A Framework for Monitoring Data Warehousing Applications

Article · June 2020

CITATIONS
5

READS
626

2 authors:

Madhusudhan Reddy Sureddy
Santander bank
9 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Prathyusha Yallamula
7 PUBLICATIONS **11** CITATIONS
SEE PROFILE

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

A Framework for Monitoring Data Warehousing Applications

Madhusudhan Reddy Sureddy[1], Prathyusha Yallamula[2]

[1]*Associate Director, Application Development, Santander, Holmdel, NJ, USA*
[2]*Vice President, Risk Treasury, Natixis CIB Americas, New York, NY, USA*

Abstract - *Data warehousing environmental architecture tend to become very complex and can easily become an operational nightmare without a robust monitoring framework. Monitoring Framework requirements for data warehousing applications are quite unique as compared to other applications due to the various types of sub applications involved. This article provides a comprehensive monitoring framework which satisfies all these unique requirements and provides the means to build a monitoring solution for data warehouse applications.*

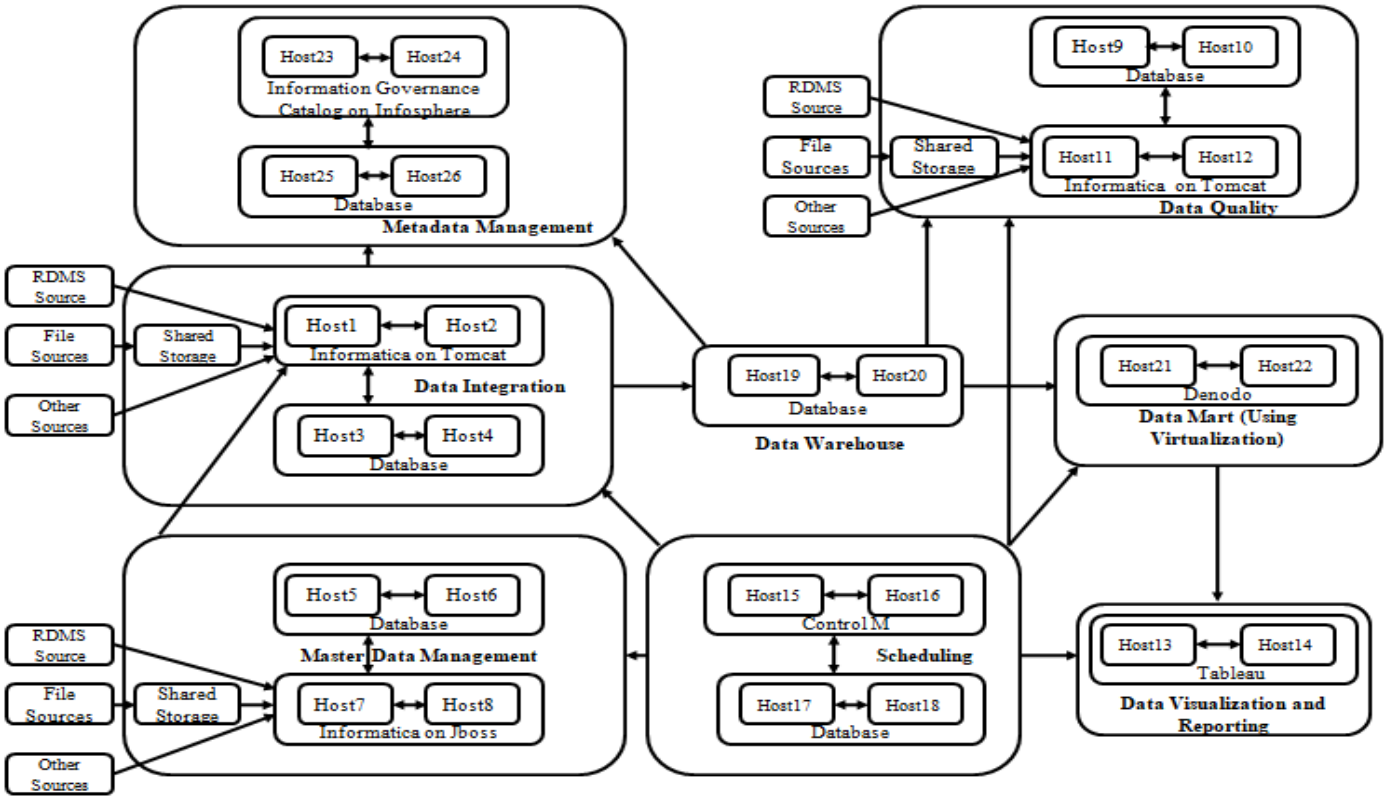
Key Words: Monitoring, Data warehousing (DW), Framework, Data Integration (DI), Architecture, Business Intelligence (BI), Applications, Enterprise data warehouse (EDW), Linux, and Service Level Agreement (SLA)

