Comput Sci Res Dev
DOI 10.1007/s00450-012-0218-0

SPECIAL ISSUE PAPER

GBench: benchmarking methodology for evaluating the energy efficiency of supercomputers

Balaji Subramaniam · Wu-chun Feng

Abstract Recent studies point to power consumption becoming the major design constraint in exascale computing systems. Current scientific benchmarks, such as LINPACK, only evaluate high-performance computing (HPC) systems when running at full throttle, i.e., 100 % workload, resulting in more of a focus on performance than on power and energy consumption. In contrast, efforts like SPECpower evaluate the energy efficiency of a server at varying workloads. This is analogous to evaluating the fuel efficiency of an automobile at varying speeds. However, the applicability of SPECpower to HPC is limited at best.

Given the absence of a scientific benchmark to evaluate the energy efficiency of HPC system at different workloads, we propose GBench (short for Green Benchmark), a methodology to evaluate the energy efficiency of supercomputers and enable a more rigorous study of energy efficiency in HPC. We use LINPACK as a case study and demonstrate the efficacy of our methodology by identifying application parameters impacting performance and providing a systematic methodology to vary the workload of LINPACK.

Keywords Benchmarking · Green supercomputing · Load-varying LINPACK · Energy efficiency · Power

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

B. Subramaniam (✉) · W.-c. Feng
Department of Computer Science, Virginia Tech, Blacksburg, USA
e-mail: balaji@cs.vt.edu

W.-c. Feng
e-mail: feng@cs.vt.edu

1 Introduction

The Top500 [16] maintains a list of the fastest supercomputers in the world by measuring their performance using the high-performance LINPACK (HPL) benchmark [6]. However, because power is fast becoming *the* major constraint in high-performance computing (HPC),[1] a benchmark that evaluates the *energy efficiency* of a HPC system is needed.

Since applications rarely execute at maximum performance due to the time that is spent waiting for data-transfer 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 [1]. The SPECpower benchmark [14] does exactly this by varying the workload of the SPEC Java Business Benchmark (SPECjbb). This is analogous to evaluating the energy efficiency (i.e., fuel efficiency) of an automobile at varying speeds or loads on the engine (e.g., miles per gallon highway versus city). SPECpower's applicability to HPC, however, is limited, as shown in Table 1.

In contrast, conventional HPC benchmarks adopt a "pedal to the metal" approach and execute at full throttle, i.e., 100 % workload. Unfortunately, calibrating the parameters of scientific benchmarks to incorporate load variation is significantly more difficult than with 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 GBench, short for *G*reen *Bench*mark, as a benchmarking

[1] Exascale systems are predicted to consume about 67 megawatts (MW) of power [2].

Published online: 23 May 2012

methodology to evaluate the energy efficiency of supercomputers. To this end, we make the following contributions:

- The identification of benchmark parameters that are critical in determining performance by using a feature-selection technique.
- A methodology to vary the workload of the benchmark by calibrating the parameters identified above, thus enabling the analysis of the power profile of the HPC system under varying workload.

We use HPL, short for High-Performance LINPACK, to demonstrate the efficacy of our methodology. However, we stress that the methodology described to create the benchmark can be used with benchmarks other than HPL as well. In this paper, we create *Load-Varying LINPACK* (LV-LINPACK), a novel adaptation of LINPACK benchmark that enables the following contributions:

- Insight into identifying the cause of different power profiles by using the "Performance Application Programming Interface" (PAPI) [11] as well as the correlation between performance-related activities and the power profile of the HPC system under different workloads.
- Demonstration of the strong correlation between the power profile of an HPC system and data movement to/from memory.

Because the power consumption of HPC systems is nonlinear under varying workload, an in-depth analysis of LV-LINPACK can lead us to identify new benchmark metrics and methodologies for energy-efficient HPC. Furthermore, the energy efficiency of an HPC system is a function of its components, which is a single compute node of a supercomputer in our case. We, therefore, start by evaluating our benchmark methodology on a pair of single compute nodes to analyze its efficacy. Then, we execute the benchmark on an HPC system to demonstrate its scalability and its ability to address the limitations of SPECpower in scientific computing, as previously noted in Table 1.

The rest of the paper is organized as follows. Section 2 identifies the parameters that affect performance using a feature-selection technique. Section 3 presents the LV-LINPACK benchmark and the methodologies that we use to create it. Section 4 describes the experimental setup used for evaluating the benchmark. Section 5 presents our results and the analysis of the power profiles for executing the LV-LINPACK benchmark. Section 6 presents related work while Sect. 7 concludes the paper.

