Tung Dang
Stat110A
Project Report
03/15/04

Computer performance analysis

Introduction

The main idea behind computer performance analysis is to determine the efficiency of a computer system. The objectives might be, for example, estimation of the performance behavior of systems under construction, or monitoring and testing of existing systems. The results of analysis can help to make decisions relating to system design, to improve the management of the system resources, or to implement additional modules to tune up the existing systems. At the present time, the performance analysis is considered an essential part of the professional construction and management of computer systems.

There are three techniques used in computer performance analysis: analytic modeling, simulation modeling, and performance measurement. The basic technique used is the performance measurement, also known as benchmarking, which analyze the time required to execute a given application program. In terms of efficiency, the system that produces the smallest total execution time for a given program has the highest performance.

The main objective of this project is to compare the performance of two different computer systems, both with similar features. The analysis includes examining closely specifications and conditions of two computers and measuring the execution time of a simple benchmark program on each of them. The project will determine if there's any statistically significant difference in performance of these two computer systems.

Benchmarking

Computer specifications:

Computer 1

Intel Pentium III Mobile CPU 1Ghz
512 kB L2 On-board Cache
256 MB SDRAM
133 Mhz memory/front bus
Windows XP Professional

Computer 2

Intel Pentium III CPU 1Ghz
512 kB L2 On-board Cache
256 MB SDRAM
133 Mhz memory/front bus
Windows XP Home Edition

Comments:
The specs of two computers are almost the same. The slight differences are the processors; although they're from the same Pentium III family, the Mobile model is for laptops, while the other runs a desktop computer. The operating systems are also from the same Windows family. The professional edition has more features than the home edition, but it's not clear whether this difference will affect the benchmark test. The other components of the computers, such as graphic cards, sound cards, ports etc. are not included above, since it's reasonable to assume that those components have no effect in running the benchmark program, which depends mainly on memory allocation and calculation power of CPUs.

Benchmark C++ Code

```
**********************************************************************

#include <stdio.h>
#include <time.h>
#include <fstream.h>

void ackerman(int m, int n, int &result, int &count)
{ count++;
  if (m == 0)
    result = n + 1;
  else if (n == 0)
    acker(m-1, 1, result, count);
  else
  { acker(m, n-1, result, count);
    acker(m-1, result, result, count);
  }
}

int main ()
{
  ofstream sample ("sample.txt");
  int n, result, count;
  int x, y;
  for (n=0; n<1000; n++)            //Loop that initializes Ackerman
  {                                 //function for 1000 times
      x = clock();
      ackerman(3, 8, result, count);
      y = clock();                  //using the CPU clock to determine
      sample << (y - x) << "\n";    //the execution time and store the
      printf ("%d\n", (y - x));     //values in a file.
  }
  sample.close();
  return 0;
}


**********************************************************************
```

Comments:

The above code contains a recursive function that implements the Ackerman function (boldfaced code) defined as:
A(m, 0) = A(m-1, 1)
A(m, n) = A(m-1, A(m, n-1))
A(0, n) = n + 1 (terminal condition)
Basically, the recursive function calls itself repeatedly until it reaches the terminal condition. In a program, each time a function is called, the system has to allocate the memory for that function till it finishes the task. In this case, every recursive call adds up to the function stack in the main memory, so even a single call from the main program: `ackerman(3, 8, result, count)` can actually generate hundreds of stacked recursive calls. The process of allocating memory and carrying out the calculations of the function itself causes the complete usage of the CPU, which mostly decides the outcome of the performance test. The main program runs the loop that initializes the Ackerman function for 1000 times, and takes the execution time of each recursive sequence using the clock function. The readings are stored in an output file, whose data are used for analysis.

Running the program

Before running the program, all Windows applications and background programs were closed through the task manager. However, some processes needed by the operating system couldn't be stopped, and took up some space in the main memory. Additionally, these processes also slowed down CPU a little bit. In general, there was above 150 MB of physical memory available for each system before running the benchmark, and the CPU usage was about 1%. During the run, no asynchronous signals (key press, mouse movement etc.) were allowed to interfere.

