IITB SUMMER INTERNSHIP 2017

Project Report

IP-based Hardware Infrastructure Management for the Cloud

Principal Investigator

Prof. D.B.Phatak

Project In-charge

Mr. Nagesh Karmali

Project Mentor

Ms. Minali Upreti

Project Members

Harsh Khatore
Mayank Gupta
Rohit Mehta
Tasneem Lightwala

Summer Internship 2017

PROJECT APPROVAL CERTIFICATE

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

The project entitled "**IP-based hardware infrastructure management for the cloud**" submitted by Mr. Harsh Khatore, Mr. Mayank Gupta, Mr. Rohit Mehta and Ms. Tasneem Lightwala, is approved for Summer Internship 2017 programme from 8th May, 2017 to 5th July, 2017 at Department of Computer Science and Engineering, Indian Institute of Technology, Bombay.

_____

Mr. Nagesh Karmali
Dept of CSE, IITB
Project In-charge

_____

Prof. D. B. Phatak
Dept of CSE, IITB
Principal Investigator

Place: IIT Bombay
Date: 5th July, 2017

DECLARATION

We declare that this written submission represents our team's ideas. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Harsh Khatore
KIIT UNIVERSITY, BHUBANESWAR

Mayank Gupta
BITS PILANI, PILANI CAMPUS

Rohit Mehta
TECHNO INDIA NJR, UDAIPUR

Tasneem Lightwala
SGSITS, INDORE

Acknowledgement

We would like to express our sincere gratitude to Dr. D. B. Phatak for allowing us to work as interns at IIT Bombay and also, for being our mentor and guiding us at all times. We would also like to thank Mr. Nagesh Karmali for being a continuous support and providing us with useful insights and challenges. Our sincere gratitude to Miss Minali Upreti, for providing direction to us at every step and helping us with understanding and implementation of the project. We would like to thank Mr. Avinash Awate for providing us with this opportunity. Lastly, we would like to thank all who have directly and indirectly helped us in moving forward with the project over time.

Contents

Abstract

Appropriate cloud management is essential. Popular management actions to handle dynamic changes inside the cloud environment are: migration and elasticity of virtual resources, load balancing, system failure detection. A one hour downtime could cost as much as 6.45 million dollars in brokerage business or 90 thousand dollars in retail business. It becomes the responsibility of IT and network engineers to make sure these servers and networks are operational 24x7. In order to achieve this 100% service availability, cloud monitoring software becomes essential as these network engineers cannot afford to manually check each device constantly, especially in a large network.

Monitoring is a core function of systems administration, and is primarily a matter of communication. A good monitoring tool communicates with users about the problems, and communicates with hosts and software to take remedial action. The better it communicates, the greater the confidence administrators will have in its view of their environment. One such open source monitoring solution for over a decade has been Nagios. At its core, Nagios is an event loop that runs through a list of checks for a set of hosts, and then executes further commands to notify or take corrective actions when the output of the check changes from the last time it was run. As checks execute, a small text file is written out by the collector process that contains the textual and numerical output as well as some statistical and timing information for the executed check. The master Nagios process periodically reads all these text files, parses the output of the checks, takes any actions (like notifying the owner of the check or taking corrective action).

But Nagios monitors the cloud at the Operating System level. So, using Nagios one cannot manage a computer system which is powered off or otherwise unresponsive by the network. The Intelligent Platform Management Interface (IPMI) is a set of computer interface specifications for an autonomous computer subsystem that provides management and monitoring capabilities independently of the host system's CPU, firmware (BIOS or UEFI) and operating system. IPMI defines a set of interfaces used by system administrators for out-of-band management of computer systems and monitoring of their operation.

The aim of this project is to build a hardware infrastructure manager using IPMI specification.

Introduction

Enterprises manage and monitor their IT environments to ensure their healthy condition. There are a variety of IT resources that need to be managed and monitored. They include infrastructure resources such as compute and storage servers, desktops, laptops, routers, switches, clusters, load balancers, and printers. In addition they also need to manage and monitor various platforms such as application servers, database servers, applications, data, and users. Enterprise's Cloud adoption further adds to the list heterogeneous virtualization and cloud infrastructure, and various services offered in either shared or isolated form on Cloud. So it is a complex environment to be managed and monitored. Once the resources have been provisioned and are in-use, one need to monitor them through periodic checks to ensure their proper functioning and availability. Monitoring also includes trend analysis and reporting.

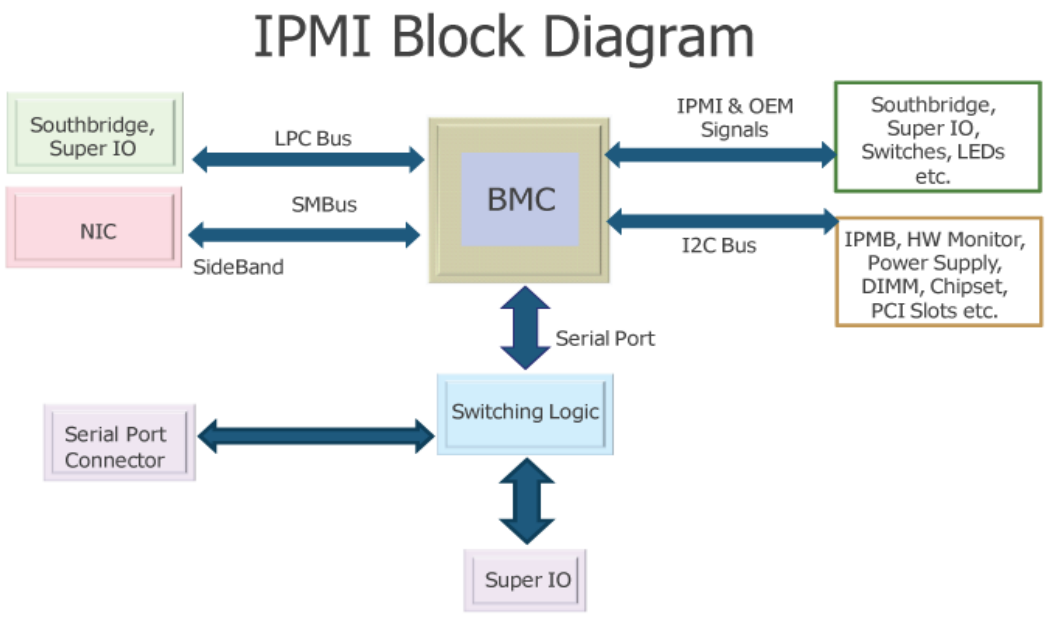
Management and monitoring can keep track of uptime, response time and latency over-time, and also provide alert notifications when the system behaviour does not comply with the defined policy. It can detect issue before it becomes a true outage, and can help to avert any problems arising out of it. It keeps track of logs and offers effective security and compliance monitoring. Management and monitoring reduces complexity, risk and overall cost. It also provides user access and permission control, cost control and allocation, and complete visibility into IT operation.

Nagios Core is a free and open source computer-software application that monitors systems, networks and infrastructure. Nagios core offers monitoring and alerting services for servers, switches, applications and services. It alerts users when things go wrong and alerts them a second time when the problem has been resolved. Nagios being customizable and flexible in nature, the customers can customize and/or extend functionality via plug- ins as per their needs. It also has the facility to integrate with customer's existing monitoring tools.

