Arista – Benchmark Dashboard

Design Document

Author: Bijan Mishra (Happiest Minds Technologies)

Bijan.mishra@happiestminds.com

Contents

[Change Log 3](#_Toc453164193)

[Current System: 4](#_Toc453164194)

[High level Design Architect: 4](#_Toc453164195)

[Low level Design split : 5](#_Toc453164196)

[Query Log: 7](#_Toc453164197)

# Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SR | Description | Date | Change by | Remarks |
| 1 | Draft | 08-06-2016 | Bijan Mishra | Creation of initial draft |
|  |  |  |  |  |

# Current System:

URL: <http://benchmark>

Script: /src/Artest/www/ArtestCgi.py

Database Name: benchmark

Tables used:

1) Table name: Benchmark

Column names: id (varchar), description (text)

2) Table name: run

Column names: benchmark (varchar), result (double), dut (varchar), project (varchar), release (varchar), client (varchar), changeNum (integer), testTime (timestamp)

Description: Bench marking of the parameters how they behaving.

# High level Design Architect:

MySQL

Node js

Client

Node Scripts

SQLite

# 

# Low level Design split :

**Step 1: Node js to MySQL Db**

* Once The code is deployed on the server, One node js script will query to MySQL DB .
* MySQL DB Details :
  + - Host : benchmark.aristanetworks.com
    - Database : benchmark, run
    - User : arastra
    - Port : 3306
* Table Details:
  + - Table name: benchmark
      * Column names: id (varchar), description (text)
    - Table name: run
      * Column names: benchmark (varchar), result (double), dut (varchar), project (varchar), release (varchar), client (varchar), changeNum (integer), testTime (timestamp)
* Firstly it will query for “Run” table to get the benchmark names which are active last year.
* Once it got the successful response from the DB. Script will create a SQLite DB instance in the local server and will create a “benchmark\_local” and “run\_local” table in the SQLite DB.
* All the response for first query will be pushed to “benchmark\_local” table.
* Secondly, it will again query for “run” table. It will fetch the last/latest 50 entries of those benchmarks, which have uploaded any data in the past year.
* Once the fetch is successful, it pushes the records to “run\_local” table.

**Step 2 and 3: update “run\_local” table in a time interval**

* Once we got all the required MySQL db values into the local SQLite, Another script will query for the new records that are added in the MySQL db.
* The script will run as a cron job at a certain time of interval.
* After the successful fetch of new records from MySQL Db, the data will be pushed/replaced into the “run\_local” table in local SQLite.

**Step 4 and 5: UI on load**

* Once the UI is loaded, an Ajax call will be fired to the node js server to get the the json data for the drop down values.
* When node js gets this request, it will query and fetch all the data from “benchmark\_local” table in local SQLite db.
* Once the data fetch is done, it will give the response to the UI.
* When UI will get the response, it will manipulate the response data in the frontend as required.
* After the manipulation, the drop down values will appear in the drop down.

**Step 6 and 7: Drop down value selection**

* Once Id of the benchmark form the drop down values are selected, another Ajax call will be fired to node js server.
* Only the id of the benchmark will be passed to the Ajax call.
* Once the node js get the request, it will get the id of the benchmark from the request.
* Using that id, one query will be fired to get all the relevant data for that benchmark id from the “run\_local” table in the local SQLite db.
* Once the data fetched successfully, it will be converted in to json and response will be given back to the UI.
* Once the UI get the json response, it will be passed to a JavaScript function to plot the graphs for the drop down selection.

**Step 8: Selection in the Filter**

* By default, user will see data for latest 50 entries. User can select other than 50 entries in the data value filter [20, 50,100, 200…], if he/she wants to display less or more data of the selected id in the graph.

**Step 9: UI to nodeJs server**

* When one of the option is selected more than 50 in the filter, An Ajax call with the selected id of the benchmark and selected number, will be fired to node js server.

**Step 10: nodeJs to MYSQL DB**

* The node js fires a query to the MYSQL database to fetch those number of records selected in the filter, for particular benchmark id from Run table.

**Step 11: nodejs to UI**

* Once the data fetched successfully, it will be converted to json and response will be given back to the UI.
* Once the UI gets the json response, it will be passed to a JavaScript function to plot the graph for the drop down selection for those many number of records selected in the filter.

**Step 12: Refresh button**

* When the refresh button clicked on the UI, both the Ajax call will be fired to get the data for the drop down and the data for the selected id.
* Once the UI get the response, it will update the dropdown values as well as the chart values. But the selection won’t change**.**

# Query Log:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SR | Query | Raised By &  Date Raised | Response | Remarks |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |