ENERGY CONSUMPTION ANALYSIS AND ENERGY OPTIMIZATION TECHNIQUES OF HPC APPLICATIONS

Rejitha R.S. *
rejitha.rs@gmail.com

C. Bency Bright *
bency@sxcce.edu.in

Dr. Shajulin Benedict **
shajubenedict@yahoo.co.in

* Department of Computer Science and Engineering, St. Xavier's Catholic College of Engineering
** Director, HPCCLoud Research Laboratory, St. Xavier's Catholic College and Engineering

***Abstract*: High Performance Computing (HPC) is used for running advanced application programs efficiently, reliably, and quickly. HPC makes use of both parallel as well as distributed computing technologies. In earlier decades, performance analysis of HPC applications was evaluated based on speed, scalability of threads, memory hierarchy. Now, it is essential to consider the energy or the power consumed by the system while executing an application. There exist performance analysis tools, such as, Periscope, Scalasca, Vampir, TAU, and Paradyn, which consider hardware based performance bottlenecks and memory hierarchy issues including EnergyAnalyzer which is a dedicated tool for energy analysis purpose. Recently, these tools have focused on doing an automatic tuning of HPC applications which require a wide study of HPC applications in terms of power consumption. This paper aims at experimenting the most commonly used HPC applications and express the HPC application developers or tool developers that power consumption will be higher in certain conditions. We have done the experiments in HPCCLoud Research Laboratory, India. The experimental results were impressive when tested for the energy consumption of HPC applications.**

***Index Terms*: high-performance computing, performance, power consumption, grid computing.**

I. INTRODUCTION

Scientific computing requires an ever-increasing number of resources to deliver results for ever-growing problem sizes in a reasonable time frame [5]. The word computing was used similar with counting and calculating, and the science and technology of mathematical calculations. Computing means using and operating, the computing hardware and also the use of the theoretical concepts of this hardware to complete a task. The computing hardware must be a computer system of any form. Most individuals use some form of computing every day whether they realize it or not. Swiping a debit card, sending an email, or using a cell phone can all be considered forms of computing [4].

Software development [3, 6, 22] is a crucial ever-lasting research domain which undergoes different stages of uncertainties to get through an objective of developing software. In general, the various stages of software development include design, generating code, and resolving bugs. Out of which, resolving bugs or post analysis of software is both time consuming and challenging. This is manifested by a research survey stating that US expends about $59.5 billion dollars each year to solving software bugs [23, 30]. To ease the purpose of debugging and analysis, many researches have been heralded by developing tools that analyze or tune simple to complex software.

Performance analysis tools [10] are important to highlight software bottlenecks and if possible to tune for obtaining better performance results. Commonly speaking, performance is debited to the computer architecture. In the current scenario, performance analysis tools of HPC applications [32] consist of various analysis approaches [8], such as, online vs offline, trace-based vs profile-based, distributed vs centralized, and so forth.

Earlier performance was evaluated based on speed, scalability of threads, memory hierarchy etc. To do maximum number of jobs in a short period of time, it is necessary to increase the number of cores or processors. When large number of processors is working, there is an increase in energy consumption. So it is essential to see of ways to reduce the energy consumption of applications while considering the performance [7].

Most of the performance analysis tools that analyze applications are not designed for analyzing energy consumption of applications. They either do programming language-based analysis or hardware-based analysis, such as, MPI analysis, OpenMP analysis [9], memory leakage analysis, pipeline stall analysis, cache misses, and so forth. However, recent computer architecture designers are more particular about diligently designing their products with energy efficiency [13]. This is because of increased electrical billing and scarce of power generation, especially in developing countries such as India. In addition, a report submitted to the US congress on “Server and Data Centre Energy Efficiency” in 2007 highlights that the energy consumption by US data centres was 61 billion kilowatts-hour in 2006 totalling \$4.5 billion [15].

It is also estimated from various energy modelling approaches [28, 29, 31] that the electrical billing of computer systems surpasses the machine cost within three years of 24x7 utilization of the machine. Energy inefficiency in compute nodes may even lead to environmental hazards, say, carbon emissions due to cooling the machines. Although researchers are aware that software development should consider energy efficiency as a prime concern these days, they are either not suggested with available solutions to mitigate energy inefficiency problem or the solutions are not final – energy aware designs [17, 18, 19, 20, 21, 26] of machines are still under research.

