I. J. Computer Network and Information Security, 2022, 1, 40-51
Published Online February 2022 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2022.01.04

System Monitoring Addon Analysis in System Load Simulation

Filip Gjorgjevikj
Faculty of Informatics, AUE-FON University Skopje, R. N. Macedonia
E-mail: filip.gorgevik@fon.mk

Kire Jakimoski
Faculty of Informatics, AUE-FON University Skopje, R. N. Macedonia
E-mail: kire.jakimoski@fon.edu.mk

Received: 05 June 2021; Revised: 10 July 2021; Accepted: 03 August 2021; Published: 08 February 2022

Abstract: The complexity of interconnected devices requires constant real-time monitoring, as failure of one part can have catastrophic consequences for the entire system. Computer-information monitoring tools enable us to always be one step ahead of potential problems that may occur in a monitored network environment, whether it is a human-caused configuration or simply an element has failed or stopped working. Not only can they report potential problems, but they can also solve the problem itself. For example, if an element needs increased resources at a given time, the tool itself can recognize it and automatically increase the resource needs of that element. By setting up a monitoring system in a virtual environment, the results can be seen and through their analysis will bring an optimal solution when it comes to what agent to use. This paper presents analysis of how network monitoring agent is responding in cases when there is increased use of shared resources. Knowing this can help in choosing what agent should be used in any given environment, and with that more resources will be saved. This leads to better utilization of resources which is an important in mid-size and big setup of computer monitoring systems.

Index Terms: Monitoring; workload; clients; analysis; system.

1. Introduction

With constant innovation and a constantly increasing trend of interconnected devices, monitoring tools are becoming more than a necessity in any information environment.

Monitoring systems [1] are responsible for controlling the operation of the information technology used to perform day-to-day operations to analyse performance, through which it can be predicted when a problem may arise in the equipment itself, whether it is hardware or software. Preventing an event that may cause operational problems is one of the primary goals of monitoring systems. A well-established monitoring system could set up monitoring of devices and their infrastructure, software that includes monitoring of applications, services and even business processes that are particularly important in the daily work of any institution [2].

A good monitoring system helps to increase productivity at work, because it can predict when and where a problem may occur. For example, if a component installed in the system shows signs of malfunction, a monitoring system will detect it. Monitoring system could even prevent a problem that has not yet occurred.

Some systems are such that they are modifiable and as such are great to use. Other systems can be modified to be used with third-party tools to create a more specialized monitoring system.

Section 2 presents a literature review. Section 3 presents configuration and setup of virtual machines used for test environment. In Section 4 it is explained what information will be collected and what kind of tool is in use for simulating workload on the CPU. Also, in Section 4, there are tables that shows results collected while system was tested. Section 5 describes what action can be taken to improve response, to improve and minimize impact on the whole system when shared resources are used. Section 6 gives conclusion of what kind of cumulative activities must be taken in consideration when the agent is used for network monitoring.

2. Literature Review

In article [1] authors want to present the monitoring systems in shared networking where accent is given to client-side monitoring where discussion is given to practical implementation issues. From the view of end-user point they want to give valuable insight into physical infrastructure without complex and ever heterogenous nature of monitored

devices. SNMP (Simple Network Messaging Protocol) still plays a vital role in any system monitoring applications. Authors in article [2] maintain the focus on Nagios evolution, as an open-source flexible monitoring tool for enterprises. Their work relies on previous work done within the UC Labs, where a new object is added to the management information base.

Articles [3,4,5] focus on Internet of Things concepts and solutions. Authors using the new concepts and technologies try to find a solution to common problems that are involved in everyday life. One of the authors says that the smarter world is a result of the smarter technology. The Internet of things concept is very dependable of system monitoring tools, as all these new added elements need to be monitored to see if they are working as it supposed to be. IoT (Internet of Things) concept will have a big benefit if the utilization of a well utilized monitoring system. The Internet of Things refers to ubiquitous end devices and facilities, including sensors with intrinsic intelligence, mobile terminals, industrial systems, building control systems, home intelligence devices, video surveillance systems, externally enabled, such as RFID-attached assets, individuals and vehicles carrying wireless terminals.

In [6] the main motive of the authors is to alert the network administrator by methods like SMS or E-mail in case of any failure in the network. This helps in securing the network by alerting the potential issues in real time.

One feature that makes the monitoring system a must have for every serious organization is proactive monitoring. Proactive monitoring using prediction [7] and monitoring agent [8] helps when we have set goals and whether they are achieved or not. The process of software defect prediction has been focused on by many researchers in the last decade. However, improving the prediction accuracy has always been the main concern [9]. For example, monitoring systems can show how much of the total potential of the hardware and software is used in an organization.

When setting up a monitoring system, care should always be taken about what should be monitored and whether the right monitoring system has been selected. In this book [10], authors are explaining in detail setting up and using Nagios monitoring system.

Every additional software installed on an already configured and working machine adds adional workload on CPU. CPU workload shows the number of instruction actuality executed by the processor through a given phase or at a certain time. This statistic shows a need for a rise in processing power of the CPU when it is loaded all the time, or a reduction in processing power if CPU usage drops under a certain threshold. Additional routine developments can be gained for the similar number of series required by an instruction for effective implementation. This can be enhanced by refining code effectiveness as authors describe in [11] and [12]. In [13] authors want to bring closer virtualization technology used for implementation of cloud computing when migration is done in online and offline mode. They conclude that downtime is less when migration is done during live migration, compared with offline migration.