Table 1 SPECpower vs. HPC benchmarks

Features	SPECpower	HPC Benchmarks
Workload	Transaction-based, i.e., SPECjbb	Scientific computing, e.g., HPL
Metric	ssj_ops (i.e., server-side Java operations per second)	FLOPS (i.e., floating point operations per second)
Cluster execution?	No	Yes

2 Identifying critical parameters in HPL

The SPECpower benchmark realizes different workloads by directly 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. This can be achieved by either controlling the arrival rate of request to be 400 ssj_ops at the start of one second and not submitting any request to be processed for the next second or submitting 200 ssj_ops every second. Currently, the SPECpower benchmark uses a negative exponential distribution for controlling the rate of arrival of work requests [14].

In contrast, the HPL scientific benchmark possesses 18 different parameters that can determine the performance of HPL [6, 15] and eight parameters have predefined values that are independent of the problem size used. Identifying the important parameters of HPL is *not* easy, as even a change in a single parameter can cause a significant variation in performance. Hence, identifying the parameters that affect HPL performance is a challenging multivariable problem. In this section, we determine the correlation between HPL parameters and performance to understand which parameters actually affect performance.

2.1 Feature selection

Feature selection is applied as a preprocessing technique to machine-learning algorithms and data-mining techniques such as neural networks. It is applied to optimally reduce the number of features that are used, based on criteria such as redundancy and degree of relevance to the class. In our case, the features are the HPL parameters and the class is its performance.

In this paper, we use a feature-selection technique called *fast correlation-based filter solution* (FCBF) [17]. This technique uses symmetrical uncertainty (SU) [12] to determine whether a feature is relevant. SU is used for evaluating the goodness of a feature for classification. It is based on the concept of information gain theory. Information gain is a measure of decrease in uncertainty of a random variable after observing another variable. $IG(X|Y)$, given by Eq. (1), is the information gained on X due to Y, where $H(X)$, given by Eq. (2), is the entropy or measure of uncertainty of a random variable X, $H(X|Y)$, given by Eq. (3) is the entropy of

X after observing Y and $P(X)$ and $P(X|Y)$ are the probability of X and probability of X given Y, respectively.

$$IG(X–Y) = H(X) - H(X|Y) \tag{1}$$

$$H(X) = \sum_{i=1}^{n} P(x_i) \log\big(P(x_i)\big) \tag{2}$$

$$H(X–Y) = \sum_{i=1}^{n} P(y_i) \sum_{j=1}^{n} P(x_i|y_i) \log\big(P(x_i|y_i)\big) \tag{3}$$

$IG(X|Y)$ is biased towards features that have more values in the analysis. Moreover, we need a metric that normalizes the output so that a fair comparison can be made between the features even though the values of the features lie in different ranges.

SU, given by Eq. (4), eliminates the bias due to using more values for one feature when compared to other features. Moreover, SU normalizes the relevance of the parameters in the range of [0, 1] with 1 indicating that the feature completely predicts the class (in our case, performance) and 0 indicating that there is no relevance between the feature and the class. By using the FCBF algorithm, we find the HPL parameters that are not only relevant to performance but also the parameters which are not redundant. In other words, FCBF finds the HPL parameters that predict performance and that do not have a high enough correlation with other parameters so that the parameter cannot be predicted by another relevant parameter.

$$\mathrm{SU}(X, Y) = 2\frac{IG(X|Y)}{H(X) + H(Y)} \tag{4}$$

We use FCBF software [4] to apply feature selection on the HPL parameters. The software identifies the features (i.e., HPL input parameters) that are relevant to the output (i.e., performance achieved) and list them in order of their SU. Table 2 shows the data set that we used. As noted earlier, we start by evaluating our methodology on individual compute nodes. The compute nodes used to collect the data set required to perform FCBF are called *Armor* and *Ice* and are described in Sect. 4. Table 3 shows the parameters and their corresponding SU.

NB, Q and N were chosen as important parameters on both the compute nodes. N is the least significant parameter. Therefore, N is used as a secondary parameter to vary the workload. NB is used as one of primary parameters as it is has the most significance. Q is used as the other primary parameter. However, Q cannot be varied independent of P since P and Q are the rows and columns of the MPI process grid, respectively, in the HPL benchmark. Moreover, P is eliminated by FCBF because of the correlation with Q (recall that FCBF reports only the significant parameters that do not have a high enough correlation with other parameters) and not because of its insignificance. As a result, we chose to use both P and Q simultaneously to run the benchmark using different MPI process configuration. We use NB, Q and P as the primary parameters to create the LV-LINPACK below.

