Volume 3, Issue 12, December 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Image Based Approch to Compressive Distributed System Monitoring

Mr. Santoshkumar Biradar
Assistant Prof. Dr D.Y Patil COE,
Computer Department, Pune University, India

Ms.Akshada Bhondave
Dr D.Y Patil COE ,
Computer Department, Pune University, India

Abstract— Federated computing infrastructures have become the important for many real-world systems like cloud computing infrastructures, enterprise data centers, web service composition ,workflow management, and huge data processing systems .distributed quality sensitive application need up-to-date dynamic information about all nodes in the hosting infrastructures. But, it is a difficult task to provide, scalable, full-coverage, fine-grained and failure-resilient monitoring for large scale hosting infrastructure .This paper present Resilient self-Compressive Monitoring (RCM) system for large scale hosting infrastructures. RCM provide scalable distributed monitoring by performing online data compression to reduce remote data collection cost. RCM provides failure resilient monitoring to achieve robust monitoring for dynamic distributed systems where host and network failures are common. RCM can achieve higher compression ratio with a very less overhead than the existing approaches.

Keywords— Online data compression, distributed system monitoring.

I INTRODUCTION

Large-Scale distributed hosting infrastructures have become the basic platforms for several real-world production systems like enterprise data centre, cloud systems and massive data processing systems, distributed stream processing, and workflow management. As these computing infrastructures still grow, a way to efficiently manage such large scale dynamic distributed systems to better support application needs has become a challenging problem A production hosting infrastructure generally consists of 1) a large number of distributed worker nodes that execute different application tasks and 2) a group of management nodes that offer varied configuration and optimization services. A distributed monitoring system generally deploys monitoring agents on distributed worker nodes and configures those agents to continuously collect various metrics and periodically report sampled metric values to management nodes.

Obtain complete and fine-grained information regarding all hosts and network connections within the hosting infrastructure is necessary to achieve efficiency and accuracy. However, it is a challenging task to deploy fine-grained monitoring for large-scale hosting infrastructures due to scalability overhead. A production hosting infrastructure usually includes lot of hosts and virtual machines (VMs), each of which can be associated with multiple dynamic metrics. Management node is almost overloaded when the number of worker nodes becomes large. Thus, while not reducing the monitoring traffic to the management node, it's impractical to use fine-grained monitoring to giant scale hosting infrastructure.

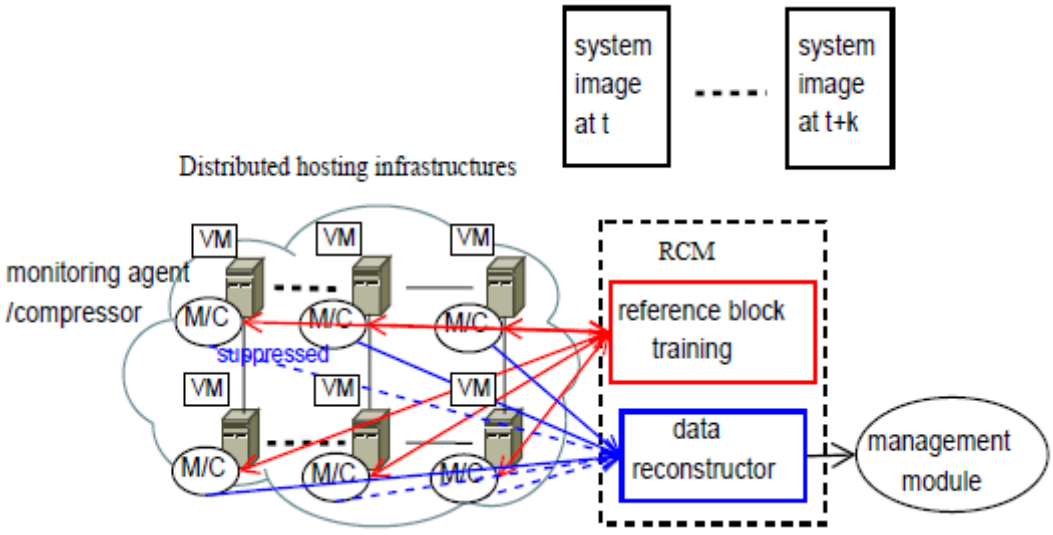

Fig 1. Distributed Monitoring System

The main goal of this paper is to achieve failure-resilient, scalable, full-coverage, fine-grained monitoring for large scale hosting infrastructure. RCM explores new image based approach and simplest failure-resilient online information compression framework to enable scalable ,full-coverage, fine-grained monitoring for large scale hosting infrastructure .RCM cab achieve higher compression ratio with lower overhead than previous compression approach.

