppManager and Nagios. Some of the tools, especially supporting large and enterprise organizations, would require vendor consultations and assistance to achieve best results. Solutions that need additional software to be installed on monitored systems would definitely take some time to deploy and configure – the delay could be due to the number of systems as well as due to change management process if the organization follows ITIL best practices [14, 53]. While reviewing the monitoring tools, it is worth checking what automation capabilities it provides. This may include auto discovery of systems and services, automation on threshold and alert assignment, adding systems in bulk uploads, or automation in monitoring software configurations and upgrades – this is essential in large deployments with hundreds of monitoring agents. Lastly, it is recommended to discuss the responsibilities in regards to daily maintenance of the monitoring solution, obtaining vendor's technical support when needed, configuring scheduled downtimes of monitored systems to avoid false alert notifications and upgrading the tool to newer version when it becomes available.

5.2 Deployment in Local Datacenter

Choosing the monitoring approach and monitoring solution means choosing between consequences as neither approach nor tool can provide full and comprehensive solution to the organization. An agent-based approach offers an in-depth view of monitored metrics but its configuration and maintenance is not an easy task. An agentless approach

is a lightweight solution which gives a holistic view of the entire IT infrastructure. On the other hand, it offers only a general overview of the monitored environment. Data streams approach gives precise details, however only for business transactions and services monitoring. The monitoring approach for distributed systems environment should also provide capabilities of being able to be deployed in the organization that has only one datacenter as well as in the companies that have multiple datacenters and the datacenters can be in multiple geographic regions or in remote countries.

In this section I focus on one, local datacenter infrastructure and monitoring solution deployment with agentless approach and my novel, hybrid approach [20]. I don't include the agent-based approach's implementation due to its deployment and maintenance complexity. The sample deployment was validated in the infrastructure with 130 servers running Windows server operating system. The purpose of this research was to:

- monitor the system's security events,
- monitor the system's operational events,
- monitor distributed systems' availability, and
- allow customizable and lightweight implementation.

Before my monitoring research started, I took factors presented in Figure 10 into consideration. Those are the external dependencies that can interfere with the monitoring solutions results and are as follows:

- *Level 1 – Electricity.* The key component that enables entire infrastructure to run. When the datacenter experiences a power outage, the monitoring server becomes unavailable as well. To ensure IT support teams are still notified in that case, external verification of monitoring server needs to be implemented. This can be accomplished by deploying a single server instance in one of the organization's offices and enabling regular ping command test.

- *Level 2 – Network.* This level provides connectivity layer between monitoring server and monitored resources like server 1 till server N. This layer consists of network devices such as routers, switches, load balancers and firewalls.

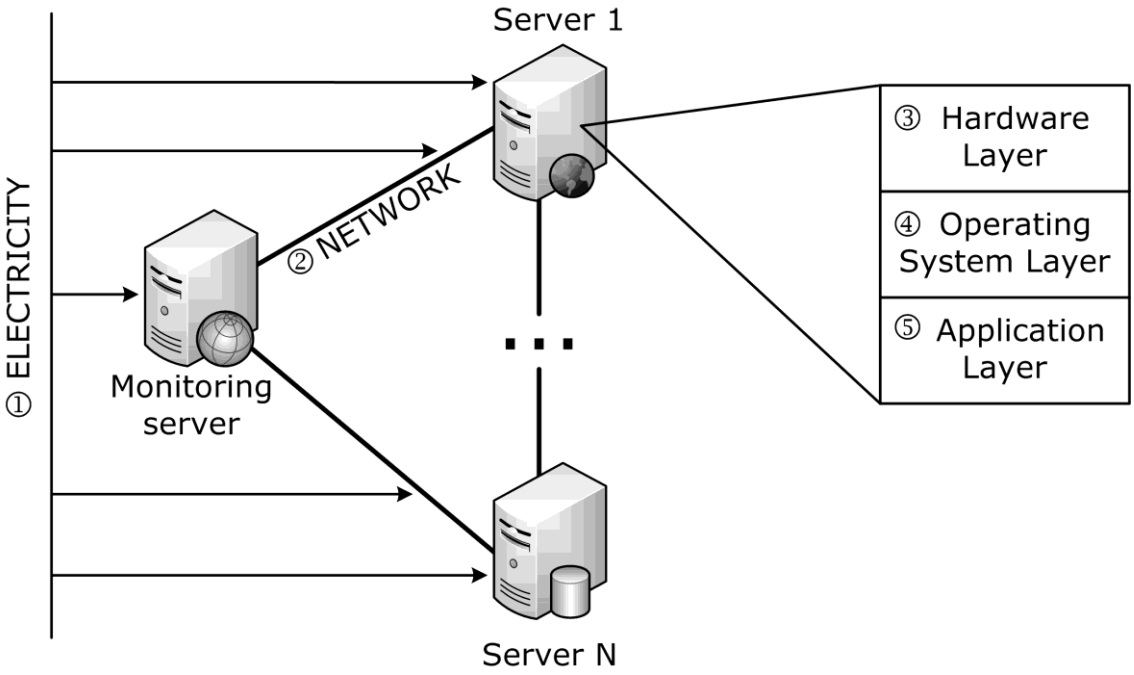


Figure 10. Dependency model for monitoring solution in local datacenter. Level 1 (Electricity) demonstrates core factor based on which all components are running. Level 5 (Application Layer) presents the top layer that is working only when previous levels are available and accessible.

Overall monitoring of those devices can be achieved by probing edge switches and routers or by checking the availability of sample servers that are connected to each of those switches.

- *Level 3 – Hardware Layer.* This level demonstrates physical devices from which the server is built upon. This includes computer chassis, motherboard, CPU, memory modules, disks, network cards, power supplies, and various fans. In enterprise monitoring solution, monitoring of listed hardware components should also be provided. Typical hardware monitoring is implemented using hardware's manufacture dedicated software throughout an agent-based approach or through custom scripts that parse the system logs.
- *Level 4 – Operating System Layer.* This level defines the environment in which applications will be running. The operating systems such as Microsoft Window server or Unix / Linux distribution can be installed

directly on the server and using all hardware resources or as a guest client through computer virtualization technology.

- *Level 5 – Application Layer*. The top level where dedicated software is deployed. Examples of the applications are database servers, web and application servers, file servers and email servers. Those applications are accessible through thick clients that are usually installed on users' computers or via web interface while using web browsers. When applications have hundreds of users and require high availability, the load balancing solution on network level needs to be deployed and connected to the application layer. The load balancing implementations will allow effective management of application's performance, as well as it will allow seamless deployments of new versions of the web applications.

First, I examined the monitoring solution with agentless approach. This approach doesn't need additional software on monitored systems. The data for required metrics can be gathered using operating system's built-in monitoring protocols and technologies. High level architecture design of this implementation is presented in Figure 11, while detailed data flow is shown in Figure 12.

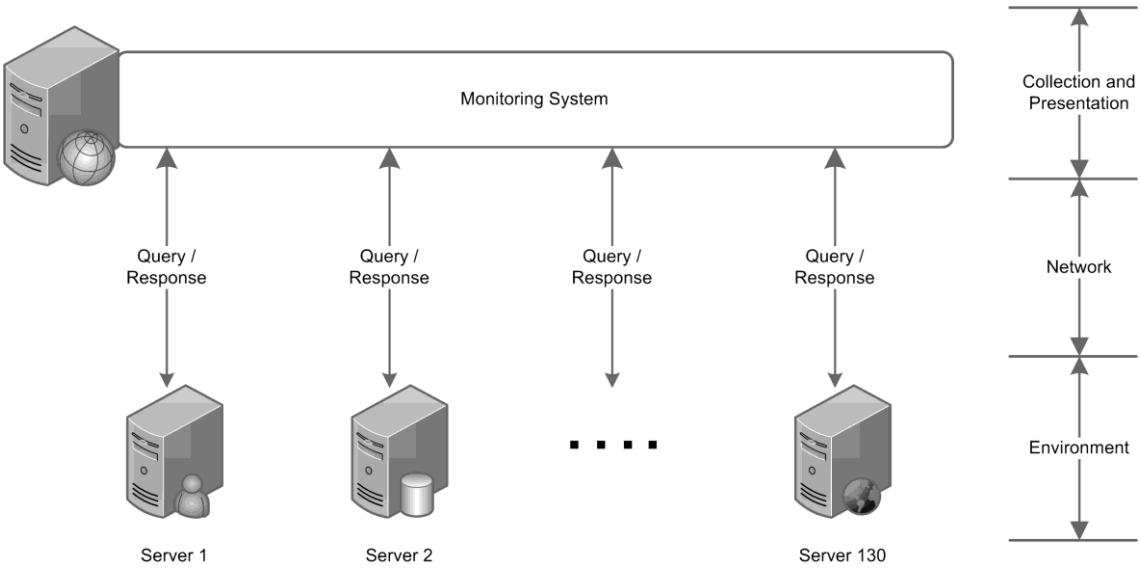


Figure 11. High level view of monitoring solution using agentless approach. Monitoring system queries individual servers to collect the data for monitored metrics.

