J. Parallel Distrib. Comput. 96 (2016) 163–171

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Prediction mechanisms for monitoring state of cloud resources using Markov chain model

Mustafa M. Al-Sayed [a,*], Sherif Khattab [b], Fatma A. Omara [b]

[a] Department of Computer Science, Faculty of Computers and Information, Minia University, Egypt
[b] Department of Computer Science, Faculty of Computers and Information, Cairo University, Egypt

ARTICLE INFO

Article history:
Received 10 March 2015
Received in revised form
3 April 2016
Accepted 21 April 2016
Available online 28 April 2016

Keywords:
Markov chains
Cloud computing
Resource monitoring

ABSTRACT

Cloud computing allows for sharing computing resources, such as CPU, application platforms, and services. Monitoring these resources would benefit from an accurate prediction model that significantly reduces the network overhead caused by unnecessary push and pull messages. However, accurate prediction of the computing resources is considered hard due to the dynamic nature of cloud computing. In this paper, two monitoring mechanisms have been developed: the first is based on a Continuous Time Markov Chain (CTMC) model and the second is based on a Discrete Time Markov Chain (DTMC) model. It is found that The CTMC-based mechanism outperformed the DTMC-based mechanism. Also, the CTMC-based mechanism outperformed the Grid Resource Information Retrieval (GRIR) mechanism, which does not employ prediction, and a prediction-based mechanism, which uses Markov Chains to predict the time interval of monitoring mobile grid resources, in monitoring cloud resources.

1. Introduction

Recently, distributed computing systems have enabled a high level of resource sharing. Types of these systems include clusters, grids, and cloud computing, which allow the users to access large amounts of computing power in a fully virtualized manner, through resource pooling and a single system view. These systems deliver resources as a utility. Cloud computing provides services to the users through a "pay-as-you-go" manner with automatic elasticity. Therefore, accurate monitoring of resources consumed by the users has a great effect on accounting, billing, and system performance [6,16,13].

Monitoring resources in cloud environments is considered the key tool for controlling and managing hardware and software infrastructures. Monitoring provides information and Key Performance Indicators (KPI) for both platforms and applications. Also, SLA violations can be detected by continuous monitoring. Therefore, monitoring resources in cloud environments is an important task for both cloud Providers and cloud Consumers [4].

The monitoring process is used to collect the resources information (e.g., CPU load, memory usage, network bandwidth), which help for job scheduling, load balancing, event prediction, fault detection and recovery tasks. Because the inaccurate information would affect the performance of these tasks, the monitoring process has to ensure the consistent state. This monitoring scheme needs to adapt to the dynamic nature of cloud computing that stems from dynamic user and system loads, and elastic resource provisioning [5,23].

A monitored resource (termed the producer in this paper) sends its monitored state to one or more master nodes (termed the consumer in this paper) over the network. The producer may proactively send its state to the consumer or wait until the consumer requests a state update. The former case follows a push model, whereas the latter is a pull model.

Monitoring cloud resources would benefit from an accurate prediction model that significantly reduces the network overhead caused by unnecessary push and pull network messages. For instance, using periodic monitoring updates with a fixed interval may cause the monitored resource to push (send) unnecessary network messages containing the same information. Fixed-period monitoring may alternatively cause the master node to send unnecessary pulls (requests). However, if the Consumer correctly predicts, up to a predefined error tolerance degree (ETD), a monitored value, the Producer would not need to send this value. However, accurate prediction is hard due to the dynamic nature of system load and resource provisioning, among other factors.

In this paper, we have developed two resource monitoring mechanisms for cloud computing that are based on Markov

* Corresponding author.
E-mail addresses: mostafamcs@gmail.com (M.M. Al-Sayed), s.khattab@fci-cu.edu.eg (S. Khattab), f.omara@fci-cu.edu.eg (F.A. Omara).

http://dx.doi.org/10.1016/j.jpdc.2016.04.012
0743-7315/

Chains. The first mechanism uses Continuous-Time Markov Chains (CTMC), whereas the second uses Discrete-Time Markov Chains (DTMC). Both mechanisms use pushing of resource state (updates) when the difference between the predicted and actual values exceeds the ETD. These updates are used to tune the prediction model at the Consumer. In both mechanisms, a K-state Markov model is developed based on training data.

The used dataset to evaluate our mechanisms has been released by Google. It contains measurement data of 29 days that were collected from a computing Cluster in May 2011 [28]. Half of these data was used for training. Furthermore, we have compared the accuracy and the network overhead of our mechanisms with the push-based mechanism Grid Resource Information Retrieval (GRIR) [7] and a prediction-based mechanism [23], which uses Markov Chain Model (MCM) to predict the time interval of monitoring mobile resources (for brevity, we will call it MTI mechanism), after deploying them to monitor cloud resources.

The implementation results showed that the CTMC-based mechanism achieved better performance than that was achieved by the DTMC-based mechanism. Also, the CTMC-based mechanism achieved better performance than that was achieved by the GRIR and the MTI mechanisms.