The tuning techniques are used mainly to optimize the code. The code is optimized by performing some kind of code re-transformation, code scheduling to give a performance tuned and a code that consumes less energy than the original code. Various optimization techniques consume different amount of energy. So it is essential to find which code sequence consumes less energy.

The rest of the paper is organized as follows: Section 2 presents the Energy Consumption Analysis on various optimization techiniques. Section 3 presents the results and findings of the analysis. Section 4 presents the conclusion to the paper.

I. ENERGY CONSUMPTION ANALYSIS

There are various optimization techniques available. These optimization techniques help in improving the performance of the application. On improving the performance, these optimization techniques, when used for tuning of the software can also reduce the energy consumption of the applications. The optimization techniques can be categorized into – loop optimization, procedure optimization, software optimization, and program optimization.

2.1 Loop Optimization

Loop optimization [1, 2, 11, 16, 24, 27] technique is done through a sequence of loop transformations. This optimization can be done on the source code or in the intermediate representation. These transformations should be legal and safe so that it does not alter the actual working of the code. There are various loop optimization techniques, such as loop interchange or loop permutation or loop re-ordering, loop fusion, loop unrolling, scalar replacement, loop invariant code motion, redundancy elimination.

Consider the above matrix multiplication code in figure 1. This code performs the matrix multiplication in the normal way without any optimization. The loop transformations will be represented using this code. The study also considered the cases of having tail-call and tail-recursion. Elimination of these are also ways of optimization.

```
void matrix(long double *a, long double *b, long
double *c, long double cons, int n)
{  int i, j, k;
   for(j=0;j<n;j++)
   {    for(i=0;i<n;i++)
            c[j×n+i] = cons × c[j×n+i];
   }
   for(k=0;k<n;k++)
   {    for(j=0;j<n;j++)
        {     for(i=0;i<n;i++)
              {
c[j×n+i]=c[j×n+i]+a[k×n+i]×b[j×n+k];}}}}
```

Fig. 1 Original code for matrix multiplication

2.1.1 Loop Interchange or Loop Re-ordering or Loop Permutation

Loop interchange or loop permutation or loop re-ordering [1, 12, 16] technique interchanges the loops between the inner and the outer loops. These interchange or re-ordering should be performed legally and safe as it must not affect the working of the loop. The reordered loop must compute exactly the same as the loop without reordering.

2.1.2 Loop Fusion

Loop fusion [1, 12, 16] technique integrates two adjacent loops may be into a single loop as long as their data’s are not referenced to each other. This optimization technique reduces the loop overhead. The two loops can be fused only if their iterations are same.

2.1.3 Loop Unrolling

Loop unrolling [1, 12, 16, 24] technique is taken from the concept of loop unroll-and-jam, where selected outer loops are unrolled by a small number of iterations, and the unrolled iterations are jammed inside the innermost loop to promote register reuse.

It is parameterized by the number of loop iterations unrolled for each outer loop. This optimization results in a long sequence of straight line code in the innermost loop body. This decreases the number of times the loop condition is tested and the number of iterations performed.

2.1.4 Scalar Replacement

Scalar replacement [16] is a technique were the array references are replaced by register references. This helps in reducing the memory references.

2.1.5 Loop Invariant Code Motion

In an imperative programming language, it may contain certain codes in which certain expressions can be moved outside the body of the loop without affecting the semantics of the program. Such types of codes are called Loop-invariant code. The movement of these codes is called Loop-invariant code motion [12]. It is a compiler optimization which performs this movement automatically.

2.1.5 Redundancy Elimination

In this technique, the redundant code[1] is eliminated. This is brought about by executing the redundant code only once thus avoiding repeated execution of the same piece of code that gives the same result. After executing the code once, only the results will be used instead of executing redundant code. This helps in reducing the execution time and also helps to reduce the power consumption while using redundancy elimination along with loop unrolling and scalar replacement.

2.2 Procedure Optimization

These optimization techniques apply to the whole procedure. The procedure optimization techniques considered for study were: tail - recursion elimination, tail-call optimization, procedure integration or in-line expansion. Unlike loop optimization techniques, the data flow analysis need not be done. These optimization techniques do not consider the data dependencies.

The following sub-sections explain in detail about the procedure optimization techniques.