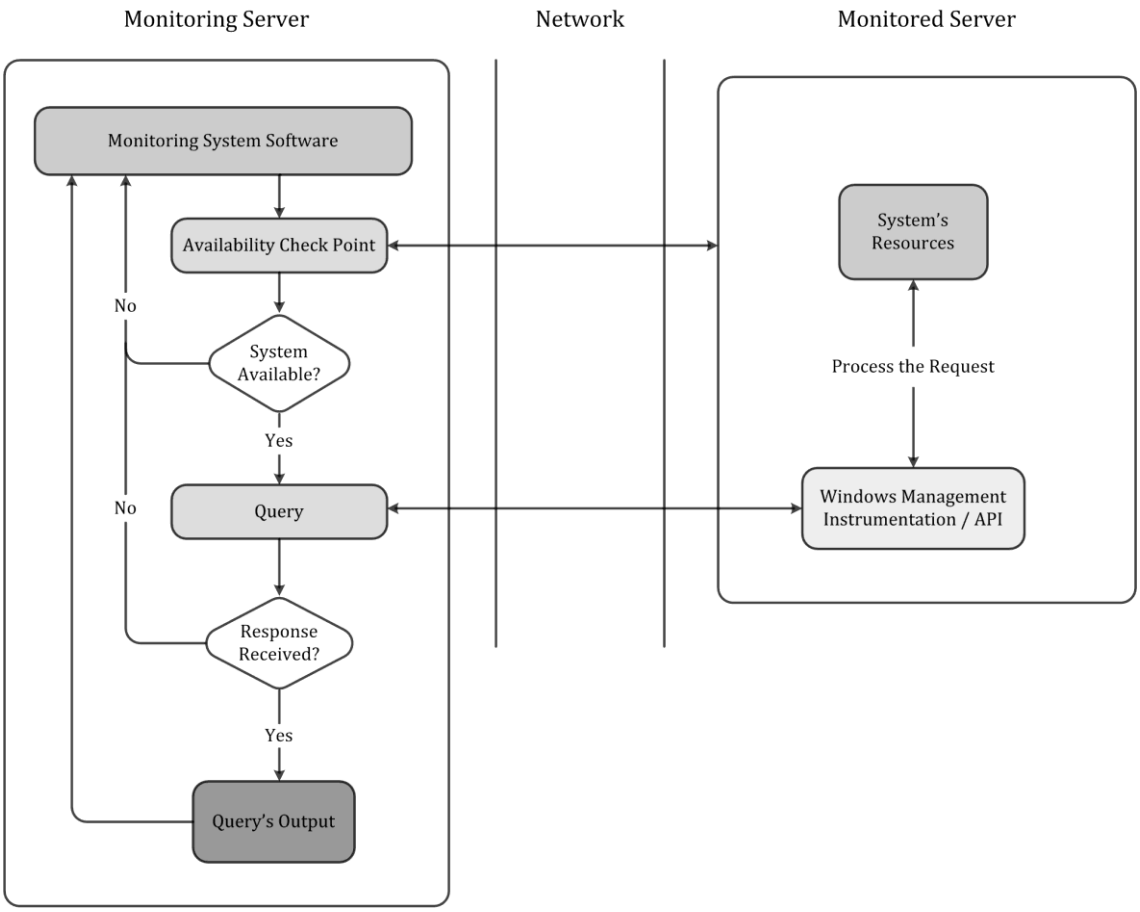


Figure 12. Details of data flow in monitoring solution using agentless approach. Monitoring system software verifies remote server availability before requesting data for monitored metrics.

The agentless monitoring starts on the server that runs monitoring system software. The first action is to verify if remote server is available. This is accomplished by using *ping* command and ensures the execution of further steps is possible or not. When availability check point fails, monitoring process finishes for that server. In situation the availability check succeeds, the monitoring system software polls remote server for one security event and three operating system events. The connection is being established through Windows Management Instrumentation (WMI) technology [61] or Sysinternals PsLogList program [62] installed on the monitoring server. Now, the monitored server receives ad-hoc instructions to report on current values of the requested metrics. The request is being processed and the data are being returned to the

monitoring server. The results can be imported directly to the monitoring system software or stored in a local file.

Secondly, I validated the same monitoring solution with a hybrid approach. During my study, I found that every time the monitoring system software requests for latest values of monitored metrics, it always calls the remote servers for the same metrics and at the same polling frequency. Thus, I proposed an order-based monitoring (OBM) - a hybrid approach based on idea of predefined and prefilled order, which eliminates the time of ad-hoc system queries as monitored metrics values are prepared in advance [20]. The sample order can be a list of following metrics, CPU usage, memory usage, verification of security logs if event 529 (Logon Failure - Unknown user name or bad password) has occurred in last 10 minutes etc. As the list and polling intervals are known, a scheduled task (Windows), cron job (Unix), or a lightweight agent can be configured with script execution to fulfill the order. The order's results are always placed in the same, shared location folder and the monitoring system software can access it anytime. The other key factor of the order is its customization capabilities and alignment with customer needs – only required metrics will be captured.

The hybrid approach with OBM integrates advantages of agent-based and agentless approaches into one solution, that

- provides in-depth information capabilities;
- is lightweight;
- is not platform or scripting language specific; and
- makes use of built-in monitoring technologies and protocols as well as allows usage of third party free products such as Windows Sysinternals [62].

High level architecture design of hybrid approach using OBM and monitoring sample set of 130 Windows serves is shown in Figure 13. In OBM, data can simply be obtained using operating system shell commands, built-in system tools and monitoring protocols, and it doesn't require software programming skills. OBM adds also business value as it focuses on exact metrics that need to be monitored and are relevant to the organization's business. This gives extra flexibility while deploying the monitoring solution with OBM as each organization will have specific, business critical applications requiring constant availability and unique metrics verifications.

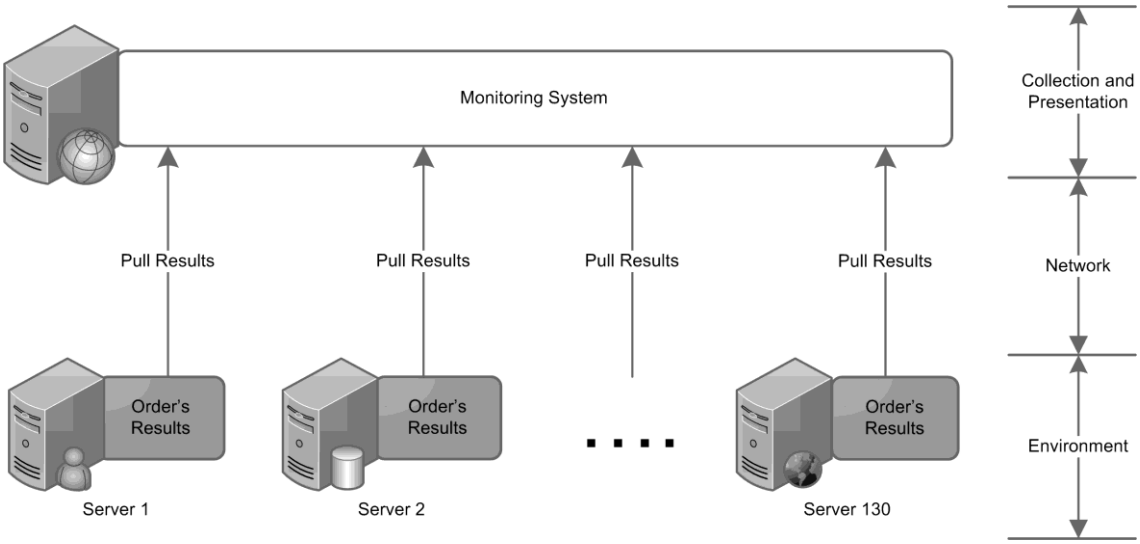


Figure 13. High level view of monitoring solution using hybrid approach order-based monitoring (OBM). The monitoring system collects already prepared latest values of monitored metrics (orders' results) from monitored servers.

As opposed to monitoring solution with the agentless approach, the monitoring process in OBM starts on the monitored server as presented in Figure 14. The job scheduler executes a shell script with order's definition on regular intervals. In this example the order was to collect the data for exactly same metrics as in agentless approach discussed above. The security events defined in the order were collected locally using WMI technology or Sysinternals PsLogList program. Once the script completes its execution, all the results are stored in one text file, in a known location available for another process running on the monitoring server.

