An interactive tool for intelligent analysis of geo-temporal backbone monitoring data

Nilson Luís Damasceno
Computing Institute
Fluminense Federal University
Rio de Janeiro, Brazil
nilsonld@id.uff.br

Antonio A de A Rocha
Computing Institute
Fluminense Federal University
Rio de Janeiro, Brazil
arocha@ic.uff.br

***Abstract*—Due to the increased usage of the internet for all sort of human activities, the demand for high-quality communication services continually grows, which makes essential the detection of events that compromise the quality of these services. The intelligent analysis of the monitoring data produced by routers can reveal relevant patterns hidden on the massive volume of data archived and, therefore, help to guide decisions of network providers. This paper presents a graphical and interactive web tool that performs intelligent analysis of visually selected geo-temporal subsets of monitoring data collected by routers. The tool uses a variation of a data structure to store and selectively retrieve geo-temporal data to feed the intelligent analysis. At this time, the analysis module can detect anomalies and perform data predictions using selected Machine Learn algorithms present on Microsoft's ML.Net framework.**

***Index Terms*—Network monitoring, GIS, Network Traffic anomalies, Artificial Intelligence.**

I. Introduction

Backbone internet routers commonly collect monitoring data, such as the status of the equipment itself and information related to the large amount of traffic that pass by their interfaces. Such information may include different kind of data, including the amount of packets received, transmitted and discarded, and other details about the packets themselves. Performing proper analysis of these data can indicate actual or potential issues about the quality of the network services/status.

With the evolution of Artificial Intelligence (A.I.) techniques, it became possible to analyze the collected data to identify hidden traffic patterns; to recover, via interpolation, missing measurement details; and even, to perform prediction, via extrapolation, of trends in some monitored values. However, A.I. monitoring data analysis is usually done in offline mode, apart of the network operation, imposing a significant delay between the registration of network events and the analysis of these events. Due to this delay, it is not possible to fix, on real-time, any detected issue.

The tool presented in this paper performs A.I. analysis over data collect from backbone network routers, using algorithms provided by Microsoft's ML.NET framework [1], on a real-time fashion. The tool offers visual and interactive ways to analyze all (or a subset of) data, considering also other factors (besides geo-temporal information) presented in the data, such as the location of routers and time of the event.

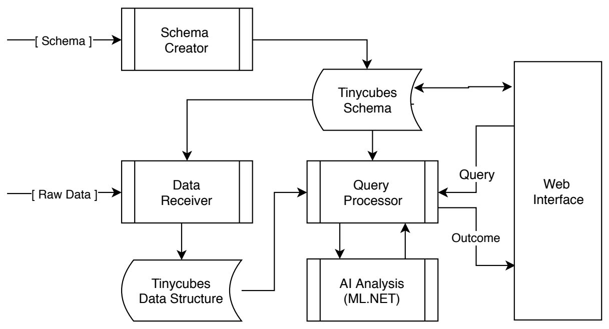

Fig. 1. Schematics

This kind of tool can be used for many different purposes, including: (i) automatic detection identification of issues regarding network performance or connectivity; (ii) generation of automatic alerts and notifications of failures; and, (iii) simultaneous visualization of failures in different monitoring points, offering a big-picture view, that may recommend further investigation of correlation among traffic anomalies.

This paper is organized as follows. Section II describes the tool structure. Section III presents a case study using actual monitoring data from two RNP Point of Presence (PoP). And, finally, Section IV lists the conclusions and opportunities for further research.

II. Tool Description

This section describes the most relevant components of the tool, as can be seen in Figure 1. The current implementation reads input data from a file, store it in a specialized data structure that performs statistical geo-temporal queries, whose outcomes feed machine learning algorithms. The final result is present in a graphical manner on a web interface.

A. Core Data Structure - Tinycubes

The tool uses a new data structure called Tinycubes. A full description of the Tinycubes data structure is beyond the scope of this paper and will be subject of a future publication. This section presents only the features of Tinycubes necessary to understand the capabilities of the tool.

