Table of Contents

A Lightweight System for Just-In-Time Aggregation of Machine Generated Data in a Distributed Network

Student : Michael Dreeling

Waterford Institute of Technology, Ireland.

Abstract. Application Data Management or ADM is a research area concerned with the aggregation, storage and presentation of information typically emitted from Enterprise Applications. ADM is a subset of Enterprise Information Management or EIM [27]. EIM usually joins together Business Intelligence and raw Application Data. Large scale commercial EIM systems started to appear in 2003, and since then many of the larger software vendors such as Tibco, Oracle and SAP [28] have created their own systems. The goal of each system is to present the Machine Generated Data [17] or MGD being emitted from applications and servers. Over the past few years, there has been a focus in EIM on dealing with the challenge of Big-Data as companies such as Amazon, Facebook and Zynga generate terabytes of application information per day. This data, when organized, can be used in the analysis of system problems, user behavior and security threats.

1 Introduction

Application Data Management or ADM is a research area concerned with the aggregation, storage and presentation of information typically emitted from Enterprise Applications. ADM is a subset of Enterprise Information Management or EIM [27]. EIM usually joins together Business Intelligence and raw Application Data. Large scale commercial EIM systems started to appear in 2003, and since then many of the larger software vendors such as Tibco, Oracle and SAP [28] have created their own systems. The goal of each system is to present the Machine Generated Data [17] or MGD being emitted from applications and servers. Over the past few years, there has been a focus in EIM on dealing with the challenge of Big-Data as companies such as Amazon, Facebook and Zynga generate terabytes of application information per day. This data, when organized, can be used in the analysis of system problems, user behavior and security threats.

Most EIM software solutions work in either an Active, Passive or Combined operation mode. An Active system requires installation of modules or agents (sometimes called collectors) on servers which are being monitored. The agents usually run with a relatively high level of access and feed information back to a central aggregator. The agents can very efficiently pass the information back to the server and have extremely fast access to the data. A Passive system on the other hand operates entirely remotely and must work harder to retrieve and analyze the data. Typically, file and directory access will need to be granted to the host system by an Administrator over a particular protocol but software is not required to be installed once access has been granted. For the purposes of this dissertation we will focus on the Passive style. The major challenges for all EIM systems are

1. Accessing and Transmitting Data
2. Indexing Data
3. Searching Data

Data access is the most straight forward issue to solve in both the Passive and Active style as Administrators must be involved in either solution or need to at least provide some level of configuration. Data indexing and Searching however are much more difficult to solve for and as such typically involve mathematical solutions (such as Bloom Filters[22,23] for instance) to resolve. This dissertation will attempt to design a lightweight Passive analysis system which can be used to quickly aggregate and view short term data within a system, rather than detailed historical analysis. Such a system could be used to quickly analyze application issues during a high severity event within an Enterprise.

This system would have several advantages over existing software

1. A Light installation foot print (1 server)
2. Low overall network utilization
3. Low overall CPU usage on server and monitored host

2 Background and related work

In this section we will outline modern approaches to data aggregation and analysis of Machine Generated Data. This area has seen considerable innovation over the past 10 years and so there are many methods being used. Many of the systems use proprietary technology and granted patents [24]

In this section we will outline modern approaches to data aggregation and analysis of Machine Generated and JVM based Data.

2.1 Distributed Data Collection methods

Information from Enterprise Applications is typically stored in log files on the disks of servers running these applications. The servers may be running a variety of Unix, Linux or Windows based operating systems.

Each server typically stores application logs in a variety of different areas on disk depending on how the application was developed. In addition to the application logs themselves, the software used to run the application may also have logs, and errors can be reported in any one of these files. Logs break down into several categories:

a) Be-spoke Application logs (Developer created);
b) Off-the-shelf Application Software Logs (Apache, Weblogic, Tomcat);
c) Logs generated from the RunTime Environment for example JVM based logs (In Java environments);
d) Logs generated by the Operating system.