II LITERATURE SURVEY

System Monitoring Tool: Large-scale distributed system management needs continuous monitoring service because it is initial step to know system runtime behaviors. There are plenty of off-the-shelf monitoring software available, such as IBM's Tivoli [1], HP's OpenView [4] and Amazon's Cloudwatch [3]. These systems perform centralized data monitoring by aggregating information from a variety of sources and presenting it to system operators through some graphical user interfaces. There also has been a lot of research work in this area, CoMon [2] is a monitoring infrastructure that collects system performance data for PlanetLab nodes and reports them through a web interface. Astrolabe [3] is a peer-to-peer monitoring system for information dissemination and aggregation in large-scale distributed systems. SDIMS [4] is a scalable distributed information management system which uses hierarchical monitoring structures. Ganglia[5] is another well-known open source monitoring tool for data collection that is also based on a hierarchical design. Compared to those works, SRDM is more flexible to be deployed since it does not have any specific requirement of the monitoring system architectures. Monalytics [6] is an integrated monitoring and data analysis system with flexible monitoring architecture to manage large-scale data centers and utility clouds.

Monitoring Scalability: A major challenge of building distributed monitoring systems is the scalability concern. Previous research work has proposed various solutions to address this problem. Some approaches employ specific monitoring topology to offload the monitoring cost. Astrolabe [3] used a peer-to-peer architecture and a gossip protocol to achieve scalable monitoring for distributed systems. SDIMS [4] aggregated information in large-scale networked systems by leveraging distributed hash table (DHT)'s internal trees for hierarchical aggregation. Other approaches trade off information coverage or precision for lower monitoring cost.

InfoEye [7] dynamically configures the operations of different monitoring sensors based on recent application query patterns and attribute distributions to achieve minimum monitoring overhead at the cost of losing some information coverage since only a subset of monitored attributes that satisfy current statistical conditions will be pushed to the management node. STAR [8] proposed a self-tuning algorithm that adaptively sets precision constraints to bound the numerical error in query results.

Correlation Based Approaches: Compressible data typically have some redundancies that can be exploited to represent the original data in a more compact way. Correlation among distributed data sources is an indicator of data redundancy. Correlation-based approaches have been studied under different contexts such as sensor network monitoring, distributed event tracking and resource discovery. Several previous work [8] has proposed to leverage correlation patterns to reduce monitoring cost. InfoTrack [9] explores spatial and temporal correlations to compress live monitoring streams in a large scale distributed system. It integrates both metric value predictors and adaptive clustering algorithms to suppress remote information update .OLIC [11], proposed a new image-based online compression approach to reducing distributed monitoring cost.

Data Compression: Compression techniques have been extensively used in video coding applications. Compression approach is inspired by the video compression technique in which large video data is encoded at the source and compressed video data is transmitted for lower communication cost, and then decodes the compressed data at the receiver to reconstruct the original data. However, challenging task is, source data are distributed on different hosts that can experience transient or persistent failures from time to time. Offline data compression, LZW and Deflate algorithm are widely used by data compression tools such as gzip and UNIX compress. VPC3 [12] is an offline trace compression algorithm for large log files. Different from those offline compression schemes that can only be applied after the data have been reported to the management node, RCM performs online compression over live monitoring data streams during monitoring runtime. Thus, RCM can reduce end-system resource and network bandwidth consumption on both worker nodes and management nodes.

In comparison with the existing system monitoring tools, the goal of RCM is to provide a compressive monitoring scheme that can be applied to any distributed hosting infrastructure so that fine-grained and full coverage monitoring with smaller transmission and storage overhead can achieved. RCM solve the scalability problem by applying online data compression techniques on the raw monitoring data to reduce the monitoring cost. Compared with those correlation-based compression approaches, RCM take an image-based approach that models snapshots of the monitored distributed system using a sequence of system images and applies lightweight online reference block search algorithms to compress distributed monitoring data. By applying a boarder search range than the existing temporal and spatial correlation based approach, RCM achieve higher compression performance. RCM is failure resilient, which can tolerate network and host failures that are common in real-world hosting infrastructures.

