Energy Efficient Parallel Matrix-Matrix Multiplication for DVFS-Enabled Clusters

Longxiang Chen, Panruo Wu, Zizhong Chen
University of California, Riverside
Email: {lchen, pwu, chen}@cs.ucr.edu

Rong Ge
Marquette University
Email: rong.ge@marquette.edu

Ziliang Zong
Texas State University-San Marcos
Email: zzll@txstate.edu

***Abstract*—Excessive energy consumption has become one of the major challenges in high performance computing. Reducing the energy consumption of frequently used high performance computing applications not only saves the energy cost but also reduces the greenhouse gas emissions. This paper focuses on developing energy efficient algorithms and software for the widely used matrix-matrix multiplication, so that it is able to consume less energy in a DVFS-enabled cluster with little sacrifice in performance. The state-of-the-art practical parallel matrix-matrix multiplication algorithm in ScaLAPACK partitions matrices into small blocks and distributes matrices using a two-dimensional block cyclic distribution approach. Experimental results demonstrate that our energy efficient matrix-matrix multiplication algorithm can save up to 26.35% of energy with about 1% performance penalty. And the modified PDGEMM of ScaLAPACK is able to save energy more than 20% with less than 2% of performance loss.**

***Index Terms*—Energy efficiency, Dynamic Voltage Frequency Scaling (DVFS), Matrix Matrix Multiplication, ScaLAPACK.**

I. Introduction

As people use the high performance computers more widely and frequently, the energy consumption of these clusters is an essential issue of the world today. A study commissioned by the U.S. Environmental Protection Agency draws a conclusion that the energy consumption of servers almost doubled in five years, from 2000 to 2005, and the electricity consumed by all the worldwide servers costs more than $7 billion. Compared to the total amount of money, about $10 billion, spent in the investment in high performance computing (HPC) technology, we have reached a point that the energy efficient high performance computing should be considered as a limitation for the future of HPC [1]. From the TOP500 list, we can calculate the performance predictions and find that a 100 Petaflops system is most likely to appear in the year 2016. And possibly, the Exasflops systems, which might be in the range of hundreds of Mega-Watts, will come out in 2020 [2]. Improvement of the energy efficiency of HPC is badly needed today and in the future. Energy consumption has a significant influence on the performance of computing. The computers consume more energy as the scale of the HPC clusters increases. Meanwhile, a large number of the scientific applications spend days or weeks in executing, which costs a vast amount of energy consumption. Also the reduction of energy consumption is beneficial to not only the economy, but also the environment.

To reduce the energy consumption and improve the energy efficiency of HPC, we can start from the most widely used operation, matrix-matrix multiplication. This operation plays a critical role in the higher level mathematical libraries, such as LU factorization, QR factorization and Cholesky factorization of LAPACK and ScaLAPACK, because it is the basic matrix operation in most of the numerical applications. The running time of square matrices multiplication is $O(n^3)$, of the rectangular matrices (one $m \times p$ matrix with one $p \times n$ *matrix*) is $O(mpn)$.

In order to improve the energy efficiency, Dynamic Voltage Frequency Scaling (DVFS) is introduced. DVFS is an effective method of reducing the CPU energy consumption by providing "just-enough" computation power. In this paper, we mainly use **Dynamic Frequency Scaling** (also known as CPU throttling) [3], which is a technique in computer architecture that sets the CPU frequency adjusted to the power exactly needed, to demonstrate our idea. Dynamic Frequency Scaling reduces CPU frequency when the process executing on this CPU is not CPU-bounded and increases the CPU frequency when the CPU is busy with computing [4].

In this paper, we measure the energy consumption of matrix-matrix multiplication with the tool – *PowerPack* [5], and compare the results of performance and energy efficiency to the version written by ourselves and the PDGEMM from ScaLAPACK. By applying the Dynamic Frequency Scaling technique to the original matrix-matrix multiplication, we achieve the goal of energy saving.

Our idea of this paper is to set the CPU frequency to the lowest possible value when the CPU is in the idle state or communication, and speeding up the CPU frequency to execute as fast as it can in the computing procedure. The consequence of this method demonstrates the validity of energy saving. The overall energy efficiency is higher and the performance does not decrease drastically. The main contribution of the present paper is the implementation of the matrix-matrix multiplication with DVFS technique and the realization of reducing energy consumption up to 20% with only 2% performance penalty.

