**IMPACT: International Journal of
Research in Engineering & Technology
ISSN (P): 2347-4599; ISSN (E): 2321-8843
Vol. 6, Issue 9, Sep 2018, 9-14
© Impact Journals**

REMOTE PROCESS AUTOMATION OF MONITORING USING NAGIOS

Venkata Rohit Gandikota & A. Mary Sowjanya

*Research Scholar, Department of Computer Science & SE, College of Engineering, Science
Andhra University, Visakhapatnam, Andhra Pradesh, India*

Received: 29 Aug 2018 **Accepted: 04 Sep 2018** **Published: 18 Sep 2018**

ABSTRACT

With the advent of powerful computing machines and advances in technology, more and more machines are being used on a daily basis. With the increase in the infrastructure, monitoring of said infrastructure has been playing a key role in keeping the working of an organization streamlined and efficient. While modern machines are more than capable of running for days together without the risk of failing, there can be no shortcuts when coming to infrastructure as any of the machines can go down without warning if not properly looked after. This can cause delays which can sometimes extend to days or even months if not attended to immediately. However, the major gripe with the existing manual system is that problem identification has become a roadblock and also there are no ways to predict a failure of a system in advance. A person is always required to constantly keep looking at a screen in order to see if alerts are generated and when indeed they are, if that person is missing from his station then there will be a delay as to when the alert is generated and when it is actually notified to the concerned team. This human involvement becomes time-consuming as the alerts may or may not be generated. Since there is not a lot of work involved and the skills of the person can be put to better use at a different place. The aim of this work is to eliminate the human factor involved by automating the monitoring of infrastructure with the help of software tools and also integrate the alert generation and notification process by using machine learning algorithms and natural language processing. The monitoring tool can be configured according to the needs of the company and the generated alerts can be sent to the concerned team immediately without any delay. As the tool displays real-time data we can predict which system needs to be taken care of immediately and which machine is likely to go down in the future.

KEYWORDS: *Automation, Infrastructure, Monitoring, Machine Learning, Nagios*

INTRODUCTION

Monitoring is crucial because problems are not always immediately evident, but tend to build up over time. As information technology continues to play an increasingly important role in the day to day operations of an organization, it has become difficult for the organization to keep an eye on all the activities at the same time. This has put an additional burden upon administrators to develop a system which is capable of keeping an eye on the activities and as well as send reports and alerts when required in order to streamline the performance of the organization. A monitoring tool can be described as a simple software that is capable of keeping an eye on all the activities at the same time, collecting data, analyze the data, generate reports, send notifications and alerts to the administrators as and when required. While collecting and analyzing the data is the easy part in any monitoring process, interpreting the data and developing a monitoring system to address the needs for that data is notoriously difficult. Our work makes use of a monitoring tool called Nagios.

It is integrated with an artificial intelligence framework which makes use of machine learning and natural language processing techniques to understand the problem and suggest a solution to that problem [1].

NAGIOS

Nagios is one of the numbers of monitoring tools that are available on the market. It is a monitoring tool that is open source and free to use. It offers services like monitoring hardware infrastructure, network devices and connections and other services. Nagios can also send alerts to the administrators when a problem arises and can also let them know when a problem has been solved. Our work makes use of the version of Nagios called Nagios XI. This is basically an extended interface of the original Nagios software and has been developed using PHP and Python. Nagios XI is one of the most powerful monitoring tools available on the market and offers a lot of benefits like a powerful engine, proactive planning, easy to use interface etc. Nagios makes use of different agents to monitor machines running on different platforms like windows. While Linux or UNIX machines make use of Nagios Remote Plugin Executor or NRPE, Windows machines use NSClient++. There is also another agent called Nagios Cross Platform Agent (NCPA) which can be used on any machine running Linux, Windows or even Mac OS X[2].

FRAMEWORK FOR AUTOMATING REMOTE PROCESS MONITORING

We have designed an artificial intelligence framework which makes use of machine learning techniques like support vector machines and hill climbing along with natural language processing. This framework helps reduce the time taken to detect problems and generate reports of the incidents. By using SVM a clear solution can be given as the algorithm divides the datasets into classes.

