ISSN: 0975-766X
CODEN: IJPTFI
Available Online through *Research Article*
www.ijptonline.com

ANALYSIS FOR ENERGY REDUCTION IN PARALLEL ALGORITHM

Ravi Sharan Gautam[1] , G. Uma Maheswari[2] , S.Subha[3]
School of Information Technology and Engineering, SITE, VIT, Vellore.
Email: g.umamaheswari@vit.ac.in

Received on 25-10-2016 *Accepted on 02-11-2016*

Abstract:

The practice of using computer resources efficiently is Green computing. The use of computers, printers and monitors consumes more than 150 watts and maximum amount of energy is wasted when the equipment is left in on condition when not in use. We are utilizing a big amount of energy and their wastage is continued still today. So here we are trying to reduce the consumption of energy it may be the energy of mobiles battery, laptop battery, desktop etc. We all know that to perform any task there is some process; each and every process executed by some cores and each core is operated at distinct frequencies. So here we are showing how an algorithm reduces energy according to the number of cores and at what frequency they are working using parallel algorithm.

Key words: Cores, Energy Consumption, Frequency. Parallel Algorithm

1. Introduction:

Green Computing is the practice of using computer resources efficiently [1]. Another definition is the one which defines green computing as the study and practice of efficient and eco-friendly computing resources and the environmentally responsible use of computers and related resources. The primary goals of green computing are to reduce the use of hazardous materials and to maximize the energy efficiency during the product's lifetime [2]. Normally computer consume so much electricity, even today according to survey of the united states the 14% of total electricity is used by computer, these are also responsible for greenhouse effect. There is one more problem with mobiles, tablets, laptops that these contains limited battery backup.

Whenever the power comes into the CPU, the computation cycle of cores will be increased and the consumption of the power will remain constant. Now we come to know that in multicore architecture, we can measure the speed of the single core or we can discard it in idle state, so the energy consumption will be reduced. Since in a sequential processor, energy loss will reduce if we decrease the frequency of a processor, because there is no relationship

between frequency & power. As we know that decreasing the frequency in a processor will increase the time of an algorithm that is it will decrease the performance. The main problem occurs when we used parallel processors. Parallel computation contains few parallel and serial sub computation therefore energy and performance are dependent on the structure of the parallel algorithm, not only the multicores. If we increase number of cores then the computation for each core will reduce so the performance will increase. In parallel algorithm there is no relationship between computations for each core, if we double the cores the time for computation will become half.

2. Materials and Methods:

The Smart2020 report predicts the emission of BAU CO2 from three sectors of ICT industry such as end-user devices, telecommunication and networks, data centers is increasing gradually. It is estimated that footprint of end-user devices will increase 2.3% every year. Therefore, to reduce the overall footprint of these ICT devices the factors, energy improvements and way of usage are essential.

Green and sustainable software is a software product that has the smallest possible economic, societal, ecological impact as well as impact on human beings [3].This has led to the introduction of various programmes and initiatives that encourages energy efficient software such as green software engineering and Eco-design software [4].

In fact the software does not consume energy directly; it is due to hardware that makes the software to run. Hence, the resource usage metric such as the CPU usage, memory and disk usage are used as the measuring criteria [5]. The observation of energy utilization towards effective work performed is more important. Therefore monitoring resource utilization at the user level has to be encountered.

We are assuming some simple assumptions, which also can be used for future research, where more specific performance model is used. Our assumptions are:-

1. All cores works at same frequency & each cores frequency can be change according to the frequency probe.
2. By measuring the frequency of the core we can compute time of the cores.
3. The time consumed for sending and receiving a message is comparatively more time consuming than the time taken to route the message between the cores.
4. The access time of memory is constant, so there is not any memory hierarchy for the cores.
5. Every core has its own memory & cores are connecting through message communication.
6. We can control process scheduling.

We know that **t** = (running time) of a core is directly proportional to the no. of cycles u which is executed by the core.

Suppose f is the frequency for one core, therefore

$$t = u * 1/f$$

Remember that if we increase voltage supply linearly then this leads to increase of the frequency for the core and it's also leads to incrementing a non-linear power consumption.

We are showing the critical factors:

$$e = ec * t * f^3$$

Where **em**= consuming energy by sending or receiving message between two cores.

f= maximum frequency for single core.

n= size of the input for parallel application.

m= allocated no. of core for parallel application.

kc= no. of cycle executed at maximum frequency for single message communication time.

Pa= consumed static power.

METHOD:

Here we are presenting our method which helps to examine energy scalability under parallel application.

Step.1:- Firstly we are discovering the critical path of the algorithm; here critical path is the largest path where the edges are representing task serialization of an algorithm.

Step.2:- now we are dividing the critical path into two steps computation and communication.

