TELEINFORMATICS REVIEW
No. 1, 2013

Real-Time Content Delivery Networks Monitoring

Krzysztof KACZMARSKI, Marcin PILARSKI

Wydział Matematyki i Nauk Informacyjnych,
Politechnika Warszawska,
ul. Koszykowa 75, 00-662 Warszawa
k.kaczmarski@mini.pw.edu.pl, m.pilarski@mini.pw.edu.pl

ABSTRACT: Modern Content Delivery Networks require both monitoring and analysis tools in real-time. The first ones deliver real time alerts and system behaviour. The second are intended to offer more complicated reports which may be used for data mining, prediction or even service accounting. In this paper we present a method of covering all these functionalities by providing a real time (or almost real time) monitoring, but based on sophisticated statistics which may be basis for deeper system analysis. Our solution is based on real time MOLAP (*Multidimensional Online Analytical Processing*) cube creation and time series stored in a dedicated database system.

KEYWORDS: CDN, Time Series, Cloud Computing, MOLAP, Big Data.

1. Introduction

Content publishing in the Internet has become centre of interest for many media companies. This task requires both huge storage capacity and satisfactory delivery time to even tens millions of users at a time. Usually no media publisher can handle network traffic of this volume. On the other hand, telecommunication companies already created good infrastructure for sending data to their clients. Cooperation with content providers allows for fast and smooth content delivery by building Content Delivery Networks (CDN) which cache data offered by publishers in distributed manner. Collection of servers of this kind is fully or partially autonomous solution for Infrastructure as a Service (IaaS) system. By migrating content from the central publishing server (often called an origin server) and distributing it to as many nodes as it is necessary, delivery path to clients is shortened, network traffic and bottlenecks are reduced. It is also much more economical to place CDN nodes in already existing Telco backbone than to build a new backbone from scratch only for one content provider. Telecommunication companies begun selling CDN as a service to many content providers which further reduces running costs. In this win-win solution each party is satisfied: media can achieve better quality of service for end users reducing internal financial engagement and the users may get high quality of streaming media on demand. However, this scenario requires tight cooperation between a content provider and a network owner. It often includes charges for certain data volume or achieved speed of data transfer and therefore requires detailed analysis of data flow, activities of users and service quality. The task of high level data delivery monitoring remains an open problem. CDN operator must also monitor the current system state for example in case of dramatically increasing number of sessions which may slowdown all operations. Both parties are interested in a tool able to both monitor current system state and deliver various statistics of users activities. However, this kind of solutions do not exist yet or are difficult to be integrated with existing telecommunication technology.

2. Problem Description

2.1. System Requirements

Currently monitoring systems must not only focus on runtime events and triggering warnings but have to create a database including detailed information, statistical as well as sensor measurements. It must be available real-time and cover any given period of time not losing any detail. For example, if a malfunction is detected one may need to track its behaviour up to the unlimited point in the past in order to find possible coincidences with other events. Therefore all relevant data concerning system operation must be stored in exact shape. The list of functional requirements includes but is not limited to:

- Querying Speed: All data and statistics must be available real-time;
- Durability: The system must be able to answer any query for any given period of time. No measurements may be lost in case of hardware failures;
- Scalability: The system must scale well in case of substantial growth number of requests and cache utilization;
- Availability: The system must be available and must continue monitoring all the time even in case of hardware failures;
- Flexible Querying: The system must not limit types of queries or aggregations. Data aggregation cannot reduce information resolution in order to be able to perform the same queries for all time periods.

3. Monitoring Process

Content delivery monitoring raises new challenges when compared to other computer monitoring systems. On one hand we require on-line almost real-time responses including detailed and broad system statistics while on the other hand we need to include detailed users behaviour tracking.

The first demand is solved by systems which track simple events, display detailed but short-term view of selected measurements. They usually aggregate this information and store it as a higher level information in a database. This approach gives perfect and very quick view of the current situation while is not able to present historical data with the same level of details.

The second approach resulted in log collecting systems which try to store all available information coming from systems' logs and then allow for unlimited querying of this data. Although they seem to solve drawbacks of the previous systems they lack in presenting the current view of the system. They must execute a query on the latest log entries to see what is going on.