Authors in article [14] make measurements of input-output performance of different file storage systems using five different configurations of SSD hard drive using different RAID levels. Article [15] gives concept of smart SSD that is used with the code that can be initiated inside the ssd drive, to take advantage of high internal bandwidth that SSD can provide.

Great growth in devices that are part of computer networks brings an increased use of system monitoring applications as authors describe in [16]. Their focus is SNMP service, and they do a comparison between different network monitoring systems, Nagios, and WhatsUp Gold. They conclude that Nagios is a better option for companies to remain competitive in area network monitoring with little investments. In article [17] authors are saying what is important when monitoring system is implemented from the point of the health services that can be seen on a dashboard. In their paper they describe the capabilities of dashboards to present heterogeneous metrics, side by side, regardless of their origin. The proof of concept in this is a tool that has been designed and implemented.

In [18] authors are saying that even that SSD drives are faster than ever, and of course faster than regular spinning disk hard drive, RAM disk are still the fastest I/O devices for data [19]. Problem with ram disks is that their memory is volatile on power outage. But using a hybrid RAM/SSD drive will be a great solution for devices that need fast I/O as well as memory that could be kept after power lost.

3. Configuration and Methods

Well-implemented monitoring system saves money and time. For example, if we look from a software point of view, when monitoring a website, the system notices that loading certain data takes longer than usual. Then, the system itself alerts and sends data that something is wrong with the website.

Simulation is executed on virtual machines that are attached to main host. The main host machine has the following configuration:

- CPU - 8th Gen. Intel i7 CPU
- RAM – 16 GB
- SSD – 480 GB

All virtual machines are preconfigured separately. This is done because all virtual machines start at the same point, and only the agent is modified. This approach was taken because reconfiguration on a machine can leave parts from

other agent and can interact with stability of the system that measurements are taken from.

Methodology that has been used for the preparation of the tables is taken from an average of three measurements for a given load at the highest peaks. The measurement will be done in a span of 5 minutes when a CPU is on specific workload. This study was performed by collecting data when client machine has no additional workload in first case, then when artificial workload is added, and CPU is at 50% load and 80% load. In any specific workload, the measurement that is taken is the one with the highest pick when CPU is at artificial workload.

Each virtual machine in Table 1 is setup with default installation of Windows 7 or Windows 10 operating system. The server itself is hosted on CentOS operating system, because Nagios cannot be directly installed on Windows operating system. This is when existing server is moved into a virtual environment [12]. Minimum specification is suggested by the authors in [13] for the host operating system. One virtual processor with an operating frequency of 2.21 Ghz is assigned to perform the tasks and it is quite enough for a test environment, where the capabilities of the system itself are tested. An exception is made only for the server itself, that has two assigned cores. This is done to have a greater response to requests that he sends himself using active monitoring, or to be able to receive data that is only sent to him through passive monitoring.

They have a virtual SSD hard drive with a capacity of 40 GB. An SSD hard drive is chosen because a regular hard drive with a rotating drive is slow, given that several virtual machines are mounted on a single drive. As authors in [14] suggest that SSDs are far superior compared to the traditional mechanical hard disks. 2 GB of working memory on each host is quite sufficient, because no additional applications are installed except the minimum required to make the analysis. Each host has its own fixed IP address, which is set manually, so that it is always available at the same address.

Table 1. Virtual machines configuration

Name, OS	CPU count	Hard Disk (GB)	RAM memory (MB)	IP Address	Agent installed
client1,Win10	1	40	2048	192.168.1.101	/
client2,Win7	1	40	2048	192.168.1.102	SNMP
client3,Win10	1	40	2048	192.168.1.103	NCPA Active
client4,Win10	1	40	2048	192.168.1.104	NCPA Passive
client5,Win10	1	40	2048	192.168.1.105	NSC++ Active
client6,Win10	1	40	2048	192.168.1.106	NSC++ Passive
NagiosXI, CentosOS 7	2	40	2048	192.168.1.91	NCPA local

Fig.1 shows the virtual environment design for testing. It consists of six client operating systems and one server operating system.

