Load-Varying LINPACK: A Benchmark for Evaluating Energy Efficiency in High-End Computing

Balaji Subramaniam and Wu-chun Feng
Department of Computer Science
Virginia Tech
{balaji, feng}@cs.vt.edu

***Abstract*—For decades, performance has driven the high-end computing (HEC) community. However, as highlighted in recent exascale studies that chart a path from petascale to exascale computing, power consumption is fast becoming *the* major design constraint in HEC. Consequently, the HEC community needs to address this issue in future petascale and exascale computing systems.**

Current scientific benchmarks, such as LINPACK and SPEChpc, only evaluate HEC systems when running at full throttle, i.e., 100% workload, resulting in a focus on performance and ignoring the issues of power and energy consumption. In contrast, efforts like SPECpower evaluate the energy efficiency of a compute server at *varying* workloads. This is analogous to evaluating the energy efficiency (i.e., fuel efficiency) of an automobile at varying speeds (e.g., miles per gallon highway versus city). SPECpower, however, only evaluates the energy efficiency of a *single* compute server rather than a HEC system; furthermore, it is based on SPEC's Java Business Benchmarks (SPECjbb) rather than a scientific benchmark. Given the absence of a load-varying scientific benchmark to evaluate the energy efficiency of HEC systems at different workloads, we propose the *load-varying LINPACK* (LV-LINPACK) benchmark. In this paper, we identify application parameters that affect performance and provide a methodology to vary the workload of LINPACK, thus enabling a more rigorous study of energy efficiency in supercomputers, or more generally, HEC.

I. Introduction

The Top500 [7] maintains a list of fastest supercomputers in the world by measuring their *performance* using the high-performance LINPACK (HPL) benchmark [2]. However, because power is fast becoming *the* major constraint in high-end computing (HEC)[1], a benchmark that evaluates the *energy efficiency* of a HEC system is needed.

Since applications rarely execute at maximum performance due to the time that is spent waiting for data- and memory-related operations, we need to analyze and understand the energy efficiency of a system at *varying* workloads, particularly in light of the observation that the power profile of a server system is *non-linear* with respect to the performance achieved [9]. The SPECpower benchmark [6] does exactly this by varying the workload of the SPEC Java Business Benchmark (SPECjbb). However, its applicability to HEC is limited for the following reasons:

- SPECpower only evaluates the energy efficiency of a *single* compute server rather than a HEC system.
- SPECpower is a Java-based business transaction workload rather than a scientific benchmark.
- The performance metric for the SPECpower is ssj_ops (i.e., server-side Java operations per second) whereas the metric widely used in the HEC community is FLOPS (i.e., floating-point operations per second).

Conventional HEC benchmarks, on the other hand, adopt a "pedal to the metal" approach and only execute at full throttle, i.e., 100% workload. The most prominent example of such a benchmark is *high-performance LINPACK* (HPL). Unfortunately, calibrating the parameters of HPL to incorporate load variation is *not* nearly as easy as the SPECpower benchmark, where the workload is varied by simply controlling the rate at which requests arrive for processing.

To address the aforementioned issues simultaneously, we propose a load-varying LINPACK (LV-LINPACK) benchmark for which the following contributions are made:

1) The identification of HPL parameters that are critical in determining HPL performance by using principal component analysis (PCA).
2) The correlation of the HPL parameters that impact performance via a feature selection technique.
3) A methodology to vary the workload of HPL by calibrating the parameters identified above, thus enabling the analysis of the power profile of the HEC system under varying workload.
4) The insight in identifying the cause of different power profiles by using the "Performance Application Programming Interface" (PAPI) [5] as well as the correlation between performance-related activities and the power profile of the HEC system under different workloads.
5) The demonstration of the strong correlation between the power profile of an HEC system and data movement from memory.

With the power consumption of HEC systems being non-linear under varying workload, the in-depth analysis of LV-LINPACK can lead us to identify new benchmark metrics and benchmarks for energy-efficient HEC.

The rest of the paper is organized as follows. We start with a brief overview of the HPL benchmark and its parameters

[1]Exascale systems are predicted to consume about 67 megawatts (MW) of power [10]

in Section II. In Section III, we identify the parameters that can have an effect on performance using principal component analysis and feature selection technique. Section IV presents the LV-LINPACK benchmark and the methodologies that we use to create it. Section V describes the experimental platforms and setup used for evaluating the benchmark. Section VI presents our results and the analysis of the power profiles for executing the LV-LINPACK benchmark. The related work is described in Section VII. Section VIII presents our conclusions and future work.

II. Overview of High-Performance LINPACK (HPL)

HPL is a popular benchmark used in the HEC community. It is used by the Top500 List [7] to rank the fastest supercomputers of the world. The benchmark is also used by the Green500 List [11] to rank the most energy-efficient supercomputers. The workload is an algorithm to solve a dense linear system of equations of the form $Ax = b$ of the order N. It uses recursive LU decomposition of matrix A and the solution x is obtained by back substitution. The data is distributed on a two dimensional PxQ grid using a cyclic scheme for better load balance and scalability. Table I shows the HPL input parameters.

#	Parameter Name	Predefined Values (If Applicable)
1	Problem Size (N)	NA
2	Block Size (NB)	NA
3	Process Mapping (PMAP)	0=Row-major, 1=Column-major
4	Rows of Process Grid (P)	NA
5	Columns of Process Grid (Q)	NA
6	Threshold	NA
7	Panel Factorization (PFACT)	0=left, 1=Crout, 2=Right
8	Minimum Columns For Recursion (NBMIN)	NA
9	Subpanel Division (NDIV)	NA
10	Recursive Factorization (RFACT)	0=left, 1=Crout, 2=Right
11	Broadcast Algorithm (BCAST)	0=1rg, 1=1rM, 2=2rg, 3=2rM, 4=Lng, 5=LnM
12	Look Ahead Depth (Depth)	NA
13	Swapping Algorithm (SWAP)	0=bin-exch, 1=long, 2=mix
14	Swapping Threshold	NA
15	Storage for Columns of Upper Triangular Matrix	0=transposed, 1=no-transposed
16	Storage for Rows of Upper Triangular Matrix	0=transposed, 1=no-transposed
17	Equilibration	0=no, 1=yes
18	Memory Alignment	NA

TABLE I
LIST OF HPL PARAMETERS

III. Identifying Important Parameters in HPL

In SPECpower benchmark, different workload is achieved by controlling the arrival rate of the request. For example, if the maximum throughput achieved by a server system is 1000 ssj_ops, then to achieve a workload of 20% throughput requires 200 ssj_ops. So, this can be achieved by either controlling the arrival rate of request to be 400 ssj operations at the start of one second and not submitting any request to be processed for the next second or submitting 200 ssj operations every second. Currently, the SPECpower benchmark uses a negative exponential distribution for controlling the rate of arrival of work requests [6]. However, as shown in Table I, there are 18 parameters that can determine the performance of HPL and eight parameters have predefined values independent of the problem size used. Identifying the important parameters of HPL is not easy as even a change in single parameter can cause huge variations in performance as shown in Fig. 1. Hence, identifying the parameters that affect HPL performance is a very challenging, multi-variable problem. In this section, we determine the HPL parameters which have the most effect on performance.