This paper is organized as follows: background about Markov chains and related works are presented in Sections 2 and 3, respectively. The proposed mechanisms are presented in Section 4. Section 5 describes the evaluation of our proposed mechanisms. Finally, the conclusion and future work are presented in Section 6.

2. Markov chains

The artificial intelligence community has adopted stochastic technologies for machine learning. Bayes' rule is the basis for these technologies where Bayesian approaches can expect future events from prior events. Markov Chain Model (MCM) is one of these technologies. In MCM, the probability of an event occurrence at any time is a function of the probabilities of the occurred events at previous time periods [20].

Markov chains may be discrete-time or continuous-time. In the following subsections, we give a brief introduction about the key main property of Discrete-Time Markov Chain (DTMC), and Continuous-Time Markov Chain (CTMC).

2.1. Discrete-Time Markov Chain (DTMC)

Markov property in DTMCs implies that [10,21]:

$$P[X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0] = P[X_{n+1} = j | X_n = i] = P_{ij}$$

where P_{ij} is the one-step transition probability from state i to state j at a specific interval. These transition probabilities could be expressed in a $K \times K$ transition matrix, where K is the number of states [15].

Assuming a transition matrix with an initial probability distribution vector π, the probability that the chain is in a state i after n steps is the ith entry in the vector π^n [14].

Although matrix self-multiplication can be computed in $O(\log(n))$ instead of $O(n)$, such performance gain would be tangible for large values of n. The value of n in our work was small ($\sim$40), which did not warrant the use of optimized multiplication.

2.2. Continuous-Time Markov Chain (CTMC)

According to the CTMC, the state transitions happen at any period of time [3]. CTMCs move from one state to another according to a DTMC model. The time spent in each state is exponentially distributed with parameter λ. The CTMC model is described by a transition matrix $T = [P_{ij}]$, as in DTMC model, together with a set of time rates $\{\lambda_{ij} : i, j \in S\}$, where S is the state space that has been expressed in a transition rate matrix $R = [\lambda_{ij}]$. Every time state i is visited, the chain spends there, on average, $E(t_i) = 1/\lambda_i$ time units before moving on [2,15].

3. Related works

The existing monitoring systems are based on three basic models: Push, Pull, and Hybrid. In Push model, Producers push the updated information to the Consumer under some trigger conditions based on a Service Level Agreement (SLA). In Pull model, Consumer requests resources information from Producers when it needs. The Hybrid model allows switching between Push and Pull models according to a specific threshold [7,18].

The monitoring systems of distributed systems can also be classified into three types based on these models: (1) Push-based monitoring systems, such as Nagios Service Check Adaptor (NSCA) [12], PCMONS [9], and Lattice [8]; (2) Pull-based monitoring systems, such as Nagios Remote Plugin Executor (NRPE) [11] and Hyperic HQ [17]; and (3) Hybrid systems, such as Ganglia [26] and Monalisa [19].

According to [1], the monitoring mechanisms based on the Push model are considered the most suitable for large distributed systems, where the update messages are pushed to the Consumer based on the Producer state. This rescues the network and the Consumer from useless messages. On the other hand, in the Pull model, each pull operation is offset by another push operation. This doubles the consumed bandwidth of the network and the load on the Consumer.

Hence, our proposed monitoring mechanisms are based on the Push model, where the Producer pushes its updates to the Consumer when the difference between the actual measurement and the predicted state values, using a Markov Chain Model (MCM), exceeds a certain *Error Tolerance Degree* (ETD) limit, which is defined according to the users' requirements.

In an attempt to minimize unnecessary and useless update messages and at the same time maximize the consistency between the Producer and the Consumer, Chung and Chang [7] has proposed a resource monitoring mechanism for Grid computing called Grid Resource Information Retrieval (GRIR). GRIR improves on the Push model. The authors there have examined a set of data delivery protocols: the Offset-Sensitive Mechanism (OSM), Time-Sensitive Mechanism (TSM), and hybrid Announcing with Change and Time Consideration (ACTC). They found that the hybrid protocol outperforms other protocols, because it is based on tuning the updating time interval and the threshold dynamically when any change occurs.

A comparative study has been conducted between the GRIR (based on the ACTC protocol), and the monitoring mechanism of Ganglia system. According to the comparative results, it is found that the GRIR mechanism achieved a high quality and a small degree of communication overhead [1]. So, the GRIR mechanism will be considered as a benchmark to evaluate the performance of our proposed mechanisms.

MCM has been widely used to model sequential events, such as natural languages, human speech, and animal behavior. Also, the Markovian property (i.e., the conditional distribution of the next state is based only on the current state) satisfies the memory less property, which would be beneficial during continuous monitoring. On the other hand, it has been used, in a limited range, to monitor resources in large distributed environments to overcome the high level of network overhead by the pull-based systems [23,3,2]. Examples of these limited attempts, which are the base of the work in this paper, are discussed in the following paragraphs.