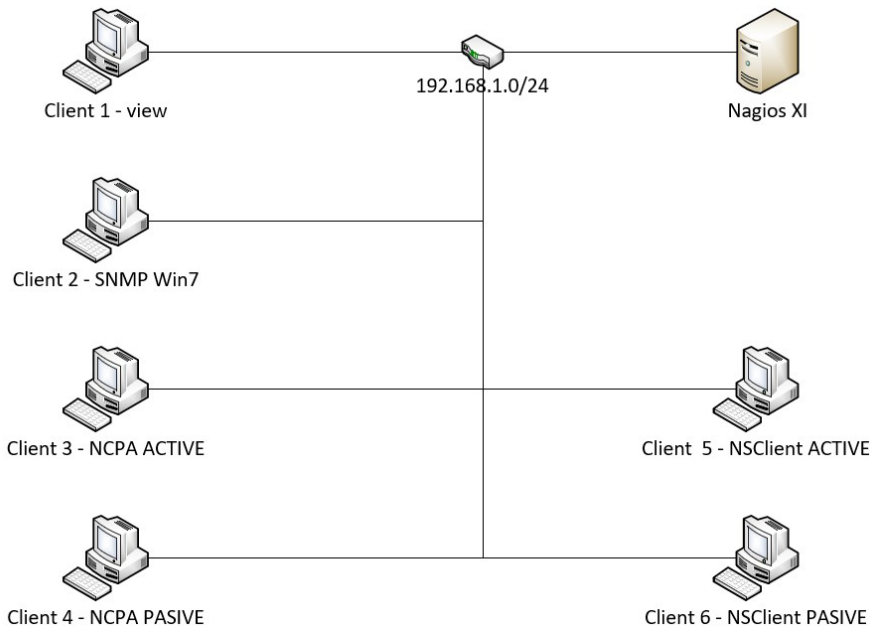

Fig.1. Layout of virtual machines

Client 1 – view: virtual machine is used for browsing through a web browser and displaying the results. This is done so that the server and the virtual machine being tested do not have additional loads on it. It is done by another independent machine, in this case client 1-view. Machines from client 2 to client 6 are installed in the same way as can

be seen in Table 1. This means that they have the same hardware performance with the same amount of RAM, the same hard drive capacity, and the same processor. The only difference is what agent they have installed on them. This is done to have an equal load from the operating system itself.

For best results and without interruption in measurements, only server and VM with specific agent is active when measurements are taken. Closed environment is created, without outside interaction, in order to produce clean results from the simulations.

4. Workload Simulation and Results

Computer systems are one of the main components in performing daily tasks in companies, educational institutions, or state institutions. The personal computer that people work with is especially important to be functional. To detect a problem before a potential operational disaster occurs, it is necessary to use computer network monitoring systems.

Additional workload is not only added to CPU itself, but also on other hardware components in the given configuration. Memory workload is also important as CPU workload. Particularly program or instruction wants some memory to stock momentary or lasting data and to do intermediate computation. The memory workload regulates the memory usage of the whole system over a specified period or at a specific time. Paging and segmentation actions use a lot of virtual memory, thus incising the use of central much needed memory. Once the number of a programs being performed become so large, the memory workload is a bottleneck for the performance. This indicates that more memory is desired, or programs need to be managed in a better way.

By collecting and analyzing data, it will be seen which client achieves the best time. VM will be throttled with additional CPU utilization for simulation. Results will be measured at:

- No extra workload on client's machines.
- With 50% workload on client's machines.
- With 80% workload on client's machines.

As example, when the CPU is at 50% load, the three picks' usages are taken, and average value is representing that measurement. This is done because primary and qualitative data is taken from simulation and is written in tables for further analysis. Data collected is descriptive, that means that no modifications has been done.

The quantity of work performed by a component in an agreed period, or the normal volume of work handled at specific instant of time is called workload. The amount of work controlled by an object provides an approximation of the efficiency of that object. In computer science, this is referring to computer capability to handle and execute work. Computer parts such as servers or other systems are frequently allocated an expected workload upon creation. Examination of their performance related to the workload that was expected is then showed over time.

These analyses are needed to see which client will give better results and under what conditions. A software tool that will simulate CPU load is called CPU Grab Ex. This software tool works quite simply, just by adjusting the required percentage to get the desired load and running it. The tool gives the CPU performance calculations and thus simulates the workload on the CPU.

- RTA Round trip average (time)
- PL Packet loss (percent)
- RTMAX Round trip max (time)
- RTMIN Round trip min (time)

RTA is the time it takes to send for verification and receive feedback. This means that you can measure the total time it takes to send and receive. PL presents the percentage of the packets that are lost when they travel from the server to the client and back. Usually in the tests here the value is zero or one hundred precents. Zero is obtained because the connection is not loaded, and if 100% are obtained it means that all packets are lost and it is a sign that either the client is unavailable, incorrectly configured, or turned off. RTMAX - represents the maximum time it takes for the server to send verification and receive feedback. RTMIN - is the shortest time it takes the server to send verification and get feedback. Time is expressed in ms (milliseconds). So, this result means how many milliseconds it takes the server to send verification and receive feedback.

4.1. SNMP

SNMP approach when choosing a monitor is sometimes a good choice because it is built-in and does not need to be further installed or configured. The server simply downloads the already available information from the monitored

system. Nagios does support SNMP [16] probing via the use of the check_snmp plug-in. This is especially important for systems where the installation of additional agents is prohibited. SNMP can be said to be less secure than other agents. But with good configuration with rule sets by which SNMP can only read information without being able to change, security is on a solid level. It is also always advisable to use version 3 where protection is added to prevent information from spilling out while being transmitted from the server to the client and vice versa.