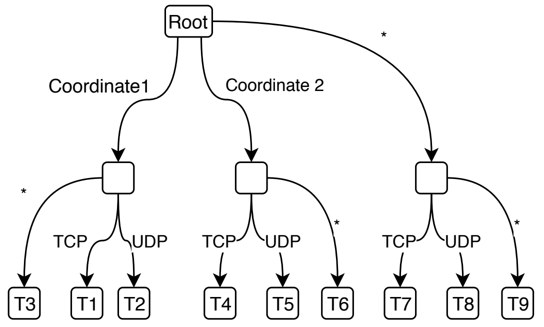

Fig. 2. Logical representation of a Tinycube with two dimensions

The Tinycubes data structure works as an in-memory database. It stores collected data and replies statistical geo-temporal queries such as "How many TPC and UDP packets was received by the POPs in specified region between date1 and date2?". The Tinycubes was derived from a previous geo-temporal data structure called Nanocubes [2]. This structure is a datacube [3] specialized in handling geo-temporal data, offering low response time for queries and moderate memory usage. Tinycubes enhances Nanocubes with new capabilities, such as even lower memory usage and online removal of stored data, which allows data stream processing.

Tinycubes stores information as a graph [4], where part of the available input data can be used to create paths to terminal nodes, whose contains a set of summarized information related to the data in the path. Typical summarized information are event counters, averages and variances. The data paths of Tinycubes can be logically viewed as datacube dimensions, which can be queried independently. As the main enhancement regarding usual datacubes, Tinycubes allows hierarchical subdivision of each dimension, which facilitates storing and retrieving information hierarchically encoded, as geographic coordinates organized as QuadTrees [5].

Figure 2 illustrates a Tinycube with a geographic coordinate dimension followed by a categorical dimension that identifies a transport protocol. The distinguish trait of the structure is given by the sided-node edges labeled with "*" (asterisk), that store the aggregation of information contained in some node's children. For example, if each terminal T_i counts how many packets were received, then node T3 stores the sum of counts of T1 and T2, which is the count of all packets received in coordinate1, despite of the protocol. Node T9, on the other hand, stores the sum of counts of all packets received in all coordinates.

B. Schema

The actual composition of the Tinycubes structure is determined by a JSON schema file, as detailed in Listing 1. From that schema, a Tinycube structure is generated similar to the one in Fig. 2. The JSON record section enumerates the information available to be either stored as dimension or summarized in terminal nodes. The dimensions section defines the datacube dimensions, used as path to reach the

```
{
   "record": {
      "fields": [
         { "id":"seconds", "type": "int" },
         { "id":"lat", "type": "double" },
         { "id":"lon", "type": "double" },
         { "id":"proto", "type": "int" },
         { "id":"sport", "type": "int" },
         { "id":"dport", "type": "int" },
         { "id":"ipackets", "type": "int" },
         { "id":"ibytes", "type": "int" }
      ]
   },
   "dimensions": [
      { "id": "location", "length": 25,
           "class": [ "geo", "lat", "lon" ] },
      { "id": "proto",
           "class": ["cat", "proto"]}
   ],
   "terminal": {
      "contents": [
         { "id": "hours",
           "class": ["binlist",
              "seconds", "1" ],
           "contents": [
              {"id": "hc",
                 "formula":["counter"] },
              {"id": "havg",
                 "formula":["avg", "ibytes"] },
              {"id": "hsd",
                 "formula":["sd", "ibytes"] }
           ]
         }
      ]
   },
}
```

Listing 1: Schema file for Figure 2

summarized data stored on terminals. Finally, the terminal field defines which summarized information is stored for future retrieving using queries. In the example, the geographical dimension location is created using the class geo, which uses "lat" and "lon" record fields, and uses the "length" field to define the geographic precision of the location (in this case, approximately $\frac{1}{2^{25}}$ of Earth circumference, around $1,3$m). Besides, the contents of field "hours" in terminal section indicates that each terminal node has a 1-second resolution binned list of a triple formed by an event counter "hc", the average and the standard deviation of "ibytes" record field.

C. A.I. Analysis - Machine Learning Algorithms

The current implementation of the tool uses three algorithms for anomaly detection (detailed in Table I) and one algorithm for regression and prediction of values (detailed in Table II), all of them available in Microsoft ML.NET framework. The tool interacts with these algorithms sending and receiving data files computed by ML.NET routines.

