International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

*Research Article*

# Real time monitoring of system counters using Map Reduce Framework for effective analysis

**Sandeep B. Aher†* and Poonam D. Lambhate‡**

†Computer Engineering, JSPM's JSCOE, Pune, India.
‡Information Technology, JSPM's JSCOE, Pune, India

Accepted 27 July 2015, Available online 28 July 2015, **Vol.5, No.4 (Aug 2015)**

***Abstract***

*Several monitoring software's provide flexibility to add various servers, performance system counters for real time monitoring. Application service managers use performance system counters to monitor servers and application health. It is constructive to detect anomalies earlier, take preventative actions and minimize the probability of failures of system to reduce uptime. The amount of data is huge for processing considering retention period and various global, process level system counters. The new approach, well known for offline processing, has been proposed for real-time data monitoring and analysis. The most notable solution that is proposed for managing and processing big data is the Map Reduce framework. Although Map-Reduce framework is very suitable for batch jobs but there is an increasing demand for non-batch processes on big data like: interactive jobs, real-time queries, and big data streams for faster and efficient monitoring.*

***Keywords:*** *Integrated monitoring, performance counters, distributed monitoring*

## 1. Introduction

With the advent of parallel processing on enormous data, processes are running with manifold instances and varied number of threads. Various processes like daemon process, batch processes, system process, backup processes etc. runs continuously on servers and utilize the resources. Some processes run for longer duration while some processes run for short duration. Monitoring helps in effectively manage system resources, making system decisions, evaluating and examining systems. If process level data is captured with data collection interval every second, the raw file size is in few GB's per hour on each server.

There is need of cost-effective monitoring solution. For monitoring application, both offline and online processing is required. Big data is being used extensively in offline processing but its use in online processing is very limited. The use of big data framework for real-time processing is explored in this paper.

In this paper, map reduce framework has been proposed for real time data processing. Instead of disk processing, memory processing is faster and it is proved with experiments that this approach is feasible to process huge offline volume as well as online volume.

*Corresponding author: **Sandeep B. Aher**

## 2. Literature Survey

### *2.1 Apache Spark*

Apache Spark is an open-source cluster computing framework originally developed in the AMP Lab at UC Berkeley (https://spark.apache.org/). In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in this scenario, Spark is running on a single machine with one executor per CPU core. Spark had over 465 contributors in 2014, making it the most active project in the Apache Software Foundation and among Big Data open source projects.

### *2.2 Comparative Study with relevant Journals*

Various international journal papers were studied on monitoring system and implementation underlying technology. In (Masahiro Yoshizawa, *et al*, 2014), Most of the monitoring tasks are being performed, not by any single monitoring software, but by suitable combination of multiple monitoring software's. A

combination of two types of monitoring has been proposed: Global level monitoring and Log level monitoring.

In (M. Yoshizawa, *et al*, 2011), Integrated monitoring software is effective for reducing the cost of monitoring SaaS-based systems. Data models for real-time data and system configurations for easier associations between the real-time data are also

In (Rakesh Bhatnagar, *et al*, 2013), the author discussed the various performance parameters and criteria of the grid monitoring system, Ganglia. But it's not integrated monitoring system and Ganglia doesn't have a built in notification or alert system.

In (Brian Laub, *et al*, 2014), Emerging research systems are focusing on scalable, distributed monitoring capable of quickly detecting and alerting administrators to anomalies. Nagios and Tivoli monitoring have centralized architectures and large data flow in network.

In (Yassine Tabaa,, *et al*, 2012), author discussed about faster and efficient computing. Hadoop Map Reduce suffers significant troubles when dealing with iterative algorithms; as a consequence, many alternative frameworks that support this class of algorithms were created. The most important enhancement of Spark is that it allows some data to be shared between the mappers while they are computing, a fact that significantly reduces required data over the network.

*2.3 Comparative Study with relevant Journals*

To monitor real time process level data, it has to be collected very frequently. Looking at number of running processes in environment and data collection interval, raw data will be very huge. Map reduce programming framework can be used for parallel processing of huge amount of data. Processing will work on different servers in distributed manner. The framework is scalable, fault tolerant and easily extendable. Map transforms a set of data into key value pairs and Reduce aggregates this data into a scalar. A reducer receives all the data for individual key from all the mappers. The Map-reduce framework operates exclusively on <key; value> pairs.