Taking data

The data collected by the program can be considered as wall-time measurements, which measure the total time needed for the system to perform tasks of only one application. As mentioned before, the processes required by operating system to operate have little impact on the outcome of the benchmark, since they utilize only 1% of CPU power, which is negligible. Besides, no interactive inputs were required nor interfered, enforcing a smooth run of the program.

Data analysis

Execution time results

Time1 (for computer 1):

230 230 221 220 230 221 230 220 221 230 220 220 231 220 220 231 220 230 221 220 230 221 230 220 231 220 220
231 220 230 221 220 230 221 220 230 221 230 220 221 230 220 230 221 230 220 221 230 220 231 220 230 221 220
230 221 220 230 221 230 220 221 230 220 231 220 230 231 220 220 231 220 220 230 221 230 220 221 230 220 231
220 220 231 220 220 221 220 230 221 230 220 221 230 220 231 220 220 231 220 220 231 220 230 220 221 230 220
221 230 220 231 220 220 231 220 230 221 220 230 221 220 220 221 230 220 221 230 220 231 220 220 231 220 230
220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 230 220 231 220 220 231 220 230 221 220
230 221 220 220 220 231 220 220 231 220 230 221 230 220 221 220 220 231 220 220 231 220 230 221 220 230 221
230 220 221 230 220 221 230 230 220 231 220 220 231 220 220 231 220 230 221 230 220 221 230 220 221 230 220
231 220 230 221 220 230 221 220 220 221 230 220 220 231 220 230 221 220 230 221 220 230 221 230 220 221 230

220 231 220 220 231 220 220 221 220 230 221 220 230 221 230 220 220 231 220 220 231 220 230 221 220 230 221
230 220 221 230 220 221 230 220 231 220 220 231 230 220 221 230 220 220 231 220 230 221 220 230 221 230 220
221 230 220 221 230 220 231 220 230 221 220 230 221 230 220 221 230 220 221 230 220 220 231 220 230 221 220
230 221 230 220 231 220 230 221 220 230 221 230 220 221 230 220 221 230 220 231 220 220 231 220 230 220 221
230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 221 230 220 221 230 220 221
230 220 230 221 220 230 221 220 230 221 230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230
221 220 230 221 230 220 230 221 220 230 221 220 230 221 230 220 221 230 220 221 230 220 231 220 220 231 220
230 221 230 220 221 230 220 220 231 220 230 221 220 230 221 220 230 221 230 220 221 230 220 231 220 230 221
230 220 221 230 220 221 230 220 231 220 220 231 220 220 230 221 220 230 221 230 220 231 220 220 231 220 220
221 220 230 221 230 220 230 221 220 230 221 230 220 231 220 220 230 221 220 230 221 230 220 231 220 220 230 221
220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 221 230 220 231 220 230 221 220
220 230 221 230 220 221 230 220 221 230 220 231 220 220 231 220 230 221 230 220 231 220 220 230 221 220 230
221 220 230 221 230 220 221 230 220 221 230 220 231 230 220 221 230 220 231 220 220 231 220 230 221 220 230
220 221 230 220 231 220 220 231 220 230 221 220 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220
220 231 220 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 230 220 221
230 220 221 230 220 231 220 220 230 221 220 230 221 230 220 221 230 220 231 220 230 221 220 230 221 220 230
221 230 220 221 230 220 221 230 220 230 221 220 230 221 230 220 221 230 220 231 220 220 231 220 220 231 220
230 221 220 230 221 220 230 221 230 220 231 220 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231
220 230 221 230 220 221 230 220 231 220 220 231 220 230 221 220 230 220 221 230 220 221 230 220 231 220 220
231 220 220 231 220 230 221 220 230 221 220 230 221 230 220 221 230 220 221 230 220 231 220 230 221 230 220
221 220 230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 230 220 221 230 220 231 220 230 221 220
230 221 230 220 221 230 220 231 220 220 231 220 220 230 221 230 220 221 230 220 231 220 230 221 220 230 221
230 220 221 230 220 231 220 220 231 220 220 230 221 230 220 221 230 220 231 220 220 231 220 230 221 230 220
221 230 220 221 230 220 221 230 220 231 220 220 231 220 220 230 221 230 220 231 220 230 221 220 230 221 230
220 221 230 220 231 220 220 231 220 220 231 220 230 221 230 220 221 230 220 220 231 220 230 221 220 230 221
220