In the next phase, another job scheduler on the monitoring server starts a set of processes to measure remote servers' availability and collect their order's results file. First the availability checkpoint process is run to validate if remote server is running. If not, this is recorded in completion status file and the collection process continues with remaining servers. When the monitored server is available, the download results process starts and using *xcopy* command it downloads order's result file onto the monitoring server. After this step, the download results process calculates the time difference between current time and the time when the result's file was created or refreshed. This is to verify if the job scheduler is updating the result's file on regular basis. The result of time difference is being added to the downloaded order's result file. In situations

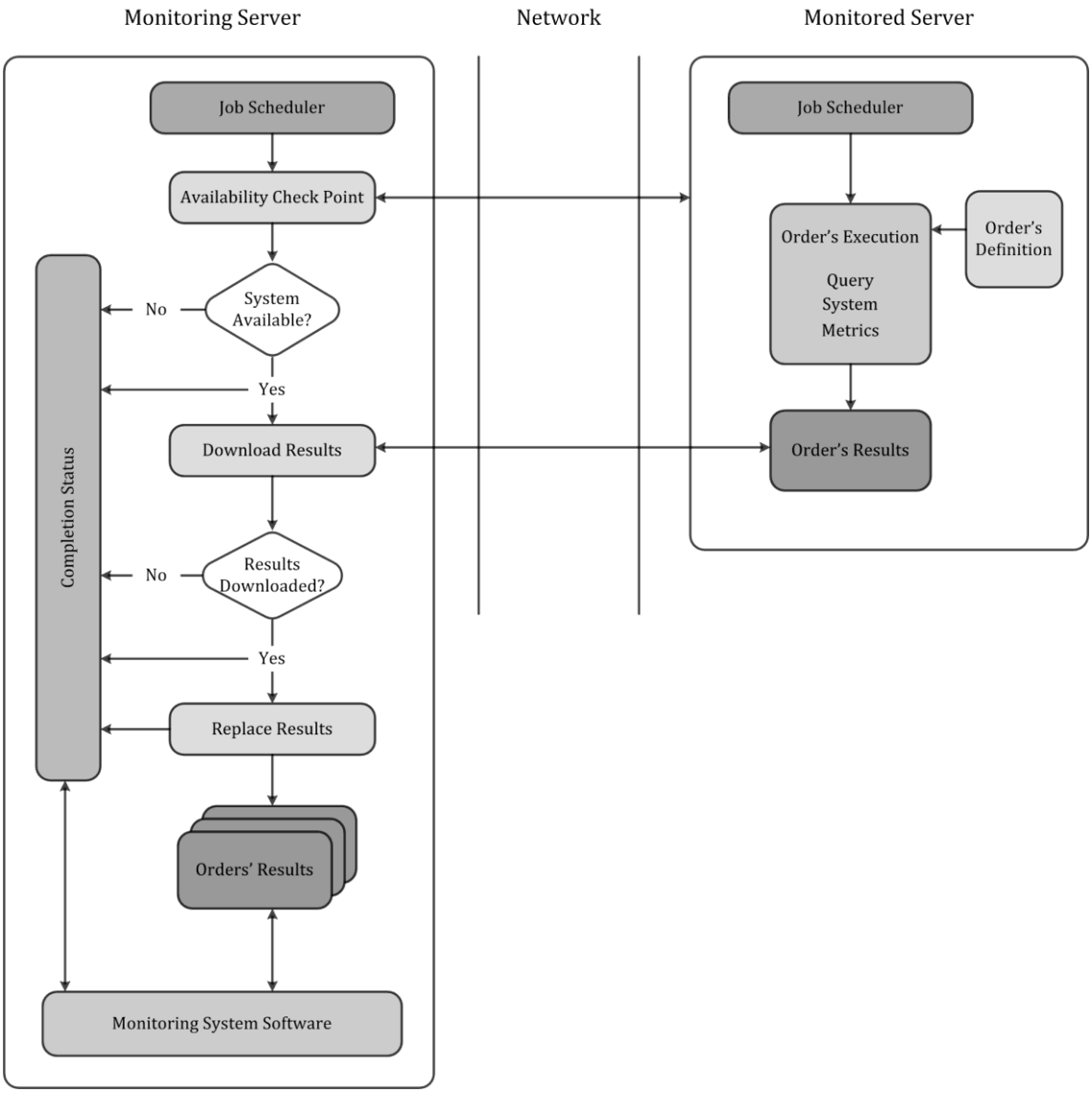


Figure 14. Implementation details of monitoring solution with OBM. Monitoring server verifies availability of remote server and then collects already prepared data of monitored metrics (order's results).

the order's file could not be transferred due to for example temporary network glitch or the file was missing, the relevant entry is provided to the completion status file. The last module, replace results, overwrites previously downloaded files so that the monitoring system software has the latest details about monitored metrics. The replacement action takes place when all the order's results files are downloaded – this improves performance of entire process and prevents the files from being locked. The final status is being updated in the completion status file that is periodically checked by monitoring

system software. There is only one completion status file and as many order's results files as many servers are monitored.

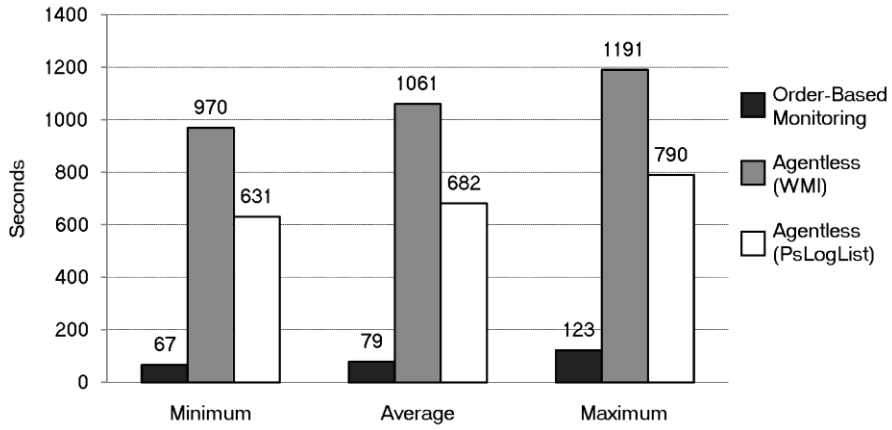


Figure 15. Comparison of cumulative collection time through various monitoring approaches. Data gathered for three months from 130 servers located in one datacenter.

I chose ManageEngine AppManager as monitoring system software for my research. The AppManager offers agentless monitoring approach and has scripting module that gives interface to integrate with hybrid solution like OBM. All monitored servers including monitoring server were in one datacenter. For three months, I measured the cumulative time, in seconds, required to run sequential monitoring process, starting from first server, carried on with the next one until the 130th server. The results of my work are shown in Figure 15. The cumulative time indicates how quickly can the monitored metrics be collected and analyzed. OBM approach accomplished much better results than the agentless approaches, as it was just collecting prepared data by the monitored servers. The cumulative time in OBM approach is the time it takes to copy the files from monitored servers onto the monitoring server. In the agentless approaches, the cumulative time also includes the time the monitored server requires to gather the data for the ad-hoc monitoring requests.

The OBM approach I discussed in this section gives flexibility that will be beneficial in each organization. The customization is achieved by adjusting order's definition and execution script(s) and because of that:

- It's scalable. Requesting new metrics to be monitored or removing not used metrics is easy and does not impact the cumulative time of collection process.

To keep the cumulative collection time unchanged when adding more servers to be monitored, it may be required to setup a second or more processes to collect order's result files in parallel.

- It supports in-house developed applications. Because of OBM scripting capabilities, it only requires the output of monitored metrics to be stored in predefined format, in order's result file. The data can be gathered through shell scripts, third party programs or by using built-in monitoring technologies and protocols.
- It's optimized on architecture level. It uses only one session connection to gather data of all the monitored metrics – the connection to download the order's results file. This feature is very helpful when deploying OBM approach in Unix environments where traditional monitoring approaches occupy multiple session connections.
- It's fast. In my sample research, it was eight times faster than the traditional agentless approach when comparing on the cumulative time needed for monitored metrics collection.

In the next section, I'm going to discuss OBM approach deployment in datacenters located in multiple geographic locations.

5.3 Deployment in Multiple Datacenters

Setting up monitoring solution in one, local datacenter is not a trivial task. However, deploying monitoring solution across multiple datacenters, located in remote geographic regions and countries adds additional complexity and the same methodology like the one used in local datacenter will not be as effective [23]. In local datacenter with local monitoring server(s) the response times to monitored resources are usually around one millisecond. This allows the monitoring tool to have as many connections to the servers as needed and the performance as well as network bandwidth will be negligibly affected. In typical datacenter, the bandwidth of internal network is measured in gigabits per second. The remote locations are connected using the wide area network (WAN) links (see Figure 16) where bandwidth is generally measured in megabits per

second. Additionally, due to the physics of the network media (aka network latency), the response times are from few milliseconds up to hundreds of milliseconds despite of available network throughput. Thus, the number of connections from monitoring server to remotely monitored resources needs to be as low as possible and the monitoring frequency needs to be adjusted to the level that it still meets the business requirements.

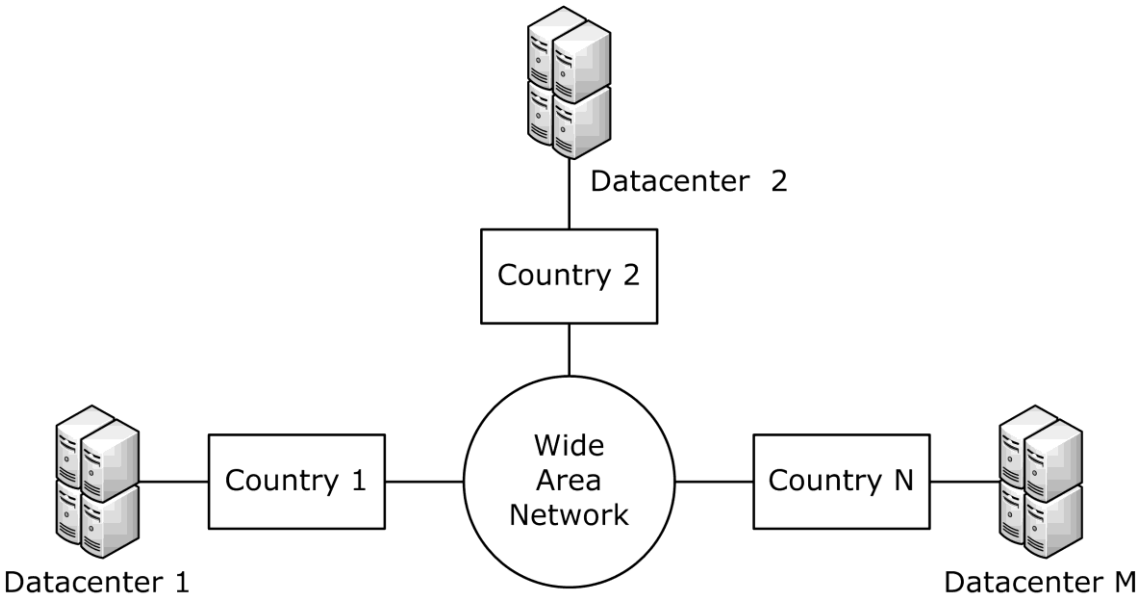


Figure 16. Monitoring solution architecture for datacenters located in multiple geographic regions. Monitoring server is for example located in Datacenter 1 and monitored servers are in Datacenter M.

