Cpts 411 Programming Project 4

Clancy Andrews Alex Shirk

Introduction

For this project, our goal was to use a multithreaded program to approximate the value of π. Given a unit square with center at $(0.5, 0.5)$, overlay a unit circle in the unit square with origin at $(0.5, 0.5)$. From there, we can randomly select a coordinate inside the unit square (the same idea as throwing a dart at the square). If the coordinates are located inside the unit circle, then we can increment the number of hits by one. If the coordinates are not located inside the unit circle, then we do not increment the number of hits. It is worth noting that the coordinates are always located inside the unit square. Letting n be the number of random coordinates generated and h be the number of those generated coordinates also being inside the unit circle, we can use the following equation to approximate π:

$$\pi = \frac{4h}{n} \tag{1}$$

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

Analysis

For comparing the estimated value of π with the actual value, we used

$$\pi = 3.14159265358979323846264338327950288419 7$$

as the actual value. We got this number from the following webpage: https://www.britannica.com/science/pi-mathematics. Below we can observe the speedup table where the values of n are the row labels and the values of p are the column labels.

Table 1: Speed up for corrsponding n (rows) and p (columns) values.

	1	2	4	8
1024	1	0.2385321	0.1333333	0.1830986
2048	1	0.2897727	0.3090909	0.1360000
4096	1	0.4488189	0.5181818	0.3149171
8192	1	0.3853073	0.3329016	0.2014107
16384	1	0.3291139	0.3506033	0.1972843
32768	1	0.3572349	0.2851268	0.1745332
65536	1	0.2905215	0.2814954	0.1608937
131072	1	0.2146532	0.2670989	0.1529466
262144	1	0.1663780	0.2853384	0.1498652
524288	1	0.1649269	0.1975436	0.1428007
1048576	1	0.1638797	0.2086935	0.1510394

Using our previously presented true value of π to compare, we can observe the estimated values of π approximated by our program in the table below.

Table 2: Estimations for the value of π for corrsponding n (rows) and p (columns) values.

	1	2	4	8	16
1024	3.08203	3.10156	3.07031	3.14453	3.12891
2048	3.11328	3.14648	3.10742	3.08398	3.07617
4096	3.15234	3.16016	3.14160	3.17383	3.16211
8192	3.14307	3.14062	3.13770	3.15381	3.12500
16384	3.14185	3.15088	3.14282	3.13989	3.15649
32768	3.14282	3.13660	3.14172	3.14417	3.13684
65536	3.13977	3.14807	3.14392	3.14520	3.14270
131072	3.14679	3.14087	3.14386	3.14404	3.14023
262144	3.13927	3.13904	3.13815	3.13802	3.13959
524288	3.14467	3.14394	3.14259	3.14350	3.14474
1048576	3.14133	3.14146	3.14038	3.14141	3.14060

From observation, we can see that as n increased in size, our approximations would get closer to the true value of π. The next table shows the difference between the true and estimated π values at each corresponding n and p value.

Table 3: Differences in the true value versus the estimated value of π for corrsponding n (rows) and p (columns) values.

	1	2	4	8	16
1024	0.05956	0.04003	0.07128	0.00294	0.01269
2048	0.02831	0.00489	0.03417	0.05761	0.06542
4096	0.01075	0.01856	0.00001	0.03224	0.02052
8192	0.00147	0.00097	0.00390	0.01222	0.01659
16384	0.00025	0.00929	0.00123	0.00170	0.01490
32768	0.00123	0.00500	0.00013	0.00257	0.00475
65536	0.00182	0.00648	0.00233	0.00361	0.00111
131072	0.00520	0.00072	0.00227	0.00245	0.00136
262144	0.00233	0.00255	0.00344	0.00358	0.00200
524288	0.00308	0.00234	0.00099	0.00191	0.00315
1048576	0.00026	0.00014	0.00122	0.00019	0.00099

It is more evident now that as we increase n and p together, we get a closer approximation to the true value of π.

Analysis Code

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

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

#Import data for analysis
df = read.csv2("output.csv", header = FALSE, sep = ",")
colnames(df) = c("Pi","P", "N", "Difference", "Time")

#Extract Serial Time from Data
serial_time = subset(df, P == 1)$Time

#Get the parallel times for each P value
parallel_time = data.frame(matrix(ncol = length(unique(df$P)) - 1, nrow = 11))
colnames(parallel_time) = c("1", "2", "4", "8")

for (i in colnames(parallel_time)) {
  p_value = as.numeric(i)
  data = df$Time[df$P == p_value]
  parallel_time[, i] = data
}

#Calculate the speed up
speedup= data.frame(matrix(ncol = length(unique(df$P)) - 1, nrow = 11))
colnames(speedup) = c("1", "2", "4", "8")

for (i in colnames(speedup)) {
  speedup[,i] = as.numeric(serial_time)/as.numeric(parallel_time[,i])
}

row.names(speedup) = unique(df$N)

#Precision data
pi = data.frame(matrix(ncol = length(unique(df$P)), nrow = 11))
colnames(pi) = c("1", "2", "4", "8", "16")

for (i in colnames(pi)) {
  p_value = as.numeric(i)
  data = df$Pi[df$P == p_value]
  data = round(as.numeric(data),5)
  pi[, i] = data
}
row.names(pi) = unique(df$N)
```

```
#Difference in the pi values
pi_diff = data.frame(matrix(ncol = length(unique(df$P)), nrow = 11))
colnames(pi_diff) = c("1", "2", "4", "8", "16")

for (i in colnames(pi)) {
  p_value = as.numeric(i)
  data = df$Difference[df$P == p_value]
  data = round(as.numeric(data),5)
  pi_diff[, i] = data
}
row.names(pi_diff) = unique(df$N)

#Tables for speedup and pi_estimates
kable(speedup, row.names = TRUE, caption = "Speed up for corrsponding $n$ (rows) an

kable(pi, row.names = TRUE, caption = "Estimations for the value of $\\pi$ for corr

kable(pi_diff, row.names = TRUE, caption = "Differences in the true value versus th
```

Session Info

```r
sessionInfo()
```

```
## R version 4.3.0 (2023-04-21 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22621)
##
## 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
```