The remaining parts of this paper are organized as follows. Section II presents the related work. Section III introduces our energy efficient algorithm of matrix-matrix multiplication. Section IV evaluates our approach experimentally. Section V shows the experimental performance of modified PDGEMM of ScaLAPACK. Section VI concludes the paper.

1530-2016/12 $26.00 © 2012 IEEE
DOI 10.1109/ICPPW.2012.36

II. Related Work

In the past several years, there were few tools could measure the energy consumption of computers, because people focused more on the performance of HPC rather than the energy efficiency. As we mentioned above, today, people start to move part of their attention from performance to energy efficiency due to the increasing energy consumption of HPC. *PowerPack* is the first tool to isolate the power consumption of devices in a high performance cluster and correlate these measurements to application functions [6] [7].

PowerPack is designed and implemented by the SCAPE Laboratory at Virginia Tech, and is an effective tool for energy measurement and analysis. The infrastructure of *PowerPack* is composed of both hardware and software components: the hardware measures the energy consumption directly from the system and the software collects the energy information from the hardware automatically and writes to the file for further processing [8] [9].

With the use of *PowerPack*. we can analyze the effect of Dynamic Frequency Scaling on the energy efficiency and overall performance of the programs. The *PowerPack* gives the information as follows:

- Node name. It specifies the name of the node we run the program on, so we can calculate the energy consumption according to the node we use.
- Time stamp. *PowerPack* contains the time stamps of the program running on the node. The last time stamp minus the first time stamp is the time duration the *PowerPack* measures.
- Energy consumption. The energy consumption of the node records the energy consumed by the node at the time instant of the time stamps.

To calculate the total amount of energy consumption for a specified node, we sum up the energy consumption over the time duration of this node.

The nodal power **P** represents the rate of energy consumption at a discrete point in time, energy **E** describes the total number of joules spent in time interval (t_1, t_2), as a sum of products of average power $\overline{P}$ and delay $(D = t_2 - t_1)$ over all the computing nodes [6]:

$$E = \sum_{1}^{\#node} \int_{t_1}^{t_2} P(t)dt = \sum_{1}^{\#node} \overline{P} \times D$$

From the formula above, we are able to calculate the total amount of energy consumption of each node, and make a comparison between different algorithms.

Two methods were proposed to save energy of multi-core processors with simulation. In the paper of Perdo Alonso and Manuel F. Dolz [10], Slack Reduction Algorithm (SRA) and Race-to-Idle Algorithm (RIA), adjust the CPU frequencies during the execution of the computation to obtain the goal of energy saving. The SRA introduces the dependency DAG to choose the task that can be slowed down without increasing the total execution time of the algorithm. The RIA, on the other hand, executes the tasks at the highest possible frequency, while reduces CPU frequency to the lowest possible value during the CPU idle periods. Both of the methods lead to the energy saving result. But in their paper, they tested the experiment in simulation with a multi-core environment, which would ignore some problems caused by the real computer cluster [11] [12].

In the execution of the parallel programs, the reduction of the CPU frequency can result in the energy consumption. However, it also slows down the computation speed that lowers the overall performance of the program. So it is critical to choose the tradeoff between the performance and energy efficiency of the programs [13] [14]. As we increase the CPU frequency, the performance increases, while the energy efficiency becomes lower as well (lower energy efficiency means higher energy consumption). But if the CPU frequency is too low to execute the program properly, the execution time is much longer than the case in the normal CPU frequency, with the trend of increasing energy consumption [15] [16].

III. Energy Efficient Matrix-matrix Multiplication

In this paper, we first write a simple version of matrix-matrix multiplication for testing, just to see the efficiency of our idea only. If it obtains a significant result, then we can implement this method to the PDGEMM function of ScaLAPACK. We focus on improving the energy efficiency of the original matrix-matrix multiplication running in the highest frequency during the whole procedure. By applying Dynamic Frequency Scaling to the execution suitably, we achieve the goal of energy saving.