When establishing WAN links, especially those that are transcontinental, the organization would need to allocate additional funding to cover the costs of network packets transmission over infrastructure of third party companies known as global network service providers. As already mentioned, even the link bandwidth may be hundreds of megabits; there is tangible network latency that may significantly delay overall monitoring process [8, 58]. As defined by Svoboda and his colleagues, network latency is *"a metric comprised of the sum of all small delay contributions along the data path between the two interfaces defined (...). Many parameters and variables influence the delay, especially if the measurement interfaces are separated by many hops."* [38]. Table 8 presents monthly average network latency in milliseconds on various worldwide network links that should be taken into consideration when designing monitoring solution for geographically distributed datacenters.

Table 8. Monthly average network latency measured in milliseconds (ms) using 64-byte packets and round trip times via the Internet Control Message Protocol (ICMP). Samples were collected in five minutes intervals [59].

Link Description	Median (ms)	Dec. 2016	Nov. 2016	Oct. 2016	Sep. 2016	Aug. 2016	Jul. 2016
Australia to UK	284.98	284.28	285.27	284.94	285.00	284.96	289.85
Australia to US	153.73	153.58	153.89	153.60	154.01	153.86	153.60
Brazil to US	113.16	116.79	113.87	111.84	113.08	113.23	111.88
Chile to US	103.35	103.32	105.63	99.88	107.18	101.97	103.37
Colombia to US	57.86	68.64	59.93	57.38	57.13	57.43	58.29
Hong Kong to US	158.20	156.95	156.31	155.79	161.65	168.49	159.46
Hong Kong to Singapore	32.36	32.48	32.14	32.16	32.24	35.75	32.90
Hong Kong to Sydney	112.33	114.13	112.35	112.30	112.22	112.30	127.41
Hong Kong to Tokyo	53.63	57.19	57.46	57.72	50.07	48.53	46.81
Hong Kong to UK	224.46	212.02	233.02	223.59	225.34	210.89	236.39
India to Hong Kong	97.86	98.43	97.28	95.11	100.44	101.42	95.53
India to Sydney	167.81	191.33	167.24	167.37	166.79	168.52	168.25
India to Tokyo	140.82	137.59	140.11	141.54	144.73	144.90	139.19
India to UK	116.79	116.05	115.16	117.53	119.30	113.34	151.07
Japan to UK	231.90	229.06	231.43	232.72	232.38	231.03	243.28
New Zealand to UK	276.12	282.14	275.44	259.61	268.89	276.81	306.29
North America to India	245.01	241.76	243.05	243.87	250.62	246.15	260.64
North America to Europe	85.90	84.76	85.32	86.06	85.86	85.95	86.24
Singapore to UK	181.02	180.88	182.40	180.81	181.15	180.62	183.11
Singapore to US	182.05	183.28	180.82	180.01	186.96	184.10	179.27
Sydney to Melbourne	12.36	12.32	12.37	12.56	12.37	12.35	12.35
Sydney to Perth	46.48	46.48	46.49	46.48	46.54	46.45	46.42
Sydney to Tokyo	119.08	118.82	118.24	120.68	119.34	119.38	115.61
Taiwan to UK	243.09	236.56	240.60	241.97	250.77	244.20	259.66
Trans Atlantic	74.22	74.22	74.21	74.20	74.23	74.21	74.22
Trans Pacific	102.88	102.89	102.88	102.76	102.89	102.76	114.53
Within Europe	11.21	11.16	11.21	11.21	11.21	11.26	11.30
Within North America	35.96	35.87	35.60	35.65	36.06	36.23	36.13

Table 9. Comparison of network parameters in local area and wide area networks.

Parameter	Local Collection	Remote Collection
Network type	LAN	WAN (VPN, MPLS)
Ping Response / Latency	Less than 2 ms	More than 2 ms
Bandwidth	100 Mbps - 10 Gbps	1 Mbps – 1 Gbps
Interruptions	Negligible	Multiple, External
Costs and Maintenance	Internal	Third Party

A quick comparison of local area and wide area network parameters is shown in Table 9. *Network type* demonstrates how IT equipment such as servers, storage and network devices is connected to the monitoring system server. This parameter has direct correlation with all the other parameters and determines the overall network performance. In LAN networks, the *latency* and *interruptions* such as packet drops are almost negligible due to the fact of short distance between all the devices. In opposite, in WAN networks latency and interruptions are more likely to impact the overall network performance and increase the time required to transmit the same number of packets comparing to LAN networks. Lastly, the *cost and maintenance* of the networks may have an influence on where the monitoring solution will be deployed.

In this section I focus on OBM hybrid monitoring approach deployment in multiple remote locations [19]. The main goal of my experiment was to evaluate how the network latency may impact the entire monitoring process. To examine this, I conducted a following experiment:

- A location was chosen in the USA (state of Arizona) with set of 100 servers to be monitored;
- Two other locations were selected where monitoring system could be installed, one in the USA (state of Washington) and second in Europe (Ireland).

All the servers in my experiment were running Microsoft Windows Server operating system. To simplify the test, each server to be monitored had a shared folder with three files (order's results) of various sizes: 1 kB, 4 kB, and 8 kB. Each file had sample security and system events details and access to those files was controlled by share folder's permissions and file system's access control list (ACL).

At the beginning, I measured the network latency by running ping command and gathering details of packets round trip response times from two locations where monitoring system was installed to the location where a set of 100 servers was situated. The results are as follows:

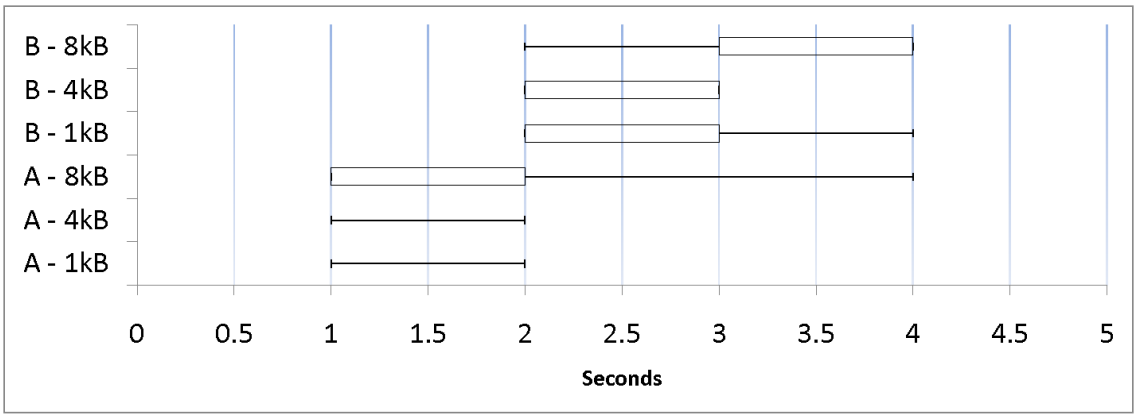
- From the USA location (Washington state) to the USA location (Arizona state) the average response time was 38.81 ms, based on 63,386 samples;
- From Irish location to the USA location (Arizona) the average response time was 144.31 ms, based on 31,966 samples.

The measurements from ping command revealed, that the location theoretically 4.4 times further away from the monitored servers had a network latency approximately only 3.7 times higher. The discrepancy appears to be due to a fact that, it is very difficult to measure the exact distance the packets traverse through the computer networks. Hence, I continued with the experiment and measured the time required to transfer the files from monitored servers onto the monitoring system server.