Table 2 Data set used for feature selection

HPL parameter	Data set considered
$P \times Q$	Values of $P \times Q \; \forall P * Q \leq$ number of cores
N	5,10, 15 and 20 percent of memory
NB	16, 32, 64, 96, 128
PFACT	0, 1, 2
NBMIN	2, 4
NDIV	2, 4
RFACT	0, 1, 2
Depth	0, 1, 2

Table 3 Result of FCBF on HPL parameters

System name	HPL parameter	Symmetrical uncertainty
Armor	NB	0.233
	Q	0.144
	N	0.018
Ice	Q	0.229
	NB	0.123
	N	0.090

Note: All the other parameters had SU less than 0.01

3 LV-LINPACK

In previous section, we identified the parameters that are important for determining the performance of HPL using the FCBF algorithm. Based on this identification, we propose two methodologies to vary the workload of HPL. The first methodology fixes the $P \times Q$ configuration and varies the parameter NB while the second methodology fixes the parameter NB and varies the $P \times Q$ configuration. We present an algorithm used for executing the LV-LINPACK at different workloads in this section.

The LV-LINPACK benchmark executes a series of HPL runs with different input configurations to achieve different workloads. 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 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 [14].

The algorithm to execute 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

4 Experimental setup

Armor is a dual quad-core Intel Xeon E5405 processor at 2 GHz with 4 GB of 667-MHz DDR2 SDRAM. Each processor has 12 MB of L2 cache shared between 4 cores and 32 KB L1 cache for each core. Ice is a dual dual-core AMD Opteron 2218 running at 2.6 GHz. It has 4 GB of DRAM. Each core has 1-MB L2 cache and 64-KB 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 a mid-sized cluster named SystemG. The cluster consists of 324 Mac Pros, each with dual quad-core 2.8-GHz Intel Xeon 5462 processors and 8-GB of RAM. The nodes are connected over a QDR InfiniBand interconnect technology. We use 64 nodes for a total of 512 cores from SystemG. We use a "Watts Up? PRO ES" power meter to measure the energy consumption, as shown in Fig. 1. All the power values reported in this paper are average power.

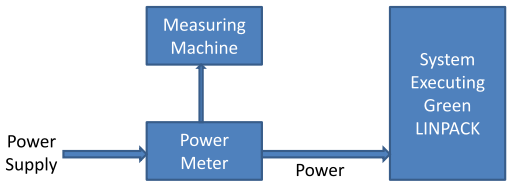


Fig. 1 Experimental setup

5 Experimental evaluation

In this section, we evaluate the methodologies described in earlier sections to vary the workload of HPL. Section 5.1 describes the LV-LINPACK with fixed $P \times Q$, Sect. 5.2 describes the LV-LINPACK with fixed NB, and finally we discuss the execution of LV-LINPACK with fixed $P \times Q$ on our SystemG supercomputer in Sect. 5.3.

5.1 LV-LINPACK with Fixed $P \times Q$

In this section, we discuss about changing the workload of HPL by fixing $P \times Q$ 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 $P \times Q$ for Armor and 8 different $P \times Q$ configurations for Ice. Such