Our solution has joined both properties by allowing for current system view and also for unlimited querying in the past data.

The most important part of the system is data gathering and storing in the repository. Its efficiency and flexibility influences capabilities of all the rest of the system. In our case it contains the following steps:

1. Data fetching and initial treatment. A single collector instance gathers data from a single node and combines it with other available information. For example a geographical location may be assigned to a client IP number. A collector may work on a node's side or may be running in a dedicated machine together with other collectors. At this point no data reduction is done.

2. Local node aggregation. The collector calculates aggregates upon collected data. Kind of aggregations depend on measured metrics. As a result of this process each collector produces a multidimensional cube containing aggregated values for the analysed time period. Obviously, different metrics may require different time periods and aggregating functions. For example, upon a fragment of a data transfer log a collector may calculate an average bandwidth for five minutes. See the next section for details.

3. Global aggregation. Some metrics (like a number of unique client IP numbers for the system) cannot be calculated separately from other nodes. This measurements have to be done in a centralized place, where some form of data from all nodes is collected, or using a kind of a distributed map-reduce procedure. The later is necessary if high efficiency of the measurements is required. Tools like Hadoop [3] may be applied.

4. Inserting time series into the repository. Data calculated by collectors and represented by values equipped with timestamps for a time series. All collectors send all their time series into the repository and the process begins again for the next time period.

Figure 1 presents a schema of data flow between monitored machines, collectors and the data repository. Local collectors use node logs and external data sources. Local aggregators consume data prepared by collectors and then send results to the repository or to reduction workers which perform global aggregation. Since this paper focuses on time series and we do not intend to reduce dimensionality of our data as it happens in global aggregation reduction workers and global aggregation are not described here. This problem will be addressed in another paper.

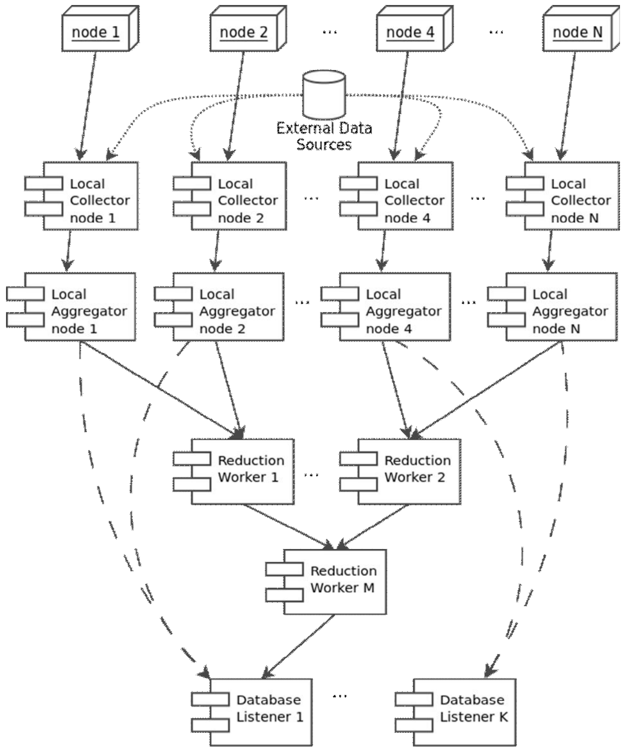

Figure 1. Data flow between nodes and repository

3.1. Local Node Aggregation

Log files containing entries for each event may grow up to several hundreds thousands of records. The first level of aggregation done locally in a node must reduce this very detailed information by aggregating events which apply to the same data category or measured time period. For example, if we calculate number of requests for certain URL all records referring to that URL may participate but will produce only one value as an output. Also time granularity may be reduced to achieve proper metrics.

Local data aggregation processes raw data taken from a node collector and produces a time limited MOLAP cube. The cube may contain arbitrary number of dimensions and metrics depending on client needs. Figure 2 presents a sample cube in form of a tree. Each dimension defines a separate level. Any possible value of a dimension generates children in the next level. Leafs contain metrics. In the example the cube has three dimensions (timestamp, language and provider) and two metrics (requests and Mbps).