To reduce the challenges in map reduce for real-time processing, In-memory computing solutions will be used. Some of them are: Apache Spark, GridGain and XAP. (https://www.safaribooksonline.com/)

## 3. Problem Statement

There are restrictions to use Map-reduce (Hadoop) for real time process level monitoring due to following:

1. Map reduce is basically process offline files stored in file system. The proposed solution is for real time data analysis.
2. All of the input must be ready before job starts and this prevents Map Reduce from real time processing use cases.
3. Map-reduce cannot run continuous computations and queries. The processing is disk based and hence slower for online or real time data processing.

Since Map Reduce framework (Hadoop) is not proper for these non-batch workloads, new solutions are proposed to these new challenges.

## 4. Proposed System

*4.1 Objective*

The main objectives is to offer efficient and conclusive monitoring to different users in IT system including Monitoring team, Performance Test Team and Capacity planning team.

Integrated monitoring system will have following objectives:

- Faster and efficient processing of data
- To identify the problematic process or application earlier
- To report the problem immediately to appropriate team
- To take temporary precautions such that minimal impact in current system
- To save cost, effort and time in fixing the issue
- To gain more confidence on the system and monitoring capabilities
- Improve high availability of applications
- Lower number of anomalies

Proposed system is experimental and can be replacement of existing monitoring systems. Existing systems have mostly standalone global level statistic and use SNMP protocol for data processing (Anshul Kaushik, 2010). SNMP has support to standard performance counters and has security issues in previous versions.

The proposed monitoring application with Map Reduce framework has been developed with Real time processing - Process level statistics.

*4.2 System Architecture*

Spark applications run as independent sets of processes on a cluster, coordinated by the Spark Context object in main program as shown in Figure 1.

Specifically, to run on a cluster, the Spark-Context can connect to several types of cluster managers, which allocate resources across applications. Once connected, Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for the application. Next, it sends application code (defined by JAR or Python files passed to Spark-Context) to the executors. Finally, Spar-Context sends tasks for the executors to run (Matei. Zaharia). Current architecture diagram, performance is better up to 50 servers. If servers are more, then loosely coupled architecture option will have to be considered.

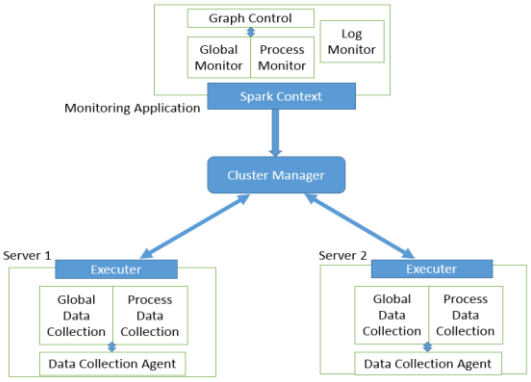


**Fig.1** Architecture Diagram – Experimental system

*4.3 Mathematical Model*

Let set S' is defined as: S'= {I, O, P, F, S}
I = {Sn, Pc, t, It}
Where:
Sn – Server name to be monitored.
Pc – Performance counter to be measured.
t - Time period for which output should be displayed.
It - Interval time, when output updates regularly.
P = {P1, P2, P3, P4, P5}
P = {Data collection, Data processing, Clustering, Data Conversion, Display)

1. Collect data using data collection agent on each server.
2. Process data as per input criteria received from application service manager
3. Cluster data by process ID of high resource consumption.
4. Convert received data to array format
5. Display output in graphical view.

F = Fail to display real time streaming graph.
S = Implementation of real time map reduce framework to display streaming process level performance counter data.

Figure 2 shows state transition diagram representing major states of the proposed system and transitions between them

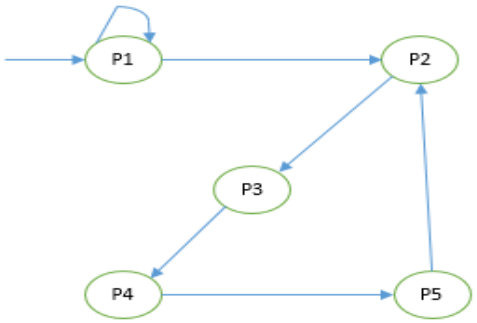


**Fig.2** State transition diagram

## 5. Implementation Details

The implemented system has mainly divided into two modules. First is data collection agent on servers to be monitored (offline processing) and second is data processing and display module.