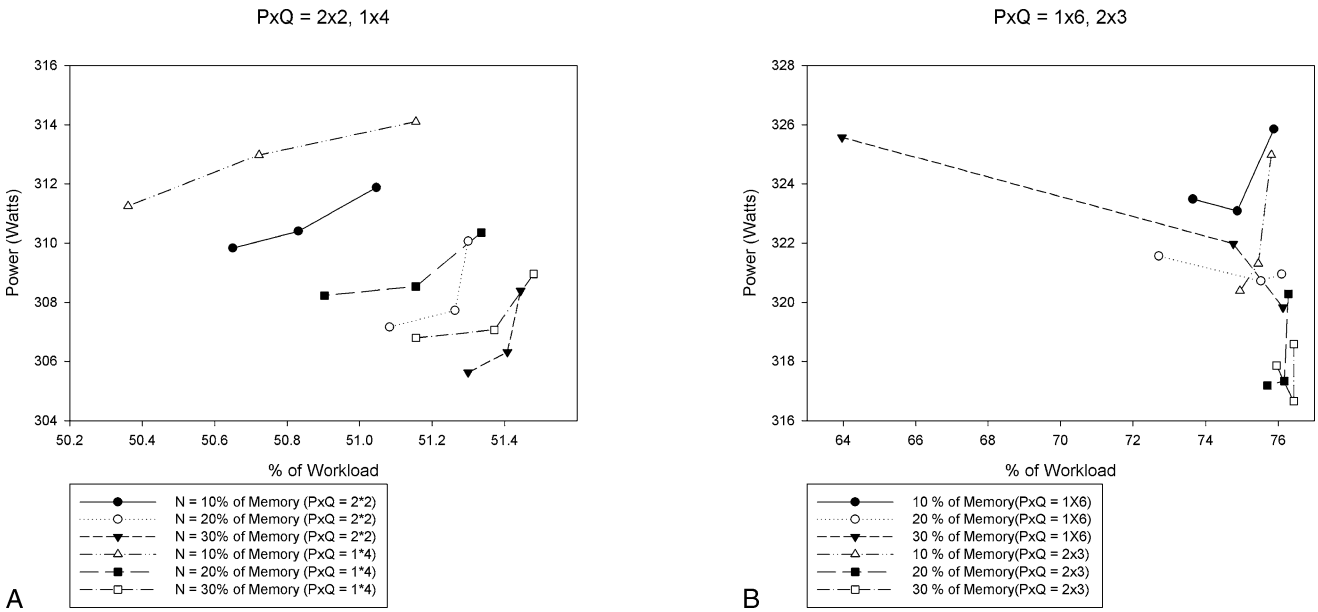


Fig. 2 LV-LINPACK with fixed $P \times Q$ on Armor, (**A**) Configurations 1×4 and 2×2 (**B**) Configurations 1×6 and 2×3

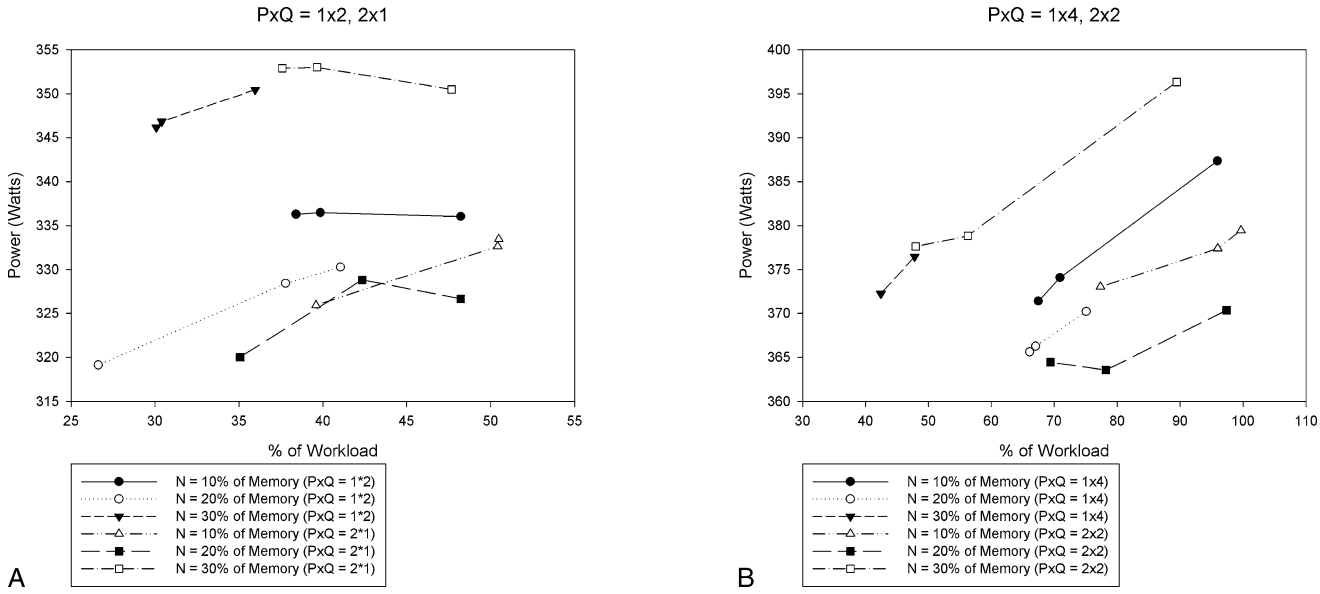


Fig. 3 LV-LINPACK with fixed $P \times Q$ on Ice, (**A**) Configurations 1×2 and 2×1 (**B**) Configurations 1×4 and 2×2

