International Journal of Computer Applications (0975 – 8887)
Volume 96 – No.23, June 2014

Resource Aware Monitoring in Distributed System using Tabu Search Algorithm

Sonali L. Vidhate
P.G.Student at MET's BKC IOE
Nasik, Maharashtra, Pune University, India

M.U.Kharat, PhD
Professor at MET's BKC IOE
Nasik, Maharashtra, India

ABSTRACT

Tabu search algorithm like simulated annealing or evolutionary algorithm or genetic algorithm and guided local search algorithm is a effective solution of optimization problem. This is the most comprehensive combinatorial optimization technique available for treating difficult problems. It is a neighborhood based search method which is very useful in distributed system for monitoring application. Distributed operation of Applications involve: Multiple applications deployed over different sets of hosts e.g. Datacenters. Application State monitored the performance of both systems and applications running on large-scale distributed systems. It is constantly collecting detailed performance attribute values as a large number of nodes & a large number of attributes. Tricky task of Resource aware application state monitoring is the monitoring overlay construction. In this method first, it jointly considers inter-task cost sharing opportunity and node-level resource constraints. Further, it clearly models the per-message processing overhead which can be extensive but is often ignored by earlier works. Second, REMO produces a forest of optimized monitoring trees through iterations of two phases. One stage explores cost-sharing opportunities between tasks, and the other refines the tree with resource-sensitive construction schemes. REMO also included an adaptive algorithm that balances the profit and costs of cover adaptation. This is helpful for large systems with continuously changing monitoring tasks.

General Terms

Remo, Algorithm, Resource etc

Keywords

Resource-Aware, State Monitoring, Datacenter, Adaption.

1. INTRODUCTION

Monitoring system that gather values of different status of attribute and visualize them in interesting ways can often lead to an increased understanding of a System's detailed behavior. For examination, analysis and control of distributed applications and systems, application state monitoring is required. For e.g. data stream applications may need monitoring the data receiving/sending rate, tracked data entities, captured events, signature of internal states and any number of application-specific attributes on participating computing nodes to guarantee constant operation in the look of highly bursty workloads. A fast increasing set of distributed applications ranging from stream processing to applications running in Cloud datacenters also increases. Monitoring of such applications involves collecting values of metrics, e.g. performance related metrics, from a big number of member nodes to locate out the state of the application or the system. Such monitoring tasks are called as application state monitoring.

Application state monitoring is organizing nodes into a monitoring topology where metric values from different nodes can be together and delivered is the main problem. In many cases, it is useful to collect detailed performance attributes at a prohibited gathering frequency. As an example, fine-grained performance characterization information is necessary to build various system models and to examine hypotheses on system performance. Similarly, the data rate and buffer occupancy in each element of a distributed application may be compulsory for judgment purposes when there is a seeming restricted access. So, REMO is the first system that promotes resource-aware attitude to hold up and extent several applications state monitoring tasks in extensive distributed system. Under dissimilar environments REMO employs efficient monitoring topologies. For that basic topology planning algorithm is used. At a high level, REMO operates as a guided local search approach [1].Using this approach, optimized monitoring trees for a set of moving monitoring tasks. So the existing works either construct monitoring tasks or use a fixed monitoring topology for all monitoring tasks [3] [6].In mostly scalable monitoring System involves design and operation, including what kinds of data gathered and the scale of the processing involved[2].So the monitoring system designed specific monitor activity. At any given time it collects and reports on active node. Reporting is done via a Web interface that provides the ability to sort data. Application requires collecting routine of attribute values. The application state monitoring tasks occupy collecting values of various position attributes from a large number of nodes. So its implementation is the systems and applications successively on large-scale distributed systems that always collecting complete performance attributes values as a large number of nodes & a large number of attributes.