A. Principal Component Analysis (PCA)

PCA is a dimension extraction technique which reduces the number of dimensions in a data set while preserving the variance of the data set as much as possible. The features extracted are expected to provide the relevant information from the input data in order to produce an output using only the reduced number of parameters. To achieve this, PCA transforms the data into principal components and finds an order in which the fewer dimensions accounts for the variation in the data set [15]. The first identified principal component accounts for the most variability and each of the succeeding components account for as much as variability as possible.

HPL Parameter	Data Set considered
PxQ	Values of PxQ $\forall P * Q \leq$ number of cores
N	10, 20 and 30 percent of memory
NB	16, 32, 64, 96, 128
PFACT	0, 1, 2
NBMIN	2, 4
NDIV	2, 4
RFACT	0, 1, 2
BCAST	0, 1, 2, 3, 4, 5

TABLE II
DATA SET USED FOR PCA

In this section, we apply PCA to HPL parameters. Table II shows the data set that we used. The systems that we used to collect the data set required to perform PCA are called Armor and Ice, respectively, and are described in Section V.

The first step in PCA is to make all the data mean-centered in order to standardize the inputs. This is done by subtracting the respective mean from each of the data. Then we need to study the covariance of the data in order to extract the parameters which account for most variation. To understand the variance accounted by each of these parameters, we calculate the covariance matrix. The covariance between X and Y can be calculated from the Equation (1), where $\bar{X}$ and $\bar{Y}$ refers to the mean of the respective variables and n is the number of samples used. The covariance matrix can be directly calculated by using Equation (2) where each row in matrix Z

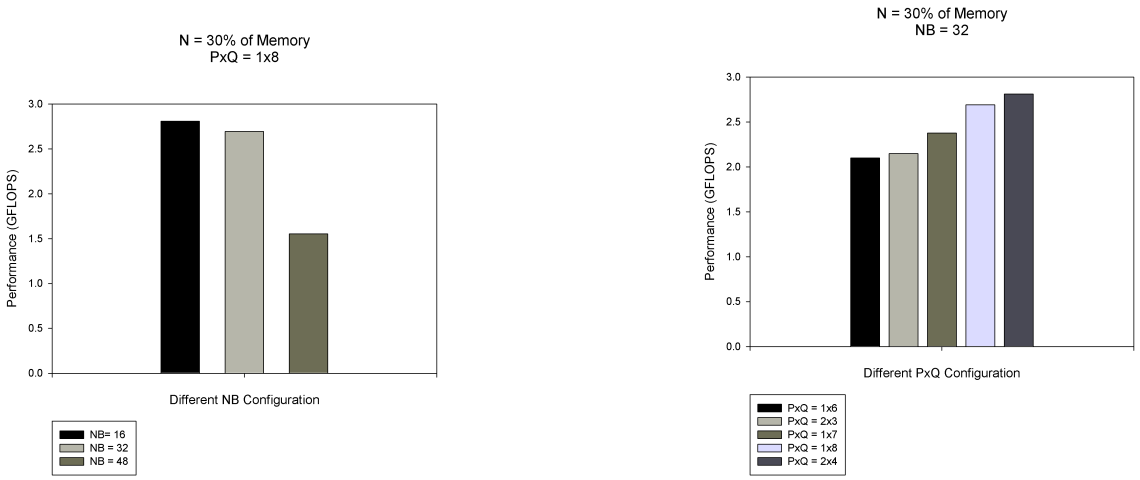

Fig. 1. HPL Performance Variation While Changing Its Parameter. Note: Other Parameters Are Set To Default

is a mean-centered data point, and each column corresponds to one of the HPL parameters.

$$cov(X,Y) = \sum_{i=1}^{n} \frac{(X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \quad (1)$$

$$CovarianceMatrix = \frac{1}{n-1} Z^T Z \quad (2)$$

We identify the principal components of HPL by finding the eigenvalues and eigenvectors of the covariance matrix where each of the eigenvector is the principal component and the corresponding eigenvalue is the strength of the component. Alternatively, we can apply Singular Value Decomposition (SVD) [15] by simply dividing each of the mean centered data in matrix Z by $\frac{1}{\sqrt{n-1}}$. This is because SVD decomposes the matrix as shown in Equation (3) where E(*) represents the eigenvectors of the matrix * and S represents the singular value. To find the principal components we need to find the eigenvectors and eigenvalues of the covariance matrix which essentially means multiplying the matrix Z by $\frac{1}{\sqrt{n-1}}$. The singular values S obtained from the SVD can be used for assessing the strength of the component as the singular values are the square root of eigenvalues of matrix Z. Table III shows the singular values for the HPL parameters on the different experimental platforms.

$$\begin{aligned} SVD(Z) &= USV^T \\ &= E(Z^T Z)\, X\, S\, X\, E(ZZ^T)^T \end{aligned} \quad (3)$$

The results show that N, NB, P and Q account for the greatest variation in the data set. As singular values shows the strength of the respective component, we can neglect the HPL parameters with low singular values. Hereafter in our analysis, we will only use parameters with singular values greater than one. PCA extracts the parameters which account for the most variation, however, it is still unclear as to how these parameters influence performance.

HPL Parameter	Armor	Ice
N	1536.78	2149.45
NB	41.0213	75.3872
P	1.77259	3.1384
Q	1.26315	1.40127
PFACT	0.795845	1.19465
NBMIN	0.298923	1.08907
NDIV	0.588981	0.507249
RFACT	0.549211	0.241647
BCAST	1.14771 e-15	2.03751 e-16

TABLE III
SINGULAR VALUES SHOWING THE STRENGTH OF HPL PARAMETERS

B. Feature Selection Technique

In this section, we show how the parameters identified by PCA influence performance by using feature selection technique. Feature selection is applied as a preprocessing technique to machine learning algorithm and data mining techniques such as neural networks. It is applied to optimally reduce the number of features used based on criterion such as redundancy and degree of relevance. In this paper we use a feature selection technique called fast correlation-based filter solution (FCBF) [19]. This technique uses symmetrical uncertainty to determine whether a feature is relevant or not. We use the FCBF software [1] to apply feature selection on the HPL parameters. The software identifies the features (HPL input parameters) that are relevant to the output (performance achieved) and list them in order of their symmetrical uncertainty. The symmetrical uncertainty normalizes the relevance of the parameters in the range of [0,1] with 1 indicating that the parameter completely predicts the output and 0 indicating that there is no relevance between the parameter and the output. By using symmetrical uncertainty we find the HPL parameters which are not only relevant to performance but also the parameters which are not redundant. In other words, the FCBF software finds the HPL parameters which predict performance and do not have a high enough correlation with other parameters so that the parameter cannot be predicted

by another relevant parameter. Table IV shows the valid parameters and their corresponding symmetrical uncertainty.

System Name	HPL Parameter	Symmetrical Uncertainty
Armor	NB	0.423886
	P	0.355749
	N	0.138793
Ice	P	0.443355
	NB	0.424584
	Q	0.420685
	N	0.173625

TABLE IV
RESULT OF FCBF ON HPL PARAMETERS

NB, P and N were chosen as the most relevant parameters for both the systems and Q was chosen as important for Ice. Since P and Q are the rows and columns of the MPI process grid in the benchmark, it makes sense to use them simultaneously. Moreover, the parameter Q is not chosen due to its redundancy with parameter P and not due to its irrelevance to performance. However, it is clear that N is the relatively less significant. Therefore, from here on the effect of parameter N on performance will be ignored as it is least relevant for both the systems. We will use NB, P and Q to create the LV-LINPACK in the subsequent sections.

