Cloud Metrics Visualization

Vir Ratanasangpunth, and Catharina Vijay

Abstract—Metrics are any time-series numeric values that servers in production emit. The Metrics data is big data; thousands of servers emit constant stream of multiple variables they are collecting. Current Cloud Metrics Visualizations do not employ novel Information Visualization techniques. In this paper we will investigate on improving current visualizations’ interactivity using D3.js library.

Index Terms—Cloud Monitoring, Metrics, Information Visualization

INTRODUCTION

Servers in production typically emit a large amount of time-series numeric values. We call these values “Metrics”. Examples of these Metrics include: the amount of pending jobs in a queue, the rate of incoming requests, and the count of 500 HTTP responses we send back.

Current visualizations of these Metrics rely heavily on the traditional 2D plots, i.e. time is encoded along the x-axis while the Metric values are displayed on the y-axis. Displaying multiple Metrics among multiple servers typically result in the following visualization:

The Cloud Metrics data serve two important roles: I) to inform developers when the servers are experiencing an outage, and II) to enable developers to obtain insights about the characteristics of their cloud.

1 TARGET AUDIENCE

Our target audience are System administrators, DevOps, and software engineers. Anyone that deals with server metrics and administering the health of can use the visualization.

2 PREVIOUS WORK

2.1 Graphite Frontend

Graphite is a distributed systems solution for real-time graphing, specifically made to handle numeric time-series data.

While Graphite is an industry-standard, the Graphite dashboard leaves a little to be desired. The dashboard will serve as our baseline implementation; our visualization should have all the functionality that the dashboard has, while improve on the existing dashboard’s interactivity and visual aesthetics.

Fig. 1. Current visualization displaying multiple Metrics from multiple servers over time.

The Cloud Metrics data serve two important roles: I) to inform developers when the servers are experiencing an outage, and II) to enable developers to obtain insights about the characteristics of their cloud.

2.2 Cubism

Cubism is an alternative front-end to Graphite’s dashboard. Cubism is made by Square, a hot new startup in mobile payments, and written using d3.js.

Cubism is a significant improvement over Graphite’s dashboard. The color scheme is well chosen. It is a good alternative for display many variables since it uses the Horizon Charts visualization technique: larger values are overplotted in successively darker colors, while negative values descend from the top.

Fig. 2. Cubism.js

3 QUESTIONS

Our users are mainly system administrators and software developers. The users’ objectives generally fall into two semi-related but orthogonal categories: I) to be aware at all time the health of their servers, and II) to explore and deduce the patterns of the collected Metrics. These two concerns are translated into two high-level questions of interest:

Q1) Are my servers performing as expected right now?
Q2) What are the patterns and/or characteristics of the Metrics?
We can break down those high-level questions into more detailed questions:

Q1) Are my servers performing as expected right now?
   Q1.1) Which servers (their IP addresses) are down right now?
   Q1.2) In what region do those servers belong to?
   Q1.3) How many servers are affected by the outage?
   Q1.4) What services/programs are running on those downed servers?

Q2) What are the patterns and/or characteristics of the Metrics?
   Q2.1) How many requests are there in a queue over time for a particular server?
   Q2.2) How much memory is a service/program consuming over time for a particular server?
   Q2.3) At what date & time did we achieve Metric maximum/minimum?

Retrieve the amount of memory that one program is consuming on one particular server.

What are the differences in the servers’ metrics between this week and last week? -> Retrieve, Compute derived value, Compare
Retrieve the server’s metrics from last week, compute the average and then compare it to the average from this week.

What are the differences in the servers’ metrics between the East coast and the West coast servers? -> Retrieve, Compute derived value, Compare
Retrieve the metrics from all the servers on the West coast, and do the same for the servers on the East coast. Compute the average of the server metrics for each individual coast and then compare the West coast average to the East coast average.

How many 500 HTTP status codes are returned to the clients over time? -> Compute derived value
Derive the number of 500 HTTP status codes are returned to the clients and then compute this value over time.

What specific program is running on one particular server? -> Retrieve
Retrieve the specific program is running on one particular server.

At what time/date did we achieve metric maximum/minimum? -> Retrieve
Retrieve the time-stamp of when we achieved a metric maximum/minimum.

What are the metrics for some period of time for one particular server? -> Retrieve, compute derived values, compare
Retrieve the metrics for a server and compute the distribution of the values over time. The metrics can also be compared with each other.

4 Abstract Operations

For the following questions we have the following abstract operations:

Which servers are down right now? -> Sort
Sort the servers that are up and working from the servers that are down.

What are the servers that are performing poorly? -> Find anomalies
Find the anomaly servers that are performing poorly.

What is the statistical distribution of how long it takes for task completion for one particular server? -> Derive, characterize distribution
Derive the statistical distribution of how long it takes for task completion for one particular server.

How many jobs are in a queue over time for one particular server? -> Derive, Retrieve
Derive (add) and retrieve the number of jobs are in a queue over time for one particular server.

What is the rate of the incoming requests over time for one particular server? -> compute derived value.
Derive the number of requests for one particular server and compute the rate of the requests.

How much memory is one program consuming on one particular server? -> Retrieve
Retrieve the amount of memory that one program is consuming on one particular server.

5 Encoding

Question I) and II) are equally important. However they require two separate views since the first question demands a top-level view while the second question demands a more selective/local view of the system.

We decide that our visualization needs interactivity so users can easily switch between the two views. The first view will address Q1), which is an overall view of the system. For this view we have the following abstract data types:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>servers</td>
<td>categorical</td>
<td>position</td>
</tr>
<tr>
<td>health</td>
<td>categorical</td>
<td>color hue</td>
</tr>
</tbody>
</table>

We choose to represent each server with a box. The box’s position can encode which “group” it belongs to, e.g. those servers within the same geographical area might be grouped together by positions. Users can configure the grouping of those servers.
We choose position encoding for the servers since it is first on the Mackinlay’s ranking. We determine that it is very important to represent each and every server with its own unique encoding. This is so that those servers that are experiencing an outage can be uniquely identified immediately.

The second most important variable is the health of those servers. For this we choose the color hue encoding since it is second on the Mackinlay’s ranking.

The Metrics values are the most important variable in our second visualization. We choose position encoding since it is first on Mackinlay’s ranking for quantitative type. Those values will be encoded in the y-position. For the x-position, we will encode time. We choose to display different Metrics with different color. Color encoding is second on the Mackinlay’s ranking for categorical type.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric values</td>
<td>quantitative</td>
<td>position</td>
</tr>
<tr>
<td>metrics</td>
<td>categorical</td>
<td>color hue</td>
</tr>
<tr>
<td>time</td>
<td>ordinal</td>
<td>position</td>
</tr>
</tbody>
</table>