Step.3:- in the critical path we are determining the computation steps so the performance requirement becomes equal to the parallel performance. We are determining the critical path's compute time, so we can find difference between the communication time of critical path and required performance and getting new less frequency at cores should run.

Step.4:- then we are calculating total no. of messages handled i.e. calculating the message complexity of a parallel algorithm. Note that the message complexity for some algorithm only depends on the no. of cores, while for other algorithms it depends upon the no. of cores and input size.

Step.5:- assume the frequency evaluated in step 3 and then evaluate all cores total idle time. Scaling the critical path may rise the idle time for other cores.

Step.6:- by using energy model we are creating an expression for energy consumption

The expression for energy consumption is the sum of the energy consumed by –

1. **e.idle** = idling.(it may increase as the busy cores take longer to finish)

2**. e.comm** = communication.(it may increase as more cores are used since the computation is more distributed)

3. **e.comp** = computation.(it is lower cores if the cores run at lower frequency)

e = e.idle + e.comm + e.comp

Step.7:- according to the input size of a function analyze the expression to find the no. of cores required for minimum energy consumption.

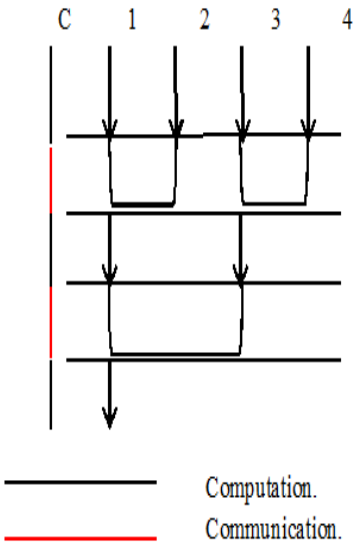
Here we are taking an example of adding numbers.

With the help of 'm' cores we are adding 'n' no's by using parallel algorithm. Normally 'n' no's are similarly distributed in 'm' cores. And for storing sum we are using one of the cores. Here we are assuming that some power of two the no. of cores available. Here we define some steps for running the algorithm.

1. Here half the cores are sending, then they are computing the remaining half therefore core is receiving sum only from one core. Then receiving cores are adding the local sum whatever the computed.
2. This step will go on till only one core is remaining.
3. At last only one core will contain the sum of n no's

3. Results:

Here we are assuming a simplest parallel algorithm, for addition of **n** numbers with the help of m cores. Primarily all **n** numbers are similarly circulated among **m** cores, at the last of computation the sum will store by any one core. We are assuming that no. of cores are in some power of 2*log(m) steps will run by algorithm. At the starting the sum computed by half of the cores send to the next half cores as a result the core is receiving sum from only one core. Then the core that receives adds the entire sum that is computed by the other cores. This process will going on until only one core is left in this process. At the last the sum of all **n** numbers will store in one core.



ANALYZING ENERGY CONSUMPTION:

Here we are examining equation of energy attain from the addition algorithm. To analyse energy scalability performance, we are breaking the function in small parts with respect to core for minimum computation. So now we will go for simple graph analysing method. Remember that expression of energy is always depends upon no. of variables where

1. **n**= input size
2. **m** = no. of cores
3. **B**= no. of instruction per addition
4. **kc**= no. of cycle execute at max. frequency
5. **em**= consumption of energy in communication between cores.
6. **Ps**= static power.

Normally no. of cycles for addition is only one in most of the architecture, so let us suppose **B**= 1, we are also setting up constant energy consumption per cycle **ps/f = 1** with the maximum frequency f. Using normalized energy value we are expressing other energy values.

Here we are making some assumptions regarding other parameters to plot the graph of the needed partition. We already worked on the analysis sensitivity to range the values for the parameters.

Let us consider the ratio to be 10, so **ec.f^2 = (ps/f).10**.but this parameter is not sufficient for analysis, in fact if parameter have large variations, it will not affect the graphs shape. So on taking different parameter '**k**' that is signifying the energy ratio that is consuming by single message sending communication '**em**' and at the maximum frequency energy consuming by executing single instruction. Therefore

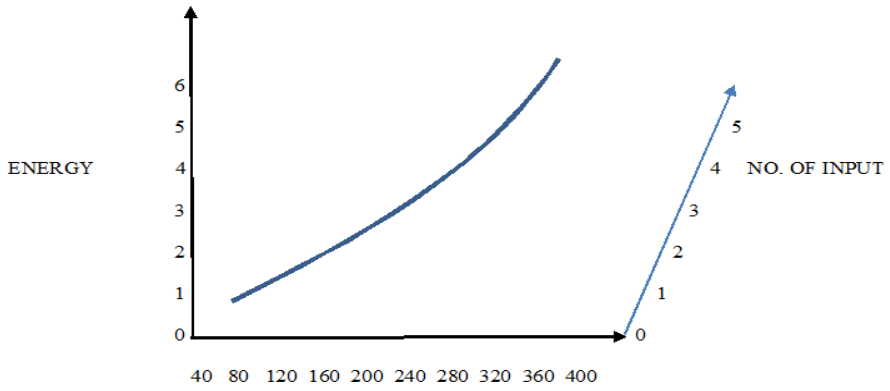
em= (k).(ec).(f^2)