C. Understanding the Important Parameters

As we have identified the important parameters, it is necessary to understand these parameters in greater detail so that we have a clear idea about the effects of these parameters on power consumed and performance achieved by HPL benchmark. In this section, we discuss about the four different important parameters (N, NB, P and Q) and how these parameters affect the execution of HPL.

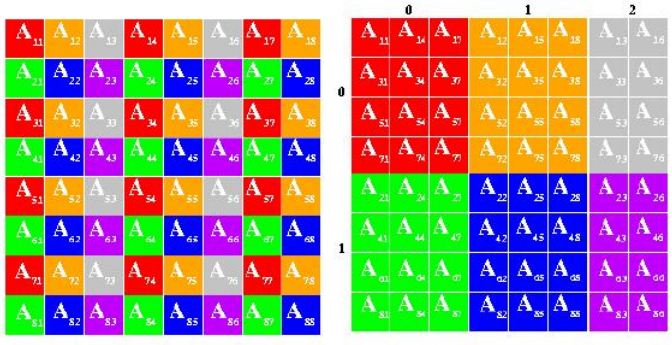

Fig. 2. Data Distribution in HPL Algorithm (A) Coefficient Matrix (B) Distribution of Data on the Process Grid [2]

The HPL benchmark is an algorithm to solve dense linear system of equations of the form $Ax = b$. The problem size N specifies the order of the matrix A which means there are NxN elements in matrix A. In order to solve for x, the coefficient matrix [A b] is first factorized by using LU factorization algorithm and the solution to x is obtained by backward substitution. To perform the factorization, the coefficient matrix is logically partitioned into NB by NB blocks and distributed on to a P by Q grid of process. To better understand the data distribution consider the Figures 2A & 2B. The Figure 2A shows a coefficient matrix of size $N/NB = 8$ and Figure 2B shows how each of these logically partitioned blocks are distributed on the process grid where PxQ = 2x3. In each iteration of the main loop in the benchmark, a panel of NB columns are factorized. Due to the data distribution scheme used in the algorithm, each panel factorization occurs in one column of the processes. The panel factorization lies in the critical path of the HPL benchmark and thus it affects the overall execution time of the algorithm.

IV. Load-Varying LINPACK (LV-LINPACK)

In previous section, we identified the parameters that are important for determining the performance of HPL using the FCBF algorithm. It is necessary that we come up with a methodology to vary the workload. With parameter N having the least relevance to performance on both Armor and Ice, it is clear that the workload of HPL can be varied by calibrating the parameters NB, P & Q. Thus we propose two methodologies to vary the workload of HPL. The first methodology is to fix the PxQ configuration and vary the parameter NB and second methodology is to fix parameter NB and vary PxQ configuration. We then present an algorithm used for executing the LV-LINPACK at different workloads.

A. LV-LINPACK With Fixed PxQ and Fixed NB

LV-LINPACK With Fixed PxQ is used to provide insights into the power profile of the system while using the same number of process. By fixing the PxQ configuration, we can clearly identify the effects of N and NB on the power profile of the system. The benchmark gives a better understanding of the power profile at particular range of workload of the application. LV-LINPACK With Fixed NB gives a better understanding of the power profile of the system while using similar configuration. By fixing NB, we will be able to gain insights into effects of PxQ configuration on the power and performance of the system. We investigate this power profile using PAPI and find that there is correlation between performance related activities such as data cache misses and the power profile of the system at different workloads.

B. Algorithm For Executing LV-LINPACK

The LV-LINPACK is executed by using a series of HPL executions with different input configuration to achieve different workload. As discussed earlier, the different workloads are achieved by calibrating the parameters of HPL. The system dissipates some power even when it is not executing any workload which we call as idle power. While executing subsequent HPL for different workload, it is important that we make sure that the system cools down to its idle power after the end of one HPL execution and before the start of the other execution. For achieving these conditions we use an algorithm similar to that used in SPECpower benchmark [6]. The algorithm followed to execute the LV-LINPACK can be summarized as follows:

1) Ready the system for power measurement
2) Record the idle power
3) Iterate for all workloads:
   - Calibrate the HPL parameters to achieve next incremental workload
   - Wait for the system to dissipate only idle power
   - Record the initial energy value
   - Execute the benchmark
   - Record the final energy value
   - Record the performance achieved
4) End

V. Experimental Platforms and Setup

In this section, we describe about the experimental platforms and power meter setup used for evaluating the LV-LINPACK benchmark.

Two compute nodes were used for the experiments named Armor and Ice. Armor is a two quad-core Intel Xeon E5405 2.00GHz processor. It has 4GB of 667MHz DDR2 SDRAM. Each processor has 12MB of L2 cache shared between 4 cores and 32KB L1 cache for each core. Armor was chosen because 122 systems on the top500 list use Intel Xeon 54xx processors [8]. Ice is a two dual-core AMD Opteron 2218 running at 2.6GHz. It has 4GB of DRAM. Each core has 1MB L2 cache and 64KB L1 cache. We chose Ice to evaluate the behavior of our benchmark on NUMA architectures. Finally, we evaluate the scalability of the proposed benchmark using mid-sized cluster named SystemG. The cluster consists of 324 Mac Pros, each with two quad-core 2.8 GHz Intel Xeon 5462 processors and 8GB of RAM. The nodes are connected over a QDR InfiniBand interconnect technology. We use 64 nodes for a total of 512 cores from SystemG. All the systems use OpenMPI 1.4.1 as the communication library.

The energy consumption of the processing node which executes the benchmark is measured using a power meter. The power meter acts as a intermediate device between the wall power outlet and the node as shown in Figure 3. We use a "Watts Up? PRO ES" power meter to measure the energy consumption. The minimum sampling rate of the power meter is one second. The measuring machine executes the device driver for the power meter and the client of the driver is invoked remotely whenever any energy measurement is required. All the power values reported in this paper are average power i.e. (final energy value - initial energy value)/total execution time.

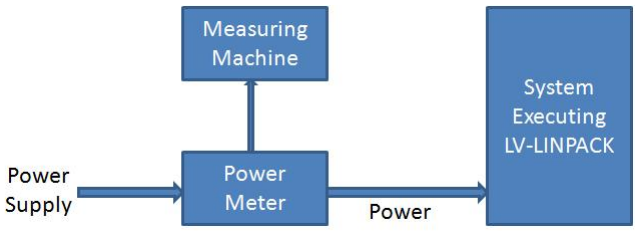

Fig. 3. Experimental Setup

VI. Experimental Evaluation

In this section we evaluate the methodologies described in earlier sections to vary the workload of HPL. The Section VI-A describes the investigation into effects of problem size (N) on power. Section VI-B describes the LV-LINPACK With Fixed PxQ, Section VI-C describes the LV-LINPACK With Fixed NB and finally we discuss about executing LV-LINPACK With Fixed PxQ on SystemG in Section VI-D.

A. Effects of Problem Size (N) on Power

The feature selection technique used in this paper listed problem size (N) as the parameter which least affects performance for both Armor and Ice. However, it is unclear as to how N can affect power. In this section, we investigate the effects of N on power.

