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. The speedup of each n, p pair in the table is relative to the serial runtime of the program.

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

	1	2	4	8
1024	1	0.6923077	0.7627119	0.8653846
2048	1	0.2705882	0.8023256	1.2321429
4096	1	0.2412371	1.0353982	1.0353982
8192	1	0.2007201	1.6518519	1.6043165
16384	1	0.1733439	1.5441696	1.5607143
32768	1	0.2013956	1.3243671	1.7770701
65536	1	0.2540193	1.3021978	1.4707447
131072	1	0.2419999	1.2886399	1.6126692
262144	1	0.2407210	1.1964164	1.6058177
524288	1	0.3152992	1.2087862	1.6426710
1048576	1	0.6300478	1.1928617	1.6330378

We can see that the speedup for the number of threads being 1 and 2 is poor. However, we can see that using 4 or more threads has improved speedup. We can now analyze the precision of estimation of the value of π. 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 corresponding n (rows) and p (columns) values.

	1	2	4	8	16
1024	3.08203	3.14062	3.14062	3.18750	3.06250
2048	3.11328	3.08203	3.14062	3.26562	3.15625
4096	3.15234	3.15039	3.10156	3.01562	3.20312
8192	3.14307	3.12500	3.13281	3.11719	3.11719
16384	3.14185	3.12793	3.15918	3.12695	3.10547
32768	3.14282	3.13501	3.14990	3.13477	3.12695
65536	3.13977	3.13281	3.14526	3.14160	3.14258
131072	3.14679	3.14648	3.13733	3.13989	3.14502
262144	3.13927	3.14340	3.13916	3.13342	3.13184
524288	3.14467	3.14291	3.14316	3.14319	3.14026
1048576	3.14133	3.14269	3.14047	3.14554	3.14374

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 corresponding n (rows) and p (columns) values.

	1	2	4	8	16
1024	0.05956	0.00097	0.00097	0.04591	0.07909
2048	0.02831	0.05956	0.00097	0.12403	0.01466
4096	0.01075	0.00880	0.04003	0.12597	0.06153
8192	0.00147	0.01659	0.00878	0.02440	0.02440
16384	0.00025	0.01366	0.01759	0.01464	0.03612
32768	0.00123	0.00658	0.00831	0.00683	0.01464
65536	0.00182	0.00878	0.00367	0.00001	0.00098
131072	0.00520	0.00489	0.00426	0.00170	0.00343
262144	0.00233	0.00181	0.00243	0.00817	0.00976
524288	0.00308	0.00132	0.00156	0.00160	0.00133
1048576	0.00026	0.00110	0.00112	0.00395	0.00215

It is more evident now that as we increase n and p together, we get a closer approximation to the true value of π. As a result of this analysis, we can expect that the speedup as we increase p will improve and increasing both n and p together will result in a more accurate approximation 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) ar

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
```