Support Vector Machines

Support Vector Machines(SVM) are a specific field in machine learning which makes use of certain algorithms for analyzing and classifying data into appropriate sets or classes. SVM is also called supervised learning models because they need a set of examples to learn from. Given a set of training examples that have already been divided into classes, the SVM algorithm learns from the existing examples and when new ones come along, it places them into the appropriate class. We have used SVM along with natural language processing in order to understand the problem that has been raised and to suggest the best solution for that particular problem. The NLP algorithm first reads the problem, understands the problem and then matches it with the best possible example in the SVM dataset that has been preloaded with examples for the automation system to understand. The examples are divided into classes based on the nature of the problem that has occurred, for example, it can be related to hardware, software, network, authorization, security etc. The main drawback of SVM is that it works efficiently when the size of the dataset is small. Since we use this framework for industrial applications, the size of the datasets can be unimaginably large. Such large datasets reduce the efficiency of the SVM models as the time taken to learn from such huge datasets is more and also some of the problems may be a combination of two or more classes [3][4].

Pseudo Code for Support Vector Machine [5]:

```
 #import Library

from sklearn import svm
```

```
# Create SVM classification object

model = svm.SVC(kernel='linear=1, gamma=1)

model.score(X, y)

#Predict Output

predicted= model.predict(x_test)
```

where gamma = Kernel coefficient for ‘rbf’, ‘poly’ and ‘sigmoid’

Hill Climbing Algorithm

Hill Climbing technique is used to solve certain optimization problems. This algorithm starts by choosing a solution that is nowhere near the optimal solution to the problem and it goes on improving the solution until the optimality condition is met. This initial solution is compared to the bottom of a hill, the process of improving the solution is akin to climbing the hill and the final solution is the top of the hill where the final solution is developed.

Hill climbing technique has the following phases −

- Selecting an initial solution which is not the optimal solution but is within the constraints of the problem
- Improving the solution step-by-step
- Finally, the optimal solution is reached and no further development can be made.

Generally, this technique is used when the problem is complex and an optimal solution cannot be developed in one go. One constraint with the hill climbing technique is that it considers only the present solution and the next immediately available solution. While this method is memory efficient, it can take a lot of time to complete as each and every solution is considered and compared.

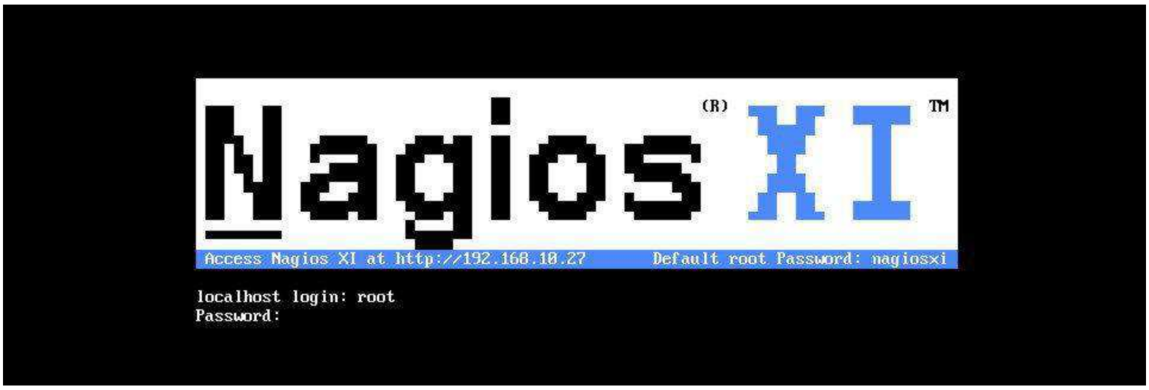
Our work makes use of a variation of the hill climbing technique called the steepest ascent hill climbing technique. The difference between this and the standard hill climbing is that steepest ascent considers all the possible solution in the given space and considers the solution which fits the problem to the maximum. This makes this more efficient in terms of time taken. Steepest ascent hill climbing is similar to best-first search, which tries all possible extensions of the current path instead of only one. As a problem can have different solutions, we have chosen the Steepest Ascent Hill Climbing algorithm as it looks at all the possible solutions and then gives the nearest apt solution[6].