1. INTRODUCTION

Data warehouse is a repository that integrates data from different sources, stores it in a structured format, and allows business intelligence to analyse this information and make informed decisions for organization growth. Data warehouses predominantly utilize databases like Oracle, Teradata and Netezza to store the data. Most organizations using business intelligence have moved or moving towards enterprise data warehouses (EDW) with an organizational strategy of single version warehouse of truth (SVOT) instead of the traditional approach of building business functionality specific data warehouses. Organizations utilizing SVOT strategy build business functionality specific physical or virtual data marts over the EDW. Enterprise data warehousing streamlines every aspect of data warehousing and allows organizations to access their data in clear and consistent manner with no duplicity. Business Intelligence done with Enterprise data warehouses include the following core functionalities - Data Integration, Data Quality, Master Data Management, Meta data management, Data mart build (using Virtualization), Reporting and Data Visualisation, Scheduling. Data Integration involves the integration of various types of sources using the methodologies - extraction-transformation-load (ETL) and extraction-load-transformation (ELT) and loading the data into the data warehouses. Data Quality involves measuring the quality of data in source, data warehouse and at all data movement points during the data integration flow. Master data management involves creation of a single master reference source for all critical business data of an enterprise, which include customers, prospects, citizens, suppliers, sites, hierarchies and chart of accounts. Metadata management involves managing metadata of the data across the organization and providing an end-to-end data lineage from source systems to the reports used by business. Data mart build (using virtualization) involves using data virtualization techniques to build the data marts on top of the enterprise data warehouse. Reporting and Data Visualisation involves the reporting of the data mart data using visualising features like graphs, charts for higher management. Scheduling involves the automation of all the needed batch activities for all data warehousing functionalities. Organizations normally utilize a separate enterprise wide scheduler as it ensures all the application jobs are in one place for monitoring and dependencies between different EDW functionalities can easily be set up. For full filling these varied functionalities, organizations tend to onboard existing tools available in the market into their ecosystem rather than building native tools. Organizations prefer market available tools as they come with great features of operational support, existing solutions and documentation. Current market leaders in Data Integration, Data Quality, Metadata Management, Master data Management functionalities include Informatica, Talend, SAS, ERWIN etc, reporting functionalities include Tableau, Business Objects, OBIEE etc. and Scheduling functionalities include Control M, Autosys, APPWORX etc.

Most data warehousing applications on Linux with high availability requirements need at least need two servers for the application execution, two servers for the applications metadata database, an application server such as JBOSS, Infosphere, tomcat, a database such as Oracle and Shared Storage such as Network File systems, Veritas Cluster File System (or VxCFS), IBM Spectrum Scale for their processing. Applications in many cases also follow service oriented architecture design and create independent processes on the server for each of its functionalities, instead of a single process running the entire application. Below illustration provides environment architecture of a data

warehouse running on Linux and utilizing Enterprise data warehouse strategy and built using market leader tools Informatica, Control M, Denodo, IBM InfoSphere Information Governance Catalogue and Tableau for various functionalities. This environment has an overhead of monitoring 26 Linux servers, about 20 application processes (for 7 core applications), 6 databases, 4 application servers, 3 shared storage clustered file systems and 26 local file systems.