The data processing and display module is common to both offline and real-time processing. Data processing flow and display algorithm has been explained below in detail:

---

**ALGORITHM**: Data Processing of Global and Process level performance monitors

---

1. Start
2. INPUT parameters:
   Server name, Performance counter, TIME
3. IF [TIME is CURRENT]
   THEN CONTINUE TO step 4,
   ELSE
   GOTO step 5
4. [Ensure Data collection agent is running]
   IF [Data collection Agent = RUNNING]
   THEN GOTO step 6
   ELSE
   MESSAGE Start data collection Agent
   GOTO step 11
5. [Ensure historical data available]
   IF [Data is present on monitoring server]
   THEN CONTINUE to step 6,
   ELSE
   MESSAGE NO DATA PRESENT
   GOTO Step 11
6. Read data from Server
   IF [TIME is CURRENT]
   THEN read streaming data
   ELSE
   Read off-line data from files
7. Invoke Map Reduce algorithm
   IF [PROCESS level data]
   Key ← process ID
   ELSE IF [GLOBAL level data]
   Key ← system time
8. IF [PROCESS level data]
   Clustering algorithm to cluster data by process ID.
   ELSE [GLOBAL level data]
   Filtering algorithm to filter data by selected time.
9. Convert selected data to JSON format
10. Display Graph
11. END

## 6. Experimental Setup and Results

*6.1 Experimental Setup – Real time processing*

The experimental setup has performed on Windows 7 operating system on desktop machines. A console application was developed using PDH libraries in visual studio to collect windows system level data. The frequency of data collection at regular interval of one

second and continues until real-time process level data requested by another machine or central server.

The monitoring server has been developed in Java and for graph display Jfreechart has been used. Two different monitoring servers were developed for comparisons. First is with usual client-server communication with Scanner class to break input into tokens. The input then passed to graph until data collection agent is running.

Another monitoring server has developed with Apache spark (https://spark.apache.org). JavaReceiver- InputDStream interface has been used for defining input stream to connect and receive data over network. The process (key) and its dynamic value to monitor selected while making connection to data collection agent. 'flatMap' transformations used instead Scanner to get sequence of tokens, which contain both key and runtime value. The tokens are collected in array and then displayed in the graph. Reduce job is to take input from above and then filter the dynamic value required to display in the graph.

*6.2 Result Analysis*

The programs were run to display real-time monitoring data using apache spark framework. The system counter Free Memory in MB was analyzed real-time graphically using Figure 3.

Fig. 3. Process level free memory (MB) monitoring system counter

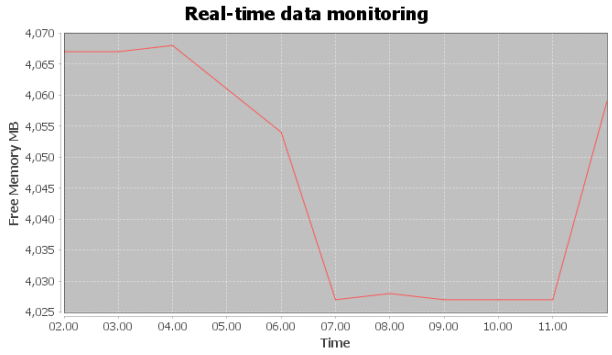


**Fig.3** Process level free memory (MB) monitoring system counter

The key contributions of the MapReduce framework are not the actual map and reduce functions, but the scalability and fault-tolerance achieved for a variety of applications by optimizing the execution engine once. As such, a single-threaded implementation of MapReduce (such as MongoDB) will usually not be faster than a traditional (non-MapReduce) implementation, any gains are usually only seen with multi-threaded implementations

## Conclusion

The implemented system is experimental and can be replacement of existing monitoring systems. Big data (Hadoop) is widely used for offline processing but its use in real-time processing is very limited.

1) Map-reduce framework can be implemented in real time processing with Apache spark.
2) It will work on commodity hardware resulting in cost-effective application system.

## Future Work

The monitoring system with map reduce framework can be further enhanced with multiple graphs and enriched features.

It can be further commercialized with low cost offering for global and process level data monitoring. For real time implementation, the optimized tuning of the application has to be done with optimized number of threads.