Table 2. Average in normal operation (SNMP)

No Extra Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,322	0	0,399	0,252
Second Measurement	0,304	0	0,417	0,216
Third Measurement	0,324	0	0,399	0,280
Average	0,316	0	0,405	0,249

Table 2 content is when the client is just turned on and there is no extra load. The RTA average is 0.316 milliseconds. RTMAX in this case ranged from 0.399 to 0.417 with an average of 0.405 milliseconds. RTMIN measurement is from a minimum of 0.216 milliseconds to a maximum of 0.280 milliseconds. The average of three measurements is 0.249 milliseconds. Table 3 presents results obtained with 50% workload on client's machines:

Table 3. 50% average load (SNMP)

50% Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,403	0	0,555	0,319
Second Measurement	0,363	0	0,482	0,288
Third Measurement	0,328	0	0,496	0,186
Average	0,364	0	0,511	0,264

At 50% CPU load, the average RTA is 0.365 milliseconds, with a minimum of 0.328 milliseconds to a maximum of 0.403 milliseconds. RTMAX at 50% load ranged from 0.555 to 0.482 with an average of these three measurements of 0.511 milliseconds. RTMIN measurement is from minimum of 0.186 milliseconds to maximum of 0.319 milliseconds. The average of three measurements is 0.264 milliseconds. For comparison, when there is no load, time is almost the same, and sometimes even smaller. With 80% workload on client's machines results are shown in Table 4.

Table 4. 80% average load (SNMP)

80% Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,368	0	0,720	0,247
Second Measurement	0,454	0	0,973	0,297
Third Measurement	0,469	0	0,663	0,381
Average	0,430	0	0,785	0,308

For the 80% CPU load, it is normal to increase the values, because the CPU resources are busy and there are fewer free resources left to perform the tasks. The minimum RTA is 0.368 and the maximum is 0.469 milliseconds, while the average of the three measurements is 0.430 milliseconds. RTMAX in this case ranged from 0.973 milliseconds to 0.663 milliseconds with an average of 0.785 milliseconds, which compared to other measurements without and with 50% load, is expected to increase. RTMIN is from a minimum of 0.247 milliseconds to a maximum of 0.381 milliseconds, and the average of three measurements is 0.308 milliseconds.

Regarding Fig. 2 when the resources for one thing increase (in this case when increasing the CPU resource), there are less resources for work in the other parts of the system. So, when CPU usage increased, RTA remained the same, even at 50% load it improved. Anyway, RTMAX and RTMIN changed dramatically and in this case the maximum return signal time doubled, but at the same time the system compensated by halving the minimum return signal time.

The graph in Fig. 3 visually represents the time it takes to execute the server request and receive the information back.

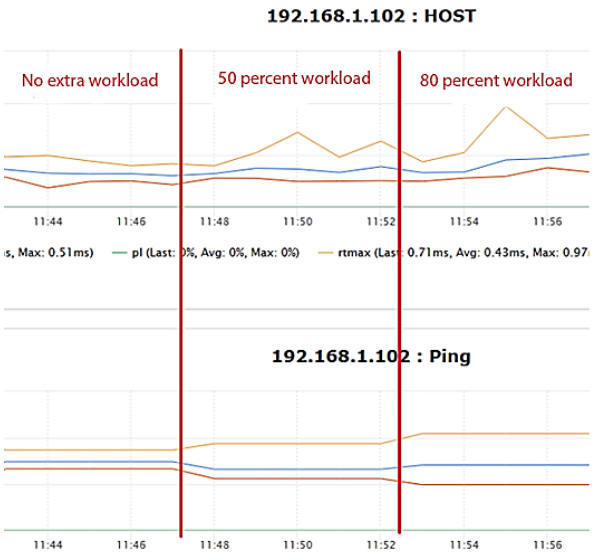

Fig.2. Print screen from server for comparison with ping command.

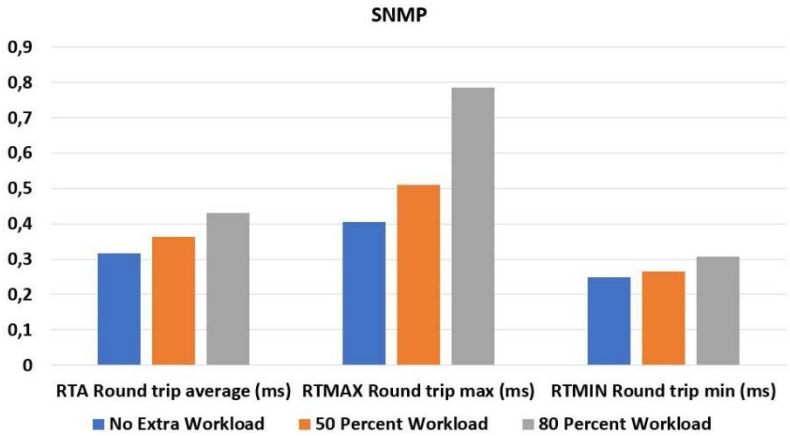

Fig.3. Print screen from server for comparison with ping command

4.2. NCPA

After installing the NCPA [20] agent, it immediately starts working without much configuration. To modify the sending time of checks, the configuration file is modified. The default check is five minutes, in this case it is modified to do one-minute checks. All checks are made on a restarted server and client, so that there are no extra burdens such as longer client work or server. With no extra workload on client's machines is shown Table 5:

Table 5. Average in normal operation (NCPA)

No Extra Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,452	0	0,660	0,359
Second Measurement	0,482	0	0,752	0,377
Third Measurement	0,446	0	0,605	0,368
Average	0,460	0	0,672	0,368

