Improving Performance of Algorithms in Distributed Computing with Perspective of Green Information Technology

Era Johri
K. J. Somaiya College of Engineering
Vidyanagar, Vidyavihar
Mumbai 400 077, India

Shameela Shaikh
K. J. Somaiya College of Engineering
Vidyanagar, Vidyavihar
Mumbai 400 077, India

ABSTRACT
In Distributed Computing approach, it is followed to assign a job to a processor if it is idle. The focus is now on how to optimize resources to decrease the energy consumption by volumes of computing equipments to deal with green and sustainability issues. So that to save environment from Global Warming and utilizing the resources efficiently. This process is twofold - One hand providing green and power efficient algorithms and on the other supporting companies green investments. In order to minimize energy consumption by processor allocation we are providing some algorithms to generalize distributed computing. In this paper we provide algorithms to green compute by calculating a threshold and sending systems to power saving modes if the processor is idle.

Categories and Subject Descriptors
Distributed Computing, Green Information Technology, Power Saving Modes and Power Consumption by different equipments in Computer Systems.

General Terms
Analysis, Documentation, Performance, Experimentation.

Keywords
Green Computing, Threshold, Distributed Computing, Power Consumption, Power saving modes

1. INTRODUCTION
A Distributed System consists of multiple autonomous computers with different topologies where individual computer are physically distributed in some geographical area that communicate with each other by message passing[1]. We can implement a "Load Transfer" policy to the process created locally on a machine and a decision has to be made weather or not it can run on the same local machine or to be transferred to somewhere else using load distribution algorithms[2]. Decision is made by the local machine, where the process is generated, to keep the process or to transfer it depending upon some threshold. If the machine load is below the threshold keep the new process for execution locally otherwise transfer it to some other processor. In this paper we calculate a threshold (decision factor in segregating the load to different processors) for determining heavy and light weight loads of processing dealing with,

- The energy requirements by various devices like printers, CPU's, monitors etc.
- Sequential verses Looping structures.

1. Calculating the difference in speed for performance check i.e. if the time taken by Processor P1 is less than the time taken collectively by using distributed computing approach[3] i.e.,

$$Processor\ (P1) < Processor\ (P1+P2+.....+P_n)$$

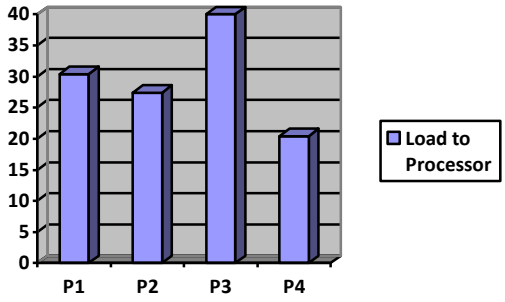

Figure 1 Heavy Loaded Processors in Distributed Environment

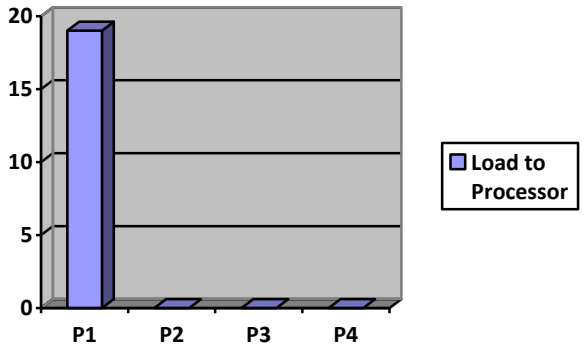

Figure 2. Only Local Processor Loaded as Load capacity is lesser than threshold

In the Algorithm we will deal with methods on how to calculate the threshold and how to take decision for number of processors to be used for computing so that to save maximum energy and contribute to environmental survival.

2. IMPORTANT FACTORS FOR CALCULATING THE THRESHOLD

2.1 Sequential Vs Looping Structure

We will take examples of different programming constructs in different languages and will try to compare them in terms of time and the resources that they hold for the amount of time to execute the instructions.