A. Mapping Processors

First of all, we write a mapping function to bind each process ID to a specified core in the cluster as:

void mapping(int myrow, int mycol, int DVFS_state)

where **myrow and mycol** are the position of the core in the processor grid, and **DVFS_state** specifies the state (HIGH or LOW). By calling this function, each core can set its CPU frequency to HIGH or LOW as needed.

B. Matrix-matrix Multiplication written for test

In order to evaluate the energy saving goal of our method easily, we write a simple version of matrix-matrix multiplication first. Each core of the computer cluster only contains one local matrix, so that every broadcast procedure transfers the whole local matrix to the cores waiting for the matrix. The simple algorithm is presented in **Algorithm 1**.

After comparing this original algorithm with energy efficient versions, we can see the difference of DVFS application. If it is verified to be beneficial to energy saving, we can then further this approach to more complicated algorithms.

C. Matrix-matrix Multiplication based on DVFS

We implement the matrix-matrix multiplication based on DVFS as **Algorithm 2**. In each loop, we set the CPU frequency to the lowest possible value when the processes are

Algorithm 1 Simple matrix-matrix multiplication

$DVFS(HIGH)$
$i \leftarrow 0$
while $i < n$ **do**
  if $i = row\_number$ **then**
    $row(i)$ $\rightarrow$ Broadcast local matrix to processors of row(i)
  end if
  if $i = column\_number$ **then**
    $column(i) \rightarrow$ Broadcast local matrix to processors of column(i)
  end if
  $Dgemm(A, B, C) \rightarrow$ Local matrix-matrix multiplication
  $Update(i) \rightarrow$ Move to next row(i+1)
  $Synchronize(processes)$ $\rightarrow$ Make sure all the local update finish
end while
where n is the row number of square processor grid.

broadcasting messages to other cores, and increase the CPU frequency to the highest value when the processes are busy in updating the local matrix-matrix multiplication.

From the pre-test we do, we find that the CPU frequency does not affect the performance of broadcast and can finish the same broadcast functions with less energy. As for the computation procedure, the lower the CPU frequency, the more time it needs to finish the computation and wastes more energy. To avoid this, we increase the CPU frequency to the highest value when the process is making the computation.

Based on the two inserted DVFS statements, we achieve the goal of reducing the energy consumption to more than 20%, with the execution time increasing about 2% than the original matrix-matrix multiplication in **Algorithm 1**.

IV. Experimental Evaluation

In this section, we experimentally evaluate the performance and energy efficiency of our approach. First, we do some pre-tests to get relative information before we modify the original programs. Then we compare our approach to the original version based on the analysis of the pre-tests.

A. The Experimental Environment

The experimental environment consists of 8 nodes with Quad-Core AMD Opteron(tm) Processor 2380, the range of frequencies = {0.80, 1.50, 1.80, 2.50} GHz. An attached node installed *PowerPack* as the meter server, is running on the background to measure the energy consumption of all nodes. The meter server detects all the energy consumption of the program running on all the nodes, and records the energy information into its file system.

B. Impact of different frequencies on energy and performance for idle, communication and computation

Before measuring the energy consumption of the programs, we take some pre-tests as follow:

Algorithm 2 Simple matrix-matrix multiplication based on DVFS

$DVFS(LOW)$
$i \leftarrow 0$
while $i < n$ **do**
  if $i = row\_number$ **then**
    $row(i)$ $\rightarrow$ Broadcast local matrix to processors of row(i)
  end if
  if $i = column\_number$ **then**
    $column(i) \rightarrow$ Broadcast local matrix to processors of column(i)
  end if
  $DVFS(HIGH)$
  $Dgemm(A, B, C) \rightarrow$ Local matrix-matrix multiplication
  $Update(i) \rightarrow$ Move to next row(i+1)
  $Synchronize(processes)$ $\rightarrow$ Make sure all the local update finish
  $DVFS(LOW)$
end while
where n is the row number of square processor grid.

