**Parallel and Distributed Systems**

159735

Assignment one

π calculation

Wei Han

20015402

Content

[Amdahl’s Law 3](#_Toc47868796)

[My program and data 5](#_Toc47868797)

[Parallel speed-ups 10](#_Toc47868798)

[Interprocess communication times 12](#_Toc47868799)

# Amdahl’s Law

A program which can be parallelized can be split up into two parts:

* A part cannot be parallelized
* Another part can be parallelized

T is called total time to execute program in serial.

The time T includes the time of both the non-parallelizable and parallelizable parts.

The non-parallelizable part is called B (serial part).

The parallizable part is T - B

**T = B + (T-B)**

**(T-B)** which can be sped up by executing it in parallel

(T-B)/N

The number of threads or CPUs is called N, which can speed up the parallelizable part.

T(N) = B + (T - B) / N

If N =1 T=1 then:

T(1) = B + (T(1)-B) which seems like just one single CPU

T(1)=T(1)=1

If N =2 T=1 B=0.6 then:

T(2) = B + (T(2)-B)/2

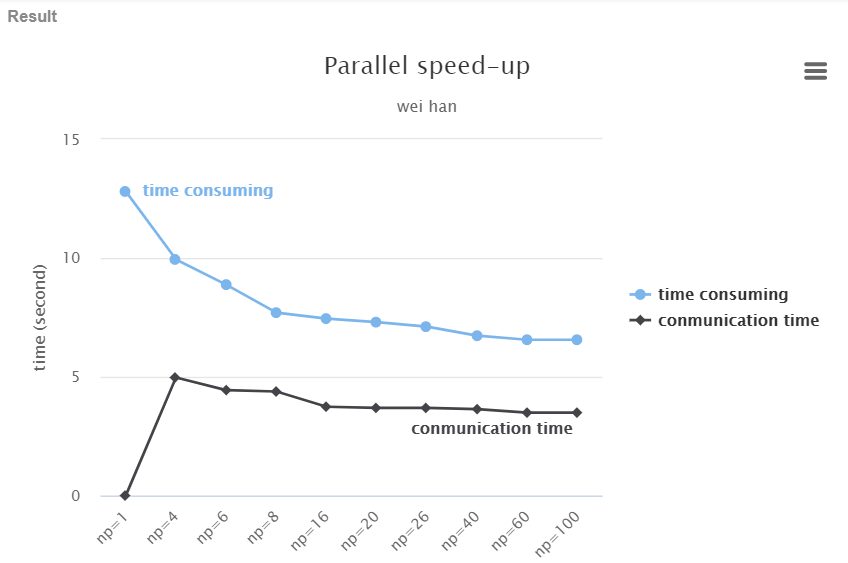
T(2) = 0.8

If N =4 T=1 B=0.6 then:

T(4) = B + (T(4)-B)/4

T(4) = 0.7

# My program and data



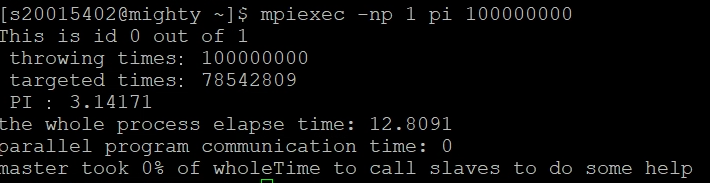
The following images were captured on the mighty cluster,

“pi” is the name of my cpp program

When np=1 throwing 100 million times,

time consuming nearly: 12.8 seconds

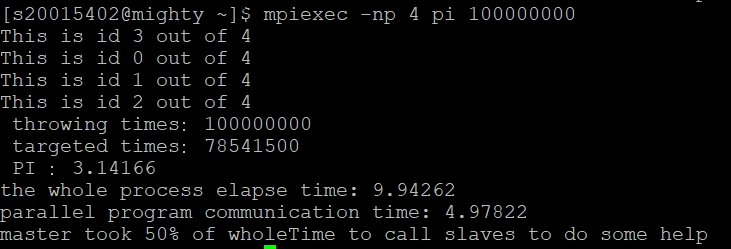
parallel program communication time: 0 second



When np=4 throwing 100 million times,

time consuming nearly: 9.94 seconds

including parallel program communication time: 4.97 seconds

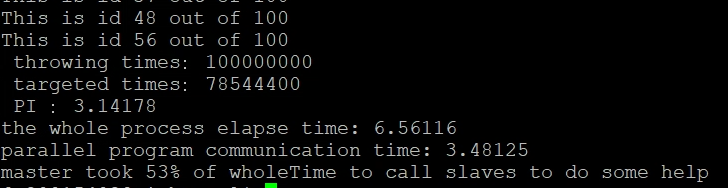


When np=100 throwing 100 million times,

time consuming nearly: 6.56 seconds

including parallel program communication time: 3.49 seconds

it might get the limit



As we can see, with the increasing number of processors, the time consuming of whole process is decreasing , which is shorten by more processors .

Meanwhile, when the number of processors get the limit such as 60 or 100, the whole time consuming is almost steady, which is reaching the maximum.

# Parallel speed-ups

When we want to optimize the sequential part of a program, we need a factor O to represent that.

T(O,N) = B/O + (1-(B/O))/N

B/O is the time of sequential part.

The factor O can shorten the program execution time in sequential part.

If the time of the old version of the program is T, then the speedup will be:

SpeedUp(SU) = T / T(O,N)

When we set T = 1 then, SU= 1/ (B/O + (1-(B/O))/N)

If B=0.6 ,O=2, N=4

SU =2.1052…..the result varies according to these parameters.

It means that the original speed has been speed up by “Speed-ups” at least 2 times.

The following list is the data generated from my program:

np=1 t=13

np=4 t=10

np=6 t=8.8

np=8 t=7.7

np=16 t=7.45

np=20 t=7.3

np=40 t=6.73

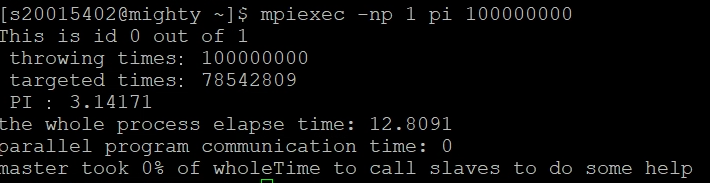
np=60 t=6.56

SpeedUP = 13/6.56 = 1.98

The SpeedUp almost get 2, which means that if we made the sequential program become parallel program by adding more processors, we can shorten the time consuming of running program to calculate PI.

Actually, the SpeedUp is what I expected, we just need to spend nearly 50% of the original time to calculate PI value.

# Interprocess communication times



If we use just 1 processor to calculate PI , the time consuming is nearly: 12.8 seconds , and the communication time is: 0,as it is just a sequential program not a parallel program.

If we employ 100 processor to do that, the communication time oppupied 53% of the whole program running time. However,the whole process elapse time decreased from 12.8 seconds to 6.56 seconds. We can use less time to calculate the PI value, which is great.