2. BACKGROUND

REMO takes available node level resources as a first class factor for building a monitoring topology. Optimization of the monitoring topology achieves best scalability and ensures that no node would be assigned with excessive monitoring workloads for their available resources. In the on hand system does not acquire node-level resource utilization as a first-class consideration. And some assumptions in existing works do not grasp in real world scenarios. For example, many works guess that the cost of inform messages is only related with the number of values within the message, while we locate that a permanent per message overhead is not negligible. Existing works frequently believe monitoring tasks to be the stage one-time topology optimization and fixed [8]. Due to which the monitoring data may be failure. Application State Monitoring is essential for monitoring topology i.e. it should keep away from monitoring nodes spend unnecessary resources on collecting and delivering attribute values. In existing works do not take node level resource consumption as a first class consideration. Due to which monitoring data may be loss. Users and administrators of major distributed applications frequently utilize application state monitoring for surveillance, debugging, and examination and manage purposes. Each application State monitoring task at times collects values of definite attributes from the place of computing nodes over which an application is administration. REMO Consist o several fundamental component:

a. Planning

1. Task Manager
2. Data collector

Monitoring Planning:

From the users' point of observation monitoring results should be correct. The monitoring network should maximize the number of node-attribute pairs established at the central node. Such a monitoring network should not cause the unnecessary use of resource at any node.

Following fig. shows system flow:

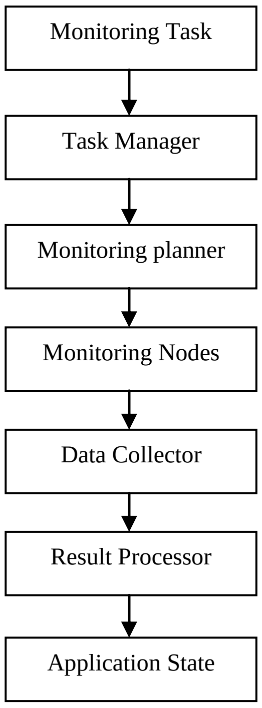

Fig 1: System Flow

1. Task Manager:

Task manager takes state monitoring tasks and removes replication among them. With such replication, node b has to send CPU utilization information twice for each update, which is clearly unnecessary. Therefore, given a set of monitoring tasks, the task manager transforms this set of tasks into a list of node-attribute pairs and eliminates all duplicated node-attribute pairs.

For instance, t1 and t2 are equivalent to the list fa-cpu utilization, b-cpu utilization and fb-cpu utilization, c-cpu utilization respectively. In this case, node-attribute pair fb-cpu utilization is duplicated, and thus, is eliminated from the output of the task manager. Management core takes de-duplicated tasks as key and schedules these tasks to sprint. One key sub-component of the management core is the monitoring planner which identify the inter relationship of monitoring nodes. For straight forwardness, we also pass on the overlay linking monitoring nodes as the monitoring topology. In accumulation, the management core also provides significant support for dependability improvement and breakdown handling.

2. Data Collector:

Data collector provides an algorithms and library of functions for powerfully collecting attribute ethics from the monitoring network. It also serves as the monitoring data and provides monitoring data entrance to users and complex applications. On a high level, a monitoring system consists of n monitoring nodes and one central node, i.e. data collector. Result processor executes the actual monitoring operations together with collecting and aggregating attribute values, triggering warnings, etc.

b. Partition Augmentation:

Through a guided iterative process the partition augmentation procedure is planned to create the attribute partitions that can potentially decrease message processing cost. At each iteration, REMO first sprint partition augmentation process to produce a list of talented runner augmentation for civilizing monitored workload between monitoring trees. If the total no. of applicant augmentations is Very huge, this procedure spick and span down the size of the applicant list for estimation by selecting the most talented ones through cost estimation.

c. Resource-aware Evaluation:

For estimation of the objective function for a known applicant partition augmentation the resource-aware evaluation process evaluates by constructing trees for nodes pretentious and method the total number of node-attribute cost pairs that can be composed using these trees.

3. DESIGN

1. Per Node Daemon:

Our System will rely on node daemon process which will monitor resources with help of system provided tasks mangers and libraries .It will also take care of communicating and sending attribute to respective monitor. It can take help of system monitoring tool like top, ps etc. for collecting resources utilization data. In this each node itself monitors and collects resource information and sending this information to assigned monitor. Also it receives information from central node or monitoring node about change.