Using a standardized interface and protocol allows systems-management software based on IPMI to manage multiple disparate servers. As a message-based, hardware-level interface specification, IPMI operates independently of the operating system (OS) to allow administrators to manage a system remotely in the absence of an operating system or of the system management software. Thus IPMI functions can work in any of three scenarios:

- before an OS has booted (allowing, for example, the remote monitoring or changing of BIOS settings).

- when the system is powered down.
- after OS or system failure – the key characteristic of IPMI compared with in-band system management such as by remote login to the operating system using SSH.



Another such Open Source monitoring software is Zabbix.

Zabbix uses a relational database to store data. Its backend is written in C and the web frontend is written in PHP. Zabbix offers several monitoring options:

- Simple checks can verify the availability and responsiveness of standard services such as SMTP or HTTP without installing any software on the monitored host.
- A Zabbix agent can also be installed on UNIX and Windows hosts to monitor statistics such as CPU load, network utilization, disk space, etc.
- As an alternative to installing an agent on hosts, Zabbix includes support for monitoring via SNMP, TCP and ICMP checks, as well as over IPMI, JMX, SSH, Telnet and using custom parameters.

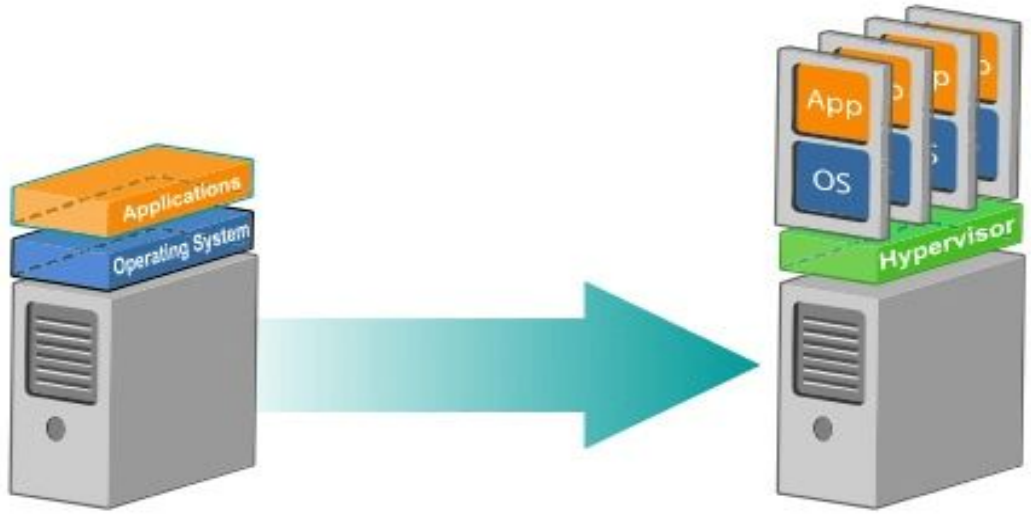
Comparison between Nagios and Zabbix

Nagios	Zabbix
First version was introduced in 1999	First version was introduced in 2001
Shows status of various parameters/ devices.	Shows performances as well as status of various devices
To add another instance or host, need to configure various files manually.	It provides a web management which allows central management without configuring files manually.
Doesn't use any database to store information	Uses database to store information
Installation and management is complex	Installation and management is comparatively easy
It follows bazaar paradigm	It follows cathedral paradigm
Provides a huge plugin library for the community	Does not have "official" plugin library
Need to add various add-ons for proper functioning	Doesn't need to add many add-ons
Large community	Relatively smaller community
Need to add plugins for visualization	For Visualizations plugins are prebuilt
Better performances because it directly use file system	Comparatively slow performance as uses database

Study

1. Virtualization

Virtualization is a kind of technology that is rapidly transforming the IT landscape and has changed the way people compute. It reduces hardware utilization, saves energy and costs and makes it possible to run multiple applications and various operating systems on the same Server at the same time. It increases the utilization, efficiency and flexibility of existing computer hardware.

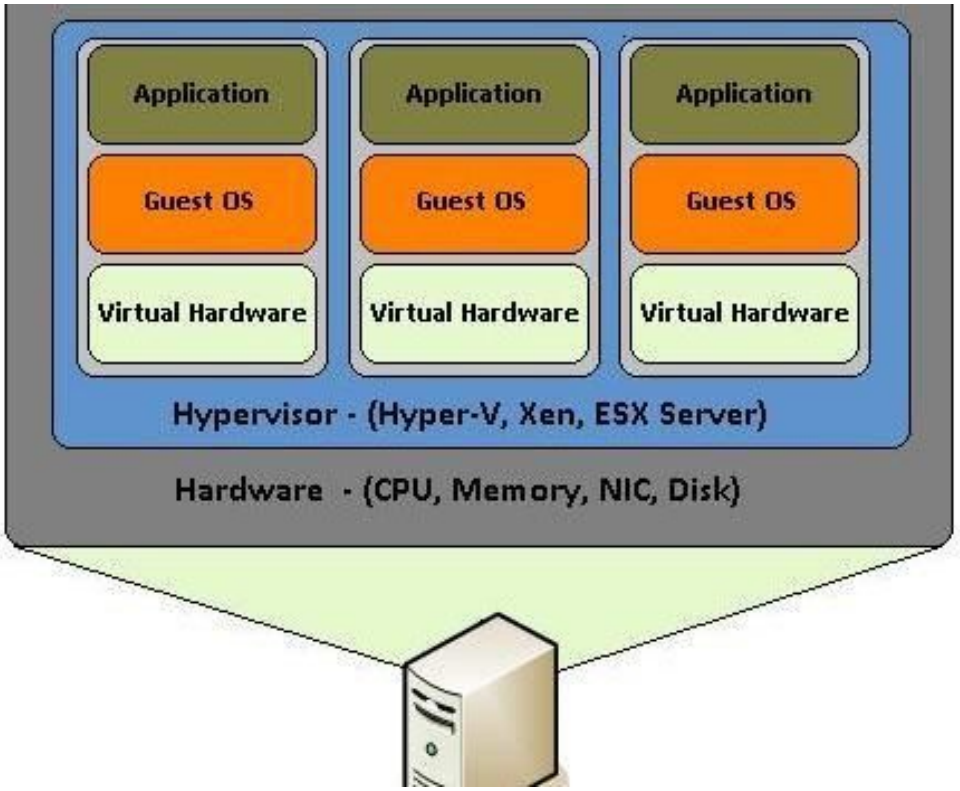


Need for Virtualization

- Provides ability to manage resources effectively.
- Increases productivity, as it provides secure remote access.
- Provides for data loss prevention.

What makes virtualization possible?

There is a software that makes virtualization possible. This software is known as a Hypervisor, also known as a virtualization manager. It sits between the hardware and the operating system, and assigns the amount of access that the applications and operating systems have with the processor and other hardware resources.



2. Cloud Computing

Cloud computing is a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. Cloud computing means "a type of Internet-based computing," where different services , such as servers, storage and applications are delivered to an organization's computers and devices through the Internet.

Cloud computing is taking services ("cloud services") and moving them outside an organization's firewall on shared systems. Applications and services are accessed via the Web, instead of your hard drive. The services are delivered and used over the Internet and are paid for by cloud customer (your business), typically on an "as-needed, pay-per-use" business model. The cloud infrastructure is maintained by the cloud provider, not the individual cloud customer.