a detailed profiling will give us insight into how the power profile of the system behaves in certain range of workload. In Fig. 2A and 2B, the LV-LINPACK with fixed $P \times Q$ configurations of 1×4, 2×2, 1×6 and 2×3 for Armor is shown. For each line graph in a plot, $P \times Q$ 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. However its effects are negligible as the effects of power lies in the range of approximately 8 watts for same $P \times Q$ and NB configurations. It is also noticeable from the graph that configurations 2×2 consumes less power for all Ns while executing at more or less the same performance as configuration 1×4 even though they use the same number of processes. In Fig. 2B, similar configuration such as 1×6 and 2×3 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 1×6 for certain block sizes. For example consider the graphs for $P \times Q$ configuration 1×6 and 2×3, the performance of 1×6 for $N = 30$ % of memory has high workload variations when compared to 2×3 configurations. This is due to the effect of panel factorization on the overall execution time as described for configurations 2×2 and 1×4.

In Fig. 3A and 3B, the LV-LINPACK with fixed $P \times Q$ for Ice is shown. We observe different workload being achieved by similar configuration such as 1×4 & 2×2 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 $P \times Q$ and N and just changing NB. Even though we can vary the workload within in certain range with fixed $P \times Q$, it will not serve as a good benchmark to profile the system at different workloads. Nevertheless, it provides clear insight into how the variation of NB can have effects on performance and power consumption of the HPL benchmark even with N and $P \times Q$ fixed.

5.2 LV-LINPACK with fixed NB

We describe about the workload variations achieved while using fixed NB and changing $P \times Q$ in this section. If we change the $P \times Q$ 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 $P \times Q$. These power profiles will provide insight into how similar $P \times Q$ configuration can have different workloads for the same N and NB.

Figure 4 shows the variation in workload on Armor for a problem size of 30 % of memory while changing $P \times Q$ and keeping the block size fixed. The variation in workload for similar configurations grows with an increase in NB size. The worst effect can be seen for 1×8 and 2×4 configuration with block size 48. Although there is a huge variation in performance, 1×8 consumes more power than 2×4 in the same example. We would expect the power consumption of 1×8 configuration to be less as the 1×8 configuration achieves less performance even while using same number of

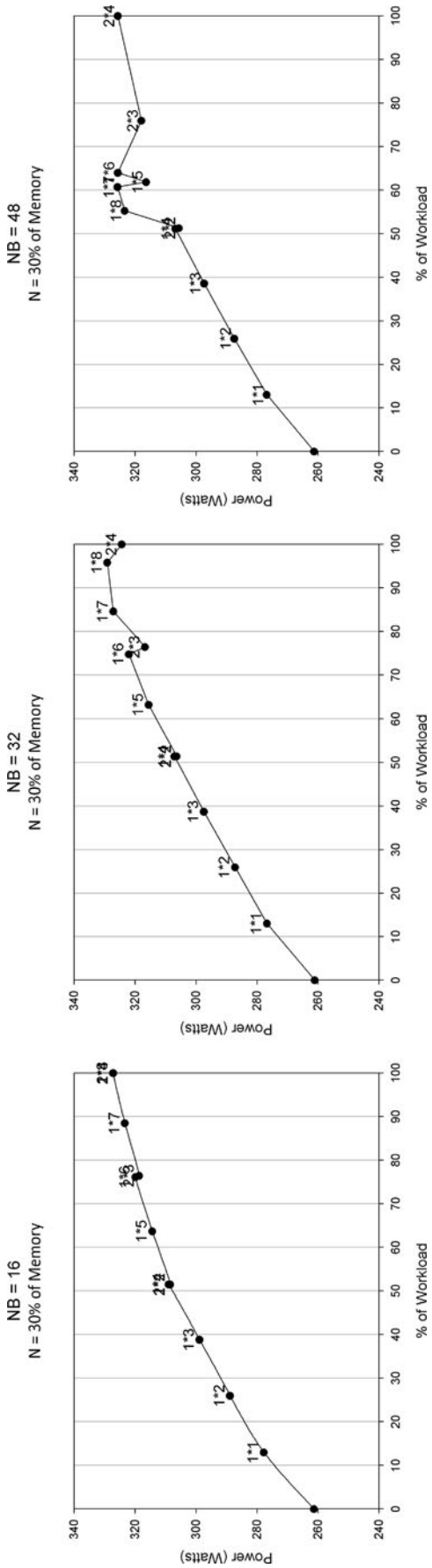


Fig. 4 LV-LINPACK with fixed NB on Armor

processes which means the functional units idle more waiting for data and thus consuming less power on average.