2. Monitoring Node:

Every monitor will be characterized by attributes it is monitoring. Attributes can be collection of one or more attributes of CPU, memory, virtual memory, network utilization parameters. Monitoring Node calculates load and self resource monitoring. It also collects resource utilization information from assigned node and sending this information to the central node.

3. Central Node:

Central node will take care of following activities:

i. Monitoring topology decision
ii. Web interface.

Monitoring Topology Decision: Deciding monitoring topology is a crucial task because scalability of system is depends on this decision. Topology planning using Tabu search algorithm which is describe in next section.

Web Interface: Reporting is done via a web interface that provides that provides the ability to sort data, run moderately complex selection queries and show graphs to recent history. Through the web interface collected information making available from monitoring nodes to admin for decision making action.

4. IMPLEMENTATION STRATEGY

Monitoring system consists of n monitoring nods and one central node i.e. Data collector. Each monitoring node has a set of observable attributes. Fig.2 shows list of node-attribute pairs. Monitoring planner organizes monitoring nodes into monitoring trees where each node collects values of a set of attribute. Also each monitoring node has separate configuration file, which is useful for communication of different nodes. Challenging tasks in this is, monitoring network should maximize the no. of node-attribute pairs received at the central node. Such a monitoring network should not cause the excessive use of resource at any node. Multiple monitoring nodes collects resource.xml file. In this xml document resource information of every monitoring nodes is available. Then this file forwarded to the central node. Now at this stage Tabu search algorithm is used. And using this algorithm load is assign to the nodes using available resource information which is collected from different monitoring nodes. Then at last topology is decide.

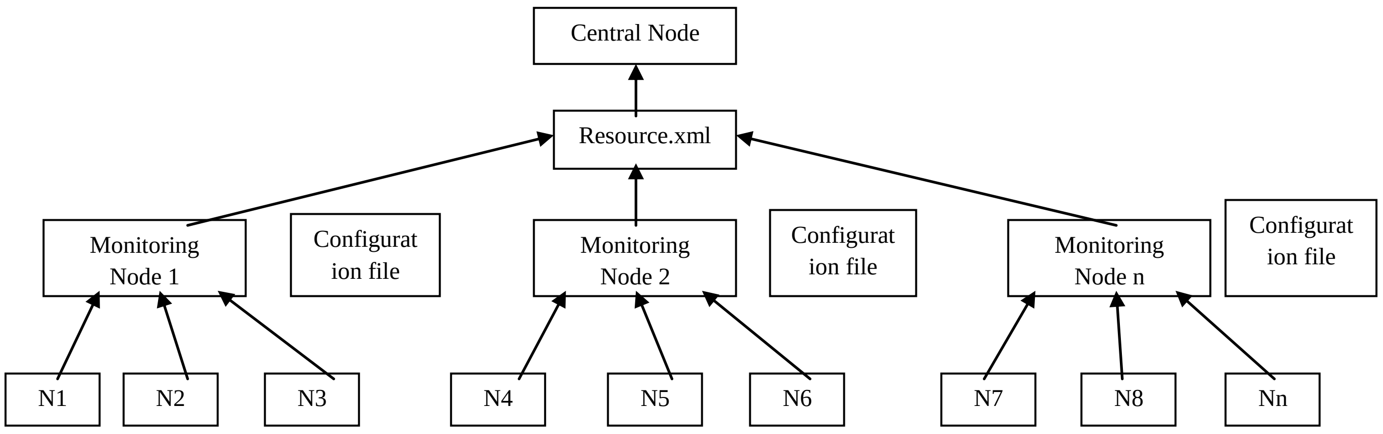

Fig 2: System Design

TABU SEARCH ALGORITHM:

The Tabu search procedure is a neighborhood based search method deterministic mechanism of avoiding local minima. The general idea of Tabu Search is to start from some initial solution and iteratively move among neighboring solutions. At each iteration move to the best solution in the neighborhood of the current one is performed. To avoid local minima, the memory of already visited solutions is introduced most frequently used type of that memory is the Tabu list. The Tabu list stores some no. of already visited solutions, its attributes, or moves leading to them. During the search process the move that leads to the solution i.e. stored in the Tabu list is forbidden. Many implementations of Tabu search method for various optimization problem shows that Tabu search can deliver optimal or near optimal solution. The efficiency of Tabu Search method is strongly dependent on the proper selection of its attributes i.e.