In a basic assembly language program when we compare an ADDITION and MULTIPLICATION instruction we see that the time taken by an addition is lesser than the time taken by multiplication operation. If we convert multiplication operation into addition operations, we can minimize the energy consumption with great extent as we can minimize the number of resources used in computing.[4]

MULTIPLICATION instructions though are just continuous addition operations but surprisingly require much more amount of time because of JUMPING, every cycle of addition requires an extra JUMP instruction[5].

Next in looping structures we see how long it takes to run a simple empty loop for a very large number of iterations approximately around 10000 billion. We will calculate it by running the following codes for different languages[6]. For example,

For example consider the following **Java code**

```
public class Loop
{
  public static void main(String[] args)
  {
    //10 000 billion iterations
    long time = System.currentTimeMillis();
    int REPEAT1 = 1000 * 1000;
    int REPEAT2 = 1000 * 1000 * 10;
    for (int i = 0; i < REPEAT1; i++)
    {
      for (int j = 0; j < REPEAT2; j++)
      {
        //do nothing
      }
    }
    time = (System.currentTimeMillis() - time)/1000;
    System.out.println("Time taken: (in seconds) " + time);
  }
```

Studies says that the above program just require less than one second for execution

Also consider the following **C Program**

```
#include <sys/time.h>
main(int argc, char *argv[])
{
  int i, j, REPEAT1, REPEAT2;
  struct timeval before, after;
  void *tzp;
  /*10 000 billion iterations*/
  tzp = 0;
  before = (struct timeval*) malloc(sizeof(struct timeval));
  after = (struct timeval*) malloc(sizeof(struct timeval));
  gettimeofday(&before, &tzp);
  REPEAT1 = 1000 * 1000;
  REPEAT2 = 1000 * 1000 * 10;
  for (i = 0; i < REPEAT1; i++)
  {
    for (j = 0; j < REPEAT2; j++)
    {
      //do nothing
    }
  }
  gettimeofday(&after, &tzp);
  printf("Time taken (in seconds): %ld\n",
          (after->tv_sec-before->tv_sec));
}
```

The time taken by the above code to execute in Windows platform will be around 1 month but it differs in case with other programming languages. For example the above code when executed on different programming language we got different results as,

Table 1: Summary of Time Taken by running an empty loop on various platforms

Language	**Time taken to run a simple empty loop test for 10 000 billion iterations**
Java	Under one second
Perl	approx. 1 month
C/C++	approx. 1 month
C#	approx. 1 month
Assembler	approx. 1 month
Fortran	approx. 1 month
Ada	approx. 1 month
Basic (e.g. Visual Basic)	approx. 1 month

Next in the case of String instructions, Let the dataset is about 38000 XML files and parse the file, and tokenized it. The output while processing these files is quite different as,

The system crunches 8200 files in the first 10 seconds, but is able to do only 5000 in the next 10, and then 3500 in the next 10 and it reduces gradually. It takes about 75 seconds totally for 38000 files, whereas if the system had proceeded at the speed with which it started, it should have taken under 50 seconds.

Therefore, all these factors have to be considered and try to incorporate all of these features in our algorithm before determining the threshold for the process.

2.2 Energy Requirements for various Devices

Once if we can determine the approximate time and the resources utilized by the program we can differentiate the process into heavy weight or light weight process. If it's a light weight process then we can compromise with speed a little bit by minimizing the requirement of resources. We can reduce the economic cost of the resources as well as Green compute the environment. Also, the resources not in use can be sent to different power saving modes like hibernation, sleep or standby. The requirement of energy by various peripherals of a computer per hour approximately at maximum at various modes can be depicted as,

Table 2: Power consumption by Computers per hour Approx.

Computers	
Desktop Computer	60-250 watts
On screen saver	60-250 watts (no difference)
Sleep / standby	1 -6 watts
Laptop	15-45 watts

Table 3: Power Consumption by Monitors per hour Approx.