2.2.1 Tail-call optimization and tail-recursion elimination

Tail-call optimization [29] and tail-recursion elimination [29] are the procedure optimizations applied on calls. By using these optimization techniques, it possible either to reduce or eliminate significant amount of procedure-call overhead.

Suppose there are two procedures f() and g(). Then a call from f() to g() is a tail-call if the only thing that f() does, after g() returns to f(), is itself return.

The call is tail-recursive if f() and g() are the same procedure; i.e. the function call itself, at the tail of the procedure.

2.2.2 Procedure Integration or In-line Expansion

In this optimization technique, the function calls are replaced with the copies of the procedure bodies. This optimization technique is otherwise known as automatic outlining [27]. It is a useful optimization technique, as it increases the possibilities to optimize the inline procedures, like the procedures having the procedure calls within the loops or a loop that calls a procedure, a procedure whose body itself is a loop or a nested loop, using the loop transformations.

The following are the several issues [27] faced by this optimization techniques:

- the size of the procedure body
- number of calls to the procedure
- whether the procedure is called inside the loop
- whether a particular call includes one or more constant valued parameters

The loop transformations especially, the unroll-and-jam, scalar replacement etc., can be applied to the code.

III. EXPERIMENTAL RESULTS

The analysis was performed in Ivy Bridge processor with 8cores. The analysis was performed for matrix multiplication as it is highly computation intensive and costly operation. Matrix multiplication was done on various sizes of matrices ranging from 100x100 to 275x275.

The power consumption was recorded using a Power Meter. The initial reading was recorded and then the reading ones the code was executed was recorded. The readings were taken in Watts(W).

The initial reading was recorded to be 26.23W without running any of the applications. It was seen that as the application was running for long period of time, the system temperature was increased which increased the power consumption of the system to around 66.74W without any application running.

3.1 Results

The graph below in figure 2 show the power consumed by the application when the size of the matrix is 100x100. It shows that the power consumption while using loop unrolling is significantly reduced.

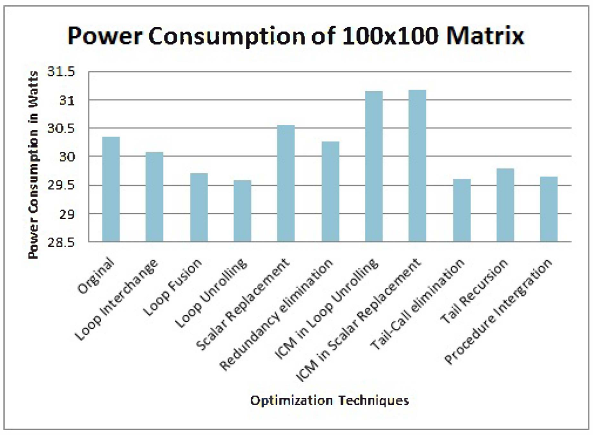

Fig 2. Power consumption of 100x100 matrix

The graph below in figure 3 shows the power consumption by a 150x150 matrix. Most of the optimization techniques have significantly reduced the power consumption when compared to the matrix multiplication without any optimizations.

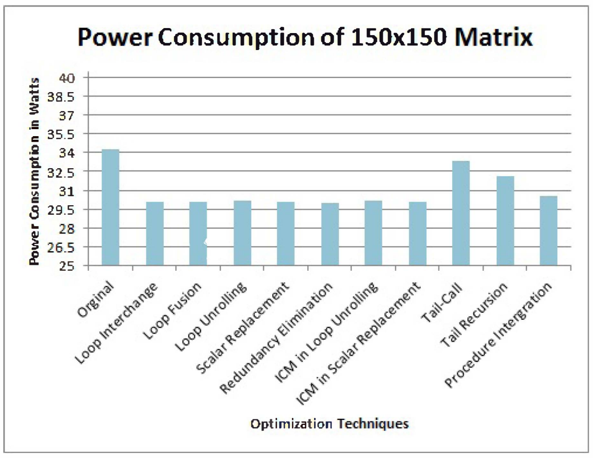

Fig 3. Power consumption of 150x150 matrix

The figure 4 below shows the graph of the power consumption of the matrix of size 175x175. It is seen that using a combination of loop unrolling along with the redundancy elimination of expressions has reduced the power consumed substantially.

