shellPing.go: Performance Study

Authors: Steve Huang, Asher Kang, Maria Ringes

I. Introduction

shellPing.go is an implementation of MP0, which demands a program that uses Go-routines and Go-channels to ping websites in parallel.

shellPing.go iteratively sets increasing values of GOMAXPROCS (GMP) and pings user-inputted websites in parallel using *Go-routines*. During each loop of the GOMAXPROCS value, the total pinging runtime is measured from start to end, with the duration stored in an int-to-float64 map called gmpToRuntime. This map is used to output an HTML graph that plots GOMAXPROCS values against the program runtime.

II. Methodology

A. Research

Our first step in research was understanding ping in general. Ping is a network tool used to measure the time it takes for messages to reach a destination computer and return from it. Typically, it sends data in the ICMP packet format, with a size of (generally) 32 or more bytes. Once the destination, usually a server, receives the packet, it automatically sends a return packet back to the caller, who will receive it and measure the time taken for the entire procedure. If there is not much network latency between the caller and the destination, a ping request should generally take about 10ms or faster. The ping time of a website is generally a good indication of how much time the user is expected to wait when opening and loading a webpage.

Our second step in research was understanding GOMAXPROCS. From the official Go documentation website for the runtime package, we learned that runtime.GOMAXPROCS() sets the maximum number of CPUs that can be executed simultaneously on the user's computer. The GOMAXPROCS value will vary depending on the user's system. It defaults to the value of runtime.NumCPU, which is the number of CPU threads on the system.

B. Data Collection

Our program iteratively called a parallelized ping function as a Go-routine while looping through a for-loop, passing in increasing GOMAXPROCS values from 1 CPU to the maximum CPU value possible, as determined by the user's individual processor. The program runtime for each pinger call, from beginning to end, was calculated and then passed to an int-to-int64 map called gmpToRuntime. This map stored GOMAXPROCS values as int keys and program runtimes as int64 values. It was used to generate an HTML graph that maps GOMAXPROCS values onto program runtimes.

To collect ping statistics, the program first spliced the user input, according to the spaces input by the user. The program then pinged each of the input websites. The ping command, as executed by the shell, returned a standard output which was subsequently spliced into the desired

time values of runtime (in milliseconds) and sent through a predefined Go-channel. These values sent to the channel were stored as field values of a custom dataset of type struct PingResults. These values were then error-checked. Once it was confirmed that the values received from the channel were of statistical value, they were passed into a global splice. If it was determined that the value was of statistical value and the ping was successfully executed, the value was then added to another predefined splice. These splice groupings were then passed through related functions of the imported stats package to determine the min, max, average and standard deviation of each related splice grouping, wherein each website had one determined grouping. The websites and their corresponding min, max, avg, and stddev values, and number of packets sent and total package success rate were then printed on the standard output.

Four identical Microsoft Excel tables were then manually created using this data. Each table contained information on network latency of various categories of websites, with measurements taken at different times in the day. For example, Table 1 stored the data collected from the nine program runs completed at 8am. Table 2 stored the data collected from the nine runs completed at 2pm. The different websites spanned across the following nine categories: foreign, government, media, eCommerce, information, social media, search engines, blogs, and personally developed websites.

Every six hours across a 24 hour cycle, the program was run nine times, each time pinging the websites from one of the nine different website categories. For example, at 8pm, the program was run nine times. The first run took in, as input, the five websites pertaining to the first website category. The second run took in, as input, the five websites pertaining to the second website category. Overall, the program was run one time per category at four points in the day.

After each individual categorical run, the data printed on the standard output line was manually transferred to the corresponding Microsoft Excel tables. Within each table, the mean() Excel function was used to calculate the average latency time across each of the nine website categories. In other words, Table 1 had nine Excel-calculated averages, one per website category. Additionally, Table 2 had nine Excel-calculated averages, and so on. Additionally, within each table, the mean() Excel function was used to calculate the average percent packet success rate.

C. Data Analysis

Using 500 concurrent Go-routines, we measured the effect of GOMAXPROCS on the runtime of these Go-routines by running the program ten times. Therefore, we received ten different numbers for the runtime of each GOMAXPROCS value. The collected data was then plotted using MATLAB and fitted using a custom equation. We tried to fit a simple logarithmic equation at first, given the overall logarithmic shape of our data. However, the best fit curve dipped below 0 at around x = 10. Therefore, we added a linear term to the general form of the fit equation. To obtain a higher r^2 value without risking to overfit, we further added a quadratic term. Finally,

MATLAB helped to define precise coefficient values so we could optimize the confidence level of our best-fit curve. The resulting best-fit curve for our data was:

$$y(x) = -2.47 * 10^6 log(x) - 1.562 * 10^4 x^2 + 5.84 * 10^5 x + 2.539 * 10^6$$

The runtime, y, is in microseconds and x is the value of GOMAXPROCS. The r^2 value of 95.53% indicates a strong effect of GMP value on the runtime.