Pseudo Code for Hill Climbing [7]:

```
i <- generate an individual randomly

best_so_far <- i

while stopping criterion has not been met:

get i's bit string and convert it to the problem representation (int or float)

increment or decrement one of the genes by the step size

if the resulting individual has higher fitness
```

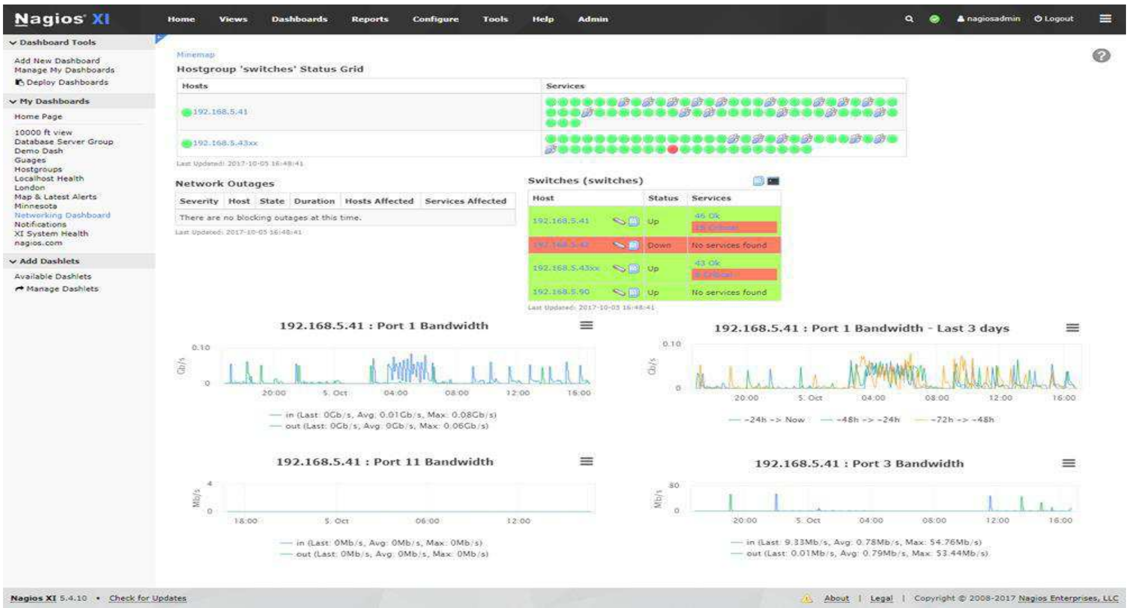
```
replace i with this individual and increase the step size
else
decrease the step size
if the step size reaches zero and increments and decrements of the current gene have been tested
move on to the next gene
if i is at a local optimum
if fitness(i) > fitness(best_so_far)
best_so_far <- i
i <- generate an individual randomly
```

RESULTS



```
Access Nagios XI at http://192.168.10.27        Default root Password: nagiosxi

localhost login: root
Password:
```

Figure 1: Nagios XI Login Screen



Hostgroup 'switches' Status Grid

Hosts	Services
192.168.5.41	
192.168.5.43xx	

Last Updated: 2017-10-05 16:48:41

Network Outages

Severity	Host	State	Duration	Hosts Affected	Services Affected
There are no blocking outages at this time.					

Last Updated: 2017-10-05 16:48:41

Switches (switches)

Host	Status	Services
192.168.5.41	Up	46 Ok [illegible]
192.168.5.42	Down	No services found
192.168.5.43xx	Up	43 Ok [illegible]
192.168.5.90	Up	No services found

Last Updated: 2017-10-05 16:48:41

192.168.5.41 : Port 1 Bandwidth
0.10
0
Gb/s
20:00
5. Oct
04:00
08:00
12:00
16:00
in (Last: 0Gb/s, Avg: 0.01Gb/s, Max: 0.08Gb/s)
out (Last: 0Gb/s, Avg: 0Gb/s, Max: 0.06Gb/s)
192.168.5.41 : Port 1 Bandwidth - Last 3 days
0.10
0
Gb/s
20:00
5. Oct
04:00
08:00
12:00
16:00
-24h -> Now
-48h -> -24h
-72h -> -48h
192.168.5.41 : Port 11 Bandwidth
4
0
Mb/s
18:00
5. Oct
06:00
12:00
in (Last: 0Mb/s, Avg: 0Mb/s, Max: 0Mb/s)
out (Last: 0Mb/s, Avg: 0Mb/s, Max: 0Mb/s)
192.168.5.41 : Port 3 Bandwidth
80
0
Mb/s
20:00
5. Oct
04:00
08:00
12:00
16:00
in (Last: 9.33Mb/s, Avg: 0.78Mb/s, Max: 54.76Mb/s)
out (Last: 0.01Mb/s, Avg: 0.79Mb/s, Max: 53.44Mb/s)
Nagios XI 5.4.10 • Check for Updates
About | Legal | Copyright © 2008-2017 Nagios Enterprises, LLC