- the energy consumption of lowest-, middle-, highest-frequencies for a specified time when the CPUs are in idle state. We compare the energy consumption with different frequencies and learn the effects on the idle. We verify that lower CPU frequency reduces the energy consumption significantly when the CPU is in idle state.
- the energy consumption and execution of lowest-, middle-, highest-frequencies during the broadcast, which is the most frequently called function in parallel communication to update the block matrices. Through this test, we find that the change of CPU frequency reduces the energy consumption and does not affect the performance of broadcast too much.
- the energy consumption and execution time of lowest-, middle-, highest-frequencies to calculate double-precision numbers addition and multiplication. The major computation in the matrix-matrix multiplication is double-precision operations, which has a critical influence on both the energy consumption and performance.

The set of data in Table I demonstrates that energy can be saved by reducing the CPU frequency when it is in idle state.

TABLE I: Idle state, all nodes sleep for 30 seconds

CPU Frequency (GHz)	Energy Consumption (J)
0.80	14862.5
1.80	17791.4
2.50	18998.0

When all nodes with different CPU frequencies sleep for a same period of time, the lower the CPU frequency is, the less energy consumption it costs. Based on this fact, we set the CPU frequency of the cores in idle state to the lowest

frequency to save energy. Table II indicates that different CPU frequencies do not have a significant impact on the execution during the broadcast procedure.

TABLE II: Broadcast, each node broadcasts for 100,000 times with a matrix size of 1,000 double-precision numbers

CPU Freq. (GHz)	Energy (J)	Time (S)
0.80	14507.7	23.41
1.80	22570.6	24.25
2.50	23973.4	22.67

We can see from Table II that broadcasts with different CPU frequencies have the close value (about 23 seconds) of execution time. This is another point to achieve energy saving when the CPU is broadcasting messages to other cores.

A comparison on the effects of different CPU frequencies on computation, we gain the results in Table III.

TABLE III: Computation, each node calculate operations for 1,000,000,000 times with double-precision numbers

CPU Freq. (GHz)	Add. Energy (J)	Time (S)
0.80	23085.1	39.63
1.80	18040.0	22.70
2.50	15248.6	15.53
CPU Freq. (GHz)	Multi. Energy (J)	Time (S)
0.80	29043.9	53.84
1.80	18367.6	22.51
2.50	15053.3	15.57

For the calculation of addition operation, the lower CPU frequency has a longer execution time, with more energy consumption totally. As for the multiplication, the similar result is shown in Table III. It is clear that the execution time is reverse proportional to the CPU frequency. For example, the highest CPU frequency (2.50 GHz) is nearly three times the lowest CPU frequency (0.80 GHz), and the execution times (39.63 seconds in addition and 53.84 seconds in multiplication) of the lowest CPU frequency are also about three times the ones (15.53 seconds in addition and 15.70 seconds in multiplication) in the highest CPU frequency. As a result, we set the CPU frequency to the highest value when it is computing.

C. Time Overhead for Changing CPU Frequency

DVFS technique needs to change the CPU frequency to control the CPU working state during the execution. Due to the frequently changing of CPU frequency, we should figure out the time overhead of DVFS.

The measurement of time overhead is implemented in **Algorithm 3 and Algorithm 4**.

We make sure the Calculate(i) of both algorithms execute in the highest frequency. In **Algorithm 3** without DVFS version, it keeps the CPU in the highest frequency, while in **Algorithm 4** with DVFS version, the CPU frequency is set to the highest value before the Calculate(i) and set back to the lowest value after the execution of the Calculate(i). The time spent in the two DVFS statements is called the time overhead for changing CPU frequency. By testing the two algorithms for the same number of loops, the difference between the two execution times is the time overhead in our experiment.

Algorithm 3 Local update w/o DVFS

$i \leftarrow 0$
while $i < n$ **do**
  $Calculate(i) \leftarrow$ Local matrix-matrix multiplication
  $Update(i) \leftarrow$ Move to next loop i++
end while
where n is the number of loop for test.

Algorithm 4 Local update w/ DVFS

$i \leftarrow 0$
while $i < n$ **do**
  DVFS(HIGH)
  $Calculate(i) \leftarrow$ Local matrix-matrix multiplication
  $Update(i) \leftarrow$ Move to next loop i++
  DVFS(LOW)
end while
where n is the number of loop for test.