How it Works

Cloud computing applies traditional supercomputing power, normally used by military and research facilities, to perform tens of trillions of computations per second. To do this, cloud computing uses networks of large groups of servers typically running low-cost consumer PC

technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

The standards for connecting the computer systems and the software needed to make cloud computing work are not fully defined at present time, leaving many companies to define their own cloud computing technologies. Organizations choose cloud providers that satisfy their needs. Cloud computing systems offered by companies, like IBM's "Blue Cloud" technologies, for example, are based on open standards and open source software which link together computers.

3. IPMI

Intelligent Platform Management Interface (IPMI) is a series of specifications that provide monitoring (system temperature, fans, power supplies and so forth) and management (booting and shutting down the server) capabilities. It is independent of host system's CPU, firmware and Operating System.

Components:

Baseband Management Controller (BMC) which is present in servers. It acts as an interface between system management software and the hardware. It monitors and create logs.
Intelligent Platform Management Bus (IPMB) for communication to and between the management controllers.
Intelligent Chassis Management Bus (ICMB): ICMB provides a standardized interface for communication and control between chasses.

Memory Areas:

System Event Log (SEL) Repository: BMC contains a central, non-volatile System Event Log (SEL). It must need to be periodically checked and deleted because of limited memory.
Field Replaceable Unit (FRU) Information: Contains information like serial number, port number, models and inventory number.

Communication Interface:

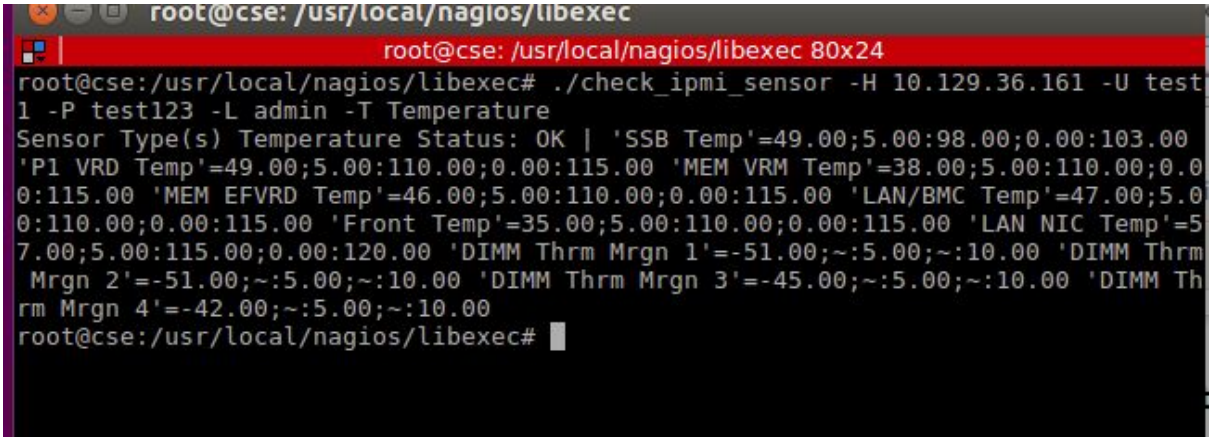
System interfaces (local access): Interface between system software to BMC.

- Keyboard controller style
- System Management Interface Chip
- Block Transfer
- SMBus system interface

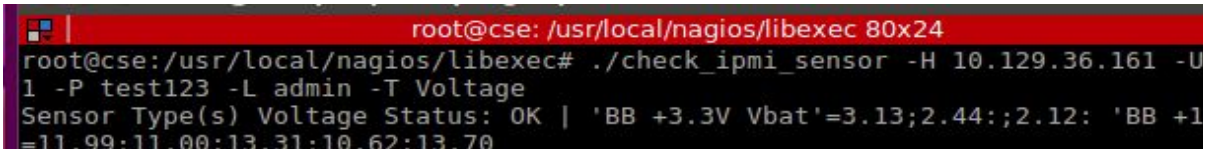
LAN Interface: Defines how IPMI message is transmitted to and from BMC in Remote Management Control Protocol (RMCP), UDP datagram.

Example of usage:

1. Temperature data:



2. Voltage data:



Similarly details of other sensors can also be determined:

Sensor Type	**FreeIPMI sensor type naming**
Current	Current
Fan	Fan
Power supply	Power_supply
Power unit	Power_unit
Memory	Memory
Watchdog 1	Watchdog_1
Button/ Switch	Button_Switch
Chassis	Chassis
LAN	LAN
Battery	Battery

4. SNMP

What is SNMP?

Simple Network Management Protocol (SNMP) is an Internet standard protocol for managing devices remotely and locally on IP network like routers, switch, servers, workstation etc. It is an application level protocol designed for application layer.

SNMP Versions

Two major versions:
1. SNMPv1 is a recommended standard
2. SNMPv2:
    2.1 SNMP v2c - SNMPv2 with user-based security
    2.2 SNMP v2*- SNMPv2 with user-based security and additional features
    2.3 SNMP v2c- SNMPv2 without security
3. SNMPv3: Started from SNMPv1 addressing security

How SNMP works?

SNMP Architecture: Manager and Agent
1. A manager checks an agent by requesting information that reflects the behavior of the agent.
2. A manager forces an agent to perform a task by resetting values in the agent database.
3. An agent contributes to management process by warning the manager of an unusual situation through traps.

The SNMP agent receives requests on UDP port 161. The manager may send requests from any available source port to port 161 in the agent. The agent response will be sent back to the source port on the manager. The manager receives notifications (Traps and Inform Requests) on port 162. The agent may generate notifications from any available port.

SNMP Components

Management information base (MIB): SNMP protocol provides information about devices and that variables information is provided by management information base (MIB). It is collection of objects and their types in hierarchical tree format.
Structure of management information (SMI): SMI defines rule for naming objects, defining object types and showing how to encode objects and data.

5. SMTP

What is SMTP?

Simple Mail Transfer Protocol is a protocol that allows to send email from email client to mail server. SMTP uses domain name in an email address to locate the recipient's email server where the email remains until the recipients receive it. It is a part of TCP/IP protocol suite. It defines the message transfer agent (MTA) and message format. Originally SMTP sends only text but MIME and other encoding enable executable program and multimedia file to attach to transported email. SMTP uses TCP port 25

Operation

1. When an SMTP client has a message to transmit, it establishes a two- way transmission channel to an SMTP server. The responsibility of an SMTP client is to transfer mail messages to one or more SMTP servers.
2. Once the transmission channel is established and initial handshaking completed, the SMTP client normally initiates a mail transaction. Such a transaction consists of a series of commands to specify the originator and destination of the mail and transmission of the message content (including any headers or other structure) itself.
3. The server responds to each command with a reply; replies may indicate that the command was accepted, that additional commands are expected, or that a temporary or permanent error condition exists.
4. Once a given mail message has been transmitted, the client may either request that the connection be shut down or may initiate other mail transactions.

Problems

1. SMTP does not require authentication. This allows anyone on internet to send email to anyone else or even large number of people. Because of this junk mails or spam mails are possible.
2. If the client and server have different timeout then one of them may give up while other is still busy, unexpectedly terminate the connection.
3. Older implementation cannot handle messages exceeding 64KB.

6. Distributed System

What is distributed system?

A distributed system is a collection of autonomous computers linked by a computer network that appear to the user of system as a single computer.

Challenges of distributed system:

1. Security is a big challenge in distributed environment.
2. Fault tolerance is tough when distributed model is built based on unreliable components
3. Coordination and resource sharing can be difficult if proper protocols are not in place.
4. Process knowledge should be put in place for the administrators and users of the distributed model.