Time2 (computer 2):

220 230 220 221 230 220 221 230 220 221 230 220 231 220 220 231 220 230 221 220 230 221 220 230 221 230 220
221 230 220 230 221 220 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 230 220 221 230
220 231 220 220 231 220 230 220 221 230 220 221 230 220 231 220 220 231 220 230 221 220 230 221 230 220 221
230 220 231 220 220 231 220 230 221 220 230 220 221 220 220 231 220 230 221 220 230 221 220 220 221 230 220
231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 220 231 220 220 221 220 230 221 220 230 221 230
220 221 230 220 231 220 220 231 220 220 221 220 230 221 230 220 231 220 230 220 221 230 220 231 220 220 231
220 220 231 220 230 221 220 230 221 230 220 221 220 220 231 220 220 231 220 230 221 220 230 220 231 220 220
231 220 230 221 220 230 221 230 220 221 220 220 231 220 220 221 220 230 221 220 230 221 230 220 221 230 220
230 221 220 230 221 230 220 221 230 220 231 220 220 231 220 220 221 220 220 221 230 220 221 230 220 221 230
220 231 220 220 230 221 230 220 221 230 220 221 230 220 231 220 220 231 220 230 221 220 230 221 230 220 221
230 220 221 230 220 231 220 220 230 221 230 220 221 230 220 231 220 230 221 220 230 221 220 230 221 230 220
221 230 220 231 220 220 231 220 230 221 220 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220
230 221 220 230 221 230 220 221 230 220 221 230 220 230 221 220 230 221 220 230 221 220 230 221 230 220 221
230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 220 221 230 220 231 220 220 231 220 220
231 220 220 231 220 230 221 220 230 221 220 230 221 230 220 221 230 220 220 231 220 230 221 220 230 221 220
230 221 220 230 221 220 220 221 230 220 221 230 220 221 220 220 231 220 220 231 220 230 220 221 230 220 221
230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 221 230 220 220 231 220 220
221 220 230 221 220 230 221 230 220 221 230 220 221 230 220 221 230 220 221 230 220 231 220 220 231 220 230
220 221 230 220 221 230 220 231 220 220 231 220 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220
230 221 230 220 220 231 220 230 221 220 230 221 220 230 221 220 230 221 230 220 221 230 220 221 230 220 231
220 220 231 220 220 230 221 230 220 221 230 220 221 230 220 221 220 220 231 220 220 231 220 220 221 220 230
221 220 230 221 220 220 221 230 220 220 231 220 230 221 220 230 221 220 230 221 220 221 230 220 221 230 220
231 220 231 220 220 231 220 220 231 220 220 221 220 230 220 221 230 220 231 220 220 231 220 220 231 220 220
221 220 230 221 220 230 221 220 230 221 220 230 221 230 220 220 231 220 220 231 220 230 221 220 230 221 220
230 221 230 220 221 230 220 221 230 220 231 220 230 220 221 230 220 221 230 220 221 230 220 231 220 220 231
230 230 231 240 220 221 230 220 221 220 220 231 220 220 231 220 220 221 220 230 221 220 230 220 221
220 220 231 220 220 231 220 220 221 220 230 221 230 220 221 230 220 221 230 220 221 230 220 231 220 220 231
220 220 230 221 230 220 221 230 220 231 220 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230

221 230 220 221 230 220 220 231 220 230 221 220 230 221 220 230 221 220 230 221 230 220 221 230 220 221 230 220 221 230 220 231 220 220 230 221 220 230 221 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220 220 231 220 230 221 220 230 221 230 220 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 221 230 220 231 220 220 231 220 230 221 220 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 221 220 230 220 231 220 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220 230 221 220 230 221 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220 220 220 221 230 220 221 230 220 231 220 230 231 220 220 231 220 220 221 220 230 221 220 230 221 220 220 221 230 220 231 220 220 230 221 220 230 221 230 220 221 230 220 221 230 220 221 230 220 231 220 220 231 220 220 231 220 230 221 220 230 221 220

Note: all the time readings above are in milliseconds.