As observed in the above diagram data warehousing environmental architecture tend to become very complex and can easily become an operational nightmare without a robust monitoring framework.

Monitoring Framework requirements for data warehousing applications are quite unique as compared to other applications due to the various types of sub applications involved and should at least have the below features:

1) Monitor the Linux server hosting the application for availability
2) Monitor the web servers hosting the application for availability
3) Monitor the application processes and application websites for availability
4) Monitor servers for CPU, RAM, DISK, Network Performance metrics
5) Monitor the performance of the application jobs and application websites
6) Monitor the performance of the databases
7) Monitor the Storage disk usage on the servers
8) Automated alert generation capability using emails, phone messaging and connectivity to IT Service management (ITSM) tools like service now with automatic ticket generation
9) Planned outage management with automatic application recycle, no alert generation during outage period and automated validation of the changes done as part of the outage
10) Monitoring of application logs for errors
11) Provide the static information such as OS version, Application version etc. related to the server, application
12) Restart application automatically during momentary glitches due to known errors.
13) Visual dashboard with the monitoring details
14) Monitoring solution should be scalable and have high availability
15) Monitoring solution should have capabilities for historical reporting

In this paper author provides a comprehensive monitoring framework for data warehouses on Linux platforms, which provides all of these features.

2. PROBLEM STATEMENT

Data warehousing applications without robust monitoring frameworks cause business user dissatisfaction as the data loading SLA's get missed frequently due to delays in solving environmental issues. These data warehouses are bound to fail as the business users search for alternate options for data needs due to the unreliability with the data and eventually causing their sunset. A comprehensive monitoring framework for data warehouses is necessary to alert the support teams during such environment issues and also facilitate a quick resolution of the issue by providing means to identify the root cause.

3. MONITORING FRAMEWORK FOR DATA WAREHOUSING APPLICATIONS

Monitoring framework proposed in this paper follows the agent-push methodology. In the agent-push methodology, a local web server runs on the monitored server and it pushes the monitored data to the monitoring server. Proposed monitoring framework has five major components.

Monitoring Servers: Monitoring server is the server which hosts the monitoring application and performs most of the monitoring services. Services performed include below:

1) Execution of a web server which hosts all the monitoring services
2) Collect the supporting details such as planned application outage information from the Document management system and push this information to the corresponding monitoring servers
3) Co-ordination with the web server agent on the monitored servers and collect the monitored data
4) Load the monitored data and outage information into the monitoring database
5) Provide web responses for the Visual dashboard website
6) Push the latest monitoring modules to the monitored servers, if not up to date.
7) Execute the "monitoring modules" and collect the "monitored data" of the websites.
8) Create alerts in the ITSM systems for any unplanned monitoring server issues and send emails/text messages to the requested operations team. Skips alerting during planned outages.
9) Provide high availability for monitoring application server. Handshake with the backup monitoring server and switching of services to backup if the primary goes down.

Monitoring Databases: Monitoring database is the repository, which stores all the monitored data and the supporting data. This database follows the "monitoring data model". Services performed include below:

1) Provide data needed for the web server running on the monitoring server. This information is in turn displayed on the visual dashboard
2) Provide historical reporting capabilities