In Figure 4, the power profile of Armor is shown for different problem sizes. In each graph, the NB size is fixed and N & PxQ is changed to investigate the power profiles. This is done in order to find the effects of N on power at different % of workloads (i.e. % of maximum HPL performance achieved). For both NB=16 and NB=32, the line graphs in each plots almost overlap indicating that there is not much effect on power as we change N. This is a clear indication that we can safely neglect the effects of N on power for Armor. This helps in reducing one more dimension from our analysis. In Figure 5, the power profile of Ice is shown for different problem size. It is observed that the power profile of Ice is not as smooth as Armor but effects of changing N on power is negligible as the line graphs in each of the plots almost overlap similar to Armor. Due to these reasons, the rest of the paper will not discuss about the effects of N on power. The effects caused due to the parameters NB and PxQ are discussed in detail in the subsequent sections.

B. LV-LINPACK With Fixed PxQ

In this section, we discuss about changing the workload of HPL by fixing PxQ and varying N & NB. To isolate and show how each of the identified parameters affect the performance and power of the system, we execute HPL for three different block sizes (NB=16, 32 & 48) for three different problem sizes by using 10 different configuration of PxQ for Armor and 8 different PxQ configurations for Ice. Such a detailed profiling will give us insights into how the power profile of the system behaves in certain range of workload.

In Figure 6A and 6B, the LV-LINPACK with fixed PxQ configurations of 1x4, 2x2, 1x6 and 2x3 for Armor is shown. For each line graph in a plot, PxQ and N are kept constant and only NB is varied. The Y-axis shows the power dissipated and X-axis shows the % of Workload (i.e. % of maximum HPL performance achieved). The power profile of runs with various N in each graph is different. Higher N gives more or less same performance with slightly lesser power consumption. This is due to the fact that the power reported is average power consumption of the system and with higher N the functional units are idling more waiting for memory related operations thus consuming less power in average. This causes the average

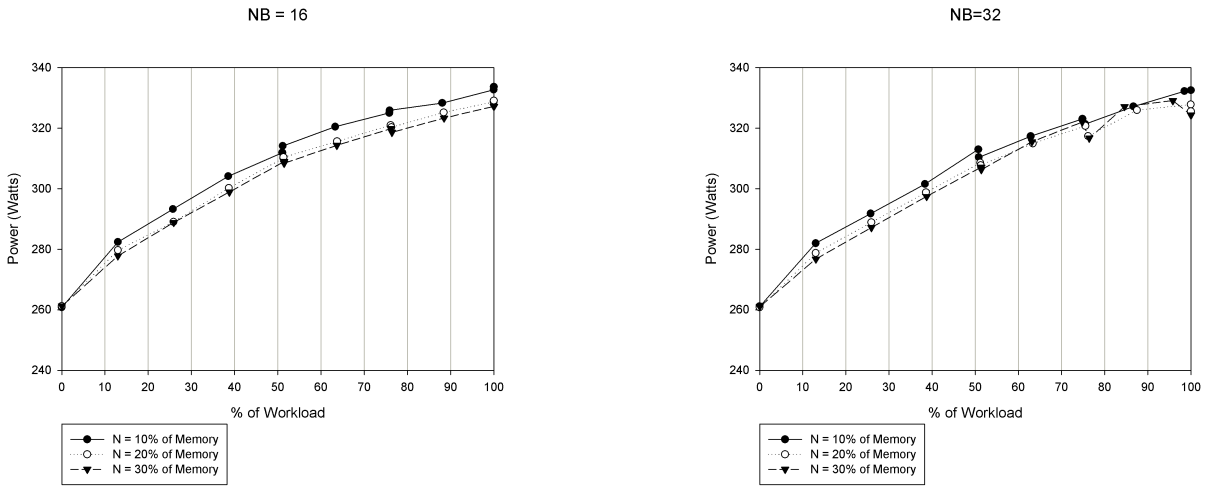

Fig. 4. Power Profile of Armor at Different Problem Sizes

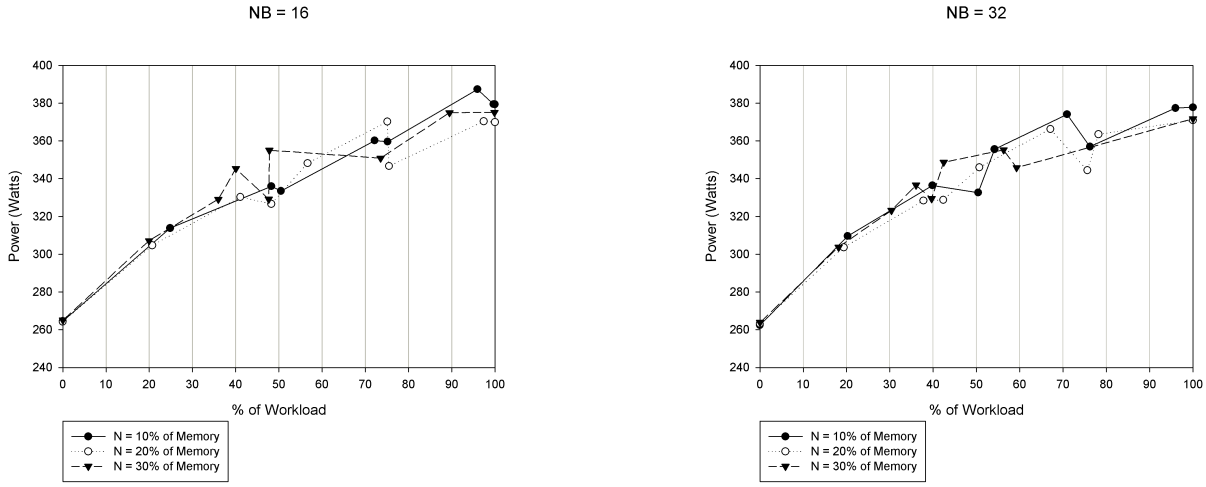

Fig. 5. Power Profile of Ice at Different Problem Sizes

power consumed by the HPL execution with higher N to be lesser. However its effects are negligible as the effects of power lies in the range of approximately 8 watts for same PxQ and NB configurations. It is also noticeable from the graph that configurations 2x2 consumes less power for all Ns while executing at more or less the same performance as configuration 1x4 even though they use the same number of process. This is due to the better load balance achieved inherently by HPL due to cyclic distribution of data [2]. For configuration 1x4, a panel from the coefficient matrix is factorized by a single process. Whereas, for configuration 2x2, a panel is factorized by two processes. Since the panel factorization is the critical path of the overall algorithm, 2x2 performs better than 1x4.

In Figure 6B, similar configuration such as 1x6 and 2x3 execute at same performance but with slightly different power consumption for even same N and NB sizes. There is also degradation of performance of configuration 1x6 for certain block sizes. For example consider the graphs for PxQ configuration 1x6 and 2x3, the performance of 1x6 for N = 30% of memory has high workload variations when compared to 2x3 configurations. This is due to the effect of panel factorization on the overall execution time as described for configurations 2x2 and 1x4.

