

Cpts 411 Programming Project 5

Clancy Andrews

Alex Shirk

Introduction

For this project, our goal was to use a multithreaded implementation of the PageRank algorithm to estimate the ranks of nodes on a graph input. The PageRank of a node u in a graph is used when determining the importance of a node throughout the network. In this particular instance, we can think of the nodes as webpages, and the edges between the nodes as hyperlinks to other webpages. It is worth noting that the graph is directed, since an edge can go from node u to node v , but node v need not necessarily have an edge (hyperlink) to node u as well. We can consider the total degree of a node as the sum between the out degree and in degree of each node. The out degree is the number of outgoing edges from the particular node and the in degree is the number of edges coming in.

The PageRank algorithm estimator is as follows. First, we start from every node in the graph. We then do a random walk of length K , keeping track of the number of times a particular node u is visited. The estimated PageRank value for node u is the number of time u was visited divided by the total number of visits in the network. When progressing through a random walk, we can expect that the next visit to a node will either be one of the neighbors of the current node, or a completely random node in the graph. To account for this, the use of a dampening ratio D will be used. It is worth noting that $D \in [0, 1]$. If we toss a coin with D probability of landing on heads and $1 - D$ probability of landing on tails, we can say that if the coin lands on heads, we randomly select a node in the network to continue the walk on. If the coin lands tails, then we randomly select a neighboring node to continue the walk on.

With this logic, our goal was to observe how the algorithm will perform given different dampening ratios, different walk lengths, and different number of threads being used. Since the graphs that are being tested on are sparse, we used an adjacency list to hold the data provided. This allows the program to run faster than the alternative adjacency matrix.

Analysis

The data was collected on a personal computer, which utilizes an Intel i9-10900K CPU at 3.70GHz, containing 10 cores and 20 logical processors (threads).

```
kable(speedup_damp, row.names = TRUE, caption = "Speed up for corresponding $D$ (rows) and $p$ (columns) number of processes.")
```

Table 1: Speed up for corresponding D (rows) dampening ratios and p (columns) number of processes.

	1	2	4	8	16
0.250000	1	1.760130	3.195537	4.665270	6.825224
0.500000	1	1.624037	2.996322	4.417045	6.567925
0.750000	1	1.597379	2.554468	4.210800	5.586382

```
kable(speedup_length, row.names = TRUE, caption = "Speed up for corresponding  $K$  and  $p$ ")
```

Table 2: Speed up for corresponding K (rows) number of steps in the random walk and p (columns) number of processes.

		1	2	4	8	16
10	1	1.806916	3.154034	4.792946	6.638435	
25	1	1.858882	3.186192	4.670256	6.955717	
50	1	1.826806	2.805881	4.493696	7.026313	
100	1	1.797802	3.226833	4.685590	7.314490	
500	1	1.820012	3.271690	4.916508	7.286049	

Analysis Code

The following is the code used for analyzing the collected data from the program:

```
#Libraries
library(ggplot2)
library(knitr)
library(stats)

#Import data for analysis
damp = read.csv2("dampening.csv", header = FALSE, sep = ",")
colnames(damp) = c("Time", "K", "p", "D")

length = read.csv2("length.csv", header=FALSE, sep = ",")
colnames(length) = c("Time", "K", "p", "D")

#Extract Serial Time from Data
serial_time_damp = subset(damp, p == 1)$Time
serial_time_length = subset(length, p == 1)$Time

#Get the parallel times for each P value
parallel_time_damp = data.frame(matrix(ncol = length(unique(damp$p)), nrow = dim(damp)[1]),
colnames(parallel_time_damp) = c("1", "2", "4", "8", "16"))

parallel_time_length = data.frame(matrix(ncol = length(unique(length$p)), nrow = dim(length)[1]),
colnames(parallel_time_length) = c("1", "2", "4", "8", "16"))

for (i in colnames(parallel_time_damp)) {
  p_value = as.numeric(i)
  data = damp$Time[damp$p == p_value]
  parallel_time_damp[, i] = data
}

for (i in colnames(parallel_time_length)) {
  p_value = as.numeric(i)
  data = length$Time[length$p == p_value]
  parallel_time_length[, i] = data
}

#Calculate the speed up
speedup_damp= data.frame(matrix(ncol = length(unique(damp$p)), nrow = 3))
colnames(speedup_damp) = c("1", "2", "4", "8", "16")

for (i in colnames(speedup_damp)) {
  speedup_damp[,i] = (as.numeric(serial_time_damp)/as.numeric(parallel_time_damp[,i]))
}
```

```

row.names(speedup_damp) = unique(damp$D)

speedup_length= data.frame(matrix(ncol = length(unique(length$p)), nrow = 5))
colnames(speedup_length) = c("1", "2", "4", "8", "16")

for (i in colnames(speedup_length)) {
  speedup_length[,i] = (as.numeric(serial_time_length)/as.numeric(parallel_time_len
})

row.names(speedup_length) = unique(length$K)

#Tables for speedup and timings
kable(speedup, row.names = TRUE, caption = "")

kable(pi, row.names = TRUE, caption = "")

kable(pi_diff, row.names = TRUE, caption = "")

```

Session Info

```
sessionInfo()
```

```
## R version 4.3.0 (2023-04-21 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
## locale:
##  [1] LC_COLLATE=English_United States.utf8
##  [2] LC_CTYPE=English_United States.utf8
##  [3] LC_MONETARY=English_United States.utf8
##  [4] LC_NUMERIC=C
##  [5] LC_TIME=English_United States.utf8
##
## time zone: America/Los_Angeles
## tzcode source: internal
##
## attached base packages:
## [1] stats      graphics  grDevices  utils      datasets  methods   base
##
## other attached packages:
## [1] knitr_1.44      ggplot2_3.4.3
##
## loaded via a namespace (and not attached):
##  [1] vctrs_0.6.3      cli_3.6.1        rlang_1.1.1      xfun_0.40
##  [5] glue_1.6.2       colorspace_2.1-0 htmltools_0.5.6  scales_1.2.1
##  [9] fansi_1.0.4      rmarkdown_2.25   grid_4.3.0       evaluate_0.22
## [13] munsell_0.5.0    tibble_3.2.1     fastmap_1.1.1    yaml_2.3.7
## [17] lifecycle_1.0.3  compiler_4.3.0   pkgconfig_2.0.3  digest_0.6.33
## [21] R6_2.5.1         utf8_1.2.3       pillar_1.9.0     magrittr_2.0.3
## [25] tools_4.3.0      withr_2.5.1      gtable_0.3.4
```

D = 0.75,0.5,0.25 p = 16 k = 100

163075 0.000020 537039 0.000017 597621 0.000017 605856 0.000015 819223 0.000014

163075 0.000043 537039 0.000039 597621 0.000039 605856 0.000034 551829 0.000034

597621 0.000078 163075 0.000077 537039 0.000077 41909 0.000070 504140 0.000064

p = 16 k = 100 d = 0.25

597621 0.000078 163075 0.000077 537039 0.000077 41909 0.000070 504140 0.000064