TABLE IV: Time Overhead Measurement

Number of loops	Time of Algorithm 3 w/o DVFS (S)	Time of Algorithm 4 w/ DVFS (S)
10,000	14.56	26.41
25,000	36.42	65.91
50,000	72.76	131.28

From Table IV above, we calculate the time overhead for each DVFS loop from the time difference divided by the number of loops.

TABLE V: Time Overhead Result

Number of loops	Time difference (S)	Average time overhead (S)
10,000	11.85	0.001185
25,000	29.49	0.001180
50,000	58.52	0.001170

As shown in Table V, the time overhead of DVFS per loop is so small (less than 0.0012 seconds) that we ignore the time overhead when the execution time is long enough with a limited number of loops in changing CPU frequency. Under these conditions, we choose the large block size of matrix (reduce the times to execute DVFS statements) to perform matrix-matrix multiplication without considering too much of the time overhead of DVFS.

D. Performance of Energy Efficient Matrix-matrix Multiplication

Based on the experimental results above, we apply the DVFS to the program to create the energy efficient matrix-matrix multiplication. We run our programs on a computer

cluster with 8 nodes, each node has 8 threads, at Marquette University. We measure the energy consumption and execution time of two versions (**Algorithm 1** without any modification and **Algorithm 2 with DVFS technique**) of matrix-matrix multiplication, and gain the following Results in Table VI. We run all the programs in 8 nodes (64 cores) cluster, so the size of global matrix is the size of local matrix times 8. And the processor grid is 8 $\times$ 8. **The number of elements in the experiment is the square of size of global matrix.** For example, when the size of the global matrix is 20,000, then the number of elements in this global matrix is 40,0000,000. Here, we use the global size to lable the matrix size, instead of the number of the elements.

TABLE VI: Performance of Execution Time and Energy Consumption

Size of Global Matrix	Time of Algorithm 1 original version (S)	Time of Algorithm 2 with DVFS technique (S)
8,000	21.48	21.80
12,000	51.51	52.02
16,000	98.92	99.72
20,000	165.11	166.58
Size of Global Matrix	Energy of Algorithm 1 original version (J)	Energy of Algorithm 2 with DVFS technique (J)
8,000	42546.20	31392.16
12,000	104438.84	76912.90
16,000	201688.90	153396.52
20,000	261100.26	338673.10

We can see that the energy consumption is reduced significantly with little performance penalty in the energy efficient version. Further analysis will be placed in the following section.

E. Performance and Energy Efficiency Analysis

In this section, we briefly describe the performance model and the energy efficiency model for evaluating the efficiency of the matrix-matrix multiplication based on DVFS.

1) Performance Analysis:

Let T_{tot} be the total execution time of the program, defined the following time-related parameters:

- t_{DVFS} represents the total execution time of the overhead of DVFS.
- t_{dgemm} is the total execution time to call the dgemm() function from ATLAS
- The total execution time of communication between processors is t_{bcast} [17].
- The total execution time of other procedures–write/read a file, memory access and initialization, are included in t_{other}.

We now obtain the total execution time T_{tot} as:

$$T_{tot} = \alpha(n)t_{DVFS} + \beta(s,n,f)t_{dgemm} + \gamma(s,n)t_{bcast} + t_{other}$$

where $\alpha(n), \beta(s,n,f), \gamma(s,n)$ are coefficient functions of **s**, **n** and **f**, related to the size of the local matrix and the number of processors used in the execution procedure respectively. And **f** is the current CPU frequency.

But in this paper, we only analyze the overall performance of the algorithms due to the limitation of the measurement tool.

We define *performance efficiency* (PE) as:

$$\begin{aligned} PE &= \frac{O(N^3)}{T_{mmm}} \\ &= \frac{2 \times N^3}{T_{tot} - t_{other}} \end{aligned}$$

where T_{mmm} is the total execution time of matrix-matrix multiplication and N is the global matrix size.

2) Energy Efficiency Analysis:

Let E_{tot} be the total energy consumption of the programs, defined the following energy-related parameters:

- e_{DVFS} is the total energy consumption of the overhead of DVFS.
- e_{dgemm} is the energy consumption to execute the dgemm() from ATLAS.
- The energy consumption of communication between processes is e_{bcast}.
- The total energy consumption of other procedures are included in e_{other}.