The minimum at RTA is 0.446 while the maximum is 0.482, with an average of 0.460. RTMAX when measuring showed a result of 0.605 minimum time and maximum 0.752. The average is 0.672 milliseconds. RTMIN with a minimum time of 0.359 and a maximum time of 0.377 milliseconds. The average load rate of 50 percent on the RTMIN is 0.368 milliseconds. With a CPU load of 50 percent, a simulation is performed in order to increase the performance of the CPU, which plays a major role in reading data and returning to the server. With 50 percent workload on client's machines is shown in Table 6:

Table 6. 50% average load (NCPA)

50% Workload	RTA	PL)	RTMAX	RTMIN
First Measurement	0,595	0	0,863	0,506
Second Measurement	0,481	0	0,964	0,340
Third Measurement	0,442	0	0,775	0,337
Average	0,506	0	0,867	0,394

With a CPU load of 50%, the RTA has a minimum time of 0.442 milliseconds and a maximum of 0.595 milliseconds. The average here is 0.506 milliseconds. RTMAX shows a maximum time of 0.964 milliseconds and a minimum time of 0.775 milliseconds, while the average is 0.867 milliseconds. RTMIN average value is 0.394 milliseconds, with a maximum measured time of 0.506 milliseconds and a minimum time of 0.337 milliseconds. At 80% CPU load, the time it takes for the server to send a request and receive an increase can be seen in Table 7.

Table 7. 80% average load (NCPA)

80% Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,568	0	0,820	0,431
Second Measurement	0,561	0	0,952	0,367
Third Measurement	0,364	0	0,675	0,255
Average	0,498	0	0,816	0,351

In this case, too, the results are as expected, so the RTA minimum time value is 0.364, while the maximum value is 0.569 milliseconds with an average of 0.498 milliseconds. When it comes to RTMAX and RTMIN, results are also expected. RTMAX with minimum and maximum values of 0.675 and 0.952 milliseconds, while the average is 0.816 milliseconds. From the graph of Fig. 4 visually is shown the time expressed in milliseconds

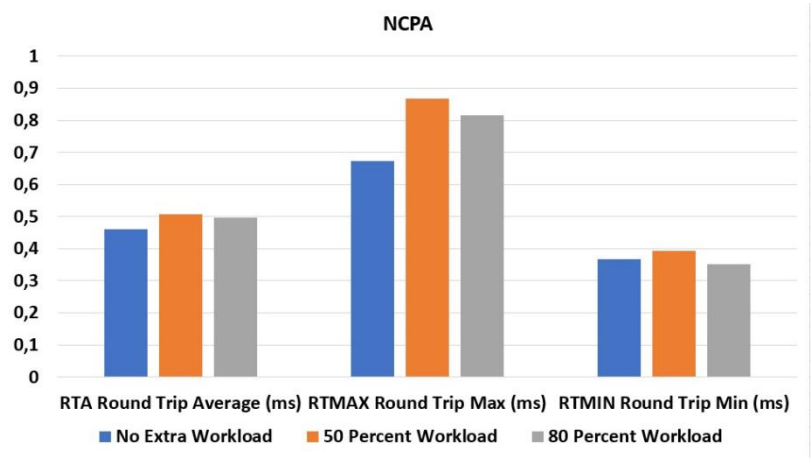

Fig.4. NCPA chart

4.3. NSClient++

NSClient++ is one of the most versatile agents for network monitoring. The client side is installed with NSClient++ which act as Nagios service for the client side and send data to the Nagios Server [17]. With no extra workload on client's machines is shown in the Table 8:

Table 8. Average in normal operation (NSClient++)

No Extra Workload	RTA	PL)	RTMAX	RTMIN
First Measurement	0,424	0	0,550	0,335
Second Measurement	0,426	0	0,576	0,340
Third Measurement	0,382	0	0,533	0,268
Average	0,411	0	0,553	0,314

From Table 8, which presents the results obtained during normal operation, without load it can be seen that the RTA averages 0.411 milliseconds, while the maximum and minimum measured times are 0.426 and 0.382 milliseconds. RTMAX with an average of up to 0.553 milliseconds and that minimum measured time is 0.533 and maximum 0.576

milliseconds. RTMIN averages 0.314 milliseconds, while the maximum and minimum response times here are 0.340 and 0.268 milliseconds. Table 9 shows results with 50% workload on clients' machines.

Table 9. 50% average load (NSClient++)

50% Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,404	0	0,805	0,244
Second Measurement	0,430	0	0,737	0,259
Third Measurement	0,442	0	0,813	0,241
Average	0,425	0	0,785	0,248

NSClient ++ with a load of 50% showed RTA results with an average of 0.425 milliseconds, while the minimum and maximum times are 0.404 and 0.442 milliseconds. RTMAX time averages 0.785 milliseconds with a minimum measured time of 0.737 and a maximum time of 0.813 milliseconds. RTMIN averages 0.248 milliseconds, while the minimum and maximum measured times are 0.241 and 0.259 milliseconds. Table 10 shows 80% workload on clients' machines.