TABLE I
ANOMALY DETECTION ALGORITHMS

Algorithm	Short Description
DetectIidSpike	predicts spikes in independent identically distributed (i.i.d.) [6] time series based on adaptive kernel density estimations and martingale scores.
DetectSpikeBySsa	predicts spikes in time series using Singular Spectrum Analysis (SSA) [7].
DetectAnomalyBySrCnn	detects time series' anomalies using SR-CNN algorithm [8].

TABLE II
REGRESSION/PREDICTION ALGORITHMS

Algorithm	Description
Predict	linear regression algorithm [9]

The anomaly detection algorithms take as input a list of values and generate, as outcome, a list with the position of the values that preceded each anomaly detected. The prediction algorithm takes as input a list of pairs (time, value being analyzed), ordered by time, and the value of the time to be predicted. The result is the value predicted itself.

D. Query Processor

The tool uses a JSON based language that emulates a simplified SQL syntax to retrieve data, as can be seen in Listing 2. It offers the traditional "select", "where" and "group by" statements to, respectively, select, filter and group/fold data. The language also offers the command "ml.net" to request some A.I. analysis over the outcome data retrieved by the previous statements. In the example, the query initially requests a time based histogram (group-by over hours) of the "hsd" values ("select": "hsd") with events occurred in a specific time interval ("where" clause) and subsequently submits the outcome as a time series to the ML.NET anomaly detection algorithm DetectSpikeBySsa.

```
{
    "select": "hsd",
    "where":[["hours", "between",
                1561935756, 1561982738]],
    "group-by": "hours",
    "ml.net": [ "DetectSpikeBySsa", "hsd",
                  "98 12 10 3" ]
}
```

Listing 2: Query example

E. User Interface

The Figure 3 illustrates how the collected geo-temporal data can be displayed by the tool. The image reflects five minutes of RNP backbone network traffic in Brazil. The data being displayed corresponds to the amount of input bytes (ibytes) that each POP collects with 1-second resolution. On the temporal chart, the tool summarizes, for each second, the value of ibytes collected in all POPs being displayed. On the map, the tool summarizes, for each POP location, the value

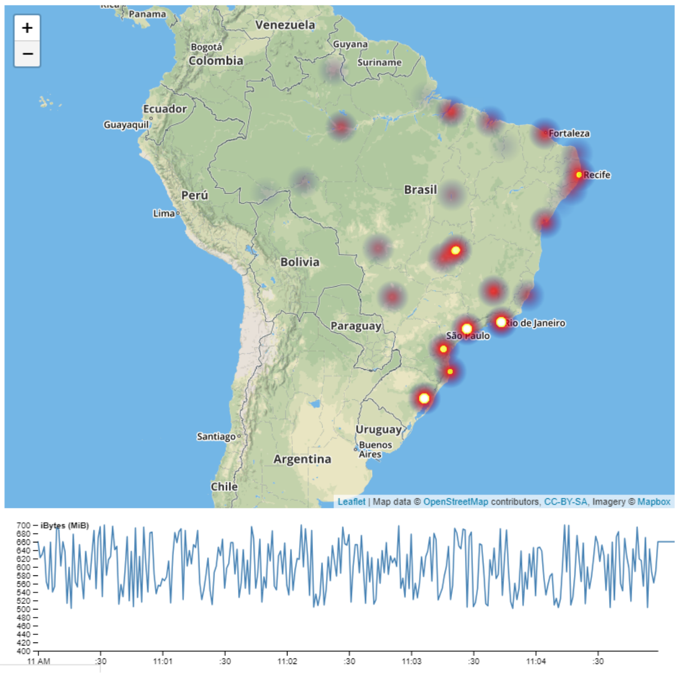

Fig. 3. Heatmap for input bytes (ibytes)

of ibytes collected by the POP in the period of time selected. After that, the total amount calculated is shown over the map as colored spot forming a heatmap.