Figure 2: Nagios Network Dashboard

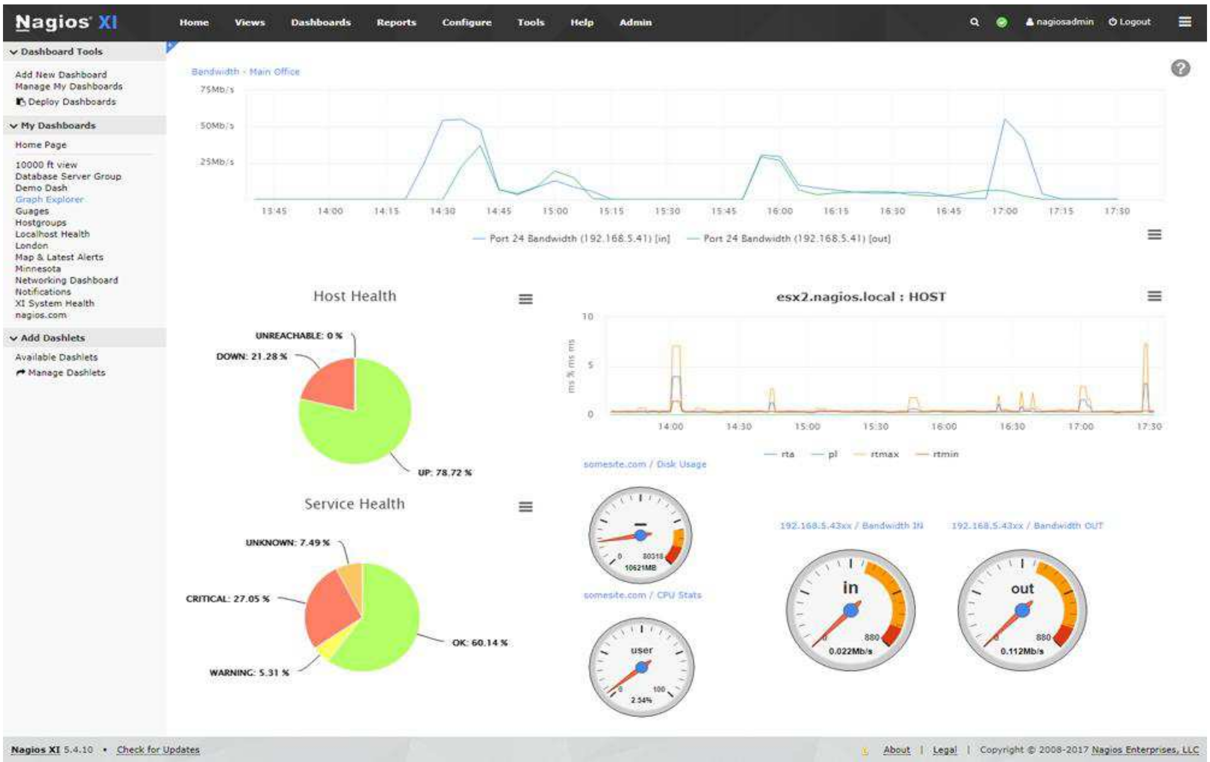


Figure 3: Nagios Performance Graphs

CONCLUSIONS

This paper discusses the possibility of monitoring infrastructure at a person or an organization level without any real difficulty by eliminating most of the manual processing involved. It is evident from our work that the remote process automation of monitoring combined with artificial intelligence framework helps reduce the human costs, the performance overheads of the organization and also leads to quick resolution of the problems that may arise.

REFERENCES

1. *http://cyborginstitute.org/projects/administration/monitoring-tactics/*
2. *https://assets.nagios.com/downloads/nagiosxi/guides/administrator*
3. *JIAWEI HAN, MICHELINE KAMBER, JIAN PEI, Data Mining Concepts and Techniques (Waltham: Morgan-Kaufmann, 2012)*
4. *https://core.ac.uk/download/pdf/6302770.pdf*
5. *https://heyitskoushik.wordpress.com/2017/12/25/support-vector-machine-algorithm-with-example/*
6. *ELAINE RICH, KEVIN KNIGHT, SHIVASHANKAR B NAIR, Artificial Intelligence (New Delhi: Tata-McGraw Hill, 2009)*
7. *http://www.cs.unm.edu/~neal.holts/dga/optimizationAlgorithms/hillclimber.html*