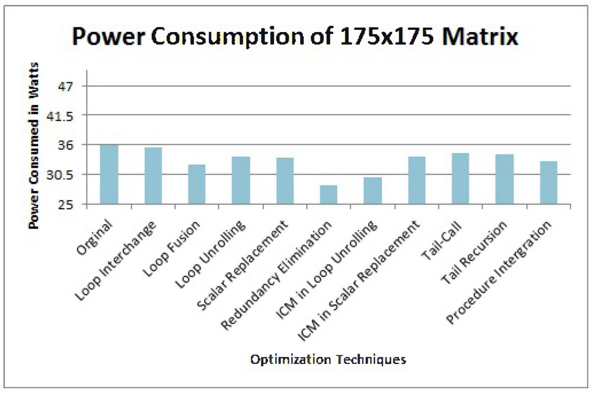

Fig 4. Power consumption of 175x175 matrix

The graph showed in figure 5 is the 200x200 matrix’s power consumption. It is seen that using a combination of loop unrolling along with the invariant code motion has reduced the power consumed substantially when compared to the other optimization techniques. There is only slight reduction in the power consumption of the other optimization techniques.

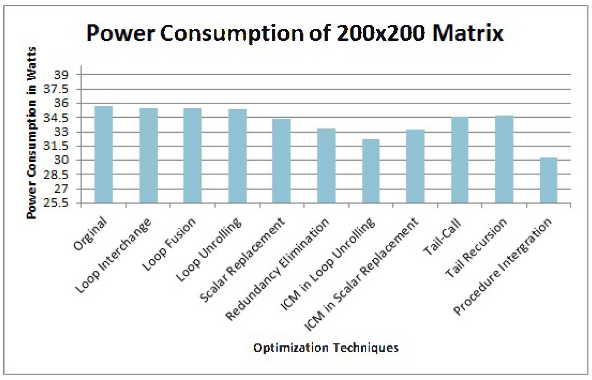

Fig 5. Power consumption of 200x200 matrix

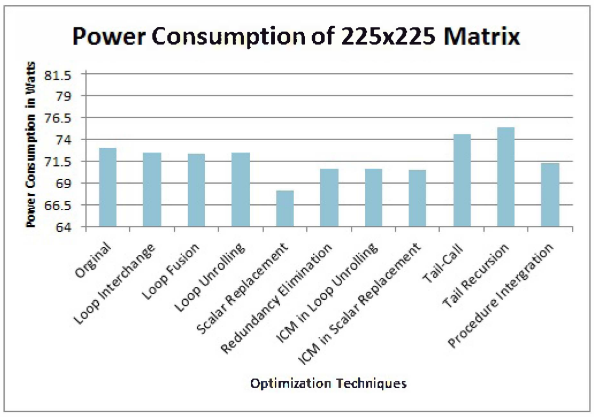

Fig 6. Power Consumption of 225x225 matrix

The graph is figure 6 shows the power consumption of the 225x225 matrix. There is a large reduction in power consumption when using scalar replacement. Larger matrices make use of this optimization technique as this technique also reduces the execution time.

The consumption of power by a 250x250 matrix is shown in the graph in figure 7. Loop unrolling and scalar replacement along with the combinations with redundancy elimination and invariant code motion shows a substantial reduction in power consumption.

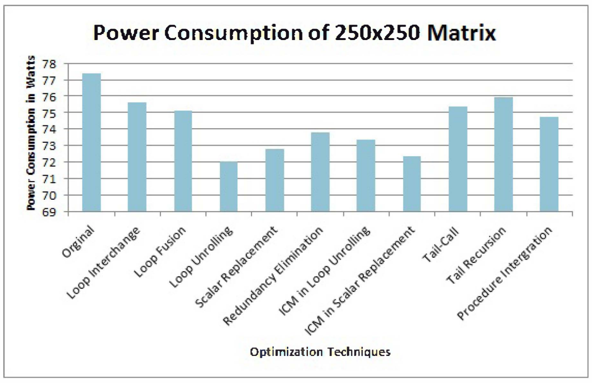

Fig 7. Power Consumption of 250x250 matrix

The power consumption graph of the matrix of size 275x275 is shown below in figure8. This again shows that loop unrolling and scalar replacement along with redundancy elimination and invariant code motion has a major effect in the power consumption. Tail-call and Tail-recursive code have higher power consumption.

