Page-Based Anomaly Detection in Large Scale Web Clusters Using Adaptive MapReduce (Extended Abstract)

Junsup Lee[1] and Sungdeok Cha[2]

[1] The Attached Institute of ETRI, Daejeon, Republic of Korea
jslee@dependable.kaist.ac.kr
[2] Department of CSE, Korea University, Seoul, 136-701, Republic of Korea
scha@korea.ac.kr

Abstract. While anomaly detection systems typically work on single server, most commercial web sites operate cluster environments, and user queries trigger transactions scattered through multiple servers. For this reason, anomaly detectors in a same server farm should communicate with each other to integrate their partial profile. In this paper, we describe a real-time distributed anomaly detection system that can deal with over one billion transactions per day. In our system, base on Google MapReduce algorithm, an anomaly detector in each node shares profiles of user behaviors and propagates intruder information to reduce false alarms. We evaluated our system using web log data from www.microsoft.com. The web log data, about 250GB in size, contains over one billion transactions recorded in a day.

Anomaly detection systems are often considered impractical solutions because of two major limitations. One is a difficulty of collaboration among anomaly detectors from web servers. The other limitation is real-time consideration. Existing systems fail to deliver real-time performance and often require expensive computational cost during training and evaluation. Conventional ADSs implicitly assume that all activities related to an event have been completed before the event may be inspected. To satisfy such assumption, the systems usually have timing windows to ensure that an event is not inspected until complete information is available. Consequently, the systems fail to satisfy real-time constraints.

To overcome these issues, we developed a page-based anomaly detection system (PADS). The PADS keeps track of access patterns on each service object such as web page and generate models per pages. In this page-profile based ADS, compare with other user-based ADS, timing windows are unnecessary during operation. An anomaly detector in each node, base on Google MapReduce algorithm [1], shares self-learned profiles and propagates intruder information to reduce false alarms.

The PADS architecture employs a combination of self-learning and profile-based anomaly detection techniques. Self-learning methodology enables the PADS to study the usage and traffic patterns of web service objects over time. In

R. Lippmann, E. Kirda, and A. Trachtenberg (Eds.): RAID 2008, LNCS 5230, pp. 404–405, 2008.

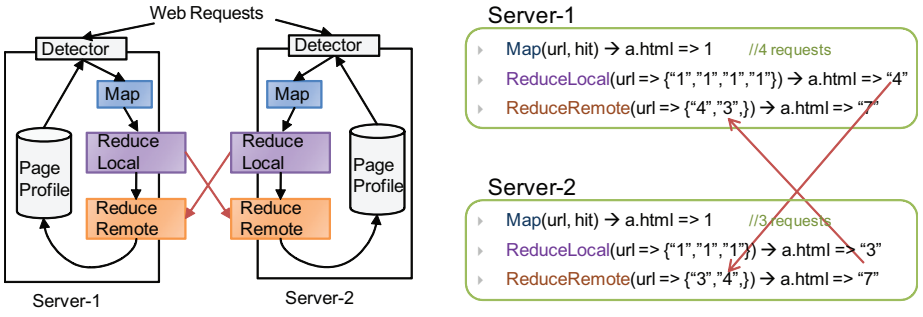

Fig. 1. Overview of PADS, Example of MapReduce in PADS (Page hit)

our model, profiles that summarize statistical features (such as exchanged bytes, query related behavior and HTTP traffic features using Chebyshev inequality, relative frequency and clustering method) per web service objects (e.q., html, asp, aspx, mspx and so on). Because popular web sites do experience legitimate and sometimes unexpected surge on particular web pages, the PADS automatically specifies access thresholds per web pages to understand unusual patterns that may occur during legitimate web query operations.

To keep pace with the latest usage and traffic pattern per web pages, each PADS node in a server farm propagates the profiles to other nodes using adaptive MapReduce technique[2]. We redesigned the technique which aware memory and network traffic. Using MapReduce algorithm, user requests are summarized into page profiles and share with other server simultaneously (Fig 1). According to our experiment on Microsoft web log data, the PADS only generate 4.44% of network traffic compare with sharing all requests among servers. In a half hour, a server receives 0.42 million requests on average. In the meantime, only 7.2 thousand page profiles are produced by LocalReduce. While total size of these profiles is 3.6MB, all requests are 81MB relatively.

In our system, each web page profile is generated dynamically by the system initially and subsequently updated and shared with other servers in real time by Google MapReduce algorithm. Currently, we are evaluating PADS using a web log data from 'www.microsoft.com'. While every web servers maintain same latest web page profiles which is about million, communication overhead between nodes is dramatically low.

References

1. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. Operating Systems Design and Implementation, 137–149 (2004)
2. Ranger, C., Raghuraman, R., Penmetsa, A., Bradski, G., Kozyrakis, C.: Evaluating MapReduce for Multi-core and Multiprocessor Systems. In: Proceedings of the 13th Intl. Symposium on HPCA, Phoenix, AZ (February 2007)