The LV-LINPACK with fixed NB for Ice is shown in Fig. 5. The worst effect on performance can be seen with NB = 48. In all, the graphs shown $1 \times Y$ always performs worse than $Y \times 1$. This is due to the way the data is distributed to each process in HPL algorithm. By using configuration $1 \times Y$, a panel from the original coefficient matrix is factorized by a single process but in case of $Y \times 1$ the factorization is divided between Y processes. As observed even in these plots, though identical configurations of $P \times Q$ achieve different performance, they consume more or less the same power which should not be the case. Then, 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.

To investigate these power profiles, we profiled the benchmark for data-cache misses as they can be directly correlated to the performance loss. Figures 6 and 7 show the L2 data-cache misses for the power profiles of the LV-LINPACK with fixed NB shown earlier. We expect that the configuration that achieves less performance consumes less power on average as they use same number of processes. When compared with the 2×4 configuration, the 1×8 configuration should dissipate less power on 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 degrades the performance of that execution.

Consumption of greater power for configurations that achieve less performance can be directly correlated to L2 data-cache misses. For example, consider data points between 55 to 70 % workload for NB = 48 in Fig. 4. All of these executions consume greater power than the 2×4 configuration which achieves 100 % workload and the power consumption can be directly correlated to the difference in L2 data-cache misses (Fig. 6) which is about a order of magnitude. Such behavior can also be seen with Ice. Consider the LV-LINPACK with NB = 32 in Fig. 5, there is a difference in the power consumed between configurations 1×3 and 3×1 even though 1×3 achieves far less performance. Like Armor, there is a order of magnitude difference in the L2 data-cache misses (Fig. 7) 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 Fig. 4,

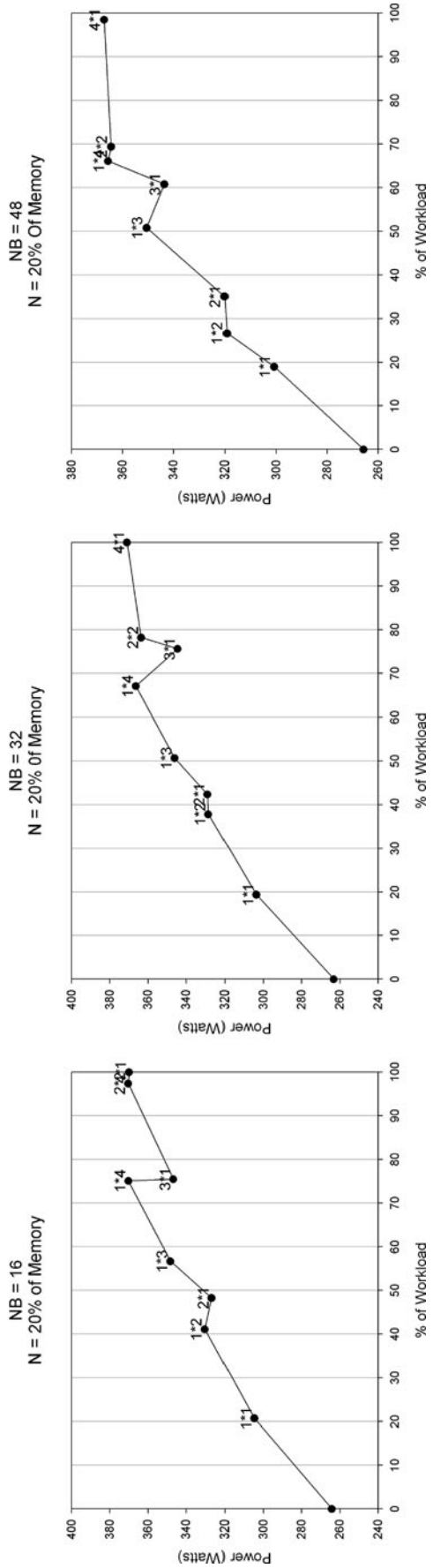


Fig. 5 LV-LINPACK with fixed NB on Ice

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 energy proportional with respect to percentage of workload achieved (i.e., the power consumed at different workloads does not increase perfectly linearly with respect to performance achieved). These observations stress on the need for energy proportional design of system [1]. Such insight 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.

