DGPAPP Tutorial

Daniel Centeno Einloft
Vinicius Meirelles Pereira

20 May 2014

Preamble

In parallel programming, the main goal is to decrease the execution time of an algorithm through its division into separate flows of execution, that will then be run by several processors or cores. There are many ways to test if the parallel algorithm is better than the sequential one, and they all depend on their times of execution. Since each test must be executed at least 30 times, in order to obtain an acceptable confidence interval, huge amounts of data are generated.

The Data Generator for Performance Analysis of Parallel Programs (DG-PAPP) is a free software designed to collect the data from performance tests, organize it, and perform several calculations. After that, the program saves the results and the organized data into two different files.

Before DGPAPP was created, in order to obtain the desired results, it was necessary to collect and filter all the data manually for each test. Later, the average time of each experiment would be calculated, and with it, speedup and efficiency were estimated. These results still needed to be organized in a way that the selected tool for generating graphics could use it.

Calculations

In its current version, DGPAPP is capable of calculating speedup and efficiency, aside from data collection and organization. We intend to add new features in the future, such as variance, standard deviation and other statistical data.

Speedup

Speedup measures how fast a parallel algorithm is, when compared with its sequential version. It is defined by:

$$S_t = \frac{T_1}{T_t}, \tag{1}$$

where:
$_t$ is the number of threads or processes;
S_t is the speedup;
T_1 is the time taken by the sequential algorithm;
T_t is the time taken by the parallel algorithm.

The ideal speedup is obtained when $S_t = t$.

As you can see, the speedup only depends on the average times of the sequential algorithm and those of the parallel one.

Efficiency

The Efficiency of a parallel algorithm shows how well used were the cores or processors when executing it. This value is helpful because the speedup can only measure the speed of an algorithm, without taking into account the number of processors or cores used. Therefore, if an algorithm runs twice as fast as the sequential one, but is being divided into 8 threads, its speedup is high, but its efficiency is not, and it probably is not a good option.

The efficiency can be obtained through the following equation:

$$E_t = \frac{S_t}{t} = \frac{T_1}{tT_t}, \tag{2}$$

where:
$_t$ is the number of threads or processes;
E_t is the efficiency;
S_t is the speedup;
T_1 is the time taken by the sequential algorithm;
T_t is the time taken by the parallel algorithm.

Installation

How to Install

To install DGPAPP, open the folder where it is, and run the command:

```
$ sudo make
```

Then, just execute:

```
$ sudo make install
```

After that, DGPAPP will have been installed to /usr/local/DGPAPP/.

How to Uninstall

If you wish to uninstall the program, while in the DGPAPP folder, you should run the command:

```
$ sudo make uninstall
```

How to Use

List of Commands

- **-h or –help** : displays help information.
- **-v or –version** : displays the program's version.

- **-f** : specifies the filter.
- **-l** : specifies the name of the log.
- **-t** : specifies the thread number of the current log file.
- **-c** : specifies the that will be displayed at the bottom of the data column.
- **-o** : specifies the output file name.
- **-p** : specifies the number of decimal places in the output files.

Using in the Command Line

The commands can be inserted in any order, as long as after a log file, comes a number of threads, since this number of threads will be associated with that log.

An example of the syntax is:

```
$ dgpapp -f filter -l log -t number_of_threads -l log -t number_of_threads
-l ... -c comment -o output_filename -p decimal_places
```

The filter is the word that DGPAPP will look for in the log files. It will save the value next to it in the logs as one of the execution times of the test, and will look for all occurrences of this word in the file. The log is the name of the log file, which must be succeeded by its number of threads. Any amount of logs can be inserted. The comment is the name that will be added to the end of each column, used to identify the test. Since the columns might have to be reorganized, in order for them to have decreasing size, from left to right, it is very important, although not necessary, to add this command. The file name is a generic name for all the files that will be generated. Their names will contain the name chosen, followed by the name of the variable calculated in that file. The number of decimal places is not a necessary command, and if it has not been informed, the program will use its standard value, which is 2. The order of the commands is irrelevant.

The log files, as well as the output name, can have their paths included (e.g., -l /home/log.txt, or -o /home/testname). The filter can be a string. In that case, it should be written between quotation marks.

Common Mistakes

- Number of threads not separated from the "-t" by a space character: returns an error message and the program is terminated.
- No sequential log: returns an error message and the program is terminated.
- Not using the comment: it's not a mistake, but could result in confusion while identifying the tests.