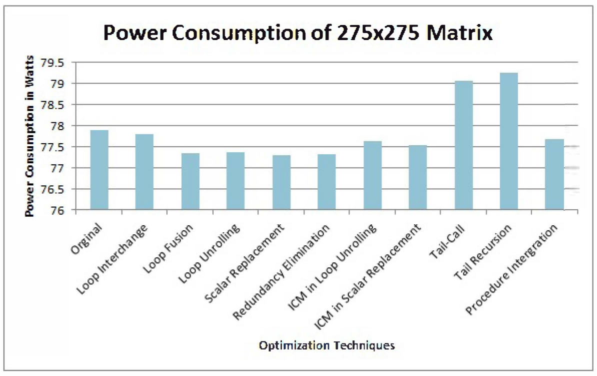

Fig 8. Power Consumption of 275x275 matrix

3.2 Findings

It can be inferred from the above results that applying optimization techniques especially loop unrolling, scalar replacement, redundancy elimination and invariant code motion, helps reduce power consumption of compute intensive applications like matrix multiplication.

IV. CONCLUSION

In the modern era, the need for high performance of application is considered to be one of the most important requirements. While considering the performance, it is also necessary to consider the energy consumption for running the particular application. If the application is modified based on the optimization techniques considered for study a substantial amount of energy can be saved as well as the performance can be improved. The work can be extended by proposing a system that implements these optimization techniques.

REFERENCES

[1] Ananta Tiwari, Hollingsworth,J.K., Chen,C., Hall,M., Liao,C., Quinlan,D.J., Applications: A Case Study', Intl. Journal of High Performance Computing Applications, vol. 25, no. 3, pp. 286-294.

[2] Ananta Tiwari, Chun Chen, Jacqueline Chame, Mary Hall, and Jeffrey K. Hollingsworth (2009) 'A scalable auto-tuning framework for compiler optimization', IEEE Symposium on Parallel and Distributed Processing (IPDPS), pp. 1-12, Rome.

[3] Anuradha Purohit, Arpit Bhardwaj, Aruna Tiwari, Narendra S. Chaudhari (2011) 'Handling the Problem of Code Bloating to Enhance the Performance of Classifier Designed Using Genetic Programming', Proc. of IICAI 2011, pp. 333-342.

[4] Anthony M.Middleton (2010) 'Data-Intensive Technologies for Cloud Computing', Handbook of Cloud Computing, Springer, pp. 83-186.

[5] Alexandru Iosup, Simon Ostermann, M. Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick H.J. Epema (2011) 'Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing', IEEE Transactions On Parallel And Distributed Systems, vol. 22,no. 6,pp. 931-945.

[6] Avadhesh Kumar, Rajesh Kumar, P. S. Grover (2011) 'Unified Cohesion Measures for Aspect-Oriented Systems', International Journal of Software Engineering and Knowledge Engineering, vol. 21, no. 1, pp. 143-163.

[7] Shajulin Benedict (2012), 'Energy-Aware Performance Analysis Methodologies for HPC Architectures - An Exploratory Study' Vol. 35, No. 6, Journal of Network and Computer Applications, Elsevier, pages 1709 - 1719, November 2012.

[8] Benedict,S., Rejitha R.S., Bency Bright,C. (2012) 'Energy Consumption - based Performance Tuning of Software and Applications using Particle Swarm Optimization', IEEE Proc. of CSI Sixth Intl. Conf. on Software Engineering, pp. 1-6, Indore.

[9] Benedict,S., and Michael Gerndt.(2012) 'Automatic Performance Analysis of OpenMP Codes on a Scalable Shared Memory System using Periscope', Applied Parallel and Scientific Computing, LNCS, Springer Publishers, vol. 7134, pp 452- 462.

[10] Benedict,S., Brehm,M., Michael Gerndt, Guillen,C., Hesse,W., Petkov,V. (2010) 'Automatic Performance Analysis of Large Scale Simulations', Euro-Par Workshops 2009, Springer, LCNS, vol. 6043, pp.199-207, Dresden.

[11] Cai,C., Wang,L., Khan,S.U., Tao,J. (2011) 'Energy-aware High Performance Computing – A Taxonomy Study', Proc. of IEEE 17th Intl. Conference on Parallel and Distributed Systems.

[12] Chowdhuri,A., Babu, M.R. (2011) 'Analysis of Loop Optimization Techniques in Multi-Core Environment using MPI-C', International Journal of Computer Information Systems, vol. 2, no. 4, pp.59-65.