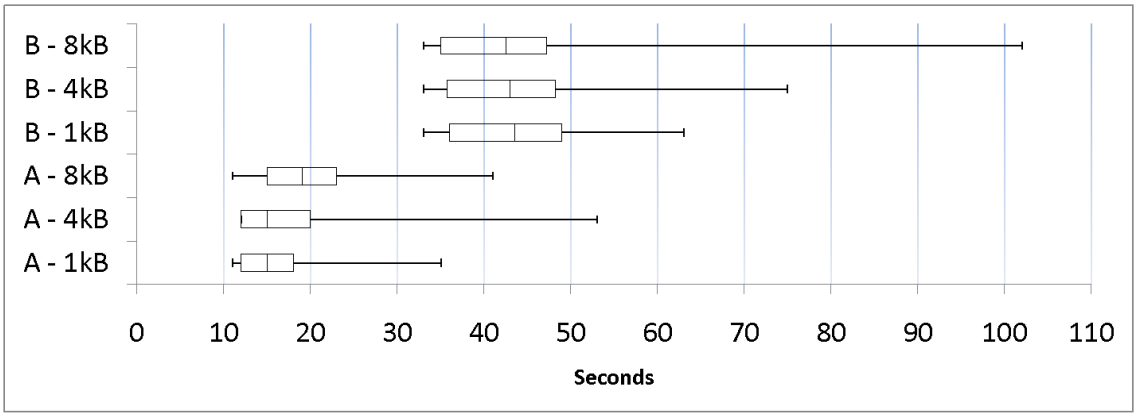
The process of transferring the order's results files in sequential manner was initiated simultaneously on the monitoring server located in the USA location (Washington state) and on the server in Irish location. The copying process was using *xcopy* command and was configured to run in three stages. The first stage was to copy the three files (1 kB, 4 kB, and 8 kB) from one server, then in the second stage files with same sizes from 10 servers, and eventually same size files from the entire set of 100 servers. WAN traffic optimization feature was disabled on the network devices during the experiment time that took two days. The copying process was executed 192 times from the monitoring server located in the USA (Washington state) and 96 times from the server hosted in Ireland.

The results presented in Figure 17 prove that transferring the same files over a longer distance will significantly delay the entire monitoring process. While the ping response times differed approximately 3.7 times (Washington state to Arizona state versus Ireland to Arizona state), the sequential process of transferring the files varied around 2.3 times. Furthermore, in my experiment sending the 4 kB files with events data took nearly same time as transferring the files of 1 kB data. An 8 kB files needed only around 10% more time to complete the copy process than 1 kB and 4 kB files.

A) Results from 1 server



B) Results from 10 servers



C) Results from 100 servers

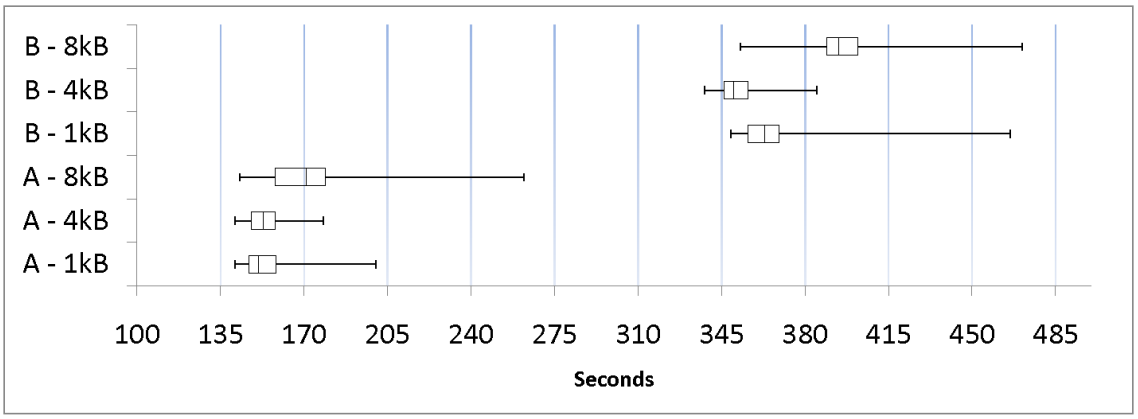


Figure 17. Results of transferring order's results files over geographically distributed locations. Figure A) shows minimum, median, and maximum time in seconds for files collection from one server. A-1kB represents the time of copying 1 kB files between two USA locations. B-8kB represents the time of copying 8 kB files from the USA location (Arizona) onto the server located in Ireland. Figures B) and C) show minimum, median, and maximum collection time in seconds from sets on 10 and 100 servers, respectively.

Additionally, the results revealed that establishing a monitoring process for a set of 100 servers located in the USA from a monitoring server located in Europe will take more than five minutes, regardless of the file size. The main culprit is the network latency and the number of connections that needs to be established to transfer the files from individual servers. In situations where there are more the one file to be copied from each server, the collection time will increase respectively as additional connections would need to be established. In my experiment, the five minutes polling interval recommended by US National Institute of Standards and Technology [18] could only be accomplished for a set of approximately 50 servers.

As network latency cannot be avoided, my research focused further on reducing the number of connections that need to be established with remote servers [6, 34]. Minimizing the connections count would limit the amount of SYN, SYN-ACK, and ACK messages between computers, as well as reduce the number of DNS and authorization requests; thus, reduce the time in the collecting the files. In [19] I proposed a concept of local Distributor (see Figure 18) that would play a role of an intermediary gateway.

The Distributor is available in a local datacenter and only establishes local connections with the servers. It collects the files on behalf of the monitoring system running in a remote location and once all the files are gathered, only one connection is being established with remote monitoring system server. This aggregation method allows fast collection of order's results files as it is using local-area networks instead of wide-area network. On the other hand, the Distributor introduces an extra layer in the entire events collection process, and additional mechanisms need to be built and maintained to ensure the processes on Distributor server are always up and running. It is also worth considering deploying secondary Distributor server in case of any scheduled or unscheduled maintenances on the primary Distributor node. Even the Distributor adds some delay to the events collection time, the benefit is tangible since only one connection is being established to exchange the data with remote monitoring system server.

Moreover, one of the Distributor processes can compress the data before sending them to the remote monitoring system server and reduce the amount of data being transmitted. In this situation, an extra process will be required on the monitoring system

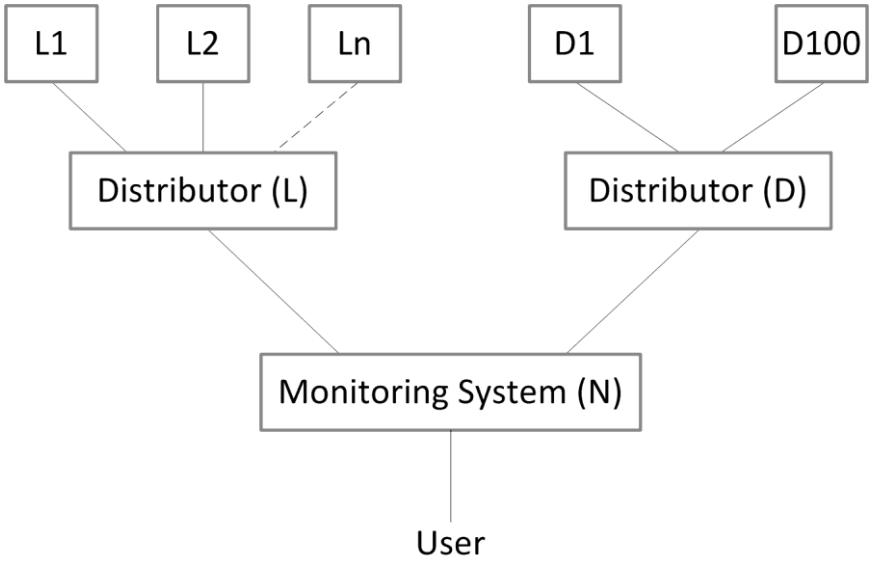


Figure 18. A concept of local Distributor in events collection process. L1, L2, and Ln are servers hosted in location L, while D1, D100 are in location D. Each location has a local intermediary server (Distributor) that collects data locally and sends them to monitoring system placed in location N.

server to uncompress the file received. Finally, the Distributor may filter the events already collected from monitored servers and using threshold policies it can further reduce the number of events and amount of data being sent over to the monitoring system server. A sample threshold policy may define actions to be taken immediately for a group of events collected such as forward the events to the monitoring system without waiting on the collection process to complete or send the events once per hour in a cumulative batch.

When assuming the monitoring process runs every five minutes for 100 servers hosted in a remote location, the monitoring system server would need to establish 28,800 connections on a daily basis. With implementation of local Distributor concept for the same set of servers, the number of connections will be reduced to 288 per day (reduction by 99%).

5.4 Deployment in Cloud

During my research period, distributed systems evolved to new architectural designs and started to be deployed in the cloud environments. This has created additional challenges to the traditional monitoring approaches and monitoring solution

deployments, as systems infrastructure in the cloud can be very dynamic and by its nature it will always introduce some network latency. Moreover, the cloud providers usually offer their own monitoring solution, that introduces another platform to learn, maintain, configure, and additional cost is required to get full access to its functionality.

In this section I focus on monitoring solution deployment in the cloud environment. To examine that scenario, I conducted a sample monitoring experiment with three popular tools: Nagios XI, Prometheus, and Sensu. The tools were chosen by following criteria: a) the software is available with open source license, b) at least one tool uses agent-based monitoring approach and one uses data streams approach, and c) the tools have established a wide community support. The experiment's goal was to verify how the tools I selected can monitor systems running in the cloud as well as how quickly can they notify support teams when a failure occurs.

All the tools were deployed in one datacenter, and each of them was installed on individual server using identical operating system version and configuration - CentOS 6.8. The installation process of Nagios XI was the easiest and relatively the quickest among all the tools. It only required one file to be downloaded from vendor's website and only one command to be executed to install entire solution. Configuring Nagios XI is fairly intuitive through its user interface (UI) available via web browser. Alert notification module, as in all selected tools, required a local or external SMTP server name and email account credentials. Nagios XI offers both agent-based and agentless monitoring approaches. The agent-based approach requires dedicated software named Nagios Remote Plugin Executor (NRPE) to be installed on monitored system, and also dedicated TCP ports to be opened. During NRPE installation, system's unique scripts are also installed to gather the data for monitored metrics. The NRPE module acts as a message broker agent that communicates between central Nagios server and the monitored system where it is installed.

Deployments of Prometheus and Sensu were more advanced due to their modular architecture and flexible configuration capabilities. Prometheus key components include:

- *core module* that collects the data from monitored systems, stores the data in time-series, integrates all modules into one system, and provides flexible query language to analyze gathered metrics;

- *Node Exporter* that extracts and publishes local monitored metrics in a time-series key=value format. This module needs to be installed on all systems in the monitoring scope. The list of available Node Exporters is constantly growing and includes SQL and multiple no-SQL databases, messaging systems such as RabbitMQ, storage solutions such as Ceph, HTTP web servers such as Apache, nginx, Varnish, and many APIs for AWS, Docker and others;
- *PromDash* that main goal is to provide visualization and dashboard capabilities of data stored by core Prometheus module;
- *Pushgateway* module to allow ephemeral and batch jobs to expose their metrics to Prometheus core module; this module helps collecting metrics of jobs that may not exist long enough to be accessed on regular polling intervals;
- And *Alertmanager* module that manages entire event management workflow when failures are detected; this module has also multiple filtering and event grouping capabilities to prevent from generating floods of event notifications to support teams.

Every Prometheus module has its own configuration file following human-readable data serialization language YAML and uses dedicated TCP port for communication. Due to YAML language requirements, semantics in configuration files are very sensitive and for example additional white space in alerts rules file may change the logic of the alert condition. To prevent this kind of situations, especially in alerts configuration rules, Prometheus website offers a syntax checker page that would moreover draw a picture of alert rules dependencies and workflow.

Deployment of Sensu monitoring solution was very similar to Prometheus installation. Sensu deployment started with installation of two key components such as Redis data-store (acting as database engine and cache) and robust messaging broker RabbitMQ. Then the installation process continued with installation of Sensu core package that includes Sensu client module. Finally, application Uchiwa was installed to provide simple dashboard visualization for Sensu monitoring framework. Similarly to Prometheus, each Sensu module has its own configuration file and dedicated TCP port for communication. The configuration files are written in JSON format as opposed to YAML in Prometheus. To complete monitoring solution's deployment process, Sensu

core package needed to be deployed on all monitored systems, however only Sensu client module was required to be activated. This step was essential as Sensu uses agent-based monitoring approach. The Sensu core package is available for all popular operating systems including Ubuntu/Debian Linux, FreeBSD, CentOS, IBM AIX, Solaris, Mac OS X and Microsoft Windows. Furthermore, Sensu operates similarly to Nagios XI as it collects the data for monitored metrics by executing shell scripts on remote system. The scripts are initiated by local Sensu client running on remote system, instead of central monitoring instance like in Nagios XI. Once the data are gathered, results are transmitted over RabbitMQ broker channel to central Sensu instance. The RabbitMQ is also used to push configuration updates from central Sensu instance to all connected Sensu clients running on remote systems – this expedites solution's management process and allows seamless configuration updates on all the nodes.

After three monitoring solutions were installed and running, I moved to the next phase of the experiment, to measure the time between failure occurrence in the systems running in the cloud environments and notifying support teams about it. The experiment was conducted on systems in Amazon AWS cloud and in Google Cloud. The systems were running CentOS or Linux operating system with required agents such as Nagios Remote Plugin Executor, Sensu client and Prometheus Node Exporter. The network traffic was secured by allowing inbound and outbound packets to traverse only between defined IP addresses, i.e. public IP addresses of servers in the clouds and public IP address of the datacenter where the monitoring solutions were hosted. As a sample failure, I selected system's CPU usage exceeding defined threshold value for a minimum period of one minute. To generate such an event, I used relevant cpuburn package [49] that constantly runs intensive floating-point unit (FPU) functions. The cpuburn application was scheduled to run every 12 or 15 minutes for the duration of four minutes. Each time cpuburn started, it started at random second of scheduled minute. The minimum monitoring polling interval available in Nagios XI is one minute, thus both Sensu client and Prometheus were configured to match that time accordingly.

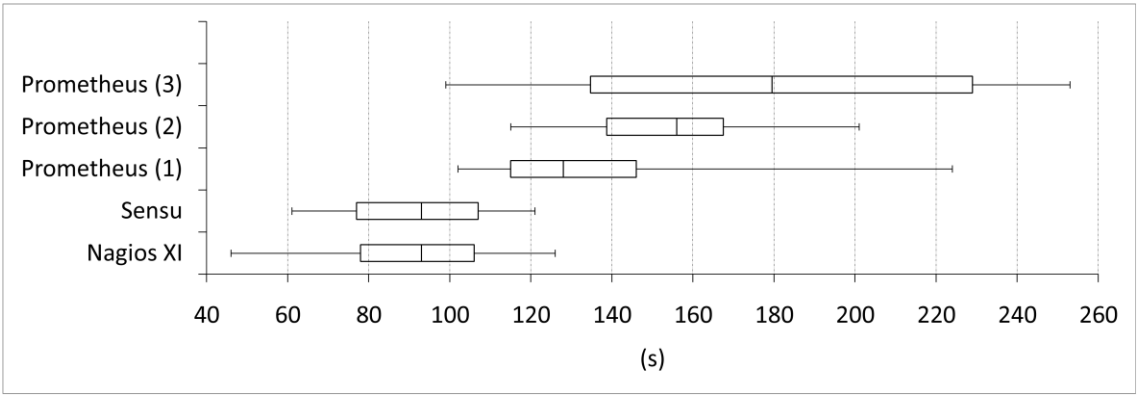


Figure 19. Experiment results of measuring duration between sample failure occurrence and notifying support teams. There are three sets of Prometheus results due to various alert configurations: (1) group wait 10s, group interval 2m, (2) group wait 30s, group interval 5m, (3) grouping disabled.

To configure polling interval of one minute in Prometheus, it was actually set to 30 seconds due to its design and storing data in time-series format. Alerting feature in all the monitoring tools was configured to trigger an email notification when CPU usage exceeds 95% threshold and stays in that condition for a minimum of one minute.

The experiment's results (see Figure 19) were collected from more than 400 scenarios and demonstrate that Nagios XI and Sensu median time to notify support team was 93 seconds since high CPU event occurred. The duration was measured as a difference between cpuburn start time and the time when notification was delivered to the email account. For Prometheus, the median time is higher due to the mechanism that calculates CPU usage metric based on time-series data, and varies as there were three different configurations tested in Alertmanager module. The Prometheus median times to notify are as follows:

- 128 seconds for group_wait configured as 10s and group_interval as 2m,
- 156 seconds for group_wait configured as 30s and group_interval as 5m,
- 179.5 seconds when grouping was disabled.

Group_wait and group_interval variables in Prometheus Alertmanager are used to group alert events and prevent a storm or flood of alert notifications that are sent to the support teams. The group_wait variable gives the extra time to wait before sending an initial alert notification, while group_interval holds particular time after initial notification execution to send a batch of new alert events that belong to the same group.

Overall, the monitoring solution deployment process to monitor systems in the cloud environment is very similar to the one when systems are distributed in multiple geographic locations. The key differentiating factor is to secure the network traffic to traverse only via allowed path(s) and TCP/UDP ports.

5.5 Common Issues

In this section, I discuss common issues that may occur during monitoring solution deployment. This includes issues that may arise due to the size of infrastructure that will be monitored or due to broad variety and complexity of organization's IT systems, and in-house developed business critical applications. Finally, monitoring solution's self-awareness or self-diagnostics is evaluated.

5.5.1 Scalability

During my research, I observed that deployment of monitoring solution for up to 50 systems located in one datacenter is relatively easy and does not require much infrastructure planning, network traffic analysis and proof of concept implementations. That is because any monitoring solution I reviewed, will only require one dedicated server and will be capable enough to monitoring all 50 systems, even at one minute polling interval for every metric. The situation changes when the infrastructure consists of more devices and may get very complicated in scenarios when more than 1,000 systems or even more than 10,000 systems need to be monitored.

The key factor the organization needs to consider while deploying monitoring solution in large infrastructure environments is the number of systems that the individual server with monitoring tool can support. This number will be higher in deployments using agent-based or data streams monitoring approach comparing to the agentless approach, and in agentless approach should not be greater than 300 systems per monitoring server. More than 300 systems in agentless approach is not recommended due to simultaneous connections the server needs to make to gather the data from monitored systems.

The other aspect in large and very large environments is a management of the platform that includes: configuration updates, adding large number of systems to be monitored, data maintenance and retention, central portal with names and references to all monitored systems, as well as resiliency capabilities (auto-failovers, planned maintenances) and performance of reporting module on collected data. The management of monitoring solution without templates or bulk edit module or lacking integration with automated configuration deployment software like Puppet or Chef, will be difficult in day-to-day operations as for example enormous amount of manual work will be required to perform any configuration update task at scale.

Finally, when assessing the scaling capabilities of a monitoring solution, the organization should also establish naming convention guidance for internal names of monitored systems (aka monitors) – the name, systems and their instances are being referenced to in the monitoring tool. This will help maintaining a monitoring platform at scale and allow support teams identifying required actions faster when alert notifications are being generated - the alert name and/or its details usually contain a monitor name.