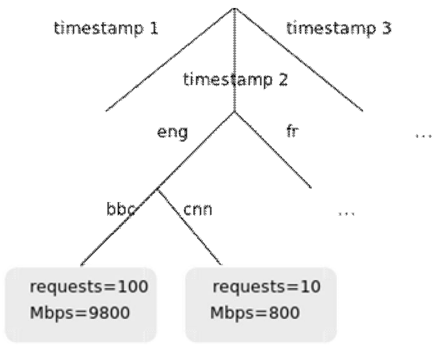

Figure 2. A Tree-based MOLAP cube structure for three dimensions (time, language, provider)

Time stamp value distinguishes the measured values in time. Its presence in the cube may vary. In the example it is placed as the first dimension but in other cases it may be better to place it in other level. Order of dimensions may significantly influence cube creation time and should be carefully tuned.

Let us assume that the input data for the cube creation contains n log entries. The cube also requires m dimensions and for each dimension $i \in 1..m$, k_i denotes number of possible values. In our example $k_2 = 2$ since we have two possible values on level 2: ‘eng’ and ‘fr’. This process has time complexity T_C described by equation (1).

$$T_C = O\left(n\sum_{i=1}^{m} k_i\right). \tag{1}$$

For each log line we need to go through all levels and need to find appropriate child for each node. Approximation of k_i with m may be wrong because it is possible that k_i is much larger than m.

If l denotes number of metrics to be measured the memory complexity S_C of the cube in the worst case is given by formula (2).

$$S_C = O\left(l\prod_{i=1}^{m} k_i + \sum_{i=1}^{m}\prod_{j=1}^{i} k_j\right). \tag{2}$$

The cube contains all measured aggregations for a set of time stamps in all dimensions. Each time series is defined by all metrics values and a collection of dimensional values which are denoting path in the tree from the root to one of the leafs. In most cases database is fed with kind of a raw data containing one time stamp with one value and all dimensions for single entry. This kind of output may be produced by going through the tree depth-first. In our example this would produce the following fragment of an output presented in figure 3.

```
timestamp1 …
timestamp2 lang=eng provider=bbc request=100   Mbps=9800
timestamp2 lang=eng provider=cnn requests=10   Mbps=800
timestamp2 lang=fr …
…
```

Figure 3. Fragment of possible output sent to a database

3.2. Time Series for Content Delivery

Content delivery network monitoring focuses on several aspects of system parameters:

– delivery metrics (number of requests, average megabits per second for 10 mins/1 h time window, number of sent gigabytes, number of particular file types requests, time to first byte, video buffering events, and many more,...);
– cloud life metrics (number of unique connections, maximum number of connections from a single ip, number of nodes alive, storage size currently occupied, number of cached files, and more,…);

- monitoring system self-state metrics (time of data processing, memory occupied, number of nodes monitored, number of communication errors, collectors heartbeat, and more,...);
- internal node hardware metrics (bytes sent/received over network, network errors, disk/memory usage, disk/memory free, processor load, etc.,…).

For each metric as many time series as possible dimensions value may be generated. For example if we consider gigabytes sent to a client in a cloud of 50 nodes (one dimension with 50 possible values) we may imagine 50 time series, one for each node. If we add a dimension of transfer speed with three possible values (<1Mbps, 1-5Mbps, >5Mbps) we get 150 time series defining volume of data sent to clients connecting to all available nodes with all available speeds.

Time series aggregation is a mechanism to answer global queries upon local time series. If we again consider gigabytes being sent we may get total number of gigabytes by adding all available time series values for given time interval. In our case of not synchronized time periods and possibly different granularity of measurements this would not be possible without linear interpolation. An example of time series aggregation is presented in figure 4.