We can gain the total energy consumption E_{tot} as:

$$E_{tot} = \lambda(n)e_{DVFS} + \mu(s,n,f)e_{dgemm} + \nu(s,n)e_{bcast} + e_{other}$$

where $\lambda(n), \mu(s,n,f), \nu(s,n)$ are coefficient function of **s**, **n** and **f**.

We can define *energy efficiency* (EE) as:

$$\begin{aligned} EE &= \frac{O(N^3)}{E_{mmm}} \\ &= \frac{2 \times N^3}{E_{tot} - e_{other}} \end{aligned}$$

where E_{mmm} is the energy consumption caused by the matrix-matrix multiplication and N is the size of the global matrix A and B.

3) Two Evaluation Rates:

Here, we also define two evaluation rates: execution time penalty (ETP) and energy saving rate (ESR) as follow:

$$ETP = \frac{T_{w/o_{DVFS}} - T_{w/_{DVFS}}}{T_{w/o_{DVFS}}}$$

$$ESP = \frac{E_{w/_{DVFS}} - E_{w/o_{DVFS}}}{E_{w/o_{DVFS}}}$$

From all the formulas above, we know that the larger the matrix size **s**, the more processors **n** and the higher the CPU frequency **f** are, the more time it needs to execute, and of course, the more energy it costs. However, if **f** is lower, it saves more energy per time unit, though it consumes more time to run the programs. So a tradeoff is demanded to make a balance between the performance and energy consumption.

4) Analysis of our Results:

As we mentioned above, we calculate the related data for analysis from our results in Table VII.

TABLE VII: Related Data of Execution Time and Energy Consumption

Size of Global Matrix	Algorithm 1 PE (Gflops)	Algorithm 2 PE (Gflops)
8,000	47.67	46.97
ETP = 1.31%		
12,000	67.09	66.44
ETP = 0.99%		
16,000	82.81	82.15
ETP = 0.81%		
20,000	96.91	96.05
ETP = 0.89%		
Size of Global Matrix	Algorithm 1 EE (Mflop/J)	Algorithm 2 EE (Mflop/J)
8,000	24.07	32.62
ESR = 26.21%		
12,000	33.09	44.93
ESR = 26.35%		
16,000	40.62	53.40
ESR = 23.93%		
20,000	47.24	61.28
ESR = 22.91%		

From the result in Table VII, we find the energy efficient matrix-matrix multiplication save energy consumption up to 26.35% with a less than 1% execution time penalty, which is our goal of this paper.

V. Implementation of DVFS in PDGEMM of ScaLAPACK

From the acceptable results we gain above, we start to implement the DVFS technique to the more practical numerical application – ScaLAPACK [18].

In [18], the definition of PDGEMM is Matrix-Matrix Product for a General Matrix. PDGEMM of ScaLAPACK performs combined matrix-matrix operations:

$$\mathbf{C} \leftarrow \alpha\mathbf{AB} + \beta\mathbf{C}$$

It performs as matrix **A** multiples matrix **B** with coefficient α, then plus the original matrix **C** with coefficient β.

A. Performance of PDGEMM

We still test the size of global matrix = {8,000, 12,000, 16,000, 20,000}. The first version of PDGEMM is the original algorithm of ScaLAPACK, and the other version of PDGEMM is modified by ourselves with DVFS technique. We present the results in Table VIII and Table IX.

From Table VIII and Table IX, Figure 1 and Figure 2, we still achieve a significant improvement of the PDGEMM of ScaLAPACK. Even though the Execution Time Penalty(ETP) is a little high (about 2%), the Energy Saving Rate (ESR) is higher than 20%, which means we are able to achieve a more than 20% of energy saving with only little performance penalty.