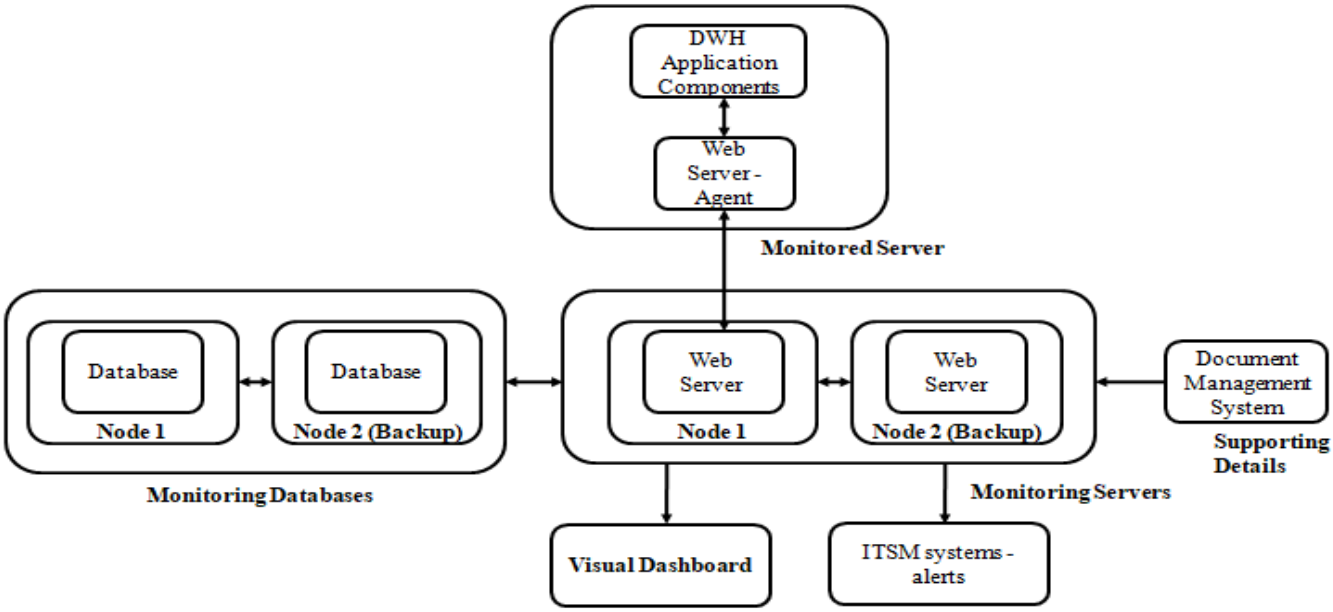
Monitored Servers: Monitored servers are the servers, which host the data warehousing applications and require monitoring. Services performed include below:

1) Execution of a web server – agent which hosts all the monitoring services
2) Execute the "monitoring modules" and collect the "monitored data" of the server and the data warehousing application
3) Provide response back to the monitoring web server and push the monitored data to it.
4) Execute the "maintenance modules" to start up the monitored applications during unplanned outages, if the issue falls in the list of known issues.
5) Execute the "maintenance modules" to shut down and restart the monitored application during planned outages of the applications metadata repositories.

Document Management Systems: Document management systems provide the supporting information needed for the monitoring servers. Supporting information include the information of the planned outages scheduled on the monitored servers, monitored applications, monitored applications databases and their servers. This information is critical to skip alerting during planned outages and automatic restart of applications.

Visual Dashboard: Visual dashboard is the front-end website for the monitoring applications. This receives the web response from the web server running on the monitoring servers. Visual dashboard provides the below information on the page for each monitoring server when requested - Server Information, Application Process status for the data warehouse application being monitored, Server Health Monitor, CPU / Memory / Disk space Usage Graph, top application user list, file system health

Below illustrations provides the details of the proposed monitoring framework with the components involved:



Monitoring and maintenance modules are the key for the monitoring framework and do the task of generating the monitored data for alert generation and also automatically restart the application during planned and unplanned outages.

Server health Module (1): This module executes on the monitored application server and collects the below statistics of the monitored application server. Primary task of this module is to check the server performance and facilitate the easy debugging of data warehousing application issues, when they arise.

Server CPU usage, CPU Usage at core level, Memory usage, TMP space usage, Disk Usage – "Read throughput, write throughput, Read IOPS, Write IOPS", File systems disk space usage, Network usage – "Transmit/ Receive usage, packets transferred and dropped", Active directory or LDAP response time, Source system response time, Target system response time and metadata databases response time, open file usage

Server health Module (2): This module executes on the monitoring server and collects the below statistics of the monitored application server. Primary task of this module is to check if the data warehousing application server is available or not.