III THE SYSTEM MODEL

Large-scale distributed hosting infrastructure that consists of N worker nodes, denoted by {v1, . . .vN}. Monitoring agents on all worker nodes report their local attribute values to the management node using a certain sampling rate. Distributed system attributes are classified into two categories 1) intra-node attributes that contain information concerning every node (e.g. CPU load, memory usage, disk I/O statistics) 2) inter-node attributes that denote measurement between different nodes (e.g. network delay and bandwidth). On each worker node, the monitoring agent

periodically samples each intra-node attribute to form a time series$\{a_i^1, {}_k, \ldots, a_i^t, {}_k \ldots\ldots a_{i,k}^{t+m}\}$, where a_i^t, k denotes the sampled value for the intranode attribute a_k collected on node v_i at time *t*. Similarly, the monitoring agent periodically samples each inter-node attribute to form a time series $d_{i,j,k}^1 \ldots\ldots, d_{i,j,k}^t \ldots\ldots, d_{i,j,k}^{t+m}$ where $d_{i,j,k}^t$ denotes the inter-node attribute d_k between node *i* and j at time *t*.
monitoring cost reduction can be obtain when compression ratio is large.

For achieving fine-grained monitoring with low cost, suppress the update of the attribute value from a worker node to the management node at time *t* if the management node can predict the worker node's attribute value using a known reference value. Let a_i denote one real attribute value and a_i' denote its reference value. System allows the user to specify an error bound $e_i(e_i \geq 0)$. If $\frac{|ai - a_i'|}{ai} \leq ei$, RCM can suppress the update of this attribute if its reference value within the error bound *ei*.

Compression ratio (CR) as follows:

$$CR = \frac{Us}{Uorig} \quad (0 \leq CR \leq 1), \tag{1}$$

where U_s is the number of attribute values whose updates are suppressed by system in bytes per second and U_{orig} is the number of original attribute updates without using any compression in bytes per second. More monitoring cost reduction can be obtained when compression ratio is large.

A.Online Compression

RCM needs to find the optimal reference value for each attribute to maximize the compression ratio .It model snapshots of dynamic monitoring attributes collected on distributed worker nodes in the hosting infrastructure as a sequence of system images as shown in Fig 2 and Fig 3.

$$\begin{bmatrix} 0 & \cdots & d_{1,j}^t & \cdots & d_{1,N}^t \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{i,1}^t & \cdots & 0 & \cdots & d_{i,N}^t \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{N,1}^t & \cdots & d_{N,j}^t & \cdots & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & \cdots & d_{1,j}^{t+1} & \cdots & d_{1,N}^{t+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{i,1}^{t+1} & \cdots & 0 & \cdots & d_{i,N}^{t+1} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ d_{N,1}^{t+1} & \cdots & d_{N,j}^{t+1} & \cdots & 0 \end{bmatrix}$$

Image at time t Image at time $t+1$

$(d_{i,j}^t$: inter-node attribute between node i to node $j)$

Fig 2. System image sequence for an inter-node attribute of N nodes.

$$\begin{bmatrix} a_{i,1}^t & \cdots & a_{i,k}^t \\ a_{i,k+1}^t & \cdots & a_{i,2k}^t \\ \vdots & \vdots & \vdots \\ a_{i,N-k}^t & \cdots & a_{i,N}^t \end{bmatrix} \quad \begin{bmatrix} a_{i,1}^{t+1} & \cdots & a_{i,k}^{t+1} \\ a_{i,k+1}^{t+1} & \cdots & a_{i,2k}^{t+1} \\ \vdots & \vdots & \vdots \\ a_{i,N-k}^{t+1} & \cdots & a_{i,N}^{t+1} \end{bmatrix}$$

Image at time t Image at time $t+1$

$(a_{i,k}^t =$ attribute a_i of the node $k)$

Fig 3.System Image Sequence For An Intra-Node Attribute Of N Nodes.

B. System Architecture

RCM performs online compression over distributed monitoring data, which is illustrated by Fig.4. The runtime operation of RCM alternates between two phases: the training phase and the compression phase. All worker nodes send complete monitoring data to the management node during the training phase. System builds a sequence of system images using the uncompressed monitoring data. For each system image block, monitoring system finds the optimal reference based on the compression ratio. The management node stores the optimal reference block information about image index and block position locally and also sends corresponding reference block information to different worker nodes.

When the training phase is over, monitoring system starts to perform compression using the reference block information obtains during the training phase. Current attribute value with its reference value are compared by each

worker node, the worker node suppresses the attribute update for reducing the monitoring data traffic if the difference is within the predefined error bound. If the management node does not receive the updated value of attribute from a worker node, attribute's reference value is used by management node to restore the suppressed attribute update. RCM on the management node reconstructs all the monitoring data and delivers the complete monitoring information to different system management modules.