## References

Sandeep Aher, Poonam Lambhate, (2015) Integrated Monitoring System *International Journal on Recent and Innovation Trends in Computing and Communication* ISSN: 2321-8169 Volume: 2 Issue: 12

Sandeep Aher, Poonam Lambhate, (2015) Data Mining and Information Retrieval track, *cPGCON 2015-Paper 687* https://easychair.org/conferences/submission.cgi?submission=2181370;a=8193048

Masahiro Yoshizawa, Tatsuya Sato, Ken Naono (September 2014), Integrated Monitoring Software for Application Service Managers *IEEE transactions on network and service management*, vol. 11, No. 3

M. Yoshizawa and K. Naono, (2011) Design and evaluation of integrated monitoring software for SaaS-based systems, in Proc. APNOMS, pp. 14.

Rakesh Bhatnagar, Dr. Jayesh Patel, (2013) Performance Analysis of A Grid Monitoring System - Ganglia *International Journal of Emerging Technology and Advanced Engineering* Volume 3,Issue 8

Brian Laub, Chengwei Wang, Karsten Schwan, Chad Huneycutt, (2014) Towards Combining Online and Offline Management for Big Data Applications, Autonomic Computing, 2014 *11th International Conference on Autonomic Computing (ICAC '14)*

Yassine Tabaa, Abdellatif Medou, (2012) Towards a next generation of scientific computing in the Cloud *IJCSI International Journal of Computer Science Issues*, Vol. 9, issue 6, no 3

Anshul Kaushik, (2010) Use of open source technologies for enterprise server monitoring using SNMP *International Journal on Computer Science and Engineering (IJCSE)* Vol. 02, No.07, 2246-2252

Performance management across the application lifecycle http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA5-0929ENW.pdf

http://en.wikipedia.org/wiki/System Monitor

A. van Hoorn *et al.*, (2009) Continuous monitoring of software services: Design and application of the Kieker framework, Dept. Comput. Sci., Univ. Kiel, Kiel, Germany, TR0921.

B. Krishnamurthy, A. Neogi, B. Sengupta, and R. Singh, (2008) Data tagging architecture for system monitoring in dynamic environments, in Proc. NOMS, pp. 395402.

J.Wei, Y. Zhao, K. Jiang, R. Xie, and Y. Jin,(2011) Analysis farm: A cloudbased scalable aggregation and query platform for network log analysis, in Proc. Int. Conf. CSC, pp. 354359.

Dr. Evangelos Bekiaris, Christos Petsos, Kostas Kalogirou, (2012) Energy Management Web Application: Brokerage Agent Front End Application, *36 International Conference on Power and Energy Systems Lecture Notes in Information Technology*, Vol.13

Manoj V ,(2014) Comparative study of nosql document, column store databases and evaluation of CASSANDRA,*International Journal of Database Management Systems* ( IJDMS ) Vol.6,No.4

Jaspreet kaur , Harpreet Kaur , Kamaljeet kaur, (2013) A review on document oriented and column oriented, *International Journal of Computer Trends and Technology* - volume4 Issue3

MapReduce Programming Model for .NETbased Cloud Computing, (2009) http://cloudbus.org/papers/MapReduce-Aneka2009.pdf

Architecture Governance Logical Architectural Diagramming Guidelines, http:// pro.iankoenig.com/ docs/LogicalArchitecturalDrawingGuidelinesv2.pdf

Saeed Shahrivari, Saeed Jalili, Beyond Batch Processing: Towards Real - Time and Streaming Big Data, Computer Engineering Department, Tarbiat Modares University (TMU)

https://www.safaribooksonline.com/library/view/learningspark/9781449359034/ch01.html,Chapter 1. Introduction to Data Analysis with Spark

https://datapsyche.wordpress.com/2014/06/05/events-over-time-chartusing-flot-charts/, Events over time chart using Flot Charts

Jitendra Kumar, Richard T. Mills, Forrest M. Homan, William W. Hargrove, (2011) Parallel k-Means Clustering for Quantitative Ecoregion Delineation using Large Data Sets, *International Conference on Computational Science*

Weizhong Zhao, Huifang Ma, Qing He, Parallel K-Means Clustering Based on MapReduce, *Chinese Academy of Sciences Graduate University of Chinese Academy of Sciences*

https://spark.apache.org/ Apache Spark, fast and general engine for large-scale data processing