Distributed Monitoring Solutions for Nagios

The goal of distributed monitoring is to allow your Nagios environment to monitor a large infrastructure. No single solution is the "right" solution for every environment. The method you choose to implement should be based on your end-goals, as well as the time and effort required to deploy and maintain the chosen solution.

Various distributed monitoring models

1. Mod-Gearman:

Mod-Gearman approaches the problem of scalability and complexity of distributed Nagios setups with a master/worker configuration, where one Nagios server hands out jobs to worker nodes. These worker nodes have the Nagios plugins installed, but do not have any configuration of their own, and no checks actually "assigned" to them. This relieves the administrator of the problem of having to maintain configurations on multiple machines and losing service checks if a worker node goes down. Performance graphing and other I/O intensive tasks are handled on the central server, which may limit scalability.

2. Nagios Fusion:

Fusion allows you to scale your monitoring environment by deploying additional Nagios Core servers to monitor additional hosts, services, and applications. Each Core server monitors a portion of the entire infrastructure, and Fusion provides a central dashboard that allows you to quickly see the status of everything from a single page.

Problems Identified

1. **No implicit implementation of IPMI protocol in Nagios core:**
   Nagios Core does not have inbuilt ipmi functionality. Additional plugins were needed to be integrated, for ipmi to work.

2. **No proper output for IPMI:**
   Nagios core does not support the required output format needed to display the results properly given by the IPMI plugin. It was dumping the output into a string, which was displayed abruptly and difficult to interpret.

3. **Setting up POSTFIX and SSMTP in the intranet:**
   Due to proxy constraint in the intranet, proper configuration of postfix and ssmtp was required, to send notifications and proper functioning of Nagios Core.

4. **Notifications were not sent when the System Event Log in the server increases:**
   Nagios core does not send notifications or allow you to log changes in the output service and host checks if the state of the host or service does not change.

5. **Adding a new monitoring host or service required going through source code and making changes in many files:**
   Adding a new host to be monitored or adding a new service to an existing host required us to make changes in many configuration files.

Technologies Used

1. **Nagios:**
   Nagios Core, is a free and open source computer-software application that monitors systems, networks and infrastructure. Nagios offers monitoring and alerting services for servers, switches, applications and services. It alerts users when things go wrong and alerts them a second time when the problem has been resolved.

2. **SMTP:**
   Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (email) transmission. First defined by RFC 821 in 1982, it was last updated in 2008 with the Extended SMTP additions by RFC 5321—which is the protocol in widespread use today. SMTP by default uses TCP port 25. The protocol for mail submission is the same, but uses port 587. SMTP connections secured by SSL. Although electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for relaying. For retrieving messages, client applications usually use either POP3 or IMAP.

3. **FreeIPMI:**
   FreeIPMI is a collection of Intelligent Platform Management IPMI system software. It provides in-band and out-of-band software and a development library conforming to the Intelligent Platform Management Interface (IPMI v1.5 and v2.0) standards. FreeIPMI also supports IPMI-related specifications such as the Data Center Management Interface (DCMI) and Intel Node Manager.

4. **Languages:**
   1. C (nagios core source code) : The source code of Nagios was written mostly in C language. Some of the main files in which we made major changes are status.c and extinfo.c.
   2. Perl (IPMI plugin) : The plugin which we used to integrate the Nagios Core with IPMI is written in perl language. The same plugin can be found on our github repo as check_ipmi.pl
   3. PHP(ipmi details file and adding new host): To display the individual ipmi details of each monitored device, separate php file was created. To simplify the task of adding new monitoring host, a separate form was created, asking the user, details about the new host.

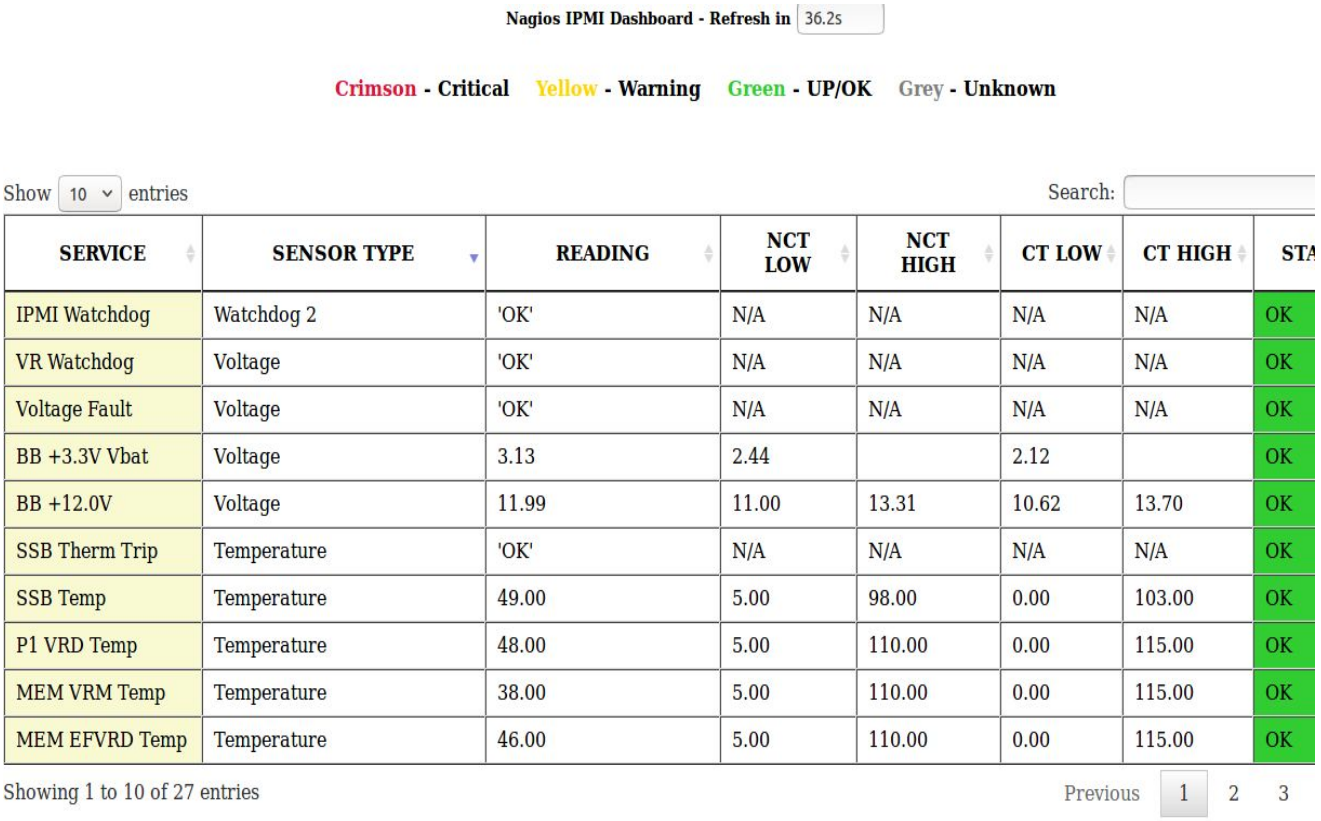
4. HTML-CSS (modify frontend) : At some places we were required to modify the frontend of the Nagios portal. This was done was writing and altering some HTML-CSS statements in the C and PHP files.
5. Bash (shell script) : Many a times we were required to run some terminal commands by writing them in files. Bash was used for the same.