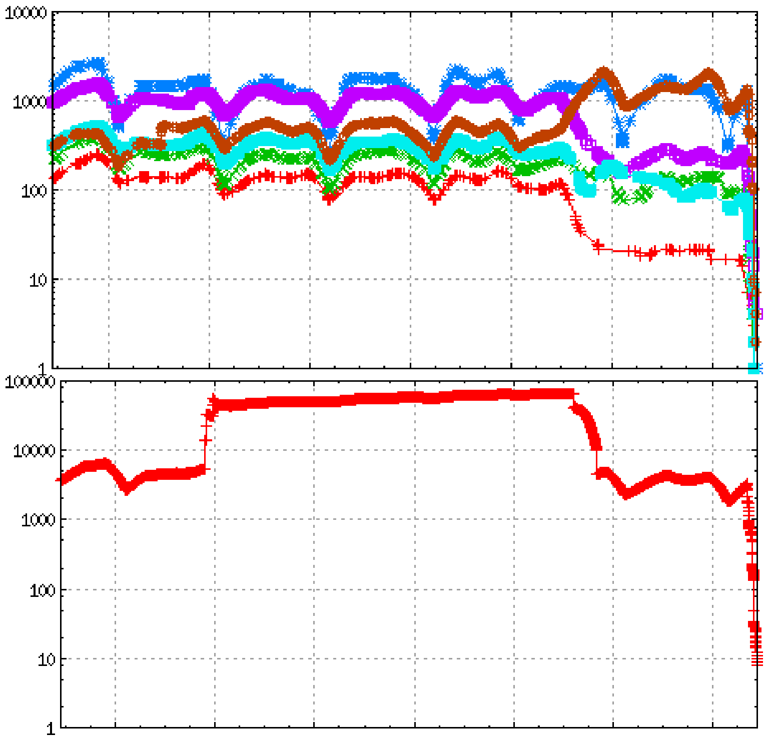

Figure 4. Time series without aggregation (top) and with aggregation found by linear adding interpolated times series (bottom)

Five time series on top are aggregated to one presented in the bottom. New time stamps come from merged time stamps from all the series. Values in these points come from linear interpolation.

4. Working Prototype

We build a working prototype on top of OpenTSDB [1]. It is a front-end to a HBase [2] distributed database system which provides very good scalability and transparency of distribution. Node collectors constantly analyze content delivery logs. Currently we monitor one of a commercial CDN systems composed of about a hundred nodes. An example of log format of this system is presented in table 1.

Table 1. An example of log format loaded by node collectors. Please note that in Apache log format no field length is needed. Fields are separated by white spaces

No	Field Name	Field Type
1	LOG VERSION	NUMBER
2	ACCOUNT_NAME	TEXT
3	REQUEST_ID	NUMBER
4	START_TIME	EPOCH [MS]
5	ELAPSED_TIME	SECONDS.MSECONS
6	CLIENT_IP	DEC.DEC.DEC.DEC
7	HTTP_METHOD	TEXT
8	URL	TEXT
9	HTTP_VERSION	TEXT
10	RESPONSE_CODE	NUMBER
11	TOTAL_OBJECT_SIZE	NUMBER IN BYTES
12	BYTES_SENT_TO_CLIENT	NUMBER IN BYTES
13	CACHE_HIT_BYTES	NUMBER IN BYTES
14	RANGE_FROM	NUMBER IN BYTES
15	RANGE_TO	NUMBER IN BYTES
16	CONTENT_TYPE	ENUMERATION
18	REDIRECTION_LOCATION	ENUMERATION

For the constant number of dimensions and metrics collected we achieved a liner processing time of cube creation, transformation to database input format and sending. In current configuration including about 20 metrics with 12 dimensions each we manage to process around 1000 log entries per second on single CPU core.

5. Related Works

There appeared many different solutions which are able to track events and parameters of distributed nodes and present them in a feasible administration console. They must process millions or billions of records efficiently. Often companies collect all possible data for future analysis even not being sure what kind of data extraction is possible and what kind of analysis is necessary. In many cases advanced data mining algorithms are necessary to get appropriate information from this kind of huge data sets.

Ntop is a network traffic measurement system that shows the network interface usage [4]. It can show many different views of network utilization and display statistics limited to the last 24 hours. It is quite low level tool and cannot collect CDN content flow information. Our solution may both perform ntop-like tasks and also provide high level data analysis.

Our solution is a bit similar to RMON monitoring standard [5] in the aspect of monitoring agents. In RMON agents are located in many places and track Internet packets going through that place. In our solution agents are located in all CDN nodes.

