Distributed Automatic Parameter Testing

Ben Duggan

February 10, 2019

1 Preface

This code is designed to provide a common tool chain which gets a parameter set, runs that tool chain and uploads the results. The system is currently designed to grab the next parameter set from Google Sheets, run the tool chain, upload the results to Box and repeat. The code can be expanded beyond this or not perform some of these steps, this is just how it was designed to be used.

2 Setup

2.1 Python dependencies

The python code uses gspread, oauth2client, flask and boxsdk. You can install them using ‘pip install gspread oauth2client flask boxsdk boxsdk[jwt]==2.0.0‘.

2.2 config.txt

This is the configuration file for the main file. It consists of key value pairs separated by a colon. All fields are optional but you need several to get any functionality. Here are the keys and what they are for:

- userName this is the user name of the system. It is not required but gets saved in the Google Sheet and allows for runs to be traced back to a particular system. It's default value is "default".
- resetTime is the time in seconds at which a parameter set will be added as available if the status hasn't changed from "in progress" to "finished". This is to ensure that if the computer shuts down or an error occurs the parameter set will still be run. The default value is "172800" or 2 days.
- numOfRuns is the number of trials you want main to run. If you set it to -1 it will run until there are no more parameters to run. The default value is "-1".
- lastTest is used by main.py to keep track of if a test didn't finish correctly. If it did not then it will try this set of parameters the next time. Its default value is "None".

  lastTest:None userName:None spreedsheetID:None client_id:None client_secret:None boxFolderID:None removeZip:None resetTime:None numOfRuns:None computerStrength:None accessToken:None refressToken:None

2.3 Google Sheet

2.3.1 Create credentials

Using the https://developers.google.com/sheets/api/quickstart/pythonGoogle Sheets API Python Startup Guide and https://www.twilio.com/blog/2017/02/an-easy-way-to-read-and-write-to-a-google-spreadsheet-in-python.html?utm$_{s}ource = youtubeutm_{m}edium = videoutm_{c}ampaign = youtube_{p}ython_{g}oogle$

Follow the following steps:

1. Ensure that the dependencies are installed.

2. Create the credentials (this is only needed to create a new set of parameters).
   (a) Go to https://console.developers.google.com/https://console.developers.google.com/.
   (b) Create a new project.
   (c) Click Enable API. Search for and enable the Google Drive API and the Google Sheets API.
   (d) Create credentials for service account key. Create a service account and select a Key type of JSON.
   (e) Name the service account and grant it a Project Role of Editor.
   (f) Download the JSON file.
   (g) Copy the JSON file into DistParam.

3. Go to the spreadsheet you wish to use, copy spreedsheet ID (The ID in https://docs.google.com/spreads is **17QJFFXto0MbOX5dH9GFP3NevNHiuxj7eZf7Pevcg96U**) and set it equal to spreedsheetID in config.txt.

2.3.2 How to use Sheet

The Google Sheet can be have any name you wish. The Google Sheet is used like a database with different sheets (tabs) acting as table, the first row of a sheet being the header and the following rows being the entries. You are only required to have one column which contains "id" and is used to reference specific parameter sets. There are several reserved column names which can be used and will be referenced or populated automatically. They are shown below:

- id: the trial id and must be unique.
- status: the current status of the parameter set. It can be empty meaning not started, in progress or finished.
- startTime
- endTime

- performedBy: the computer that performed the computation as defined by config.txt
- comment: any comments either added by the person entering values or main.py
- computerStrength: if the computer strength value for a parameter set is greater than the computer strength setting in the config.txt then the set won't be ran on that computer.

You can add any other columns you wish and can access them in the parameter set returned by dap.param.Param.requestParameters().

2.4 Box

There are two ways to access the Box API: using Oauth 2.0 or Oauth 2.0 with Java Web Tokens (JWT). Currently only the former is implemented as you need admin access to implement JWT, which individuals don't have if they are using an institution run Box service. The Oauth 2.0 process works by having an application pass a client id and secret to the server, letting the user login to Box using their credentials and then Box will give an access and refresh token. The access token is valid for 60 minutes which the refresh token, which gets a new access and refresh token, is good for 60 days. This means that once you get an access and refresh token you don't have to do it again for 60 days.

2.4.1 Create credentials

2.4.2 How to use Box

To implement this the code must get the access and refresh token from Box. Once you setup the API and have your client id and secret added to config.txt and you run ...., the code will start a Flask server. Open a web browser and go to http://127.0.0.1:5000. Click the link and you will then be asked to enter your Box username and password. This step is secure and done using Box's services. Once you click authorize the code will get the access and refresh token, save them to config.txt (assuming the keys are present), close the Flask server and return to the code. If you are on a server you will need to run the above code on a machine with a display to get an access and refresh token (displayed in your browser or seen in the config file) and then manually enter it in the config file, unless you have access to a display with a browser. The code can now use Box!

3 Usage

To run you must simply type python main.py

3.1 Command line arguments

4 How it works

When you run sheet.py you enter a while loop that breaks when either there are no more parameters or you have reached the maximum number of parameters to run as defined in config.txt. At the beginning of the while loop block code checks to see if the DB has errors. Currently this is just checking to see if the code has been running for more than the time given in config.txt. Next the code fetches the next set of parameters to use.

5 DAP Documentation

This section will go over each file of the DAP module and what they do.

5.1 __init__

This file contains all code to setup the module and also contains the code to parse the command line arguments.

5.2 box

5.3 config

5.4 csv

5.5 param

5.6 sheet

5.7 tools