5.5.2 Adaptability

Monitoring solution is usually being deployed as the last component in the IT infrastructure. When the organization is designing, and putting datacenter into operation, its major focus is on core services such as network, servers, operating systems, critical business applications including in-house developed ones, and adjusting existing processes and workflows to use new infrastructure. Hence, the monitoring solution needs to offer several integration capabilities, especially integration with organization's service desk solution that is used by multiple teams to manage queues of alerts from IT systems, service requests such as account provisioning, account membership, and IT support for the users.

Furthermore, today's distributed systems are running heterogeneous operating systems in multiple versions. This fact may affect monitoring solution's ability to provide consistent and the same results from monitored metrics, for example multi-core CPU usage on virtual and physical servers, while using Windows or Linux operating systems.

Next aspect in the monitoring solution's adaptability is integration with existing monitoring tools as well as with in-house developed applications. In some situations, organization might be already using one monitoring solution for basic infrastructure metrics, such as systems availability, CPU usage, memory usage and disk space usage. However, that solution is missing end-to-end transactions monitoring feature to monitor recently developed business critical application. In those cases, the monitoring solution should offer custom scripting capabilities, or integration with external plugins. The other scenario to consider is deploying second monitoring solution offering business transactions monitoring. Both solutions should cooperate and work in parallel, but never collect the same metrics twice.

Lastly, the more intuitive user interface the monitoring solution can offer, the shorter its entire deployment and staff training processes would take. Unfortunately, some of the monitoring tools I reviewed were requiring advanced and expert knowledge in systems operations and in popular scripting languages. Therefore, during the monitoring solution's proof of concept (POC) phase, the team that would maintain monitoring platform, should review and evaluate vendor's and users' tutorials, demos and opinions shared on solution's forums or blogs. This may prevent from making mistakes at initial stage of monitoring solution deployment as well as would give great learning opportunity, even before the solution is in production operation.

5.5.3 Self-diagnostics

After the monitoring solution is deployed and in operation, the organization measures its IT infrastructure health through the dashboards and alert notifications generated by the selected monitoring tool. The shift in trust from verifying status of individual servers to checking monitoring dashboards and consoles imposes great responsibility on the monitoring solution. Because of that, the organization needs to perform a final assessment, to make sure the monitoring solution is:

a) constantly running,

b) constantly collecting the data for monitored metrics,

c) reporting any anomalies within its operating status, and

d) taking immediate actions to prevent or minimize a downtime of its operation.

Achieving high availability of monitoring solution itself can be accomplished by setting up additional monitoring tool instances (at least one) in geographically dispersed locations. To ensure continues operation, those instances must monitor each other's critical processes such as availability through TCP/UDP ports validation, operational status of monitoring agents, database status through execution of sample queries and its server key metrics for instance CPU, memory and disk usage. The monitoring tool and agents' self-validations can be implemented within local monitoring instance. These checks guarantee the tool has access to the monitored systems; the systems are responding in expected behavior and in agent-based monitoring approach the agents are up and running.

Lastly, the organization may conduct a periodic failure scenarios tests such as disabling a monitoring agent service or revoking access to the randomly monitored system. Those examinations will prove monitoring solution's resiliency and will be invaluable in real case situations, especially in large infrastructure environments. Reducing the number of "surprise" cases will increase the trust and reliability factor of the chosen monitoring solution.

Chapter 6

Summary and Conclusions

6.1 Main Contributions of the Thesis

In this thesis, I presented the problem of selection, design and deployment of monitoring solution for security and system events monitoring in distributed systems environments, and ways of solving that problem. The distributed systems include heterogeneous platforms running Windows and Unix based operating systems in one or multiple geographically dispersed datacenters as well as in the cloud environments.

I defined the key factors to take into consideration when choosing a monitoring solution. I designed, developed and implemented a novel hybrid monitoring approach, order-based monitoring (OBM), to monitor on demand security and system events in distributed systems environments. Furthermore, I measured the network latency impact on overall monitoring process when distributed systems are geographically dispersed, and proposed a concept of local Distributor to improve monitoring solution's performance. When infrastructure failure occurs, I showed that the data streams monitoring approach would need more time to notify the support teams about it than the other monitoring approaches; assuming all the approaches use the same polling intervals.

All the research goals set in Section 1.2 were achieved and the main contributions of this thesis are as follows:

- Design and implementation of lightweight hybrid monitoring approach based on advanced order placement idea – order-based monitoring (OBM).
  [Section 5.2]
- Comprehensive survey of security and system events monitoring (SIEM) tools and infrastructure and application monitoring tools.
  [Chapter 4]

- Comparison and review of traditional and recently introduced monitoring approaches.
  [Chapter 3]
- Definition of key factors in the process of selecting a monitoring solution.
  [Section 5.1]
- Deployment of selected monitoring solutions in distributed systems environments, in one datacenter, in multiple datacenters, and in the clouds.
  [Sections 5.2 - 5.4]
- Experimental analysis of network latency and its impact on overall monitoring process performance in multiple datacenters scenario.
  [Section 5.3]
- Proposition of local Distributor concept to reduce network latency impact on remotely monitored systems.
  [Section 5.3]
- Introduction of and discussion about common issues that may occur during deployment and maintenance of a monitoring solution.
  [Section 5.5]

6.2 Direction of Future Research

The distributed systems are evolving and now more often are being deployed in the cloud environments. The clouds offer an auto-scaling mechanism that gives new infrastructure flexibility, where virtual systems can be created or abandoned almost instantly. On the other hand, that feature creates new challenges to the monitoring solutions which traditionally are configured to monitor systems with lifespan of months or years. This area is worth further investigation and study.

In this thesis, I proposed and implemented OBM approach that was storing order's results files locally, on the monitored system. Further improvements and reductions of events collection time can be explored by introducing a shared folder on the monitoring system server, and then configuring the OBM job scheduler to put the

order’s results file there. This would eliminate the sequential phase of copying the order’s results files from all the monitored servers. Moreover, it would allow additional research towards OBM approach gathering the data below one minute interval. The polling interval of one minute was a common practice until the data streams approach was introduced to the market.

List of Figures

List of Tables

Monitoring Tools Web Pages

AlienVault Unified Security Management - https://www.alienvault.com

AppDynamics - https://www.appdynamics.com

BlackStratus - http://blackstratus.com

Datadog - https://www.datadoghq.com

EMC (RSA) - https://www.rsa.com

EventTracker - http://www.eventtracker.com

Fortinet (AccelOps) - https://www.fortinet.com

Ganglia - http://ganglia.sourceforge.net

Graphite - http://graphite.readthedocs.io

HP ArcSight - http://hp.com/go/ArcSight

HP Operations Manager - http://hp.com/go/Ops

Hyperic - http://www.vmware.com/products/vrealize-hyperic.html

IBM QRadar Security Intelligence Platform - http://www.ibm.com/software/products/en/qradar

IBM SmartCloud Monitoring - http://www.ibm.com/software/products/en/ibmsmarmoni

Intel Security Enterprise Security Manager - http://www.mcafee.com/us/products/siem/index.aspx

LogRhythm - https://logrhythm.com

ManageEngine AppManager - http://www.appmanager.com

ManageEngine Log360 - https://www.manageengine.com/log-management

Micro Focus (NetIQ) - https://www.netiq.com

Nagios - https://www.nagios.org

New Relic - https://newrelic.com

Prometheus - https://prometheus.io

Riemann - http://riemann.io

Sensu - https://sensuapp.org

SolarWinds Log & Event Manager - http://www.solarwinds.com/it-security

Splunk Security Intelligence Platform - https://www.splunk.com

TICK by InfluxData - https://www.influxdata.com

Trustwave - https://www.trustwave.com

Zabbix - http://www.zabbix.com

Bibliography

[1] Aceto G., Botta A., De Donato W., Pescape A., Cloud monitoring: A survey, Computer Networks, vol. 57, pp. 2093-2115, 2013.

[2] Andreozzia S., De Bortoli N., Fantinel S., Ghiselli A., Rubini G. L., Tortone G., Vistoli M. C., GridICE: A monitoring service for grid systems, Future Generation Computer Systems, vol. 21, pp. 559-571, 2005.

[3] Azemoon T., Becla J., Hanushevsky A., Turri M., Real-time data access monitoring in distributed, multipetabyte systems, Int'l Conf. Computing in High Energy and Nuclear Physics, Journal of Physics: CS 119, 2008.

[4] Bearden M., Bianchini R. Jr., Efficient and fault-tolerant distributed host monitoring using system-level diagnosis, Research Report No. CMUCSC-96-1, Dep. of Electrical and Computer Eng., Carnegie Mellon Univ., 1996.

[5] Bellavista P., Corradi A., Stefanelli C., Java for on-line distributed monitoring of heterogeneous systems and services, Computer Journal, vol. 45, no. 6, pp. 595-607, 2002.

[6] Casalicchio E., Caselli M., Coletta A., Measuring the global Domain Name System, IEEE Network, vol. 27, no. 1, pp. 25-31, 2013.

[7] Dautriche R., Termier A., Blanch R., Santana M., Towards visualizing hidden structures, International Conference on Data Mining (ICDM) / PhD Forum, Spain, hal-01407664, 2016.

[8] De Vito L., Rapuano S., Tomaciello L., One-way delay measurement: State of the art, IEEE Trans. Instrumentation and Measurement, vol. 57, no. 12, pp. 2742-2750, 2008.