Table 10. 80% average load (NSClient++)

80% Workload	RTA	PL	RTMAX	RTMIN
First Measurement	0,520	0	1,066	0,342
Second Measurement	0,720	0	0,982	0,309
Third Measurement	0,568	0	0,919	0,432
Average	0,603	0	0,989	0,361

When it comes to 80 percent CPU load, the average RTA time is 0.603 milliseconds. The minimum time measured is 0.520 milliseconds and the maximum is 0.720 milliseconds. RTMAX time measured at 80 percent CPU load averages 0.989, while minimum and maximum times are 0.919 and 1.066 milliseconds. RTMIN records an average time of 0.361 milliseconds, with a minimum time of 0.309 and a maximum time of 0.432 milliseconds. From the graph of Fig. 5, visually can be seen the time expressed in milliseconds.

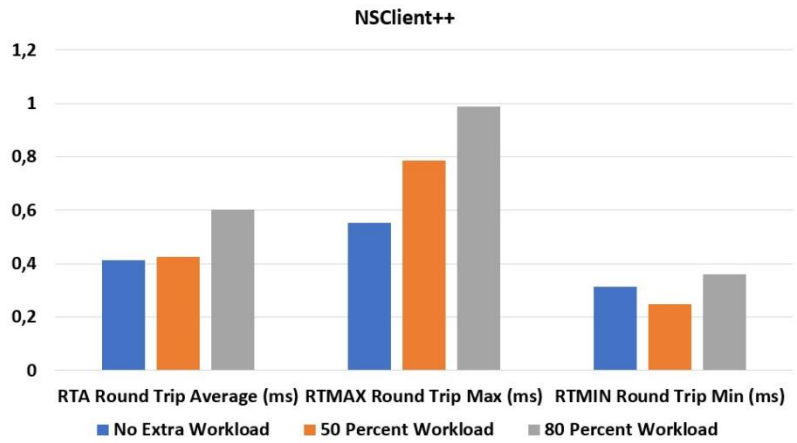

Fig.5. NSClient++ chart

5. Analysis and Improvement

Best application tends to use substantial period collecting input and creating output. As a result, the job of input-output mixtures on system must be examined carefully to confirm that suitable load performance limitation is always met. An amount on the number of inputs collected by a system and the number of outputs created over a specific period is named as input-output workload.

For result analysis we will first look at the RTA Round trip average (ms) parameter. From the graph on Fig. 6 we will see the comparative values obtained during the simulation of the agent's work.

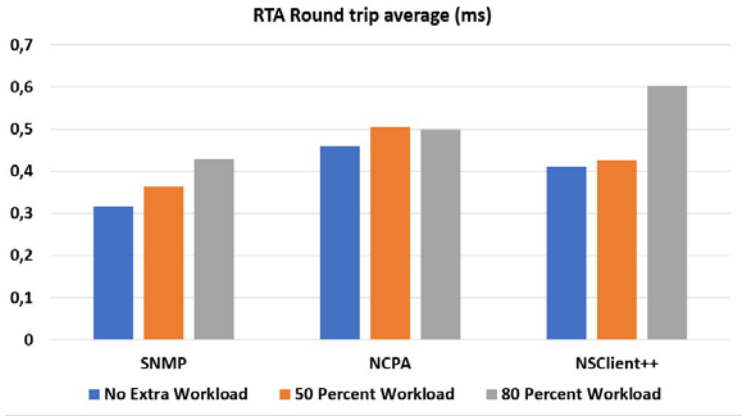

Fig.6. RTA round trip average (ms) chart

Comparative analysis in this paper is done by using the performance parameters on each virtual machine that has been monitored. This is further analysed whit detailed description. Normal operation is when system in this case the CPU is in minimum usage stage. This means that CPU has done all the operation that was given for calculation, and no further instruction is given.

During normal operation, without CPU load, the best results are visible in the SNMP agent, while the weakest results are in the NCPA agent. With 50% CPU usage, the results are the same as during normal operation, so SNMP and NSClient ++ have better results than NCPA. When CPU usage comes to 80%, there will be a downtime in NSClient ++ agent, while there is an improvement in results for NCPA agent. SNMP has the best results here as well. The good results of SNMP are since it is built into the system itself, but with a lack of modification, NCPA would still be the best agent application solution. To get improved results, it is necessary to act on both the server and the client. For clients, as hard drives it is necessary to use SSD hard drives. SSDs have 10 times faster data access than regular rotating disks. The same goes for servers. By replacing ordinary hard drives with SSDs, it dramatically increases the read and write performance. SSD technology has advanced to reverse the widening trend of performance gap between processors and storage devices [15].

When it comes to programs that have a very large number of read and write operations, such as monitoring programs that receive information at thousands of nodes per second, it is best to use RAM disks. This kind of disks are especially recommended for the Central Monitoring Server [18,19].

The media used in the physical network that is installed also plays a big role. Copper cable is widespread and is laid everywhere. It is not easy to change it at all levels, but for better results that is the only way.

Replacing a copper cable with an optical cable plays a very important role from the beginning to the end. For example, CAT6 network cable has a data flow of 10 gigabits per second, while fibre optic cable has a potential of 100 terabits per second.

