School of Computer Science and Engineering

VIT Chennai

Vandalur - Kelambakkam Road, Chennai - 600 127

Project Report

DISTRIBUTED OPERATING SYSTEM

Determining operating system performance using multithreading in java

Nikhil Saraogi [21MCA1080]

Master of Computer Application (MCA)

(nikhilsarawgi9616@gmail.com)

Under the guidance of

Dr. J.V. Thomas Abraham

School of Computer Science and Engineering

ABSTRACT

The word multithreading can be translated as many threads of control. While a traditional process always contains a single thread of control, multithreading separates a process into many execution threads, each of which runs independently. The intention is to find out multithreading mechanism and structure and how to improve the performance on a multithreading environmental operating system.

WHERE THIS MULTITHREADING CONCEPT COMES FROM?

In Computer Science, Operating System has the ability to perform more than one task parallelly. this multitasking is further divided intotwo types:

1. **Process Based Multitasking: -**

Process means running program or program under execution.
This process-based multitasking is done by Operating system and it isknown as an ability to execute multiple processes at the same time. Example: -
While surfing browser we can also execute the antivirus to detectviruses.
While typing a code in java IDE we can play music in media playeretc.

2. **Thread Based Multitasking**

A Thread is an execution path and so if a program has a multiple execution path, then we can say that it is exhibiting Multithreading orThread Based Multitasking.
It has to be done by Software Developer.

Example Of Multithreading:

- We can open multiple tabs in a same browser
- We can change volume level or add/delete songs from theplaylist while playing a song in a media player
- We can search files/folders in Windows OS and minimize it tohappen in background and do the other task etc.

INTRODUCTION

Multithreading allows multiple concurrent tasks to run within a single process for the maximum utilization of a CPU. A thread is the basic unit of the process code and is called a lightweight process within a process that cannot exist outside a process.
The main benefits that arise from multithreading are:

- improved application responsiveness and better program structure - any program in which many activities do not depend upon each other can be redesigned so that each activity is executed as a thread,
- efficient use of multiple processors - numerical algorithms and applications with a high degree of parallelism, such as matrix multiplication, can run much faster when implemented with threads on a multiprocessor,
- use fewer system resources - the cost of creating and maintaining threads is much smaller than the cost for processes, both in system resources and time

All threads created from the same initial thread (standard process) exist as a part of the same process, sharing its resources (address space, operating system state ...). Beside that the multithreaded applications use fewer system resources than multiprocessor applications, communication between threads can be made without involving the operating system, thus improving performance over standard inter-process communication. From these reasons multithreading is so popular today, and modern operating systems support it.
Multithreading also brings some problems, like signal handling, function safety under possible parallel threads execution (parallel use and change of global variables), alarms, interval timers and profiling. The problem is how to change this process oriented and defined element to support threads and to be defined on a thread level. One of the main problems in this work was a time measurement (real, user and system time) for a single thread in a multithreaded application.

METHODOLOGY USED

- We are executing different Algorithms together in a Multithreaded way as to make the Operating System Busy so that we can utilize its full potential and thus we can determineits performance.

- First, we will write the Algorithms without the use of Multithreading and then we could be able to see that OS is idledespite of the fact that all the algorithms are independent of each other and also it takes more time to execute all.

- Then we will make use of Multithreading and then we will see the performance of OS as it will execute the code fast also theexecution time will be less.

- We are using the features and syntaxes of MULTITHREADING given by JAVA SE. We will write the algorithms in NOTEPAD andat command prompt we will show the output. Also, we will include a java predefined method which will tell us at the end that how much time is taken by OPERATING SYSTEM to execute our source code.

- Thus, we can determine the performance of Operating Systemin a Multithreaded Environment and non-Multithreaded Environment.

- And at last, we will achieve our expected results that with the help of Multithreading we will be fully Utilizing CPU/OS idle time,also we will get our results in Minimum Amount of time i.e., within fraction of seconds.