Declaring the random variable

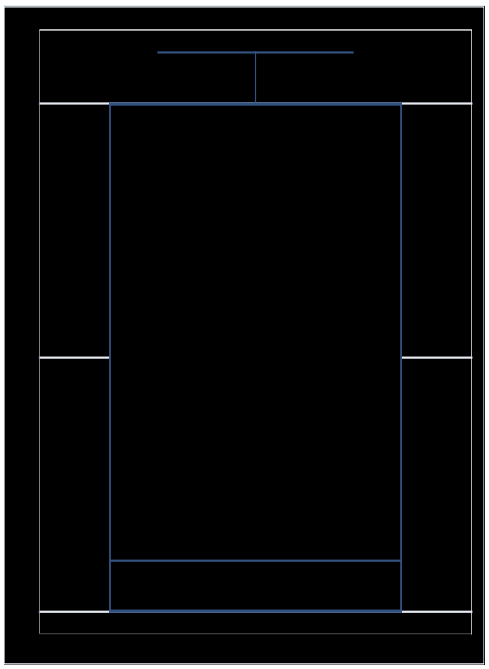
X – execution time of the Ackerman function

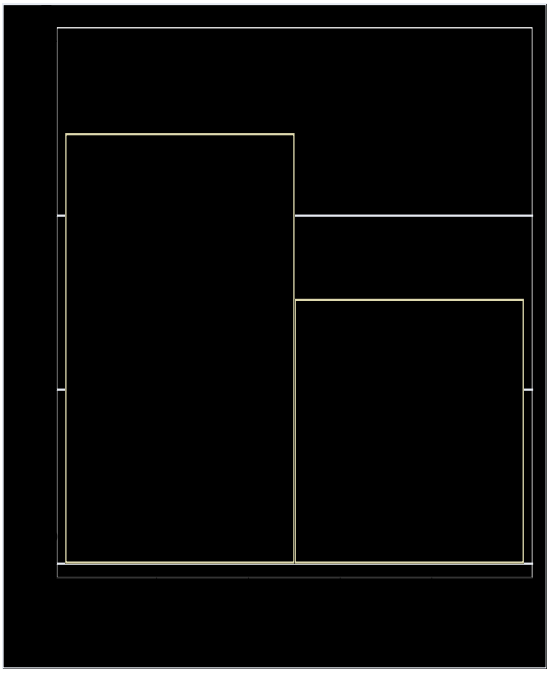
Consequently, the data for each computer system gives a random sample with a certain probability distribution.