In active EIM systems collection of this data is usually performed by Agents and centrally forwarded to a Log Aggregation server where it can be searched. In highly secured environments with change management concerns, a Passive approach is preferable where a centralized logging server pulls data from each server using common protocols and aggregates it afterwards.

2.2 Active Analysis (Push)

Compared to systems which do not require custom client software installation, Active Analysis systems typically require a large amount of setup,design and configuration, requiring Agents to be installed on the operating system which is running the monitored application. The system must then be configured to read and forward information to a central location where it is collated for presentation. Agents usually forward data over a network port (UDP/TCP or HTTPS) to the central aggregator. This method of collection is the very efficient as the agents have direct access to the data under analysis. This method also scales as the number of nodes under management increases. This is due to the fact that the client machines are running some of the required software, and as such the CPU usage required to retrieve the data from them is more evenly distributed.

2.3 Passive Analysis (Pull)

Passive Analysis systems typically require less setup and design, with more or less the same amount of configuration as Active systems. One or more central servers are installed on the network and they are configured to remotely read the logs of the monitored application. Drawbacks of this approach are that it is less efficient than having direct disk access to the files in question. The reason for this is that active analysis systems can more easily be configured to filter on certain events and feed the central server when these events occur, passive systems must analyze the data once it has been retrieved, and then decide which data to keep. Also, as it uses existing operating system software (such as SSH) to make connections to remote machines, it is also highly dependent on the operating system under analysis.

2.4 Machine Generated Data

This article [10], created relatively recently (Dec 2010) describes machine generated data as information automatically created from a computer process, application, or other machine without the intervention of a human. This statement is of course partially true, and is acknowledged as such, due to the fact that information generated by processes or machines is inherently tied to the humans which are interacting with it or issuing commands to the system.

It is more accurately described by Monash [17], in a provisional definition, as information that was produced entirely by machines OR data that is more about observing humans than recording their choices. The article goes on to further describe the categories under which MGD falls, such as the output of network, telemetry and other computer equipment appliances.

The text also categorizes MGD as information which can be, and probably should be, frequently discarded. This statement fits with the purpose of this dissertation, which is to develop a method to quickly review, but not store MGD in a sliding window fashion.

2.5 Available Data Collection and Aggregation Systems

GrayLog2 [1] is an open sourced centralized logging system intended for use by organizations in order to provide a window onto real-time and historical log data. The data is pushed to Graylogs server (graylog-server) using syslog from each registered node in data pipelines known as streams. The nodes themselves must be registered within GrayLog using its UI interface. Gray log records the Date, Host, and Level of each of the incoming messages. It reports the number of message per second which are coming into the system in the UI. It has advanced searching features using Elastic Search as its backend, which even allows some natural language to be used when searching records (i.e you can search a Timeframe such as from yesterday). This UI also provides advanced graphing and tracking of multiple sources at once. It does not support a centralized Multi-SSH server siphon as outlined in the dissertation.

LogStash [5] is also an open sourced tool for managing events and logs. It is written in Ruby and runs on a Java runtime using JRuby. Again with LogStash it uses a push model in order to Ship events from each node to Brokers which cache the information for the Indexers so that they do not become overloaded, this is then finally redirected to the Storage and Search server that is accessible via a Web UI. LogStash has a highly configurable UI (utilizing Kibana) which allows you to create your own dashboards for your application. LogStash needs to be installed on the nodes which contain the logs if they are not being presented over the standard Syslog port 514. [19]

OpenTSDB [2] provides another way to push log and metrics events to a central location. It is a highly scalable system which is in use by many companies [3] at the time of writing. The base components for the system is the Time Series Daemon or TSD, several of which is installed on the network being monitored. Each TSD uses HBase [4] to store and retrieve data. Collectors, which are basically scripts that are scheduled to run on each monitored node, send data to each TSD which in turn writes the data to HBase. Each monitored node must be aware of the TSDs on the network so that it can send data to them. Again, OpenTSDB is mainly is focused on aggregation and storage of large amounts of historical data and at massive scale, so it utilizes the common push method to achieve this. From a UI perspective, OpenTSDB provides only basic functionality [6] when compared to Kibana based solutions.