In Figure 7A and 7B, the LV-LINPACK with fixed PxQ for Ice is shown. We observe higher range of workload being achieved by similar configuration such as 1x4 & 2x2 than Armor. Such effects can be explained by the fact that each core in Ice executes at a faster rate. When block size is increased there is more data to be fetched to perform the panel factorization and thus functional units wait more for data since the operating frequency is higher. This creates the increase in the range of workloads achieved by fixing PxQ and N and just changing NB.

Even though we can vary the workload within in certain range with fixed PxQ, it will not serve as a good benchmark to profile the system at different workloads. Nevertheless, it provides clear insights into how the variation of NB can have

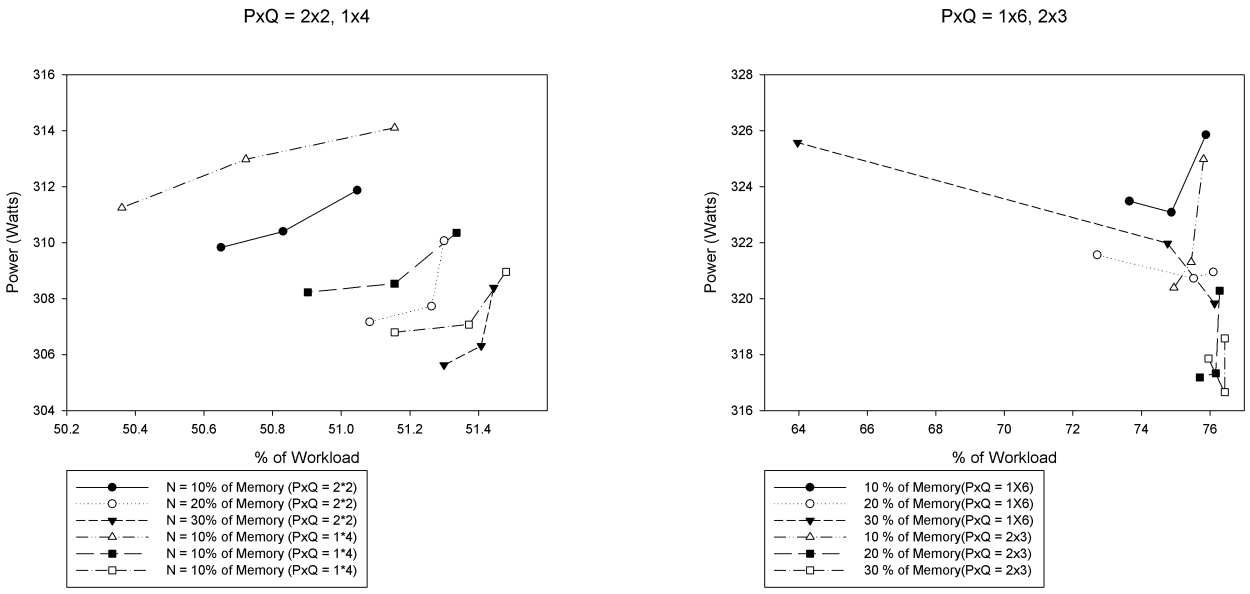

Fig. 6. LV-LINPACK With Fixed PxQ On Armor, (A) Configurations 1X4 and 2x2 (B) Configurations 1x6 and 2x3

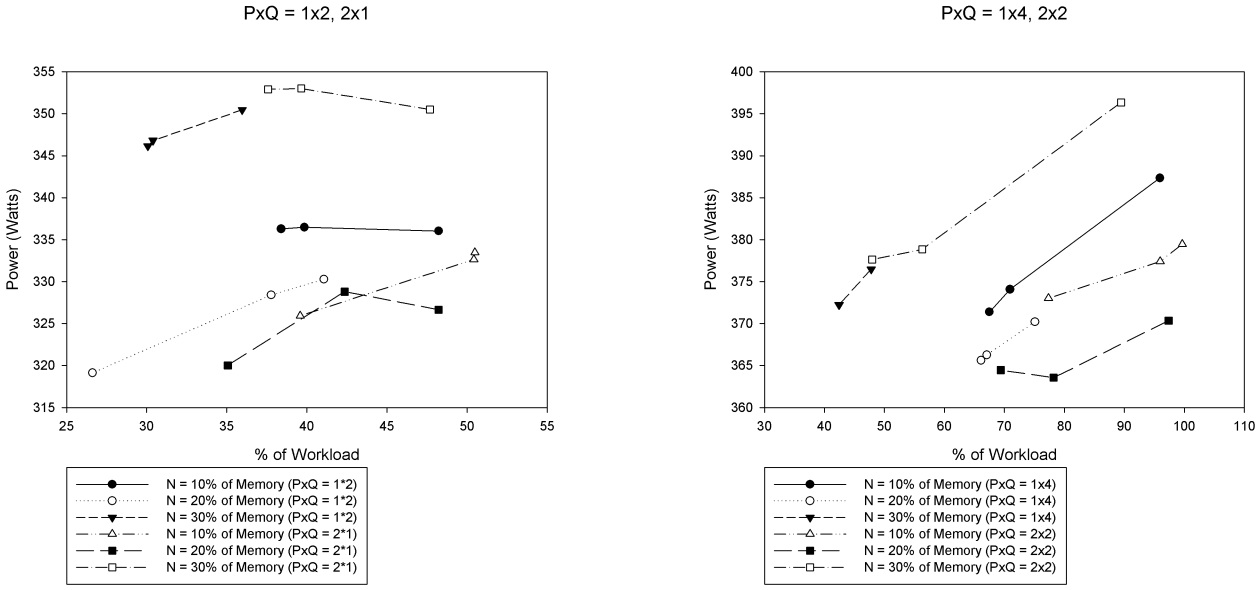

Fig. 7. LV-LINPACK With Fixed PxQ On Ice, (A) Configurations 1x2 and 2x1 (B) Configurations 1X4, 4x1 and 2x2

effects on performance and power consumption of the HPL benchmark even with N and PxQ fixed.

C. LV-LINPACK With Fixed NB

We describe about the workload variations achieved while using fixed NB and changing PxQ in this section. If we change the PxQ configuration, we will be able to achieve greater variation in the workload of HPL. So this benchmark can be actually viewed as connecting the graphs that were shown for LV-LINPACK with Fixed PxQ. These power profiles will provide insights into how similar PxQ configuration can have different workloads for the same N and NB. We investigate these power profiles to find correlation between performance related activities and the power consumption using PAPI.