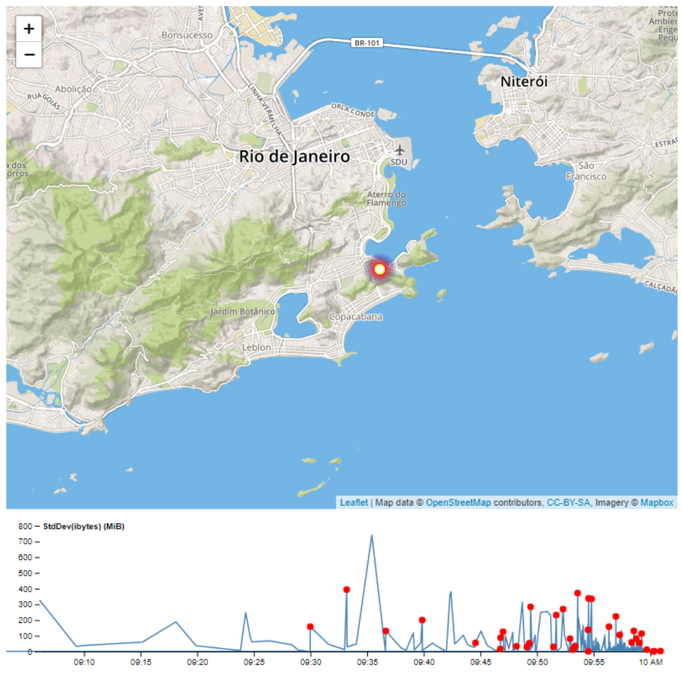

Fig. 4. Anomalies (spikes) detected

III. CASE STUDY

The tool has been used with data collected by RNP routers from POP-RJ. The fields listed on Listing 1 were extract from the actual Netflow files and written in a CSV format using the "nfdump" tool. The coordinates of the POP-RJ were added into data for geographic reference.

Figure 4 shows the visual result of applying the query in Listing 2 when the tool is loaded with POP-RJ data. The "ml.net" field of the query requires that the DetectSpikeBySsa algorithm must be applied over the raw query outcome. The result of this class of anomaly detection algorithm is a series of points whose are plotted as markers over the chart.

IV. Conclusion and Further works

In this paper, we presented a tool with the capability to explore different machine learning algorithms over geo-temporal data collected from network backbone operation. The ability of acting as a statistic fast in-memory database, allows the centralization of real-time collect data that can be visually explored and analyzed with A.I. techniques.

The tool offers many scientific and technical opportunities for future developments. One the scientific aspect, it is possible to explorer different existent A.I. algorithms. On technical side, it is possible to optimize the communication between the Tinycubes implementation (written in C) and the ML.NET framework (DOTNET domain), replacing the file-based interface by a faster memory-based interface. It is also possible to develop a more human-friendly query language that allow users to submit manually written queries via consoles. Another possible experiment would be the use the actual location of the communication peers obtained by GEO-IP service, instead of POPs location, what could allow the investigation of how regions of the world impact the network traffic.

Acknowledgment

This research was partially supported by the Rede Nacional de Pesquisa (RNP) through the project "Microsoft's Artificial Intelligence Challenge". We also thank the RNP team who provided monitoring data collected from the network backbone.

References

[1] "Ml.net documentation - tutorials, api reference — microsoft docs," https://docs.microsoft.com/en-us/dotnet/machine-learning/, 2019.

[2] L. Lins, J. T. Klosowski, and C. Scheildegger, "Nanocubes for real-time exploration of spatiotemporal datasets," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, 2013.

[3] J. Gray, A. Bosworth, A. Layman, and H. Pirehesh, "Data cube: A relational agregation operator generalizing group-by, cross-tab, and sub-totals," *Data Mining and Knowledge Discovery*, vol. 1, pp. 29–53, 1997.

[4] D. B. West, *Introduction to Graph Theory*. Pearson Education Inc., 2001.

[5] R. A. Finkel and J. L. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica. Springer-Verlag*, vol. 4, pp. 1–9, 1974.

[6] T. J. A. Cover, T. M., *Elements Of Information Theory*. Wiley-Interscience., 2006.

[7] V. N. Golyandina, N. and A. Zhigljavsky, *Analysis of Time Series Structure: SSA and related techniques*. Chapman and Hall/CRC, 2001.

[8] X. K. H. S. L. Y. Y. Xie, F. Xing, "Be-yond classification: Structured regression for robust cell detection using convolutional neural network," *Medical Image Computing and Computer-Assisted Intervention - MICCAI*, pp. pp. 358–365, 2015.

[9] S. H. Draper, N.R., *Applied Regression Analysis*, 3rd ed. John Wiley, 1998.