5. Github:

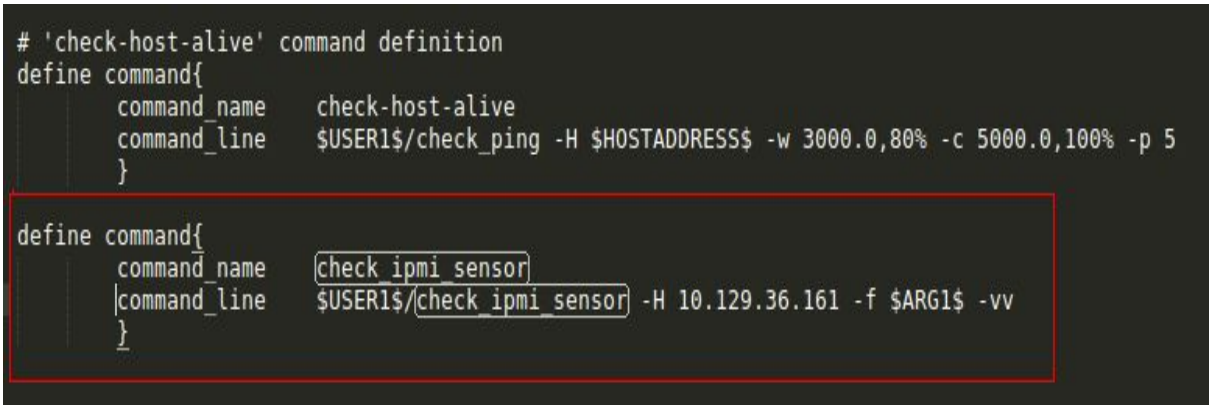
GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management(SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project. GitHub offers both plans for private repositories and free accounts, which are usually used to host open-source software projects. As of April 2016, GitHub reports having more than 14 million users and more than 35 million repositories, making it the largest host of source code in the world. Projects on GitHub can be accessed and manipulated using the standard Git command-line interface and all of the standard Git commands work with it. GitHub also allows registered and non-registered users to browse public repositories on the site. Multiple desktop clients and Git plugins have also been created by GitHub and other third parties that integrate with the platform.

Experimental Result

1. **Display IPMI details in a separate tab**:
   1.1. For all the services provided by Nagios for a monitoring host, it creates a common status.dat file, through which it extracts the information about hardware health and displays appropriately.
   1.2. Using this status.dat file, which auto-updates itself after every sixty seconds (refresh time can be changed and the lowest refresh time is 1 second), we take the values given by IPMI plugin as output and present it in a separate window, containing a detailed information about the hardware services which can be monitored by IPMI including system temperature, fan speed, etc..
   1.3. We have used languages such as HTML, CSS and PHP to develop the plugin and AJAX to implement the search functionality.
   1.4. Hardware service reading, threshold values and status have been displayed for easy monitoring. We can search for a particular category of service, a specific service etc..



2. **Show complete output of the IPMI plugin**:
   2.1. Previously, from the ipmi plugin we were only able to extract constrained information, by default. There is a verbosity option in the plugin to get a detailed output about the services. Hence we made the change in the command being executed and got the required output(in the figure above).
   2.2. The change to the command was made in the 'commands.cfg' file, which the file which describes all the possible services Nagios gives information about.
   2.3. Location of the file - /usr/local/Nagios/etc/objects/
   2.4. The image below shows the changes made to the the command.

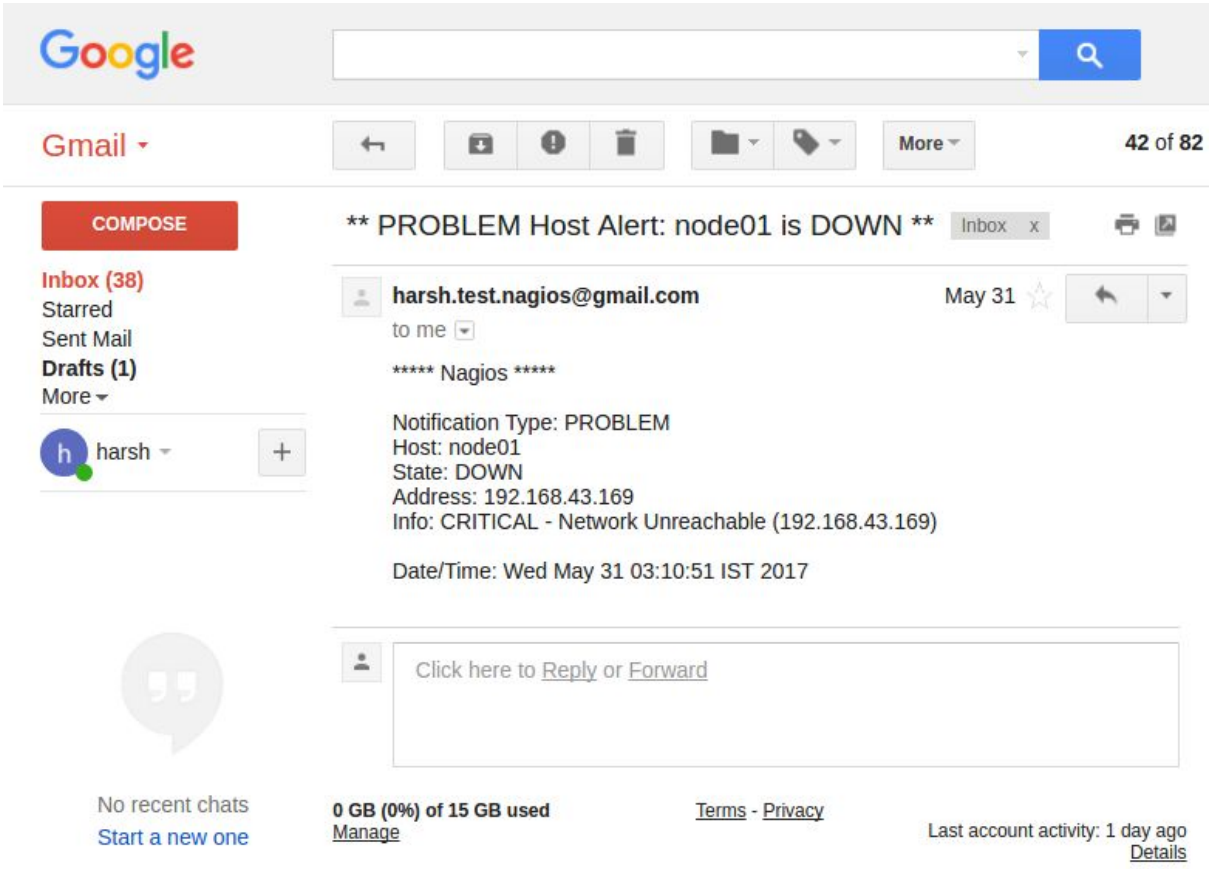


3. **Send mail for hardware services malfunction**:
   3.1. It is not just important to monitor the hardware infrastructure but we also need to alert the administrator if something goes wrong in the cloud. Nagios provides the service to send both custom and automated mail to notify the administrator about any malfunction in the cloud.
   3.2. The main challenge here was to set up the SMTP/SSMTP and POSTFIX and configure them according to the network requirements and needs. We were successful in setting up the mail service required by Nagios.
   3.3. Nagios only comes with a a set of 2 default notifications through e-mails: *notify-host-by-email* and *notify-service-by-email*. These notifications provide a minimalistic, barebone structure for sending a simple e-mail alert to defined contacts when problems occur (example screenshot below).