[13] Clark T., Yoder A. (2008) 'Best practices for energy efficient storage operations', http://www.snia.org/forums/green, pp. 1-22.

[14] Cristian Tapus, I-Hsin Chung, and Hollingsworth,J.K. (2002) 'Active Harmony: Towards Automated Performance Tuning', Proc of ACM/IEEE Conf. on Supercomputing, pp. 44-55, Baltimore.

[15] EnergyStar Data Center Report to Congress (2007) 'FINAL 7-25-07 – Energy Star Technical Report', http://www.energystar.gov/ ia/partners/proddevelopment /downloads/ EPADatacenter ReportCongressFinal1.pdf.

[16] Faizur,S., Guo,J., Yi,Q. (2011) 'Automated empirical tuning of scientific codes for performance and power consumption', ACM Proc. of Intl. Conf. on High Performance and Embedded Architecture and Compiler, pp.107-116, NY.

[17] Gaujal,B., N. Navet (2011) 'Dynamic Voltage Scaling under EDF Revisited, Real-Time Systems', Springer Verlag, vol. 37, no.1, pp. 77-97.

[18] Intel Inc., 'Data Center Energy Efficiency Using Intel Intelligent Power Node Manager and Intel Data Center Manager', White Paper in http://software. intel.com/sites/datacentre manager/whitepaper.php .

[19] Jing S., Ali S., She K., and Zong Y. (2011) 'State of the art research study for green cloud computing', Journal of SuperComputing, doi:10.1007/s11227-011-0722-1, pp. 1-24.

[20] Lin Y.C., Yi-Ping You, Chung-Wen Huang, Jenq Kuen Lee,Wei Kuan Shih, andTingTing Hwang (2007) 'Energy-aware scheduling and simulation methodologies for parallel security processors with multiple voltage domains', The Journal of Supercomputing, vol. 42, no. 2, pp. 201-223.

[21] Mehta H. (1996) 'Energy characterization based on clustering', Proc. of Design Automation Conf., pp. 702-707, New York.

[22] Neil Walkinshaw, Bernard Lambeau, Christophe Damas, Kirill Bogdanov and Pierre Dupont (2012) 'STAMINA: a competition to encourage the development and assessment of software model inference techniques', Empirical Software Engineering, DOI: 10.1007/s10664-012-9210-3.

[23] NIST Technical Annual Report, (2002) http://www.nist.gov/director/planning/ upload/report02-3.pdf

[24] Patterson, D., Hennessy,J.L. (2007) 'Computer Architecture A Quantitative Approach', 4th ed., Elsevier, Morgan Kufmann Publishers.

[25] Pietron,M., Russek,P., Wiatr,K. (2010) 'Loop Profiling Tool For HPC Code Inspection As An Efficient Method Of FPGA Based Acceleration', Int. J. Applied Mathematics and Computer Science, vol. 20, no. 3, pp. 581–589.

[26] Rizvandi, Nikzad Babaii; Taheri, Javid; Zomaya, Albert Y., Lee, Young Choon (2010) 'Linear Combinations of DVFS-Enabled Processor Frequencies to Modify the Energy-Aware Scheduling Algorithms', Proc. of Cluster, Cloud and Grid Computing (CCGrid), pp. 388 -397, Melbourne.

[27] Robert W. Sebesta, (2006) 'Concepts of Programming Languages', 7th ed., Pearson Education.

[28] Shohrab H.M. and Antiquzzaman M. (2011) 'Cost analysis of mobility protocols', Telecommunication System Journal, Springer Online, pp. 1-14.

[29] Simon Hopkins (2012) 'Shoot the Programmer', http://www.sdtimes.com /link/36712.

[30] SNIA (2012) 'SNIA: Technical Work Groups', www.snia.org/techactivities /work/twgs.

[31] Song S., Su C., Ge R., Vishnu A., and Cameron K.W. (2010) 'Isoenergy efficiency:

An approach to power constrained parallel computation', http://eprints.cs.vt.edu /archive/00001124/01/CSTechReport210.pdf, 2010.

[32] Wang,Z., Shuang,K., Yang,L.,Yang,K. (2012) 'Energy-aware and revenue-enhancing Combinatorial Scheduling in Virtualized of Cloud Datacenter', Journal of Convergence Information Technology, vol. 7, no. 1, pp. 62-70.