TABLE VIII: Performance of Execution Time and Energy Consumption of PDGEMM

Size of Global Matrix	Time of PDGEMM original version (S)	Time of PDGEMM with DVFS technique (S)
8,000	14.59	14.90
12,000	35.20	35.26
16,000	65.51	65.88
20,000	109.69	111.70
Size of Global Matrix	Energy of PDGEMM original version (J)	Energy of PDGEMM with DVFS technique (J)
8,000	28888.07	21606.67
12,000	71011.83	54868.23
16,000	134936.33	106752.93
20,000	227581.57	189178.17

TABLE IX: Related Data of Execution Time and Energy Consumption of PDGEMM

Size of Global Matrix	PDGEMM w/o DVFS PE (Gflops)	PDGEMM w/ DVFS PE (Gflops)
8,000	70.19	68.72
ETP = 2.12%		
12,000	98.18	98.01
ETP = 0.17%		
16,000	131.05	124.34
ETP = 0.56%		
20,000	145.87	143.24
ETP = 1.83%		
Size of Global Matrix	PDGEMM w/o DVFS EE (Mflopp/J)	PDGEMM w/ DVFS EE (Mflop/J)
8,000	35.45	47.39
ESR = 25.21%		
12,000	48.67	62.99
ESR = 22.73%		
16,000	60.71	76.74
ESR = 20.89%		
20,000	70.30	84.58
ESR = 16.87%		

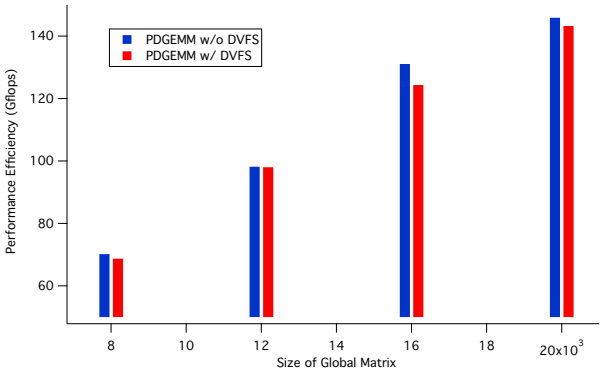

Fig. 1: Performance Efficiency of two PDGEMM

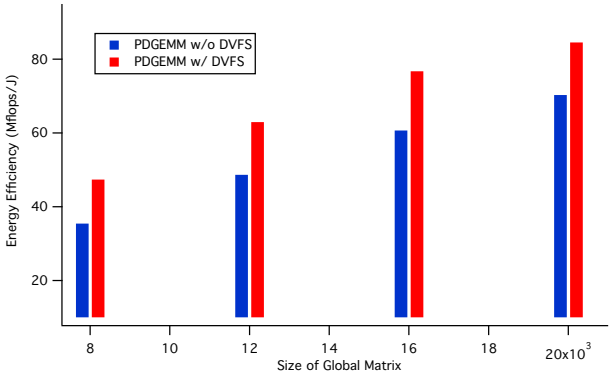

Fig. 2: Energy Efficiency of two PDGEMM

These results from the tables all demonstrate the feasibility of our approach. And we can realize the energy efficient matrix-matrix multiplication – PDGEMM of ScaLAPACK, with the implementation of DVFS technique.

VI. Conclusion

In this paper, we extended the normal matrix-matrix multiplication written by ourselves to verify the DVFS technique for energy saving. By introducing the DVFS technique, energy is saved when the CPU is in the idle state and communication procedure with the lowest CPU frequency. On the other hand, in order to speed up the computation time, we set the CPU frequency to the highest value when the CPU is busy with computing. So we implemented this strategy into the matrix-matrix multiplication to improve the energy efficiency significantly.

The experimental results demonstrate the effectiveness of DVFS for energy saving. We realized the goal of energy saving of PDGEMM of ScaLAPACK up to more than 20% with less than 2% of performance loss. In some cases, this is important because of the increasing energy consumption in the HPC clusters.

Through the comparison between **Algorithm 1 and Algorithm 2**, we can draw a conclusion that slowing down the CPU frequency by DVFS helps save energy consumption during the period of idle and communication states. While in the computation procedure, the higher the CPU frequency is, the more beneficial to both the overall performance and energy efficiency.