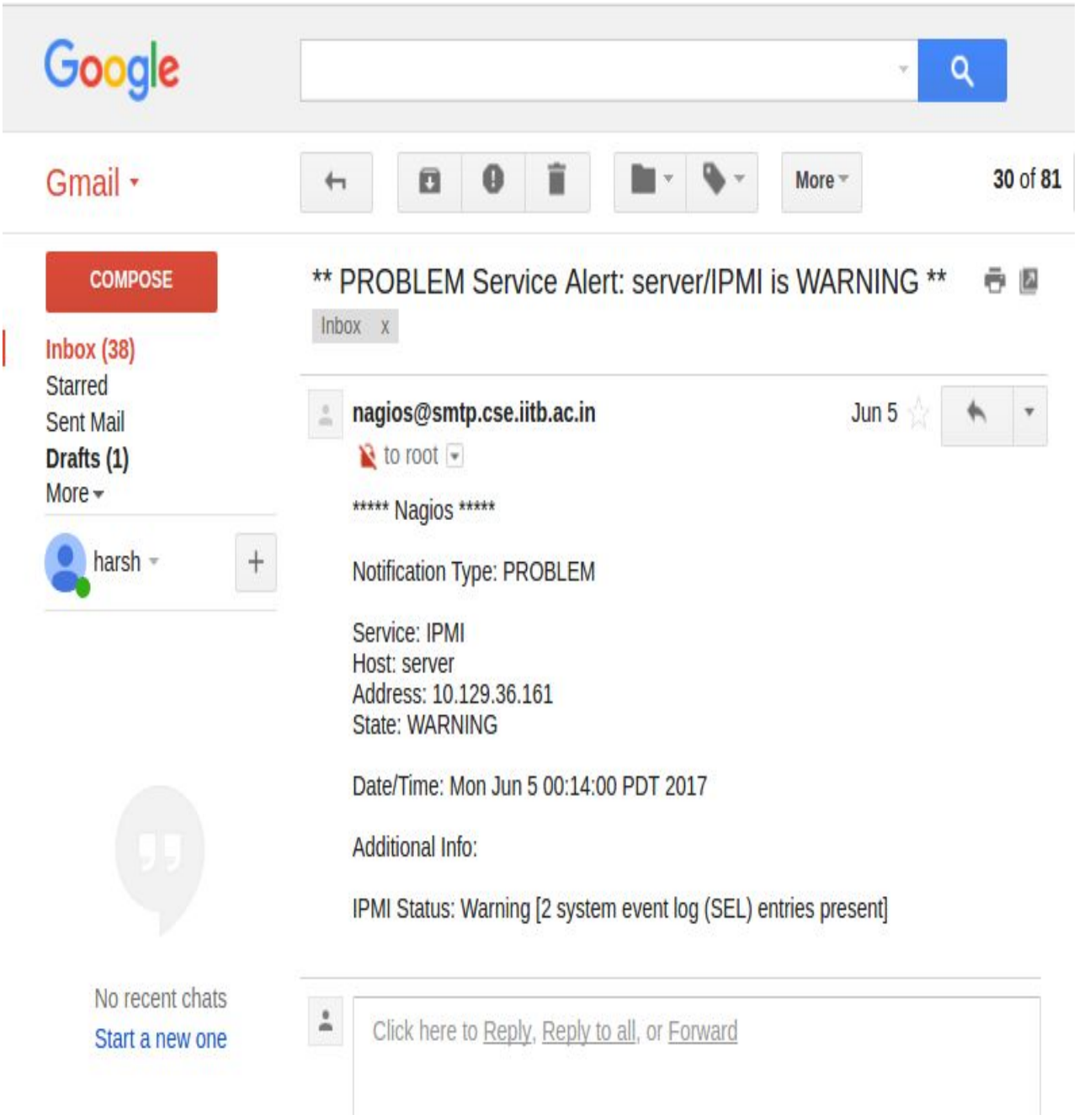
Date: January 1, 2011 11:06:41 AM GMT+09:00
From: Nagios Monitoring user <nagios@susie112.frank4dd.com>
Subject: ** PROBLEM Host Alert: dbserver1 is DOWN **
To: support@frank4dd.com

```
***** Nagios *****

Notification Type: PROBLEM
Host: dbserver1
State: DOWN
Address: 70.85.16.87
Info: CRITICAL - Host Unreachable (70.85.16.87)

Date/Time: Sat Jan 1 11:06:41 JST 2011
```

3.4. Mail examples:

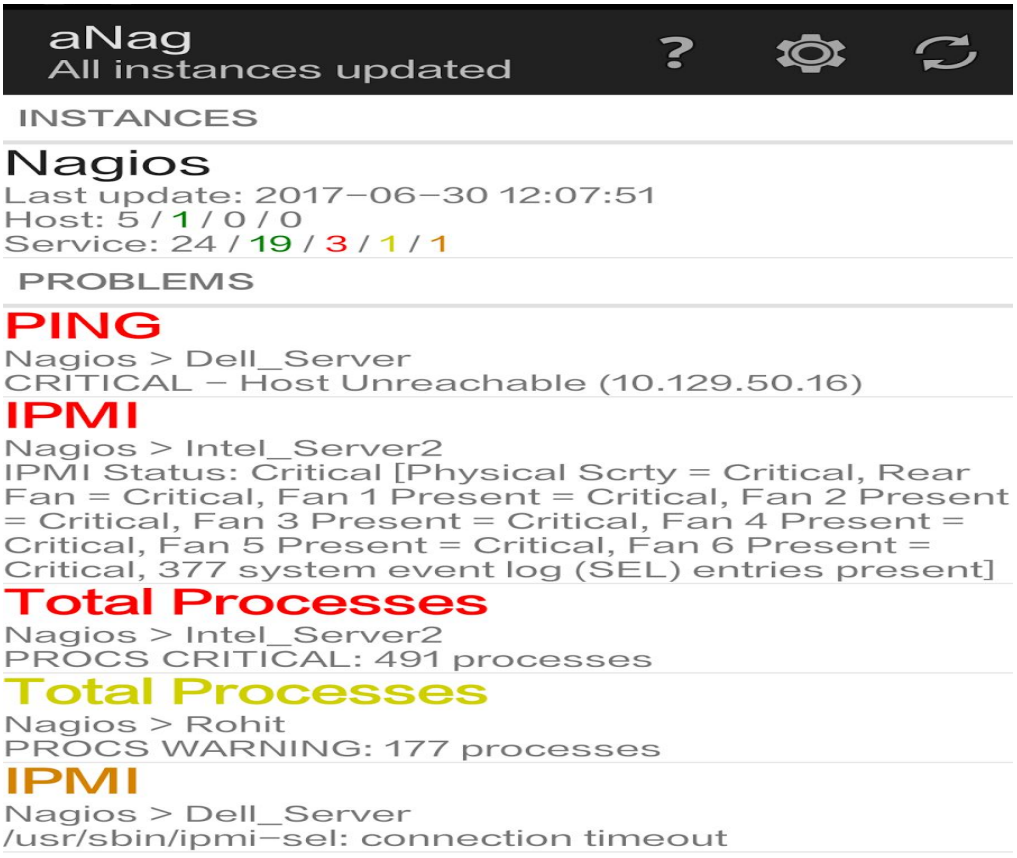


The above image notifies about a host not reachable.



The above image notifies when there are multiple system logs present in the Event log of the server.

