Make and gnuplot

Objective: You want to create an automated build, test and reporting system for a multi-threaded application. Consider there is a setup that contains application source which is compiled to build the application, configuration scripts to generate output (logs) using the built application, analysis phase to analyse the log and produce processed input for plotting graphs and a report written in latex which presents the plots as an executive summary. This is to be done using GNU Make and gnuplot utilities and performing minimal required operations when any of the relevant files change.

Setup

C source files **prog.c thread_function.c common.h** which are used to build the application (say App). The application takes two inputs: (1) number of elements is specified in param.txt
and, (2) number of threads in threads.txt This application is similar to class example for pthreads where time taken is printed as the output of the program.

Configuration files: **threads.txt, params.txt**

threads.txt ==> Specifies number of threads the built program should be executed with
params.txt ==> num_of_elements: Specifies number of elements to be passed as parameter

Steps

1. Building the application from source
2. Execute tests and generate reports as follows,
   Your statistics collections should perform the following steps (you can write this using bash or python etc.),

```
for each <num_elements> in params.txt
   for each <num_threads> in threads.txt
            execute App 100 times with <num_elements> <num_threads>
            and collect results (by redirecting the op)
```

3. Analyse generated logs and plot the results
   You are required to analyse the results to produce processed data to plot the following using gnuplot (using scripts for each),
   a. One point (scatter) graph for each thread configuration with X-axes is <num-of-elements> and Y-axes points correspond to execution time for each sample.
   b. One line graph for each thread configuration where X-axes is <num-of-elements> and Y-axes is average execution time over 100 samples
   c. One bar graph with X-axes points as <number of elements> and Y-axes as average speedup (as you have executed the same configuration 100 times) w.r.t. one thread execution. Plot bars representing <num of threads> for every point in X-axes (see the app_speedup.eps plot from the class examples).
   d. Plot the same graph as (3) but with error bars. Where error bars represent the variance calculated over 100 samples for a particular configuration.

4. A latex document that embeds the above plots to produce a pdf report. Note that each of this plot should be embedded as a separate figure with labels and some explanation of results (MAX 3 sentences).

Notes:

1) Assume threads.txt has at least one entry i.e., 1 (number of threads = 1)
2) You are **not allowed** to change the C program
3) Your Makefile should support the following,
    a. Build the latex report, which eventually required to perform all the steps (if required)
    b. Build each step separately (like $make App, $make report etc.)
    c. Incremental build, Example: if .tex file is modified only reporting step should be performed, if one single gnuplot file changes, only that plot should be recreated