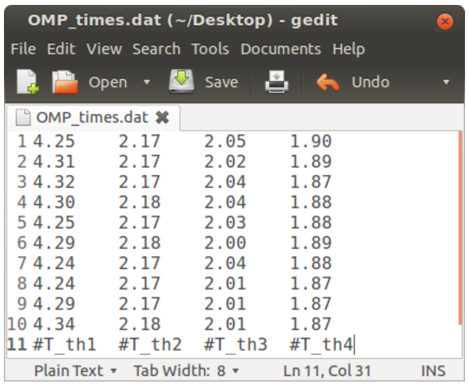
Output Files

The DGPAPP generates three output files: efficiency.dat, speedup.dat and averagetime.dat. As the file names indicate, they contain information about the efficiency, speedup and average time of the tests, respectively. The structure of these files, however, needs to be explained with more detail.

The first time the user executes the program through the command line in the linux terminal, it will generate the files and organize the data in two columns: the first is the thread column and the second is the data column. This does not apply to the times.dat output file. There, each column represents the times using a certain number of threads. The last line of each column in all the output files is the comment line, which is indicated by a `"#"`. The times.dat output file also includes the number of threads in the comment.

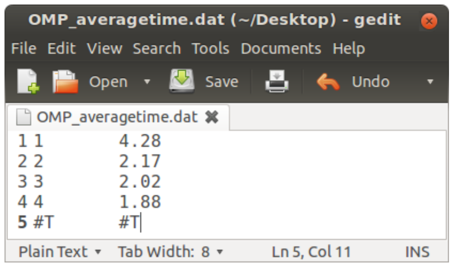
When the user executes the program again, it will generate two more columns. However, the new columns will not necessarily be located to the right of the older columns. Due to the fact that the program was meant to be used with Gnuplot for plotting graphics, the columns must be organized in a way that the two new columns have the same size or less than the next two on their left. Thus, the columns will always be arranged from the largest to smallest size, from left to right.

Below is an example of the output files.


OMP_times.dat (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
OMP_times.dat
Plain Text
Tab Width: 8
Ln 11, Col 31
INS

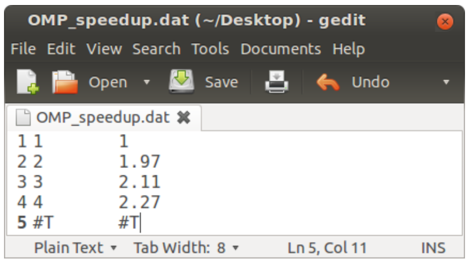
```
4.25    2.17    2.05    1.90
4.31    2.17    2.02    1.89
4.32    2.17    2.04    1.87
4.30    2.18    2.04    1.88
4.25    2.17    2.03    1.88
4.29    2.18    2.00    1.89
4.24    2.17    2.04    1.88
4.24    2.17    2.01    1.87
4.29    2.17    2.01    1.87
4.34    2.18    2.01    1.87
#T_th1  #T_th2  #T_th3  #T_th4
```

Figure 1: Example of times.dat output file


OMP_averagetime.dat (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
OMP_averagetime.dat
Plain Text
Tab Width: 8
Ln 5, Col 11
INS

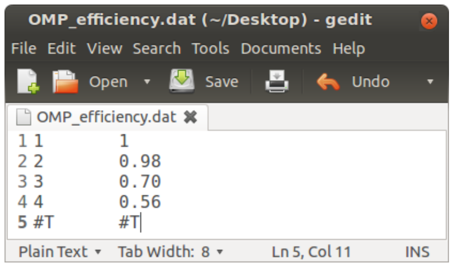
```
1       4.28
2       2.17
3       2.02
4       1.88
#T      #T
```

Figure 2: Example of averagetime.dat output file


OMP_speedup.dat (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
OMP_speedup.dat
Plain Text Tab Width: 8 Ln 5, Col 11 INS

```
1 1     1
2 2     1.97
3 3     2.11
4 4     2.27
5 #T    #T
```

Figure 3: Example of speedup.dat output file


OMP_efficiency.dat (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
OMP_efficiency.dat
Plain Text Tab Width: 8 Ln 5, Col 11 INS

```
1 1     1
2 2     0.98
3 3     0.70
4 4     0.56
5 #T    #T
```

Figure 4: Example of efficiency.dat output file

Note that in the above examples, the standard name used for the files was "OMP", and the comment for the test was "T".