We also have an android app which can be used to notify the administrators about any problem in the system:



4. **Send email when multiple event log entries are present**:
   The above image brings us to the next problem we faced, which was, we were not receiving mail notification when system log on the server incremented by more that one in number.
   The problem can be solved by two approaches:
   4.1.Delete the log on the server. This can be done whenever system log exceeds a value of one. The following command can be used to do so:
   ipmiutil sel -d -V 4 -N ip-address_of_the_server -U username -P password

   4.2. Add 'stalking_options' to the service definition.

5. **Adding a new host to monitor:** While working with nagios core one of the major issues we found was that there was no user interface for adding new host.Every Time to add a host it was required to manually create config file for them and then add its detail in the main configuration file called nagios.cfg. We decided to resolve this problem by creating a option on nagios portal to add a user by taking two inputs from the user, the **hostname** and its **IP**. Then we wrote a script in PHP which automates all the tasks that were required to add a new host. By default we added only one service

in the host, the **PING s**ervice. Which brings us to our next issue of adding new services to existing hosts.

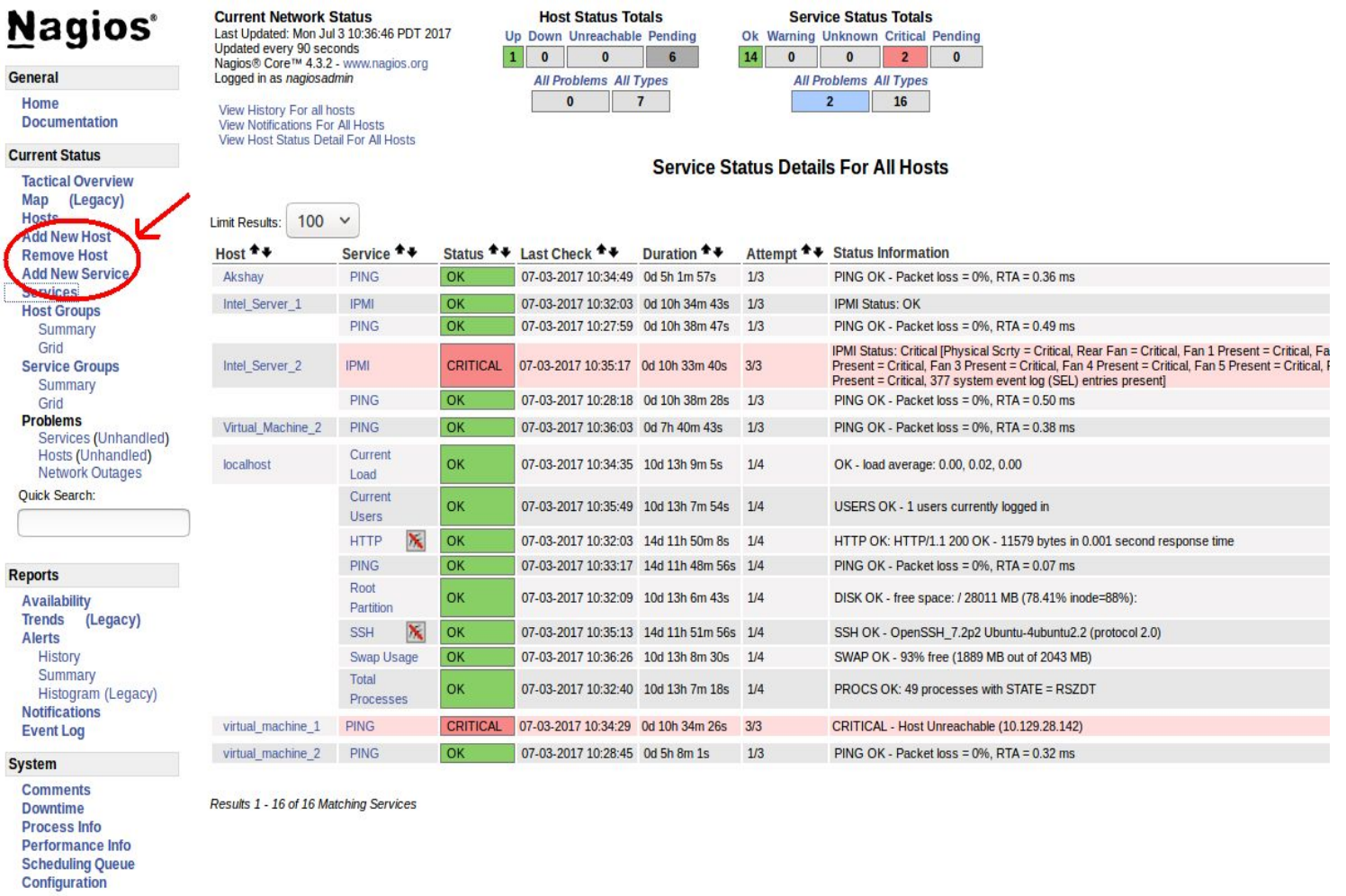
```php
<?php
    //AIM:  To add new host to nagios core
    //Save this file at location: /var/www/html/form.php
?>

<html>
<body>


<?php
    //Changing permission for directory object and files nagios.cfg and host.cfg
    shell_exec('chmod 777 /usr/local/nagios/etc/objects');
    shell_exec('chmod 777 /usr/local/nagios/etc/nagios.cfg');
    shell_exec('chmod 777 /usr/local/nagios/etc/objects/host.cfg');

    #chdir = change directory
    chdir('/usr/local/nagios/etc/objects');

    #new variable to add '.cfg' extension
    #$file_name.cfg will be the config file of new host
    $file_name = $_POST["name"].'.cfg';

    #To remove space in file name
    $file_name = str_replace(' ', '', $file_name);
    $display = $file_name." successfully added.  Restart nagios deamon to reflect changes";

    #shell command to create an empty file
    shell_exec('touch ${$file_name}');

    #shell command to change permission
    shell_exec('chmod 777 ${$file_name}');

    echo $display;

    $myfile = fopen("{$file_name}", "w")  or die("Unable to open file!");
    #To add details of host in $file_name.cfg. And adding PING as a default service
    $to_write=  "define host{
                                use          generic-host
                                host_name    ".$_POST["name"].
                                "\nalias "     .$_POST["name"].
                                "\naddress    ".$_POST["ip"].
                                "\nmax_check_attempts              3
                }


                define service{
                                use                             generic-service
                                host_name "                     .$_POST["name"].
                                "\nservice_description        PING
                                check_command                   check_ping!100.0,20%!500.0,60%
                }";


    fwrite($myfile, $to_write);
    fclose($myfile);

    #Adding the location of $file_name.cfg in nagios
    chdir('/usr/local/nagios/etc');
    $myfile = fopen("nagios.cfg", "a")  or die("Unable to open file!");
    $to_write= "\ncfg_file=/usr/local/nagios/etc/objects/".$file_name."\n";
    fwrite($myfile,$to_write);
    fclose($myfile);

?>
<a href="http://10.129.26.133/nagios/main.php">Home Page</a>
</body>
</html>
```