5.3 LV-LINPACK on SystemG

In this section, we present the results for executing the LV-LINPACK with fixed $P \times Q$ on 64 compute nodes of SystemG. The is done in order to evaluate the scalability of our benchmark. Figure 8 shows the results for executing LV-LINPACK with fixed $P \times Q$. 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 the achieved highest performance. We used performance counters to identify whether there is any correlation between power consumed and performance related activities and the results are shown in Fig. 9.

$$\text{PCC} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)S_X S_X} \tag{5}$$

There is a strong correlation between L2 data cache misses and the power consumed at certain workload. We used the 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 Eq. (5) 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 results in higher dynamic dissipation from the CPU and thus CPU contributes relatively more to the power dissipation than data movement.

6 Related work

To the best of our knowledge, the only load-varying benchmark currently in use is the SPECpower benchmark [14]. 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 HPC. In this paper we propose a load-varying benchmark for scientific computing.

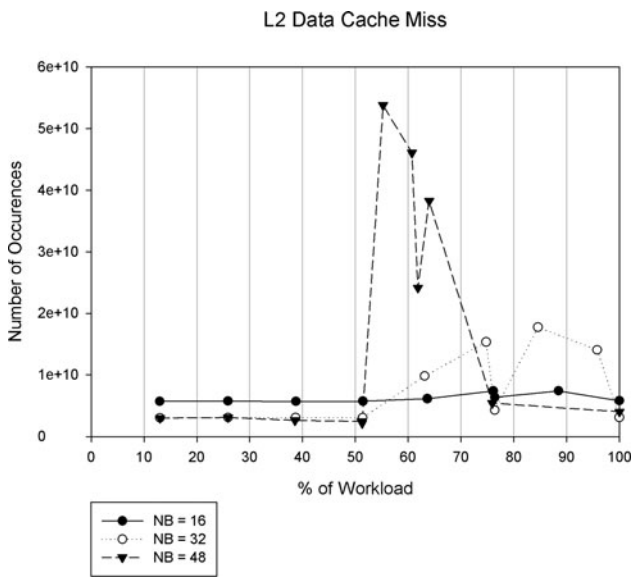


Fig. 6 L2 data cache misses for Armor

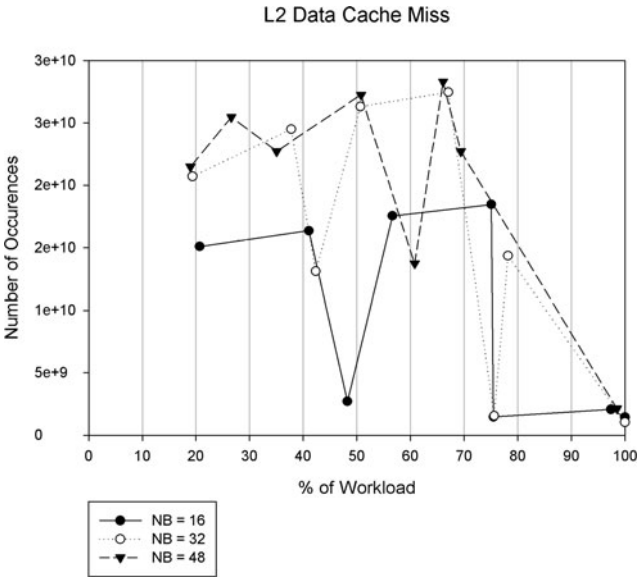
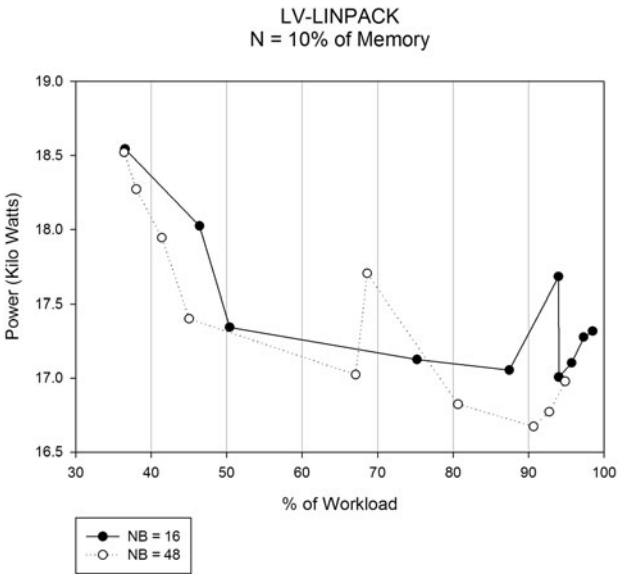