LITERATURE REVIEW

SNo.	Paper	Author	Year	Summary
1	Multithreadingin Java: Performance and Scalabilityon Multicore Systems	Kuo-Yi Chen, J. Morris Chang, and Ting-Wei Hou.	2010	Java is a fantastic programming language that is utilized on a variety ofsystems. It has many\r superior features, such as automatic memory management and cross-platform compatibility. Java's multithreading feature is also used on platforms to benefit from better throughput and faster response times. As the use of multicore processors becomes more common, software developers are

				focusing on developing multithreaded applications.
2	Thread Assignment in Multicore/Multithreaded Processors: A Statistical Approach	Petar Radojkovic, Paul M. Carpenter, Miquel Moreto, Vladimir Cakarevic, Javier Verdu, Alex Pajuelo,Francisco J. Cazorla, Mario Nemirovsky	2015	multithreaded processors, seeing as a perfect string task is associate degree obstinate issue, for functioning on models of the applications what is more, framework style. The problem is especially befuddled at the purpose once the number of apparatus settings (virtual pc chips) is big, or once processor assets are sharedat totally different levels. Since the string task issue is recalcitrant, it's in on a dailybasis troublesome to grasp the exhibition of the best task. The measurable investigation surmises the populace's greatest (or least) in sight of associate degree irregular example, inan exceedingly manner that's freed from the problem being cared for. It does not would like a major comprehension of the target framework, thus it okay could also be utilized while not a large interest in elbow grease or time. The strategy is very useful within the assessment of any new planned heuristics-basedcalculation.

3	Understandingthe energy efficiency of simultaneous multithreading	Yingmin Li, David Brooks, Zhigang Hu, Kevin Skadron, and PradipBose	2004	Simultaneous multithreading(SMT) has been shown to bea good way to boost microprocessor performance by extracting additional instruction-level parallelism from many threads. Power-efficiency is critical in today's microprocessor designs, and we present modelling extensions to an architectural simulator that allow us to investigate the power-performance efficiency of SMT. SMT may give a performance speedup of approximately 20% for a wide variety of applications with a power overhead of roughly 24%, according to a comprehensive design space study. As a result, SMT may significantly improve energy efficiency indicators like ED2. The paper also looks at the causes of the power increase, examine the influence of leakage□sensitive process technologies, and talk about our model validation technique

IMPLEMENTATION

We are executing many Algorithms concurrently in a Multithreaded manner in order to keep the Operating System busy so that we mayfully exploit its capability and hence evaluate its performance. First, we will build the algorithms without using Multithreading, and then we will be able to observe that the OS is idle despite the fact that allof the algorithms are independent of one another, and it also takes longer to run all of them. Then we'll utilise Multithreading, and we'll watch how the OS performs as the code is executed faster and the execution time is reduced. We are utilising the MULTITHREADING functionalities and syntaxes provided by JAVA SE. We will create thealgorithms in NOTEPAD and display the results at the command prompt. In addition, we will include a java predefined method that will inform us at the end how long it takes the OPERATING SYSTEM to run our source code. As a result, we may compare the performanceof an operating system in a multithreaded and non-multithreaded environment. Finally, we will accomplish our intended outcomes in that we will completely use CPU/OS idle time with the aid of Multithreading, and we will receive our results in the shortest amountof time, i.e., within fractions of seconds.

ALGORITHM

SBUBBLE

SORT

```
begin BubbleSort(list)

   for all elements of listif
      list[i] > list[i+1]
         swap(list[i], list[i+1])end if
   end for return

   list

end BubbleSort
```

MERGE SORT

MERGE_SORT(arr, beg, end)if beg

< end
set mid = (beg + end)/2
MERGE_SORT(arr, beg, mid)
MERGE_SORT(arr, mid + 1, end)
MERGE (arr, beg, mid, end) end of if

END MERGE_SORT

Code

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;


class thread_main_class{
    public static void main(String[] args){

        //checking for the time of starting
        long start = System.currentTimeMillis();