When the worker node sends attribute updates to the management node, the message is represented in form of a tuple {time stamp, a set of attribute values}. Thus, on the management node side, RCM can find out which attributes are omitted due to compression and infer which sampled values of those attributes are missing by checking the gap between different time stamps**.**

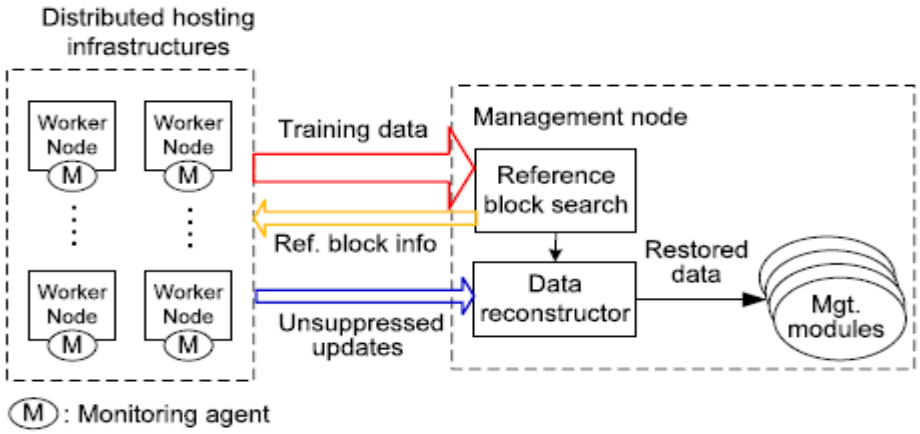

Fig. 4. RCM distributed monitoring architecture.

C. Online Reference Block Search

RCM employs a training phase to search the optimal reference block. The training phase consists of R training windows, in each training window, RCM on the management node examines *I* consecutive reference images. Each block is represented in the form of a coordinate (image index, block position). Image index denotes which reference image in the training window and the block position denotes the block sequence number inside one system image. The training phase perform coordinate mapping of each training image block to the reference block. For each block in the training image, monitoring system find a number of candidate blocks in all the reference images to find the best reference block that can give the highest compression ratio.

Monitoring system use a fast reference block search algorithm inspired by a video coding technique. The basic idea is to gradually increase the search range and terminate the search process immediately when little compression performance improvement occurs. This search Patten achieves good tradeoff between the search coverage and the search overhead. This reference block search algorithm uses a dual-diamond search pattern, illustrated by Fig. 5. It consist of 2 patterns

1. Large diamond search pattern (LDSP): consist of eight blocks surrounding the center block to form a diamond shape. In Fig. 4, blocks (i,12), (i,16), (i,4), (i,10), (i,20), (i,24), (i,26), and (i,32) form one LDSP and block (i,18) as the center. Blocks (i,2), (i,8), (i,18), (i,22), (i,24), (i,10), (i,14), and (i,30) form another LDSP with block (i,16) as the center.

2. Small diamond search pattern (SDSP): consists of four blocks surrounding the center block. In Fig. 4, blocks (i,9), (i,15), (i,17), and (i,23) compose one SDSP with block (i,16) as the center. LDSP is repeatedly used until the best reference block occurs at the center block. Then search pattern switch from LDSP to SDSP. Block that giving the highest compression ratio is selected as the final reference block among the four blocks in the SDSP and the center block.

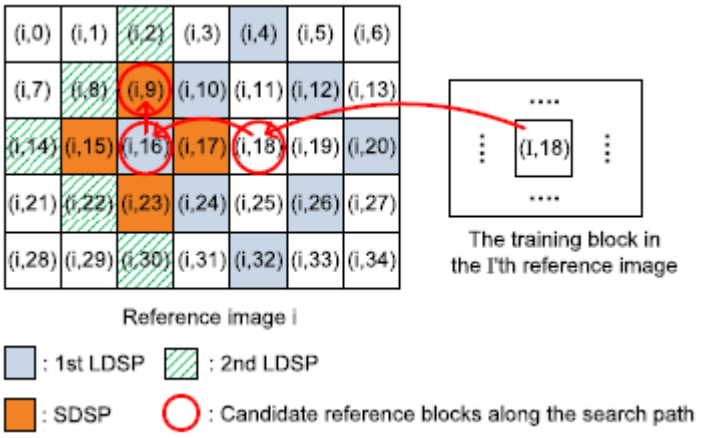

Fig. 5. RCM's Reference Block Search Algorithm.

I. Algorithm

Step 1: The initial LDSP is centered at the colocated block within the reference image i (i.e., block (i,18) in Fig. 5). The center block and the eight neighboring blocks are tested individually by checking the how much compression that can be achieved by using each of them as the reference block. If the block that gives the highest compression ratio is a center block then go to Step 3 Otherwise go to Step 2. In Fig. 5, the best block of the initial LDSP is block (i,16). Thus, search proceeds to Step 2.