Figure 8 shows the variation in workload on Armor for problem size of 30% of memory while changing PxQ and keeping block size fixed. The variation in workload for similar configuration grows with increase in NB size. The worst effect can be seen for 1x8 and 2x4 configuration with block size 48. Even though there is a huge variation in performance, 1x8 consumes more power than 2x4 in the same example. We would expect the power consumption of 1x8 configuration to be less. This is because of the fact that 1x8 configuration achieves less performance even while using same number of processes which means the functional units idles more waiting for data and thus consuming less power in average. Then,

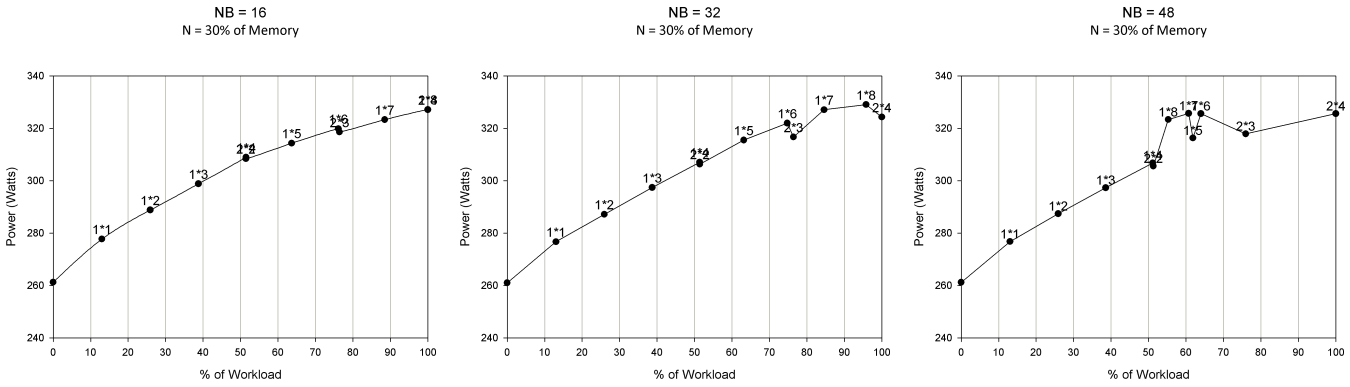

Fig. 8. LV-LINPACK With Fixed NB On Armor

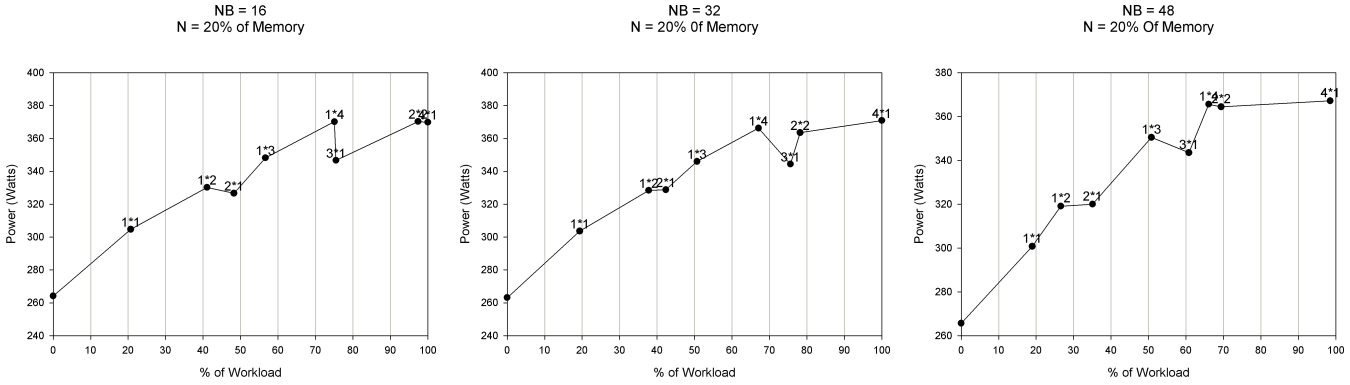

Fig. 9. LV-LINPACK With Fixed NB On Ice

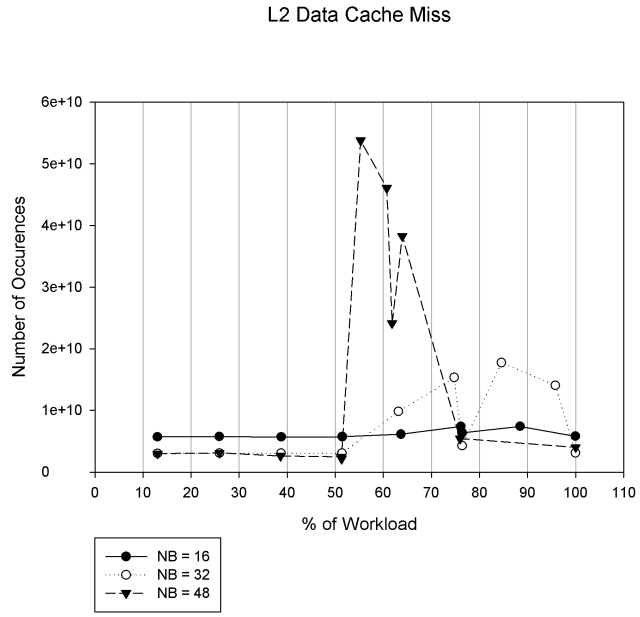

Fig. 10. L2 Data Cache Misses For Armor

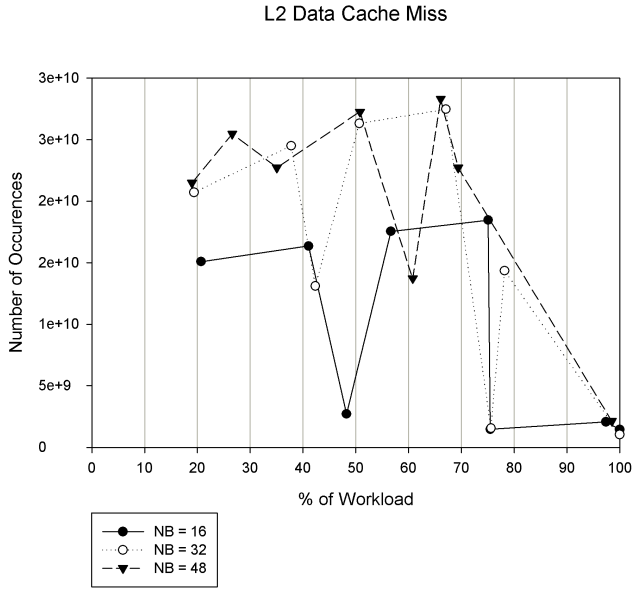

Fig. 11. L2 Data Cache Misses For Ice

why do we observe such effects on power? To investigate and identify the cause for such behavior we use PAPI to relate this power consumption to performance related activities. The investigation into the power profile is discussed later in this section.

The LV-LINPACK with fixed NB for Ice is shown in Figure 9. Similar to Armor, identical configuration have different power and performance profile. The worst effect on performance can be seen with NB = 48. In all the graphs shown 1xY always performs worse than Yx1 and this is due to the way the data is distributed to each process in HPL algorithm. By using configuration 1xY, a panel from the original coefficient matrix is factorized by a single process but in case of Yx1 the factorization is divided between Y processes. As observed even in these plots, though identical configurations of PxQ achieve different performance, they consume more or less the same power which should not be the case.

To investigate into these power profiles, we profiled the benchmark for data cache misses as they can be directly correlated to the performance loss. Figure 10 and 11 shows the L2 data cache misses for the power profiles of the LV-LINPACK with fixed NB shown earlier. It would be expected that the configuration which achieves less performance consumes less power in average as they use same number of processes. When compared with 2x4 configuration, 1x8 configuration should dissipate less power in average as more time by the functional units is spent idling but the data movement caused due to the large number of L2 data cache misses results in higher average power consumption. The L2 data cache misses and thus the memory access due to these misses consume power (power consumption due to the data movement) and also degrades the performance of that execution. Consumption of greater power for configuration which achieves less performance can be directly correlated to L2 data cache misses. For example, consider data points between 55 to 70 percent workload for NB = 48 in Figure 8. All of these executions consumes greater power than the 2x4 configuration which achieves 100% workload and the power consumption can be directly correlated to the difference in L2 data cache misses (Figure 10) which is about a order of magnitude. Such behavior can also be seen with Ice. Consider the LV-LINPACK with NB=32 in Figure 9, there is a difference in the power consumed between configurations 1x3 and 3x1 even though 1x3 achieves far less performance. Like Armor, there is a order magnitude in the L2 data cache misses (Figure 11) for these configurations. Clearly the behavior of the power profile has a strong correlation with the data movement from memory due to L2 data cache misses which suggests that power consumption due to data movement can have significant effect on the power profile of the system at different workloads.