[9] Dev Mishra A., Beer Singh Y., Big data analytics for security and privacy challenges, International Conference on Computing, Communication and Automation (ICCCA), India, pp. 50-53, 2016.

[10] Diaz M., Juanole G., Courtiat J.-P., Observer - a concept for formal on-line validation of distributed systems, IEEE Trans. Software Eng., vol. 20, no. 12, pp. 900-913, 1994.

[11] Dobre C. M., Voicu R., Muraru A., Legrand I. C., An agent based framework to monitor and control high performance data transfers, Int'l Conf. Computer as a Tool (EUROCON 07), IEEE CS, pp. 453-458, 2007.

[12] Fatema K., Emeakaroha V. C., Healy P. D., Morrison J. P., Lynn T., A survey of cloud monitoring tools: Taxonomy, capabilities and objectives, Journal of Parallel and Distributed Computing, vol. 74, no. 10, pp. 2918-2933, 2014.

[13] Guo C.-G., Li X.-L., Zhu J., A generic model for software monitoring techniques and tools, Proceedings of 2nd Int'l Conf. Networks Security Wireless Communications and Trusted Computing (NSWCTC 10), IEEE CS, pp. 61-64, 2010.

[14] Ishibashi K., Maintaining quality of service based on ITIL-based IT service management, Fujitsu Scientific & Technical Journal, vol. 43, no. 3, pp. 334-344, 2007.

[15] Jennings N. R., On agent-based software engineering, Artificial Intelligence, vol. 117, no. 2, pp. 277-296, 2000.

[16] Katsaros G., Kousiouris G., V. Gogouvitis S., Kyriazis D., Menychtas A., Varvarigou T., A self-adaptive hierarchical monitoring mechanism for clouds, Journal of Systems and Software, vol. 85, no. 5, pp. 1029-1041, 2012.

[17] Kavanagh K. M., Rochford O., Bussa T., Magic quadrant for security information and event management, Gartner, August 2016.

[18] Kent K., Souppaya M., Guide to computer security log management, US Nat'l Inst. Standards and Technology, http://dx.doi.org/10.6028/NIST.SP.800-92, 2006.

[19] Kufel L., Network latency in systems event monitoring for multiple locations, Scientific Programming, vol. 2015, article ID 371620, 2015.

[20] Kufel L., Security event monitoring in a distributed systems environment, IEEE Security & Privacy, vol. 11, no. 1, pp. 36-43, 2013.

[21] Kufel L., Tools for distributed systems monitoring, Foundations of Computing and Decision Sciences, vol. 41, no. 4, pp. 237-260, 2016.

[22] Lee S., Levanti K., Kim H., Network monitoring: Present and future, Computer Networks, vol. 65, pp. 84-98, 2014.

[23] Massie M. L., Chun B. N., Culler D. E., The Ganglia distributed monitoring system: Design, implementation, and experience, Parallel Computing, vol. 30, no. 5-6, pp. 817-840, 2004.

[24] Massie M., Li B., Nicholes B., Vuksan V., Monitoring with Ganglia, O'Reilly Media, 2013.

[25] Montesa J., Sánchez A., Memishi B., S. Pérez M., Antoniu G., GMonE: A complete approach to cloud monitoring, Future Generation Computer Systems, vol. 29, no. 8, pp. 2026-2040, 2013.

[26] Nithya M., Komathi A., Enhancing security and event management using association rule mining, International Journal of Science and Research (IJSR), vol. 5, no. 8, pp. 985-990, 2016.

[27] Obermaisser R., Peti P., Elmenreich W., Losert T., Monitoring and configuration in a smart transducer network, IEEE Real-time Embedded System Workshop, 2001.

[28] Poggi N., Carrera D., Gavaldà R., Ayguadé E., Torres J., A methodology for the evaluation of high response time on e-commerce users and sales, Information Systems Frontiers, vol. 16, no. 5, pp. 867-885, 2014.

[29] Qin Z., Rojas-Cessa R., Ansari N., Task-execution scheduling schemes for network measurement and monitoring, Computer Communications, vol. 33, no. 2, pp. 124-135, 2010.

[30] Rastogi R., S A., G S., G P., D P., Singh A., Design and development of generic web based framework for log analysis, IEEE Region 10 Conference (TENCON), Singapore, pp. 232-236, 2016.

[31] Ravindran B., Engineering dynamic real-time distributed systems: Architecture system description, language, and middleware, IEEE Trans. Software Eng., vol. 28, no. 1, pp. 30-57, 2002.

[32] Robinson W.N., A requirements monitoring framework for enterprise systems, Requirements Eng., vol. 11, no. 1, pp. 17-41, 2006.

[33] Saiedian H., Wishnie G., A complex event routing infrastructure for distributed systems, Journal of Parallel and Distributed Computing, vol. 72, no. 3, pp. 450-461, 2012.

[34] Sedlar U., Volk M., Sterle J., Kos A., Sernec R., Contextualized monitoring and root cause discovery in IPTV systems using data visualization, IEEE Network, vol. 26, no. 6, pp. 40-46, 2012.

[35] Smit M., Simmons B., Litoiu M., Distributed, application-level monitoring for heterogeneous clouds using stream processing, Future Generation Computer Systems, vol. 29, pp. 2103-2114, 2013.

[36] Subramanyan R., Miguel-Alonso J., Fortes J., Design and evaluation of a SNMP-based monitoring system for heterogeneous, distributed computing, tech. report TRECE 00-11, School of Electrical and Computer Eng., Purdue Univ., 2000.

[37] Suh-Lee C., Jo J.-Y., Kim Y., Text mining for security threat detection discovering hidden information in unstructured log messages, IEEE Conference on Communications and Network Security (CNS), USA, pp. 252-260, 2016.

[38] Svoboda P., Laner M., Fabini J., Rupp M., Ricciato F., Packet delay measurements in reactive IP networks, IEEE Instrumentation & Measurement Magazine, vol. 15, no. 6, pp. 36-44, 2012.

[39] Terenziani P., Coping with events in temporal relational databases, IEEE Trans. Knowledge and Data Eng., vol. 25, no. 5, pp. 1181-1185, 2013.

[40] Tierney B., Crowley B., Gunter D., Holding M., Lee J., Thompson M., A monitoring sensor management system for grid environments, Proceedings of

The Ninth International Symposium On High-performance Distributed Computing, IEEE CS, pp. 97-104, 2000.

[41] Vaarandi R., A data clustering algorithm for mining patterns from event logs, Proceedings of the 3rd IEEE Workshop on IP Operations & Management, pp. 119-126, 2003.

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

[43] Vaarandi R., Pihelgas M., LogCluster - A data clustering and pattern mining algorithm for event logs, 11th International Conference on Network and Service Management (CNSM), IEEE Conference Publications, pp. 1-7, 2015.

[44] Vaarandi R., Platform independent event correlation tool for network management, Network Operations and Management Symposium, IEEE Conference Publications, pp. 907-909, 2002.

[45] Vaarandi R., SEC - a lightweight event correlation tool, IEEE Workshop on IP Operations and Management, IEEE Conference Publications, pp. 111-115, 2002.

[46] Yonatany M., Platforms, ecosystems, and the internationalization of highly digitized organizations, Journal of Organization Design, 6: 2, doi:10.1186/s41469-017-0012-3, 2017.

[47] Zanikolas S., Sakellariou R., A taxonomy of grid monitoring systems, Future Generation Computer Systems, vol. 21, pp. 163-188, 2005.

[48] Zulkernine M., Seviora R.E., A compositional approach to monitoring distributed systems, Proc. Int'l Conf. Dependable Systems and Networks, IEEE CS, pp. 763-772, 2002.

[49] Cpuburn package for Linux based operating systems, https://pkgs.org/download/cpuburn, February 2017.

[50] DevOps support teams, http://theagileadmin.com/what-is-devops, February 2017.

[51] External Data Representation (XDR) - Wikipedia, https://en.wikipedia.org/wiki/External_Data_Representation, February 2017.

[52] High availability - Wikipedia, https://en.wikipedia.org/wiki/High_availability, February 2017.

[53] ITIL - Information Technology Infrastructure Library, https://www.axelos.com/best-practice-solutions/itil, February 2017.

[54] Karim Vaes, The DTAP-Street: A phased approach to a development / deployment cycle, https://kvaes.wordpress.com/2014/10/26/, February 2017.

[55] Mean time to recovery - Wikipedia, https://en.wikipedia.org/wiki/Mean_time_to_recovery, February 2017.

[56] Request for Comments (RFC) 5424 - The Syslog Protocol, http://tools.ietf.org/html/rfc5424#section-6.2.1, February 2017.

[57] Request for Comments (RFC) 5674 - Alarms in Syslog, https://tools.ietf.org/html/rfc5674.html, February 2017.

[58] Submarine cable map, http://www.submarinecablemap.com, February 2017.

[59] Verizon network latency, http://www.verizonenterprise.com/about/network/latency/, February 2017.

[60] Windows event types, http://msdn.microsoft.com/en-us/library/windows/desktop/aa363662.aspx, February 2017.

[61] Windows Management Instrumentation and Simple Network Management Protocol, Microsoft TechNet Library, online resource https://technet.microsoft.com/en-us/library/bb742612.aspx, February 2017.

[62] Windows Sysinternals, utilities for servers running Windows operating system, https://technet.microsoft.com/en-us/sysinternals, February 2017.