According to the official website of monitoring software maker Nagios [21], there are other conditions where performance could be improved, specifically for their software, but many things are the same and apply to other software vendors:

- To use statistics with MRTG, this is especially important in the short and long term to see how the system overcomes the loads in the configuration itself through graphs and allows performance to be adjusted where needed.
- Use large_installation_tweaks option during the installation. This option allows the software to find better ways to deal with poor results and poor performance. For example, it excludes unnecessary things like automatic macro calculations, because they are time consuming and at some point.
- Frequency of obtaining results where it is not needed. This means increasing and decreasing the frequency as needed in order to increase the performance of the whole system.
- To fine-tune the certain services. For example, some services need more time to return certain results than other services. With fine and precise adjustment of the response of the services, a harmony of receiving results is obtained, because as it was mentioned previously, not every service return results with the same speed. Simply put, the service itself needs more or less time to display results, which reduces the waiting time of the application itself and improves overall performance.
- To use more plugins that are already compiled and ready to use, unlike those that are in pure code and before execution should be compiled first and then executed. Here is a comparison of plugins made in C or C ++ that are already compiled, compared to scripts in Perl that are compiled before execution. Perl modules works directly over linux kernel, and it establishes a communication with Nagios through NRPE agent. This means

that the system itself will need more time, to compile the script first and then execute it. When it comes to thousands of different scripts, the time saved is obvious.

- In case you must use scripts made in Perl, it is necessary to use a compiler inserted in the plugin itself. With this Nagios scripts will not run as external scripts and no time will be wasted calling an external compiler.
- Optimization checks, with scheduled pre-tested checks and the use of non-aggressive checks play a significant role. For example, if you are working on a problem that needs more time to resolve, it is best to optimize the next check of that problem by reducing the number of subsequent checks which in some situations are not needed.

The use of distributed architecture also improves the overall monitoring environment. Nagios Load distribution into several nodes using Fusion. Distributed architecture means the parallel operation of multiple instances of a Nagios server, in order to separate tasks such as monitoring, database entry, and graphing.

This distribution of systems is done on the basis of geographical determinants, for example where the client being monitored is located and which is the closest instance to it. Also, they can then be divided according to the importance of the services being monitored. For example, highly critical points that need to be monitored more often are placed in a group that will be allowed more resources in order to always have enough resources in terms of workload.

The servers that are being reviewed can be separated from the one that sends requests and receives results, in order not to take up resources and additionally burden them. So first the products start working on the same server that sends and receives tasks, a review is made on it and the database itself was located on that server. But when it comes to increasing the number of services and clients that are being monitored increase, these things should slowly start to be allocated to dedicated servers dedicated only to specific tasks.

Start from the server first. Let the server be intended to perform its tasks and then try to free as much resources as possible. Passive checks are performed by the client, and that process do not take up much of the client's resources, but this helps the servers a lot to get rid of unnecessary tasks that would take up the much-needed resources for him.

Instead of the server performing periodic tasks such as sending commands to agents and waiting for a response from them, it would be best for the server to only receive information from agents that are pre-configured which tasks to execute and then to send information to the server. For example, instead of constantly asking the server how much free space there is on a monitored hard drive, the client computer itself can periodically send information to the server about the amount of free space on the hard disk.

Active checks when done on a remote computer need to be done in a rational way. It is true that if frequent checks are made, the condition of a client will be known better and faster. But frequent checks put a strain not only on the system on which they are executed or on the server that sends and receives messages, but also on the entire network in an IT environment that is being monitored.

6. Conclusion

There is no one-size-fits-all approach for selecting an execution agent and sending information to a server for further processing. The way how is done is always dependent on the environment where the whole system is placed and the hardware characteristics of the entire IT equipment. To optimize the performance of the monitoring system, it is necessary to go step by step and always try which solution will be best for a given environment.

After the analysis made in this paper, it can be concluded that the agents themselves in case of increased use of a resource, in this case the CPU, increase the response time to servers. Knowing this, choosing which monitoring software to use and then which agent to use should be well considered out and tested. The purpose of this paper is not to present the Nagios system and its features, but to show what happens with response time when resources are being shared. The purpose of this paper is to present the characteristics of monitoring systems and to present an analysis of how performance decreases when system is under additional workload.

The work done in this paper clearly shows that any addon take resources on a specific system where is installed. That’s why it is always important to choose an adequate addon to suit the purpose. Addon efficiency is very important to be analysed before specific addon is installed. In long run this can have a big impact in the system.

The objective of this paper was to find the best reaction when CPU resources are in use. To get the best optimum accuracy of the readings that are used in tables, top picks are being get into consideration. In our observation of results, it can be concluded that usage of the right addon is very important, as system performances can degrade if non adequate addon is used.

For future work, this can be analysed more deeply, to show how system monitoring tool that can use third party addons can be utilized in best way possible.

References

[1] Savić, M., Ljubojević, M. and Gajin, S., 2020. A Novel Approach to Client-Side Monitoring of Shared Infrastructures. IEEE Access, 8, pp.44175-44189.