1. Initial solution

2. Neighborhood

3. Tabu list

4. Stopping Condition.

Algorithm:

```
Input: $TabuList_{size}$
Output: $S_{best}$
 $S_{best} \leftarrow$ ConstructInitialSolution ();
TabuList $\leftarrow \phi$;
 while $\neg$ StopCondition () do
 CandidateList $\leftarrow \phi$;
 for $S_{candidate} \in Sbest_{neighborhood}$ do
 if $\neg$ ContainsAnyFeatures ($S_{candidate}$, TabuList) then
CandidateList $\leftarrow S_{candidate}$;
End
```

PID	Name	Status	Threads	Memory Usage	Virtual Memory
7155	python	running	1	5MB	11MB
7154	sleep	sleeping	1	0MB	4MB
7151	sleep	sleeping	1	0MB	4MB
7147	signin-u	sleeping	3	18MB	107MB
7140	mission-ctrl-5	sleeping	4	6MB	43MB
7134	telepathy-indicator	sleeping	4	10MB	83MB
7107	ubuntu-provider	sleeping	3	5MB	31MB

```
End
$S_{candidate} \leftarrow$ LocateBestCandidate (CandidateList);
if Cost ($S_{candidate}$) $\leq$ Cost ($S_{best}$) then
$S_{best} \leftarrow S_{candidate}$
TabuList $\leftarrow$ FeatureDifferences ($S_{candidate}$, $S_{best}$);
while TabuList > $TabuList_{size}$ do
DeleteFeature (TabuList);
End
End
End
return $S_{best}$ ;
```

5. RESULT

From REMO can collects a large fraction of node-attribute pairs to serve monitoring tasks presented to the system. REMO collects resources of every monitoring node. The result status is for multiple node. All resource information are available for nodes and through the web interfaces the data collects in tabular and sorted form. The Current result status is for single node. Above table shows result of single node and all resource information is available for single node.

6. CONCLUSION

Thus a resource-aware multi-task optimization framework is used for application state monitoring in distributed systems. REMO is a technique for generating the association of monitoring that optimizes various monitoring tasks and balances the resource utilization at different nodes using Tabu search algorithm. We also proposed adaptive techniques to efficiently handle continuous task updates, optimization techniques that speedup the searching process. REMO is observed as extensive technique for performance applicalability of optimization.

7. REFERENCES

[1] Shicong Meng, Student Member, IEEE, Srinivas R.Kashyap,Chitra Venkatramani, and Ling Liu, Senior Member, IEEE. Vol. 23, No.12, DECEMBER 2012

[2] K. Park and V. S. Pai, Comon: a mostly-scalable monitoring system for planetlab, Operating Systems Review, vol. 40, no. 1, pp. 6574, 2006.Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

[3] D. J. Abadi, S. Madden, and W. Lindner, Reed: Robust, efficient filtering and event detection in sensor networks, in VLDB, 2005.

[4] G. Cormode and M. N. Garofalakis, Sketching streams through the net: Distributed approximate query tracking, in VLDB, 2005, pp. 1324.

[5] P. Yalagandula and M. Dahlin, A scalable distributed information management system, in SIGCOMM, 2004, pp. 379390.

[6] U. Srivastava, K. Munagala, and J. Widom, Operator placement for in-network stream query processing, in PODS, 2005, pp. 250258.

[7] C. Olston, B. T. Loo, and J. Widom, Adaptive precision setting for cached approximate values, in SIGMOD, 2001.

[8] S. Meng, S. R. Kashyap, C. Venkatramani, and L. Liu, Remo: Resource-aware application state monitoring for large-scale distributed systems, in ICDCS, 2009, pp. 248255.

[9] R. Huebsch, B. N. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi, The architecture of pier: an internet-scale query processor, in CIDR, 2005.

[10] A. Silberstein, R. Braynard, and J. Yang, "Constraint Chaining: On Energy-Efficient Continuous Monitoring in Sensor Networks," Proc.ACM SIGMOD Int'l Conf. Management of Data (SIGMOD), 2006.