6. **Adding a new service to an already existing host:** There may be at times need to add a new service to an already existing host, or expanding the types of services to add. Nagios core has a tricky mechanism to add new services to a host. We created a PHP script and added it to the main nagios core dash, so that it can be easily done. We have also checked for cases such as adding service to a non-existent host, or adding duplicate service, and these exploitations are addressed in our script.

```
<html>
    <!---AIM: To add service to host -->
    <!-- SAVE THIS FILE AT LOCATION /var/www/html/formservices.php -->
<body>


<?php
    #chdir = change directory
    chdir('/usr/local/nagios/etc/objects');

    #new variable to add '.cfg' extension
    $file_name = $_POST["hostname"].'.cfg';

    #To Check wheather the host exist or not.
    if(file_exists($file_name) != 1){
        echo "Invalid Hostname ";

    }
    else{
        shell_exec('chmod 777 /usr/local/nagios/etc/objects/${$file_name}');
        $myfile = fopen("{$file_name}", "a")  or die("Invalid Hostname!");
        $sname = $_POST["service"];
        switch($sname){
            case "Current Users"  : $check_command="check_nrpe!check_users";
                                    break;
            case "Current Load"   : $check_command="check_nrpe!check_load";
                                    break;
            case "Root Partition" : $check_command="check_nrpe!check_disk";
                                    break;
            case "Total Processes": $check_command="check_nrpe!check_total_procs";
                                    break;
        }#end of switch
        #To define the service which we want to add
        $to_write= "\ndefine service{
                                        use                             generic-service
                                        host_name  "                    .$_POST["hostname"].
                                        "\nservice_description  "       .$_POST["service"].'
                                        check_command           {$check_command}
                    }";

        #Write the service in $file_name.cfg
        fwrite($myfile, $to_write);
        fclose($myfile);
        echo "service successfully added";
    }
?>
<a href="http://10.129.26.133/nagios/main.php">Home Page</a>
</body>
</html>
```

7. **Removing a monitoring host:** Removing monitored hosts from Nagios core was as tricky as adding a new one. To simplify this functionality, we added a PHP script to remove the host from the monitoring list of Nagios core. To achieve this, we removed the configuration file of that particular host and also removed the path to this file from the main nagios configuration file. Edge cases such as, trying to remove invalid host have been handled. These problems were addressed to simplify the user experience of administrators to add new hosts, services and delete them.

```
<html>
<body>
<!-- AIM: To remove already added host  -->
<!-- SAVE THIS FILE AT LOCATION: /var/www/html/formremove.php -->

<?php

    #chdir = change directory
    chdir('/usr/local/nagios/etc/objects');

    #new variable to add '.cfg' extension
    $file_name = $_POST["hostname"].'.cfg';

    #To check wheather the host exist or not
    if(file_exists($file_name) != 1){
        echo "Invalid Hostname ";

    }
    else{
        #To remove .cfg file of host
        shell_exec('rm -f ${$file_name}');
        chdir('/usr/local/nagios/etc');

        shell_exec('chmod 777 nagios.cfg');
        $myfile = fopen("nagios.cfg","r+") or die("File not found");

        $to_match = "cfg_file=/usr/local/nagios/etc/objects/{$file_name}\n";

        fseek($myfile, 0);
        $start = $myfile;
        $size = 0;
        //To itterate to the position in the file where location of $file_name.cfg is added
        while (($line = fgets($myfile)) !== false) {

            if($line == $to_match){
                fseek($myfile, $size);
                //comment that line
                fwrite($myfile,"#");
                fclose($myfile);
                break;
            }

            $size += strlen($line);
        }

    }

?>
<a href="http://10.129.26.133/nagios/main.php">Home Page</a>

</body>
</html>
```



The above image highlights the newly added functionalities by us in the sidebar.

Conclusion

We would like to conclude this project with the successful integration of IPMI with Nagios Core. The primary goal of this project, which was to monitor hardware components of a remote server like hard disk space, processor information, temperature details etc., has been implemented and tested. Also, we were able to display these IPMI details in a user friendly manner on a separate window in browser.

Those systems which does not include hardware for ipmi, were monitored using nrpe agent for Nagios Core.

Additional functionalities to simplify the user experience of Nagios Core have also been added, which include, adding and removing monitoring host dynamically and also, adding new services to an already existing host. Currently,we have provided option to add four types of services dynamically, but it can be extended to more number of services as and when the new service comes by adding the service definition.

Challenges and Future Scope

1. **Benchmarking Nagios core:** Nagios core can be implemented for cluster systems and distributed architecture. But, it has not be tested much for large scale installations and hence, benchmarking can be done for such a scenario.
2. **Getting hardware details without using nrpe agent:** Generate hardware details of systems without using additional service like the nrpe agent on the monitoring host.
3. **Distributed Monitoring:** The general goal of distributed monitoring is to allow a Nagios Core environment to scale to a large infrastructure.
4. **Automated response system:** An intelligent system which responds to problems accordingly, like temperature control of the cooling system, distribution of disk space as needed, etc..

References

1. **Links:**
   - **1.1. Nagios:**
     - 1.1.1. Nagios developer forum: https://support.nagios.com/forum/
     - 1.1.2. Nagios community: https://www.nagios.org/about/community/
     - 1.1.3. Nagios Installation: https://support.nagios.com/kb/article/nagios-core-installing-nagios-core-from-source.html
     - 1.1.4. Ipmi plugin: https://github.com/thomas-krenn/check_ipmi_sensor_v3
     - 1.1.5. Nagios source code: https://github.com/NagiosEnterprises/nagioscore
     - 1.1.6. Nagios nrpe: https://exchange.nagios.org/directory/Addons/Monitoring-Agents/NRPE--2D-Nagios-Remote-Plugin-Executor/details
   - **1.2. Protocols:**
     - 1.2.1. https://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface
     - 1.2.2. https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
     - 1.2.3. https://en.wikipedia.org/wiki/Simple_Mail_Transfer_Protocol
     - 1.2.4. https://en.wikipedia.org/wiki/Postfix_(software)
     - 1.2.5. https://www.thomas-krenn.com/en/wiki/IPMI_Basics
     - 1.2.6. https://www.intel.in/content/www/in/en/servers/ipmi/ipmi-second-gen-interface-spec-v2-rev1-1.html
   - **1.3. Other references:**
     - 1.3.1. https://www.thomas-krenn.com/en/wiki/Configuring_IPMI_under_Linux_using_ipmitool
     - 1.3.2. https://linux.die.net/man/1/ipmitool
     - 1.3.3. https://docs.oracle.com/cd/E19464-01/820-6850-11/IPMItool.html
     - 1.3.4. https://discuss.pivotal.io/hc/en-us/articles/206396927-How-to-work-on-IPMI-and-IPMITOOL
     - 1.3.5. https://www.ibm.com/support/knowledgecenter/en/linuxonibm/liabp/liabpcommonipmi.htm
     - 1.3.6. https://www.thomas-krenn.com/en/wiki/FreeIPMI
     - 1.3.7. https://www.gnu.org/software/freeipmi/
     - 1.3.8. https://linux.die.net/man/8/ipmiutil
     - 1.3.9. http://ipmiutil.sourceforge.net/
     - 1.3.10. http://ipmiutil.sourceforge.net/docs/ipmisw-compare.htm
     - 1.3.11. https://assets.nagios.com/downloads/nagioscore/docs/nrpe/NRPE.pdf

2. Research papers:

2.1. Rubén S. Montero and Ignacio M. Llorente, Borja Sotomayor, and Ian Foster: Virtual Infrastructure Management in Private and Hybrid Clouds, IEEE Internet Computing, Sep.-Oct. 2009.

2.2. Jiyi Wu, Lingdi Ping and Xiaoping Ge: Cloud Storage as the Infrastructure of Cloud Computing, Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference, June 2010.

2.3. Sunilkumar S.Manvi, and Gopal Krishna Shyam: Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey, Journal of Network and Computer Applications, May 2014.

2.4. Timme Katz: Intelligent platform management interface, Universität Hamburg, September 2010.