Another interesting observation from the graphs is that the systems are more power efficient at higher workloads. For example consider the graph with NB=16 from Figure 8, the difference in power consumption for workloads from 75% to 100% is about 10 watts whereas the difference in power consumption for workloads from 12% to 40% is greater than 20 watts. The dynamic power range of Armor is about 70 watts, so the increase in system power consumption is 1/7 of the dynamic power to go from 75% to 100% but 2/7 of the dynamic power to move from 12% to 40%. This indicates that these systems are not power proportional with respect to percentage of workload achieved (i.e. the power consumed at different workloads is non-linear with respect to performance achieved). These observations stresses on the need for energy proportional design of system [9]. Such insights into the power profile of the system cannot be derived from a benchmark which executes only at 100% workload.

The LV-LINPACK with fixed NB serves as a good benchmark as we are able to achieve various workloads between 0-100% . We will be able to achieve our primary goal, i.e. to identify the power efficient system at different workloads, with this benchmark.

D. LV-LINPACK on SystemG

In this section, we present the results for executing the LV-LINPACK with fixed PxQ on 64 compute nodes of SystemG. The is done in order to evaluate the scalability of our benchmark. Figure 12 shows the results for executing LV-LINPACK with fixed PxQ. All the results shown use 512 cores in the system i.e, $P * Q = 512$. The results show anomalies in power consumption similar to Ice and Armor. We are particularly interested in the power consumed while executing at performances less than Rmax. We used performance counters to identify whether there is any correlation between power consumed and performance related activities and the results are shown in Figure 13.

$$\text{PCC} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)S_X S_X} \tag{4}$$

There is a strong correlation between L2 data cache misses and the power consumed at certain workload. We used Pearson Correlation Coefficient (PCC) to further quantify the statistical significance of this correlation. PCC is commonly used to understand the degree of dependence (correlation) between two variables. The value of correlation coefficient lie between -1 and $+1$ where $-$ and $+$ imply the negative and positive correlation of the variables respectively. PCC can be calculated by using Equation 4 where X_i & Y_i are the data samples, $\bar{X}$ & $\bar{Y}$ are the respective means, S_X & S_Y are the standard deviations and n is the number of samples. Our analysis shows that the PCC for power consumed and number of L2 data cache misses are high. The PCC for executing LV-LINPACK with N = 10% of memory and NB as 16 & 48 are 0.94 and 0.97 respectively and for executing LV-LINPACK with N = 20% of memory and NB as 16 & 48 are 0.95 and 0.93 respectively. PCC being above 0.9 in all cases indicates a strong correlation. These results motivate the need for optimizing data movement in a scientific applications as a mechanism to conserve energy. It is also observed that the correlation between the power consumed and L2 data cache misses decreases as we move towards 100% workload. This is expected as more computation