Afterwards, the average latency time across each website category was summarized and plotted. Additionally, the average percent packet success rate per website category was also summarized and plotted. After all runs were completed, two summary plots were generated as seen below in the Results section. One graph allows users to understand the average network latency across different websites categories at different times of the day. The other allows users to understand the average percent packet success rate across different website categories (the same as those defined in the previous graph) at different times of the day.

D. Tools Used

The "go-stats" package allowed us to compute the min, max, average, and standard deviation of latency of a particular website from all the concurrent ping data collected.

The "e-charts" package allowed us to plot the values defining the comparison between GOMAXPROCS value versus program run time in a scatter plot.

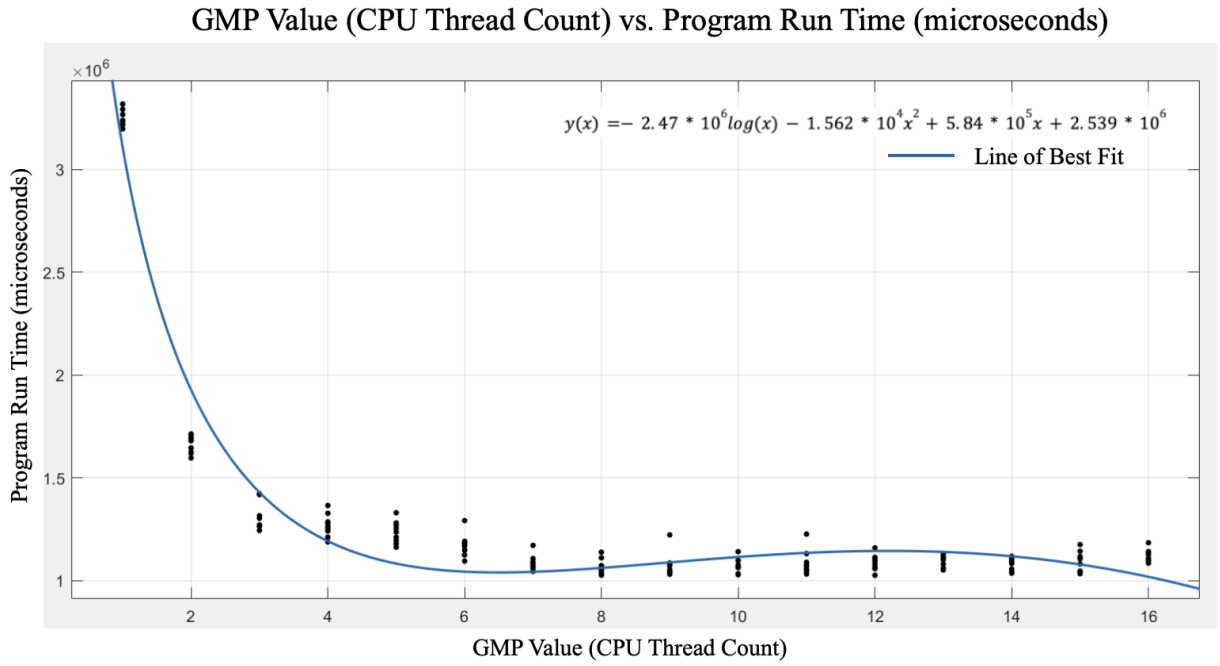
E. Design Decisions

Using 100 Go-routines, we examined three options to implement ping on Go: the packages "go-ping" and "go-FastPing" as well as executing a shell command via Go. Despite its name, "go-Fastping" was the slowest option of the three, taking about 1 second to finish the 100 routines, around five times of the other two options. On the school network, "go-ping" took around 200 milliseconds to finish, whereas shell ping took around 270 milliseconds. However, when placed in a network with lower bandwidth, or if the number of Go-routines was increased, "go-ping" took as much time as shell ping to finish. Moreover, go-ping could return data inconsistent with the actual latency of a website (tested by running ping commands without Go). For example, pinging "google.com" concurrently with "go-ping" could return a latency of 200 milliseconds while the actual latency was around 10.