Monitors	
Typical 17" CRT	80 watts
Typical 17" LCD	35 watts
Apple MS 17" CRT, mostly white (blank IE window)	63 watts
Apple MS 17" CRT, mostly black (black Windows desktop with just a few icons)	54 watts
Screen saver (any image on screen)	same as above (no difference)
Sleeping monitor (dark screen)	***0-15 watts***
Monitor turned off at switch	***0-10 watts***

As long as computer goes into different modes of Power Saving, they doesn't use squat for electricity. Of course, it should be made sure that the computer is set to different modes of Power Saving automatically when not in use hence less contributing in Global Warming and less Energy Consumption.

This feature becomes very important when used in distributed systems where the idle processors should be automatically send to the power saving modes and if we can maintain the global record of the computers in active and passive modes. Also the shared devices in network when they are not in use should be automatically sent to Power Saving modes.

3. OUR APPROACH

Based on the above two factors our approach for differentiating Heavy Load and Light Load processor is something like this…

3.1 Analyzing Types of Operations:

While the time of parsing any computer Program threshold can be calculated between 0 to 5 by assigning time units between 0 to 5 to various processes depending upon the different mathematical operations like +,_,/,*,etc and instructions like add(), sub() etc used in the program. If there are any looping structures that can be converted to sequential instructions or any other operations the numbers between '0' and '5' are assigned based on the requirement of execution time and resource requirement for each operation. For example,

Table 4: Numbering the operations

Operations	**Numbers**
Addition, Subtraction	1 for each instruction
If else	3 for each operation
Looping	5 for each operation
String Calculation	5 for each operation

It is not necessary to store the numbers; it should only be summed up based on the type of operations.

Hence we have to sum up all the points and then divide the whole by total number of operations and determine the threshold value.

If the value is equal to or more than 5 we assign that process as heavy weight and apply load balancing algorithms to it otherwise take it as light weight process and process it on the local machine.

3.2 Analyzing the Wastage of Energy in Switching of Load

If the load is lightweight and still distributed on different machines then it may happen that there is considerable switching time in for loading the process on different Processors. This is unintentionally increasing the execution time for the load. In such cases the load should not be distributed to different processors. It is possible in that case that

Time taken by,

Processor (P1) < Processor (P1+P2+…..+Pn)

Now once the process is estimated as light weight (if its lying below the threshold) or heavy weight (if lying above the threshold) process we can simply apply power saving mechanisms so as to conserve energy.[7] The load balancing can be shown as in Fig 1 and Fig 2. From this diagram we can say that compromising a little in speed we can save energy and make our systems to green compute.

4. REFERENCES

[1] M. L. Liu, "Distributed Computing Concept and Applications

[2] Niranjan G. Shivaratri, Phillip Krueger, and Mukesh Singhal, "Load Distributing for Locally Distributed Systems" *Compute*, vol.12 no.22 pp.33-44, Dec. 1992, doi: 10.1109/2.179115

[3] Andrew S. Tanenbaum, "Distributed Systems: Principles and Paradigms", *Pearsons*

[4] Jason Harris, "Green Computing and Green IT Best Practices on Regulations and Industry Initiatives, Virtualization, Power Management, Materials Recycling and Telecommuting" p.202

[5] F.C.H. Lin, R.M. Keller, "The Gradient Model Load Balancing Method," *IEEE Transactions on Software Engineering*, vol. 13, no. 1, pp. 32-38, Jan. 1987, doi:10.1109/TSE.1987.232563

[6] Jim Farley, "Java Distributed Computing", *O'Reilly Media*

[7] Georges Da Costa, Jean-Patrick Gelas, Yiannis Georgiou, Laurent Lefevre, Anne-Cecile Orgerie, Jean-Marc Pierson, Olivier Richard, Kamal Sharma, "The GREEN-NET framework: Energy efficiency in large scale distributed systems," ipdps, pp.1-8, 2009 IEEE International Symposium on Parallel & Distributed Processing, 2009