Ping response time

Application Health Module (1) –Server: This module executes on the monitored application server and collects the below statistics of the monitored application. Primary task of this module is to check if the application is available and performing well. "Application Performance" jobs are the unique application jobs created for monitoring purposes and whose successful execution under defined threshold is an identification of good application performance.

Application process check, total application jobs executing, CPU, MEM Usage at application user level, "Application Performance" job execution statistics

Application Health Module (2)-Web: This module executes on the monitoring application server and collects the below statistics of the monitored application web pages. Primary task of this module is to check if the application web pages are available and performing well.

Web page availability, Web page response time

Server Information Module: This module executes on the monitored application server and collects the below statistics of the monitored application server. Primary task of this module is to check the monitored server static information and facilitate the easy debugging of data warehousing application server issues during application server planned outages for patching.

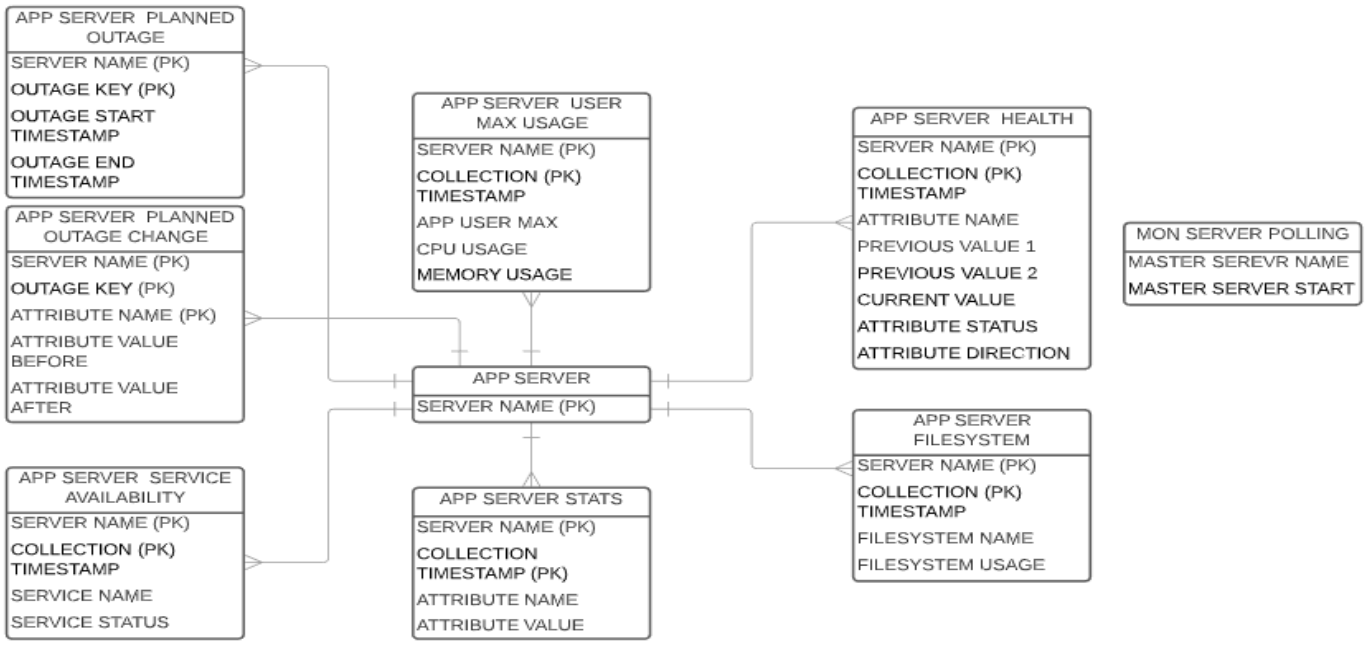
Hostname, Hardware model, Hardware location, CPU cores, CPU model, CPU speed, Total memory, OS version and latest patch, Server uptime, total open files available at application user level, stack size, File system types, Network speed

Maintenance Module: This module executes on the monitored application server and the responsibility of the module is to do below:

1) Shutdown and start-up the application process before and after maintenance window, if the metadata database of the application is being brought down for any planned outages.
2) Start up the application process during any unplanned outages due to temporary glitches of

application file systems going down, network delays between the application server and its metadata database server etc.