Of course, there are still more improvements we can make to increase the energy efficiency with little interference to the performance. When the size of block is too small, a simple application of the widely used DVFS technique, may not reduce the energy consumption because too much overhead may be introduced due to too frequently changing of the CPU frequency. In future work, We will reorganize the algorithm in several different ways to increase the length of computation before communication so that the CPU frequency can be changed less frequently and less overhead is introduced from the application-level DVFS technique. We will also try to extend more energy efficient mathematical algorithms based on this similar approach.

Acknowledgment

We would like to thank the HPCL Lab at Department of Mathematics, Statistics, and Computer Science of Marquette University for providing the computing cluster HPCL with *PowerPack* and Colorado School of Mines for providing computing cluster Alamode Lab.

This research is partly supported by National Science Foundation, under grant #CCF-1118039, #CNS-1116691 and #CNS-1118037.

References

[1] H. D. Simon, "Is HPC Going Green?, http://www.scientificcomputing.com/hpc-going-green.aspc."
[2] H. Gietl and H. Meuer, "The Top Trends in High Performance Computing, http://www.top500.org/blog/2009/05/20/."
[3] M. Chin, "Ausu EN9600GT Silent Edition Graphics Card," *Silent PC Review. p. 5*, April-21 2008.
[4] http://en.wikipedia.org/wiki/Main_Page.
[5] R. Ge, X. Feng, S. Song, H.-C. Chang, D. Li, and K. W. Cameron, "PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications," *IEEE TRANSACTIONS AND DISTRIBUTED SYSTEMS*, May 2010.
[6] S. Song, C.-Y. Su, R. G. A. Vishnu, and K. W. Cameron, "Iso-energy-efficiency: An approach to power-constrained parallel computation," *25th IEEE international Parallel & Distributed Processing Symposium,* **IPDPS**, 2011.
[7] X. Feng, R. Ge, and K. W. Cameron, "Power and Energy Profiling of Scientific Applications on Distributed Systems," *Proc. 19th IEEE Int'l Parallel and Distributed Processing Symp.(IPDPS '05)*, 2005.
[8] S. Song, R. Ge, X. Feng, and K. W. Cameron, "Energy Profiling and Analysis of the HPC Chanllenge Benchmarks," *International Journal of High Performance Computing Applications*, 2009.
[9] P. R. Luszczek, D. H. Bailey, J. J. Dongarra, J. Kepner, R. F. Lucas, R. Rabenseifner, and et al., "The HPC Challenge (HPCC) benchmark suite," *presented at the Proceedings of the 2006 ACM/IEEE conference on Supercomputing Tampa, Florida*, 2005.
[10] P. Alonso, M. F. Dolz, F. D. Igual, R. Mayo, and E. S. Quintana-Ortí, "DVFS-control techniques for dense linear algebra operations on multi-core processors," *Comput Sci Res Dev*, August 2011.
[11] L. T, "Editorial for the first international conference on energy-aware high performance computing," *Comput Sci Res Dev*, 2010.
[12] P. Alonso, M. F. Dolz, R. Mayo, and E. S. Quintana-Ortí, "Energy-aware scheduling of dense linear algebra operations on multi-core processors," *Technical report 2011-04-01, Depto. de Ingenierí a y Ciencia de los Computadores, University Jaume I*, Aprial 2011.
[13] R. Ge, F. Zhou, W.-C. Feng, and K. W. Cameron, "CPU MISER: A Performance-Directed, Run-Time System for Power-Aware Clusters," *Proc. Int'l Conf. Parallel Processing (ICPP)*, 2007.
[14] R. Ge, X. Feng, and K. W. Cameron, "Performance-Constrained Distributed DVS Scheduling for Scientific Application on Power-Aware Clusters," *Proc. ACM/IEEE Supercomputing (SC '05)*, 2005.
[15] V. F. et al, "Exploring the Energy-Time Tradeoff in MPI Programs," *Proc. 19th IEEE/ACM Int'l Parallel and Distributed Processing Symp.(IPDPS)*, 2005.
[16] C.-H. Hsu, "Compiler-directed dynamic voltage and frequency scaling for cpu power and energy reduction," *PhD dissertation*, 2003.
[17] G. Folino, G. Spezzano, and D. Talia, "Performance Evaluation and Modeling of MPI Communications on the Meiko CS-2."
[18] http://www.netlib.org/scalapack/.