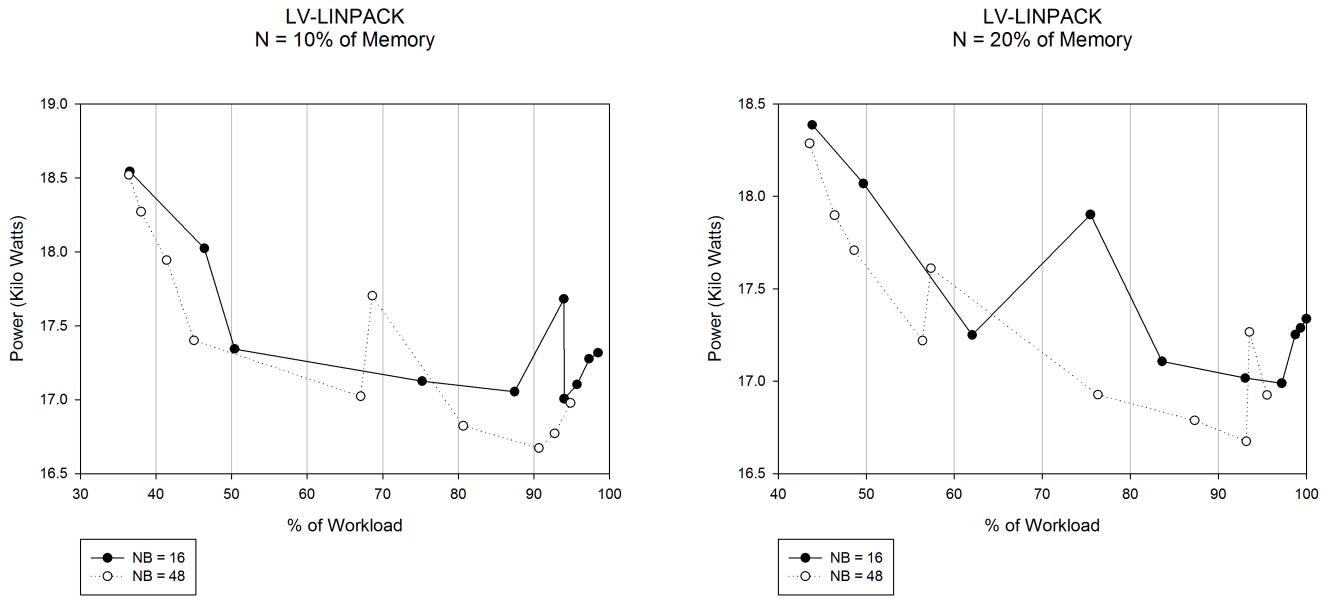

Fig. 12. LV-LINPACK with Fixed PxQ on SystemG

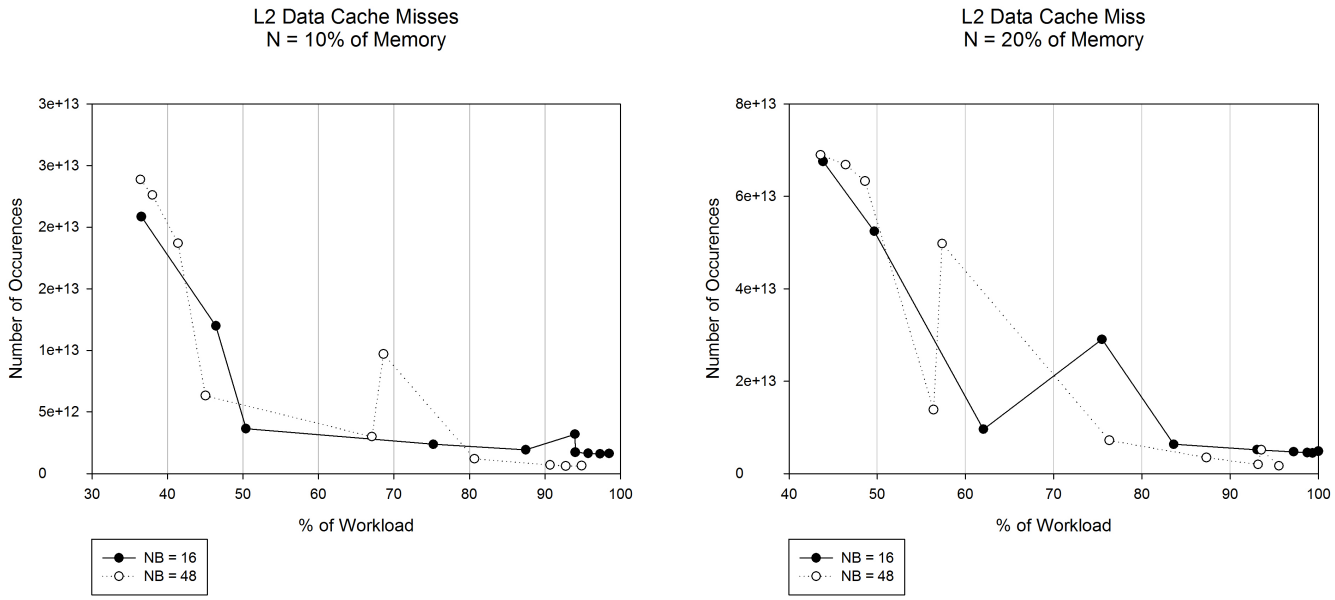

Fig. 13. L2 Data Cache Misses on SystemG

results in higher dynamic dissipation from the CPU and thus CPU contributes relatively more to the power dissipation than data movement.

VII. **Related Work**

To the best of our knowledge, the only load-varying benchmark currently in use is the SPECpower benchmark [6]. The SPECpower benchmark provides a methodology to profile the power of a single server at varying workload. However, as discussed earlier, the workload used in the benchmark is a Java-based transaction workload which is not a good representative of a typical scientific applications and has very limited relevance to the HEC. In this paper we propose a load-varying benchmark for HEC. By using HPL as the benchmark, we not only have a workload relevant to HEC but can have a good estimate of the peak power dissipated at varying workload [16].

In [18], several data mining approaches such as linear regression, M5P, multilayer perceptron and support vector machine have been applied to tune the performance of HPL. The authors use various feature selection techniques to determine the parameters that influence performance and study methodologies which automatically tune the performance of HPL. In this paper, we use principal component analysis

to identify the parameters which has the most variance and feature selection to the find their influence on the performance of HPL. We then use these parameters to create the LV-LINPACK benchmark.

There have only been few detailed studies of HPC application characteristics and its impact on power and energy consumption of the system. In [12], a detailed study of the power and energy profiles of the NAS parallel benchmarks (NPB) [4] is presented. They provide a component-level power profile analysis for executing the benchmark. Their results show that the energy efficiency has a direct correlation with parameters corresponding to performance efficiency. In another work [17], a functional and component-level study of the HPCC benchmarks [3] by using PowerPack [13] software is provided which indicates a correlation between the memory access rate and the power consumption of the system. The power consumption of large-scale HEC systems for executing benchmarks such as HPL and NPB is reported in [16]. In this paper we focus on power profile of the system at different workload to understand its trends which is not addressed in [12], [16]. In [17], the focus is on analyzing energy and power profiles of the HPCC benchmarks.

In Hsu et al. [14], a detailed of study of existing benchmark metrics for evaluating energy-efficiency is presented. Metrics like Energy Delay Product (EDP) and Performance/Power (FLOPS/watt) ratio are analyzed for suitability and the authors conclude that Performance/Power ratio metric is best for evaluating the energy efficiency. In this paper, we focus on creating a new benchmark to evaluate the energy efficiency in HEC.

VIII. Conclusion and Future Work

In this paper we created a load-varying benchmark from HPL. We first identified the parameters that influence the performance of HPL. We then presented LV-LINPACK benchmark (A load-varying version of the HPL) which is achieved by calibrating the parameters such as PxQ and NB. The power profiles of the LV-LINPACK with fixed NB was investigated and we found that there is a correlation between power and performance related activity such as L2 data cache miss at a certain workload and proposed the relation between data movement and the power profiles of the system. Finally we showed the scalability of our benchmark on SystemG and verified the statistical significance of correlation between the L2 data cache misses and the power profile of the system. As a future work, we would like to execute the LV-LINPACK benchmark on a heterogeneous cluster and incorporate load variation in benchmarks from the HPCC benchmark suite.

Acknowledgements

We thank Jeremy Archuleta for his technical input in the initial phase of this project and Yang Jiao for his assistance with the power measurement environment. We are also grateful to Mark Gardner for his critical feedback on the paper.

This project was supported by US National Science Foundation (NSF) under the grant CCF-0848670.

References

[1] Fast Correlation-Based Filter (FCBF) Solution Software. Available at http://www.public.asu.edu/~huanliu/FCBF/FCBFsoftware.html.

[2] High Performance LINPACK (HPL). Available at http://www.netlib.org/benchmark/hpl.

[3] HPC Challenge Benchmarks. Available at http://icl.cs.utk.edu/hpcc.

[4] NAS Parallel Benchmarks. Available at http://www.nas.nasa.gov/Resources/Software/npb.html.

[5] Performance Application Programming Interface (PAPI). Available at http://icl.cs.utk.edu/papi.

[6] SPECpower Benchmark. Available at http://www.spec.org/power_ssj2008.

[7] The Top500 list. Available at http://top500.org.

[8] Top500 List Statistics. Available at http://www.top500.org/stats/list/35/procgen.

[9] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, 2007.

[10] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. S. Williams, K. Yelick, and P. Kogge. Exascale Computing Study: Technology Challenges in Acheiving Exascale Systems.

[11] W. Feng and K. Cameron. The Green500 List: Encouraging Sustainable Supercomputing. *Computer*, 40(12):50–55, 2007.

[12] X. Feng, R. Ge, and K. W. Cameron. Power and Energy Profiling of Scientific Applications on Distributed Systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01*, page 34. IEEE Computer Society, 2005.

[13] R. Ge, X. Feng, S. Song, H. Chang, D. Li, and K. W. Cameron. PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications. *IEEE Transactions on Parallel and Distributed Systems*, 99(2), 5555.

[14] C. Hsu, W. Feng, and J. S. Archuleta. Towards Efficient Supercomputing: A Quest for the Right Metric. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 11 - Volume 12*, page 230.1. IEEE Computer Society, 2005.

[15] I. T. Jolliffe. *Principal Component Analysis*. Series in Statistics. Springer Verlag, 2002.

[16] S. Kamil, J. Shalf, and E. Strohmaier. Power Efficiency in High Performance Computing. In *2008 IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, Miami, FL, USA, 2008.

[17] S. Song, R. Ge, X. Feng, and K. W. Cameron. Energy Profiling and Analysis of the HPC Challenge Benchmarks. *Int. J. High Perform. Comput. Appl.*, 23(3):265–276, 2009.

[18] T. Z. Tan, R. S. M. Goh, V. March, and S. See. Data Mining Analysis to Validate Performance Tuning Practices for HPL. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–8, New Orleans, LA, USA, 2009.

[19] L. Yu and H. Liu. Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. In *The Twentieth International Conference on Machine Learning*, 2003.