Monitoring data model provides the information on how the monitoring database has to be designed. Proposed design in this paper was chosen to allow faster access of information from the monitoring application and hence not suitable for reporting. Below is a sample data model with entity names and only critical data elements. This model can be expanded based on organization needs.



APP SERVER: This table contains the information of all the servers being monitored.

APP SERVER STATS: This table contains the information from the "server information module".

APP SERVER HEALTH: This table contains the information from the modules – "server health module (1)", "server health module (2)", "Application Health Module (1)–Server", "Application Health Module (2) –Web". This excludes the File systems disk space usage, CPU, MEM Usage at application user level and Application process check attributes from these modules.

APP SERVER USER MAX USAGE: This table contains the information of the attribute - CPU, MEM Usage at application user level from the "Application Health Module (1) –Server".

APP SERVER USER PROCESS AVAILABILITY: This table contains the information of the attribute - Application process check from the "Application Health Module (1) – Server"

APP SERVER PLANNED OUTAGE: This table contains the planned outage start and end time of the monitoring server and its metadata database server.

APP SERVER PLANNED OUTAGE CHANGE: This table contains the before and after attribute information of the changes done as part of the planned outage.

APP SERVER FILE SYSTEM: This table contains the information of the attribute - File systems disk space usage from the "server health module (1)".

MON SERVER POLLING: This table contains the information of the monitoring server which is the master and is used to bring high availability for the monitoring application.

A separate reporting data model following the star schema modelling approach can be built on top of this application data model to satisfy the reporting requirements of the monitoring application. Those tables can be loaded using a daily batch cycle to ensure there is no performance impact to the monitoring application.

Below illustration provides the detailed workflow of the monitoring framework. Web server running on the Monitoring server has individual services for each of the activities and those are automatically started when the web server comes up.

1) First service to come up is the "master server polling" service. This service does the tasks of a) validating if the monitoring database is available b) utilize the monitoring database polling table to decide the master server name, the server updating the polling table first becomes the primary c) Provide high availability, polling table is updated

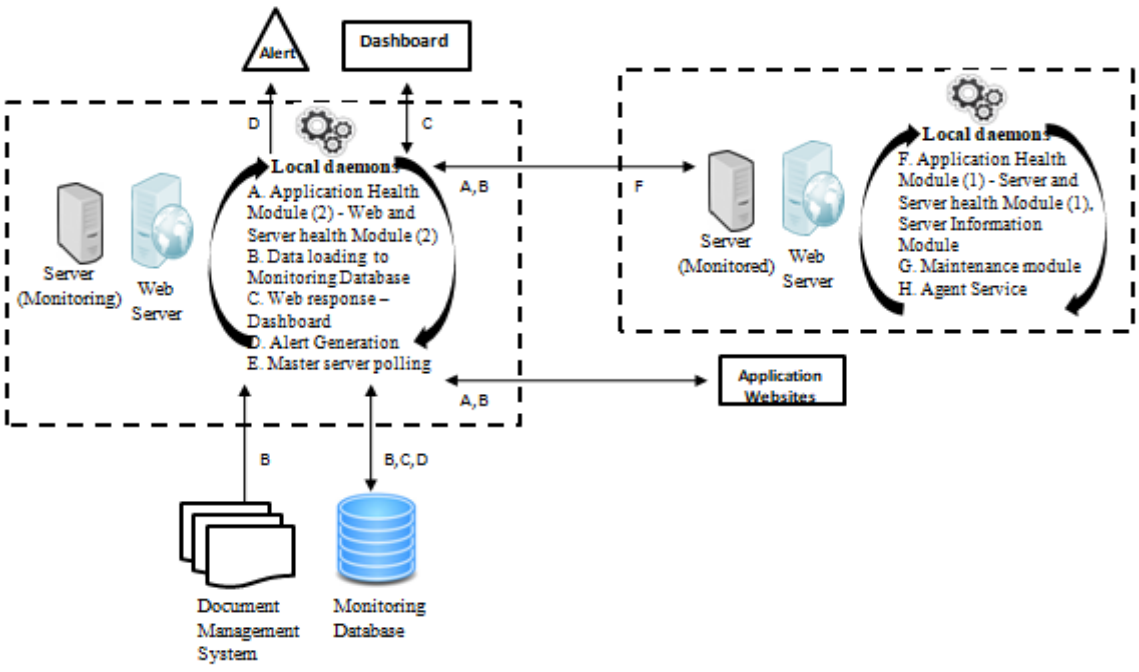
every few seconds and in case the master server name was not updated within threshold the backup node becomes primary and an alert is generated d) kick start all the other monitoring services e) Validate if the web server-agent on all the monitoring servers is available and has the latest code for web server-agent health and maintenance module, if unavailable bring the web service up on the monitoring server and if code is outdated push the latest code f) Push the latest outage schedule file to the monitoring server, if not up to date.