[2] Luchian, E., Docolin, P. and Dobrota, V., 2016, October. Advanced monitoring of the OpenStack NFV infrastructure: A Nagios approach using SNMP. In 2016 12th IEEE International Symposium on Electronics and Telecommunications (ISETC) (pp. 51-54). IEEE.
[3] Nurwijayanti. KN, Rhekaz Eka Adhytyas, " Garbage Bin Monitoring System Based on the Internet of Things at University Dirgantara Marsekal Suryadarma", International Journal of Education and Management Engineering, Vol.11, No.2, pp. 1-12, 2021.
[4] Kajol R, Akshay Kashyap K, Keerthan Kumar T G," Automated Agricultural Field Analysis and Monitoring System Using IOT", International Journal of Information Engineering and Electronic Business, Vol.10, No.2, pp. 17-24, 2018.
[5] Nalinsak Gnotthivongsa, Huangdongjun, Khamla Non Alinsavath, " Real-time Corresponding and Safety System to Monitor Home Appliances based on the Internet of Things Technology", International Journal of Modern Education and Computer Science, Vol.12, No.2, pp. 1-9, 2020.
[6] Renita, J. and Elizabeth, N.E., 2017, March. Network's server monitoring and analysis using Nagios. In 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET) (pp. 1904-1909). IEEE.
[7] Mohammad Zubair Khan, " Hybrid Ensemble Learning Technique for Software Defect Prediction ", International Journal of Modern Education and Computer Science, Vol.12, No.1, pp. 1-10, 2020.
[8] Rushba Javed, Sidra Anwar, Khadija Bibi, M. Usman Ashraf, Samia Siddique, "Prediction and Monitoring Agents using Weblogs for improved Disaster Recovery in Cloud", International Journal of Information Technology and Computer Science, Vol.11, No.4, pp.9-17, 2019.
[9] Umair Ali, Shabib Aftab, Ahmed Iqbal, Zahid Nawaz, Muhammad Salman Bashir, Muhammad Anwaar Saeed, " Software Defect Prediction Using Variant based Ensemble Learning and Feature Selection Techniques", International Journal of Modern Education and Computer Science, Vol.12, No.5, pp. 29-40, 2020.
[10] Kocjan, W. and Beltowski, P., 2016. Learning Nagios. Packt Publishing Ltd..
[11] Maas, M., Andersen, D.G., Isard, M., Javanmard, M.M., McKinley, K.S. and Raffel, C., 2020, March. Learning-based memory allocation for C++ server workloads. In Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems (pp. 541-556).
[12] Fahreza, F. and Rifqi, M., 2020. Nagios Core Optimization by Utilizing Telegram as Notification of Disturbance. Journal of Applied Science, Engineering, Technology, and Education, 2(2), pp.121-135.
[13] Garima Rastogi, Rama Sushil,"Performance Analysis of Live and Offline VM Migration Using KVM", International Journal of Modern Education and Computer Science, Vol.8, No.11, pp.50-57, 2016.
[14] Pei, Z. and Xiaoman, W., 2016. SSD performance analysis and RAID 0 scheme design. Microcomputer & Its Applications.
[15] Wang, J., Park, D., Kee, Y.S., Papakonstantinou, Y. and Swanson, S., 2016, June. Ssd in-storage computing for list intersection. In Proceedings of the 12th International Workshop on Data Management on New Hardware pp. 1-7.
[16] Bruschi, G. and Ribeiro, L., 2017. Monitoramento de Serviços de Redes usando SNMP com Nagios e WhatsUp Gold. Caderno de Estudos Tecnológicos, 5(1).
[17] Cardoso, A., Teixeira, C.J.V. and Pinto, J.S., 2018. Architecture for highly configurable dashboards for operations monitoring and support. Stud. Inform. Control, 27(3), pp.319-330.
[18] Baek, S.H. and Park, K.W., 2020. A Durable Hybrid RAM Disk with a Rapid Resilience for Sustainable IoT Devices. Sensors, 20(8), p.2159.
[19] Scott, N.,2011. “Nagios Conference 2011 - Nagios Performance Tuning”, pp.11.
[20] Enterprises, N., 2017. Nagios.
[21] Enterprises Nagios,2016, N. Tuning Nagios for Maximum Performance.

Authors’ Profiles

Filip Gjorgjevikj received BSc Degree in Informatics from the "St. Kliment Ohridski" University (UKLO), Faculty of Technical Sciences in 2009. His focus is on computer support and computer system administration. Currently working on MSc in Computer Engineering from AUE-FON University, Faculty of Information and Communication Technology.

Kire Jakimoski received his B.Sc. degree in the field of Telecommunications from the Military Academy “Mihailo Apostolski” in Skopje, R. Macedonia in 2002, M.Sc. degree in Electrical Engineering in the field of Telecommunications from the Ss. Cyril and Methodius University in Skopje, R. Macedonia in 2007, and D.Sc. in technical sciences from the Ss. Cyril and Methodius University in Skopje, R. Macedonia in 2013. From 2002 to 2006 he works as an Officer for Telecommunications in the Ministry of Defence in the Republic of Macedonia. From January 2006 to February 2012 he works as an adviser for information security in the INFOSEC Division in the Directorate for Security of Classified Information in Skopje, Republic of Macedonia. From March 2012 he is with the Faculty of Informatics, FON University in Skopje. His research interests include Wireless and Mobile Networks, Heterogeneous Networks, Computer Networks, Cyber Security, Network Security.

How to cite this paper: Filip Gjorgjevikj, Kire Jakimoski, "System Monitoring Addon Analysis in System Load Simulation", International Journal of Computer Network and Information Security(IJCNIS), Vol.14, No.1, pp.40-51, 2022. DOI: 10.5815/ijcnis.2022.01.04