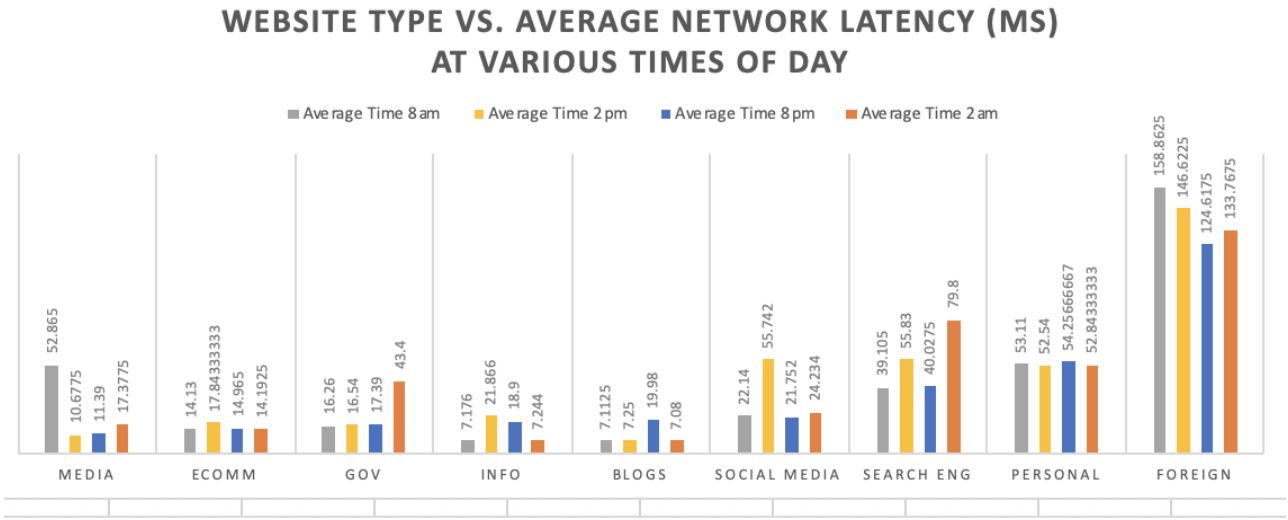
III. Results

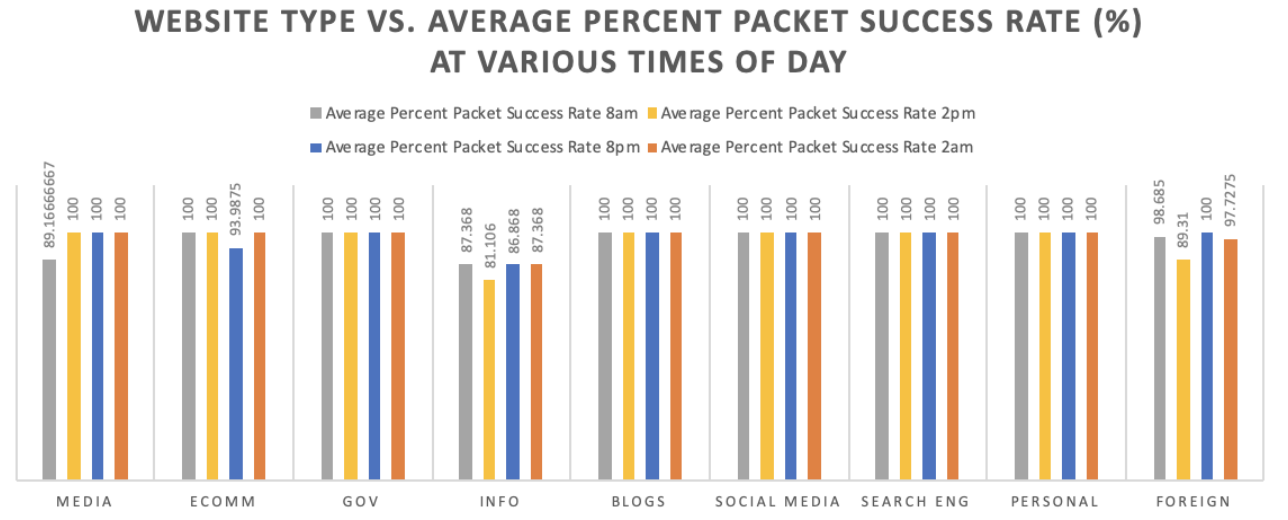
The following graph shows the comparison of GOMAXPROCS value vs. program run time (microseconds). The equation of the graph is again as follows:

$$y(x) = -2.47 * 10^{6} log(x) - 1.562 * 10^{4} x^{2} + 5.84 * 10^{5} x + 2.539 * 10^{6}$$



The following graphs show the comparison of average network latency across different types of websites at different types of the day as well as the average percent packet success rate across the different types of websites at different types of the day. Categorical pings were run in the same order at each interval. Additionally, all pings were run on the ‘eduroam’ network.





IV. Discussion

Our data analysis suggests that at low GOMAXPROCS value (around GMP<4), the runtime of the go-routines shows a logistic decay, but for higher GMP values (>4 and up), GMP value has little to no effect on the runtime.

Our results allow us to conclude that the type of website will affect ping statistics. For example, foreign websites had a higher latency, which would make sense granted their servers are at a further physical distance from the network we tested on.