Different approach is presented by Chukwa which is devoted to arbitrary logs collection and analysis [6]. Data is stored in not aggregated form and may require huge space. Reports are calculated by custom functions which must process all data. In the contrary our system stores only numerical data in appropriate resolution.

Another system which is designed to scale to large number of nodes is Scribe [7]. Again it only keeps messages in a form for key-value information as categories and strings sent to subscribers of particular types of messages. It is rather a notification systems than a database which can be queried afterwards. In the contrary our solution not focuses on data volume reduction and stores only tagged numerical values which are subject to arbitrary queries. It may also legitimate with high scalability and great robustness. We managed to store thousands of data points per second and retrieve data in queries not lasting more than a few seconds even when drilling though data collected from one month of cluster operation. In our previous works [8] we described the general architecture of this system which has been now revisited and much optimized.

6. Conclusions

In this paper we presented a new cloud monitoring solution which demonstrates two unique properties. It can run in real time and may perform complicated analysis enabling for data mining in future. This results were achieved by employing a well-known distributed storage system, common time series database and a novel data collection system based on MOLAP cube processing. Up to our knowledge it is the only system with the above properties.

The system allows for combining data from many sources and inserting partially aggregated information into a database. A user may use external interfaces for further analysis and database querying. All the architecture is open in all layers and any modifications or improvements are possible in case of new needs. The system is working in a research environment in one of the telecommunication companies.

References

[1] SIGOURE B., *OpenTSDB scalable time series database (TSDB)*, Stumble Upon, http://opentsdb.net, 2012.

[2] *Apache HBase*, The Apache Software Foundation, http://hbase.apache.org, 2012.

[3] *Apache Hadoop*, The Apache Software Foundation, http://hadoop.apache.org, 2012.

[4] DERI L., CARBONE R., SUIN S., *Monitoring networks using ntop*, in Integrated Network Management (Pavlou G., Anerousis N., and Liotta A., eds.), IEEE, 2001, pp. 199–212.

[5] WALDBUSSER S., COLE R., KALBFLEISCH C., ROMASCANU D., *Introduction to the remote monitoring (RMON) family of MIB modules*, The Internet Society, Network Working Group, 2003.

[6] BOULON J., KONWINSKI A., QI R., RABKIN A., YANG E., YANG M., *Chukwa, a large-scale monitoring system*, in Proceedings of CCA, Vol. 8, 2008.

[7] ROWSTRON A. I. T., KERMARREC A.-M., CASTRO M., DRUSCHEL P., *Scribe: The design of a large-scale event notification infrastructure*, in Networked Group Communication (Crowcroft J. and Hofmann M., eds.), Vol. 2233 of "Lecture Notes in Computer Science", Springer, 2001, pp. 30–43.

[8] KACZMARSKI K., PILARSKI M., *Content Delivery Network Monitoring*, in Proceedings of FedCSIS 2012, Wrocław, Poland, 9-12 September, 2012, pp. 633-639.

Monitoring w czasie rzeczywistym dla sieci dostarczania treści

STRESZCZENIE: Współczesne Sieci Dostarczania Treści (CDN) wymagają zarówno monitoringu, jak i narzędzi do analizy w czasie rzeczywistym. Monitoring umożliwia śledzenie zachowania systemu oraz sygnalizowanie alarmów w czasie rzeczywistym. Narzędzia do analizy umożliwiają natomiast tworzenie bardziej skomplikowanych form raportów, które mogą być wykorzystywane do eksploracji danych lub prognoz, a nawet do rozliczeń za usługi. W tym artykule prezentujemy sposób obejmujący wszystkie te funkcje, dostarczając w czasie rzeczywistym (lub prawie rzeczywistym) funkcję monitorowania, opierającą się na zaawansowanych technikach statystycznych, które mogą być podstawą do głębszej analizy systemowej. Nasze rozwiązanie oparte jest na tworzeniu kostki MOLAP (ang. *Multidimensional Online Analytical Processing*) w czasie rzeczywistym oraz analizie szeregów czasowych zapisanych w dedykowanej bazie danych.

SŁOWA KLUCZOWE: CDN, szeregi czasowe, przetwarzanie w chmurze, MOLAP, Big Data.

Praca wpłynęła do redakcji: 28.01.2013