Step 2: The neighboring block that give the highest compression ratio in the previous search step (i.e., block (i,16) in Fig. 5) is now become as the centre block of a new LDSP. If the eight neighboring blocks in this new LDSP do not giving higher compression ratio than the center block, go to Step 3. Terminate the broad search when little compression performance improvement obtained. Otherwise, repeat Step 2 to search more blocks. In Fig. 5, the best block of the second LDSP is center block (i,16) itself. Thus, search proceeds to Step 3.

Step 3: Search pattern is switch from LDSP to SDSP. The reference block found in this step is the final reference block. In Fig. 5, the final reference block is block (i,9).

For each training window, the same search process is conducted independently on all the I consecutive system images. Best reference blocks of all reference images are compared and the one that has the highest compression ratio is selected as the final best reference block. If multiple blocks have the same compression ratio then choose the best reference block in the image that is closest to the training image in time.

D .Failure Resilience

The failed hosts that serve as reference blocks can have an effect on the compression performance since RCM cannot use those reference blocks any longer. However, it's a nontrivial task is to make difference between a temporary unavailable node from a healthy node that suppresses attribute updated value for reducing the monitoring cost. For this reason, the management node cannot directly relay on the monitoring data to find out the host failures. Solution to this problem is to require each host to send heartbeat messages periodically, which introduce extra monitoring overhead. Instead, RCM use the internode attribute monitoring to detect host failures. Each monitoring agent needs to ping all the other hosts with a predefined sampling interval. If the monitoring agent on host X does not get the response from another host Y at time t, it will fill in a NULL value in its internode attribute report to indicate the host failure Y at time t. The management node can detect the host failure Y when it aggregates the internode attribute reports from all the monitoring agents.

IV CONCLUSION

RCM is a novel image-based resilient self-compressive monitoring system for large-scale hosting infrastructures. RCM models snapshots of the dynamic monitoring attributes collected on distributed worker nodes in the hosting infrastructure as a sequence of system images and apply lightweight online reference block search algorithms to compress distributed monitoring data. RCM is failure resilient, which might tolerate host and network failures that are common in real-world hosting infrastructures. This is the first attempt to adopt an image-based approach to achieving efficient and robust distributed monitoring traffic reduction. RCM is efficient, which can gain higher compression with lower overhead than previous compression approaches

REFERENCES

[1] "CoMon," http://comon.cs.princeton.edu/, 2012.
[2] "IBM Tivoli Monitoring Software," http://www-01.ibm.com/ software/tivoli/, 2012.
[3] Amazon CloudWatch. http://aws.amazon.com/cloudwatch/.
[4] HP OpenView. http://www.openview.hp.com.
[3] R. Van Renesse, K.P. Birman, and W. Vogels, "Astrolabe: A Robust and Scalable Technology for Distributed System
[4] P. Yalagandula and M. Dahlin, "A Scalable Distributed Information Management System," Proc. ACM SIGCOMM, Aug. 2004.
[5] Matthew L. Massie, Brent N. Chun, and David E. Culler. The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817 – 840, 2004.
[6] Mahendra Kutare, Greg Eisenhauer, Chengwei Wang, Karsten Schwan, Vanish Talwar, and Matthew Wolf. Monalytics: online monitoring and analytics for managing large scale data centers. In *Proc. of ICAC*, 2010.
[7] J. Liang, X. Gu, and K. Nahrstedt, "Self-Configuring Information Management for Large-Scale Service Overlays," Proc. IEEE INFOCOM, 2007.
[8] N. Jain, D. Kit, P. Mahajan, P. Yalagandula, M. Dahlin, and Y. Zhang, "STAR: Self-Tuning Aggregation for Scalable Monitoring," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2007
[9] Y. Zhao, Y. Tan, Z. Gong, X. Gu, and M. Wamboldt, "Self- Correlating Predictive Information Tracking for Large-Scale Production Systems," Proc. Int'l Conf. Autonomic Computing
[10] S. Zhu and K.K. Ma, "A New Diamond Search [8]Algorithm for Fast Block-Matching Motion Estimation," IEEE Trans. Image Processing, vol. 9, no. 2, pp. 287-290, Feb. 2000.
[11] Y. Tan, X. Gu, and V. Venkatesh, "OLIC: Online Information Compression for Scalable Hosting Infrastructure Monitoring," Proc. 19th Int'l Workshop Quality of Service (IWQoS), 2011.
[12] M. Burtscher. VPC3: A Fast and Effective Trace-Compression Algorithm. *SIGMETRICS Perform. Eval. Rev.*, 32(1):167–176, 2004.