Time 1 analysis

Variable	Obs	Mean	Std. Dev.
Time1	1000	224.132	4.87569

Min	Q1	Median	Q3	Max
220	220	221	230	231

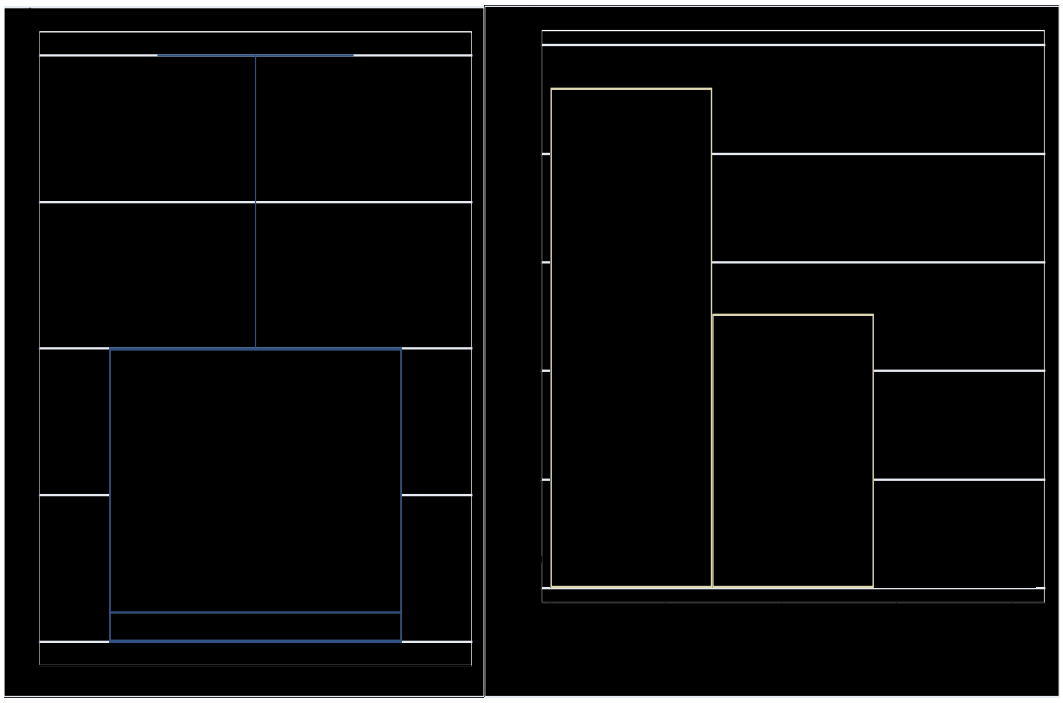




Time 2 analysis

Variable	Obs	Mean	Std. Dev.
Time2	1000	223.882	4.845158

Min	Q1	Median	Q3	Max
220	220	221	230	240



Confidence interval for difference in mean of execution time

$x_1 = 224.132$
$x_2 = 223.882$

SE = 0.2040224

95% confidence interval:
$x_1 - x_2 \pm t_{999} * SE = 224.132 – 223.882 \pm 0.4463391$

(-0.1503615, 0.6503615)

I'm 95% confident that the difference in the execution time of two computers lies between –0.15 ms and 0.65 ms.
This confidence doesn't give enough information to tell which computer performed better, since the difference might be either negative or positive.

Two sample t-test for difference in mean

```
------------------------------------------------------------------------------
Variable |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
   Time1 |    1000     224.132    .1541828     4.87569    223.8294    224.4346
   Time2 |    1000     223.882    .1532173    4.845158    223.5813    224.1827
---------+--------------------------------------------------------------------
    diff |    1000         .25    .2040224    6.451754   -.1503615    .6503615
------------------------------------------------------------------------------

                Ho: mean(Time1 - Time2) = mean(diff) = 0

  Ha: mean(diff) < 0         Ha: mean(diff) != 0         Ha: mean(diff) > 0
       t =   1.2254               t =   1.2254               t =   1.2254
   P < t =   0.8896           P > |t| =   0.2207           P > t =   0.1104
```

The above result was obtained using Stata.
Interpretation:

Ho: $\mu_1 - \mu_2 = 0$ Ha: $\mu_1 - \mu_2 \neq 0$

t = 1.2254

p-value = 2Prob(t > 1.2254) = 0.2207

At 0.05 level of significance, the t-test favors the null hypothesis.

Conclusion: There's no statistically significant difference in performance of the two computer systems.

Comments on assumptions

The measurements generated by each computer were assumed to be independent and identically distributed. This assumption should hold, once the different interferences (mainly of background operating system processes) were disregarded. Therefore, each time the tested function was run, the process of allocating memory and calculations start anew as if the function was run for the first time. Of course, there might be another type of interference, such as thermal condition of the CPU. Nevertheless, the benchmark test was run for a short time (about 4 minutes), and should not affect the CPU in physical aspect.

Reference and for more information:
http://www.frontrunnercpc.com/index.htm