Apache Flume [20] is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of streaming event data. Everything in Flume is defined in the content of an event. Events are transported, routed and stored by Flume Agents. Flume Agents are a collection of Flume Sources, Sinks and Channels, Sources can poll or wait for events, Sinks allow the data to be streamed to a destination (i.e HDFS), and channels provide the conduit between the Source and the Sink (i.e the Source sends the events to a Channel and the Sink drains the Channel).

There are no SSH connectors for Flume (although open source attempts [7] have been contributed), the reason for this is that the default solution for reading data is to install agents on the Application Server obtaining the logs and to send them to a Sink. The output of this dissertation project is to build a completely passive SSH spooling service that can be connected to a data source, so it is possible that it could be made compatible with Flume.

Splunk [21] is a closed-source commercial offering which is in use by thousands of customers worldwide. It is available in both free and enterprise edition(s). Splunk uses an architecture whereby a Forwarder sends data to a Splunk Indexer, a separate server known as the Splunk Search Head provides a UI by which to search the Indexer(s) for event data. It follows paradigms which are very similar to that of OpenTSDB and Flume but with a much richer UI and a simplified configuration. Splunk also scales horizontally extrmnely well Splunk does not provide a feature to read remote files via SSH, if you wish to do so, a script needs to be written to connect to the SSH server and then forward the data.

Kafka [22] is a newer data collection system which was not mentioned in the original proposal, at the time it was at viewed as being so similar to Flume that only one of them merited deeper analysis. It was also at version 0.7.0 having not been released from the Apache Incubator after it was contributed in 2011. Today, Kafka is one of the leading real time open source data pipelines. LinkedIn also released a companion paper [9] which describes the entire system in detail. Kafka is also horizontally scalable and supports up to 40 billion events per day at LinkedIn. [10]

Kafka uses the notion of a Producer and Consumer which send a retrieve data respectively from a Kafka cluster of Brokers. Kafka also has the concept of a channel (similar to Flume) on which you send data to Brokers which is called a Topic. Kafka is very much the just mechanism for achieving data collection at Scale, it does not provide any default Producers or Consumers, these must be written. An example of a simple tail Producer [11] is available on GitHub. Apache Storm [12] is frequently used with Kafka as a distributed computation engine which operates on data streamed from a Kafka cluster.

Suro is a data pipeline in use by both NetFlix and Riot Games [14, 13]. Suro is a collection and storage mechanism for streaming data. The version of Suro in use by Riot Games is closed source and known as Honu, but is similar in operation to the version of Suro recently open sourced by NetFlix, they were both developed by the same author independently for both companies. Suro uses a pure collect and forward architecture, requiring Collectors to be installed close to the application being monitored. Agents are not required to be installed on the monitored nodes, however any application which needs to send data to Suro needs to implement SuroSDK, which effectively generates a client side dependency. You cannot use Suro to SSH or retrieve data from nodes remotely. However, it would be possible to implement Suro support into the Multi-SSH server such that it provided its output to Suro. Suro can forward raw data to either S3 or Kafka for further processing. It does not have a real-time consume library or mechanism, hence the use of Kafka.

```
\begin{thebibliography}{}  % (do not forget {})
.
.
\bibitem[1982]{clar:eke}
Clarke, F., Ekeland, I.:
Nonlinear oscillations and boundary-value problems for
Hamiltonian systems.
Arch. Rat. Mech. Anal. 78, 315--333 (1982)
.
.
\end{thebibliography}
```

Sample Output

References

Clarke, F., Ekeland, I.: Nonlinear oscillations and boundary-value problems for Hamiltonian systems. Arch. Rat. Mech. Anal. 78, 315–333 (1982)