For this problem sequential algorithm is trivial. For computing addition of '**n**' numbers it will take '**n-1'** additions. We know that for sequential algo. The running time is

tseq= B.(n-1).(1/f)

In figure function for input size is energy **e,** no. of cores. if we will give any input size **n,** energy is decrement with incrementing mand after that on increment with **m.** we already know that energy used for computing decrement with

an increment of no. of core that are running on lower frequency & energy that is used for communication increment with increment of cores. We can evaluate that following figure increases and follow negative exponent curve with positive coefficient.



LU FACTORIZATION:

1. Here we are using the sequential algorithm which is based on n matrix. This method includes the lower triangular matrix L and upper triangular matrix U so X=LU. Due to the Gaussian elimination our method worked. The matrix U is attained through overwriting of A.
2. We already assumed that our methodology is known by user. In Gaussian elimination it needs m^2/2 paired division operation and m^3/3 multiplication and addition operation

On a single core, the time which is taken by our method is working at the higher frequency

Which showed by this equation

T seq=B*(n^3)/3*1/f

B is the no. of cycles which is used for addition and multiplication.

3. The LU factorization problem can be solved by various parallel algorithms but here we use only the coarse-grain one dimensional column algorithm. Every core is having few of columns and they used it for communication among them and attain the matrix U. At each and every core m the algorithm is

Step 1:- Start

Step 2:-load k:= 0

Step 3:-loop for k:= 1 to n-1

do

if k belongs to mycol

then

```
            for  i:=k+1 to n
          do
            lik:=aik/akk
           end  for loop
          end if condition
    for j belongs to mycol, j>k
        do
            for i:=k+1 to n
              do
                aij:=aij-lik.akj
             end for loop
        end for loop
    end for loop
```

Step 4:- stop

4. Discussions:

In earlier we used some postulations for energy scalability of the parallel addition algorithm analysis as same we use here. The sequential algorithm execution time is attained to be the needed performance during the analysis.In the above curve the value of M and the energy behaves inversely proportional, i.e. when one increases then another one decreases but after some times both increases simultaneously. The addition algorithm and the above curve are much more similar. The key size is the convolution of message. The communication through message does not diminution in usage of energy because the occurrence scaling of cores. The key size increment can increment the core's optimal number which is needed for lowest amount of energy usage. Now we know that LU factorization has same energy scalable properties as the addition algorithm.

5. Conclusion:

Now our research work showing that performance and energy properties of algorithm on scalability multiple core architecture vary significantly and it helps the programmer to make better understanding to consume the energy by algorithm. In our analysis we are using some parallel algorithms those nature are different. Interesting thing is that

our analysis totally tough for parameter values. On the other hand few of our assumptions i.e. stable energy & time for sending a message communication that not holds whatever the architecture is scale up.

Our analysis is improved in so many ways we are supposing that messages that are sends to different cores that have only one element on the other hand if we are sending so many elements in one message that can be decrease the consumption of energy for each element. Due to presence of memory hierarchy our analysis may be affected. To solve this problem we will expand our analysis which will use **logp** model of computation. Our work tells that performance and energy, no. of cores used. Here we are analysing energy scalability with the total energy consumed. This research will help to make the application performance better on a constant energy budget

References:

1. Dheera Jadhwani, Mayur Agrawal, Hemant Mande, Study of Efficient Utilization of Power using green Computing**,** International Journal of Advanced Computer Research, 2012; 2(4): 2249-7277
2. Chakraborty P., Bhattacharyya D., Nargiza Y. and Bedajna N. International Journal of Grid and Distributed Compu-ting, 2009
3. Šimunić, T., Benini, L., De Micheli, G., & Hans, M. Source code optimization and profiling of energy consumption in embedded systems. In Proceedings of the 13th international symposium on System synthesis 2000; 193-198.
4. Sagahyroon, A. Power consumption in handheld computers. In *APCCAS 2006-2006 IEEE Asia Pacific Conference on Circuits and Systems*, 2006; 1721-1724.
5. Ravi, N., Scott, J., Han, L., & Iftode, L. Context-aware battery management for mobile phones. In Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference 2008,224-233.
6. Mahmoud, S. S. & Ahmad, I. A Green Model for Sustainable Software Engineering, International Journal of Software Engineering and its Applications, 2013, 7(4), pp. 55-74
7. Ahmed,F.,Mahmood,H. & Aslam,A, Green Computing and Software Defects in Open Source Software: An Empirical Study. Lahore,IEEE, 2014

.