2) "Data loading to monitoring database" service does the tasks of a) loading the monitored data from the monitoring servers into the monitoring database b) loading the planned outage schedule of the monitoring servers and their metadata database servers from the document management system into the monitoring database c) Provide data needed for the web response service d) provide data needed for the alert generation service
3) "Health service" executes the Application Health module (2) –Web and Server health module (2) for all the monitoring servers. This service works with the "data loading to monitoring database" service to load the information.
4) Web response dashboard service displays the visual dashboard for the monitoring server being requested. This service works with "data loading to monitoring database" service to publish the information.
5) "Alert generation" service runs the rules on monitoring server data and creates the alerts in the ITSM system. This interacts with "data loading to monitoring database" service for getting the monitoring server data.

Web server running on the monitored server has individual services for each of the monitoring activities and those are automatically started when the web server comes up.

1) First service to come up is the "agent service". This service does the task of a) Providing response to the Server health module (2) b) Updating the web server "health and maintenance" module codes c) Updating the latest outage schedule file d) kick starts all the other monitoring services e) Push the monitored data file to the monitoring server
2) Maintenance service executes the maintenance module
3) Health service executes the Application Health Module (1) –Server, Server health Module (1), Server Information Module and produces the monitored data file.

Monitoring framework defined in this paper is adaptable to be coded in any technology defined by organizations. Choice of coding technology from the author is python due to availability of huge user community base and many built in class modules which can help in faster development.



4. CONCLUSION

Enterprise data warehouses requires myriad of data warehousing applications for its functioning and monitoring of each of these applications is the key to a data warehouse overall success. Failure in monitoring any of these applications causes delays in the data availability of the enterprise data warehouse. This paper provides a monitoring framework which can help in building of a tool to effectively monitor data warehouse applications on a Linux platform.

REFERENCES

[1] https://www.acronis.com/en-us/blog/posts/web-application-monitoring-basic-framework
[2] https://www.techopedia.com/definition/29993/server-monitoring
[3] https://en.wikipedia.org/wiki/Data_warehouse
[4] https://en.wikipedia.org/wiki/Metadata_management
[5] https://en.wikipedia.org/wiki/Master_data_management
[6] https://en.wikipedia.org/wiki/Data_virtualization
[7] https://en.wikipedia.org/wiki/Service-oriented_architecture

BIOGRAPHIES

I am Madhusudhan Reddy Sureddy with qualifications of Bachelors of engineering from Osmania University. I have 14 years of Information Technology experience and worked with fortune 500 companies - GE Capital Americas, American Red Cross, CVS Caremark, FannieMae and Santander Bank and in the business domains BFSI (Banking, Financial Services and Insurance), Mortgage and Healthcare. My research fields include data warehousing, data integration, data quality, data virtualization, master data management, reference data management, metadata management, big data solutions, business rule engines, data modelling, data analytics, Operational and Support Model and capacity planning.