        //start code here======================== for single thread environment >>

         //>>object to single_bubble_sort :: obj_0
         single_bubble_sort obj_0 = new single_bubble_sort();

         //System.out.println("Unsorted Array is\n ");
         //obj_0.printArray(arr);


         //>>run object
         obj_0.bubbleSort();
         //System.out.println("\n\nSorted Array is in Single Thread environment (using Bubble Sort) : \n");
         //obj_0.printArray(arr);

         //>>object to single_bubble_sort :: obj_0
         single_merge_sort obj_1 = new single_merge_sort();

         obj_1.merge_sort();
         //System.out.println("Sorted Array is in Single Thread environment (Merge Sort) : \n ");
         //obj_1.print_it(arr);
```

```
   //End code here

   //checking for the time of exiting
   long end = System.currentTimeMillis();

    //time taken by exicution of of program
    long time_taken = (end-start);
    System.out.println("\n\nTime Taken in Sorting On single Thread is in (milisecond)");
    System.out.print(time_taken);

    System.out.println("\n\nSingle thread environment ENDS here ");




   //start code here======================== for multi thread environment >>


   multi_bubble_sort obj_4 = new multi_bubble_sort();
   multi_merge_sort  obj_3 = new multi_merge_sort();



   //checking for the time of starting

   long start_m = System.currentTimeMillis();
   obj_4.start();
   try{Thread.sleep(10);} catch(Exception e){}
   obj_3.start();
   try{Thread.sleep(10);} catch(Exception e){}
   //checking for the time of end
   long end_m = System.currentTimeMillis();

   //time taken by exicution of of program
   long time_taken_m = (end_m-start_m);
   System.out.println("\n\nTime Taken in Sorting On multi Thread is in (milisecond)");
   System.out.print(time_taken_m);

   System.out.println("\n\nmulti thread environment ENDS here ");

   //------------------------------creating the file to Right the Output to it

  // File file_0 = new File("F:\\Z-MCA\\SEM-1\\ZZZ_dos_orignal_process\\result.txt");

   // if(file_0.exists()){
   //     System.out.println("Filename is " + file_0.getName());
   //     System.out.println("location is  " + file_0.getAbsolutePath());
   //     System.out.println("readable " + file_0.canRead());
```

```
//     System.out.println("writeable " + file_0.canWrite());

// }else{
//     System.out.println("file not present");
// }


String output_time_file_0_multi = String.valueOf(time_taken_m);
String output_time_file_0_single = String.valueOf(time_taken);


//for file_0
try {
    FileWriter file_0 = new FileWriter("D:\\Distibuted_environment\\result_0.txt");
    file_0.write("This Operation is done on Nikhil's PC\n\n");
    file_0.write("Time Taken in Sorting On multi Thread is in (milisecond)\n");
    file_0.write(output_time_file_0_multi);
    file_0.write("\n\nTime Taken in Sorting On single Thread is in (milisecond)\n");
    file_0.write(output_time_file_0_single);
    file_0.close();
    System.out.println("Output is sucessfull worte to file");
} catch (Exception e) {
    System.out.println("file not present");
    e.printStackTrace();
}
```

Output

```
PS D:\Distibuted_environment> javac thread_main_class.java
PS D:\Distibuted_environment> java thread_main_class


Time Taken in Sorting On single Thread is in (milisecond)
1697

Single thread environment ENDS here


Time Taken in Sorting On multi Thread is in (milisecond)
36

multi thread environment ENDS here
Output is sucessfull worte to file
PS D:\Distibuted_environment> .\result_0.txt
PS D:\Distibuted_environment>
```

Conclusion

The term "multithreading" can be interpreted as "multiple control threads." While a typical process always has a singlecontrol thread, multithreading divides a process into numerous execution threads, each of which executes independently. We successfully established that multithreading improves and optimises the performance of the system.

Future Inferences from This Project:

In future this project can be implemented on any Operating System because we have used Java and its platform independent.
This will determine the efficiency of OS and the results can becompared with the rest of OS performance.