Fig. 7 L2 Data cache misses for Ice



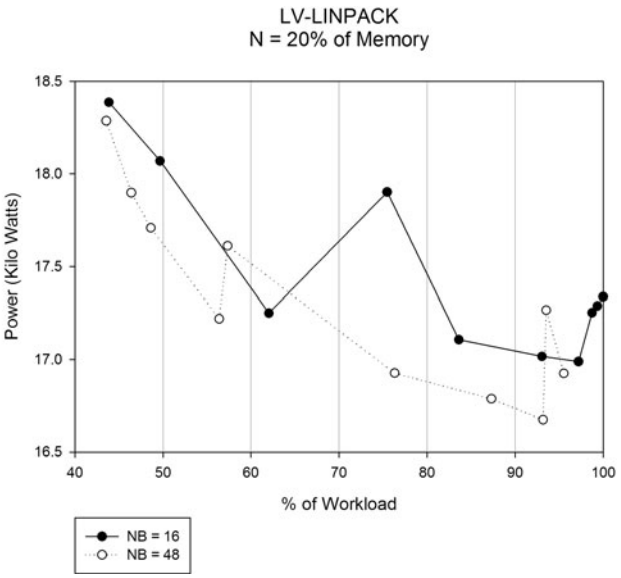


Fig. 8 LV-LINPACK with fixed $P \times Q$ on SystemG

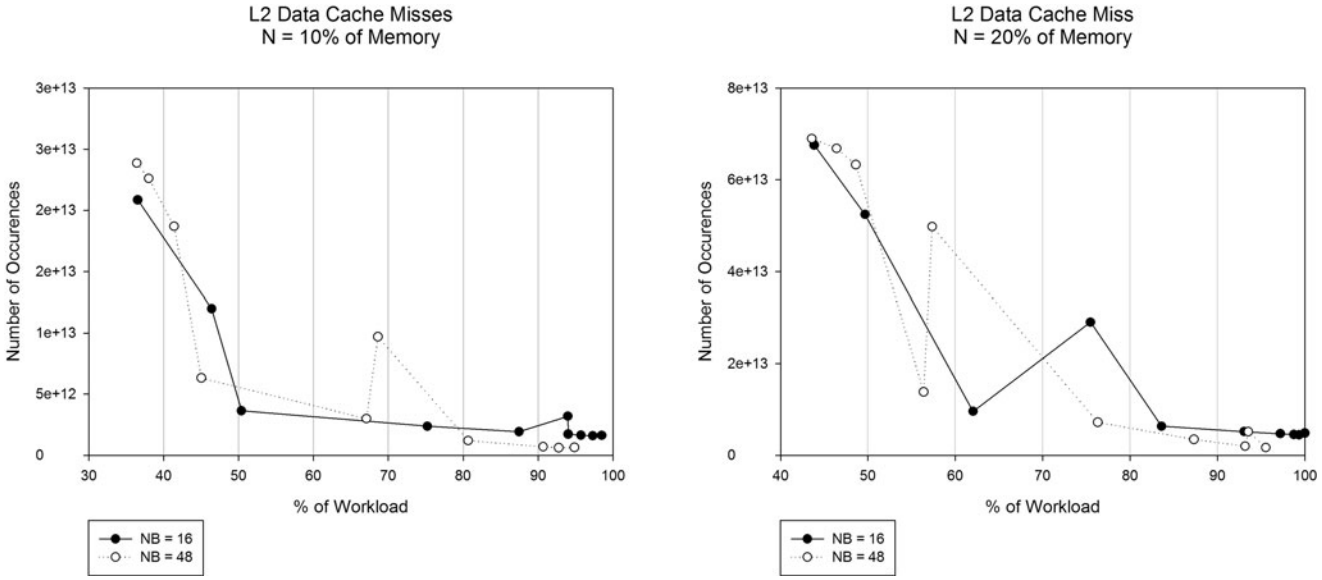


Fig. 9 L2 data cache misses on SystemG

In [15], several data mining approaches such as linear regression, M5P, multilayer perceptron and support vector machine have been applied to tune the performance of HPL. In this paper, we use feature selection to identify the parameters which has high impact on the performance of HPL. We then use these parameters to create the LV-LINPACK benchmark. In [3], a detailed study of the power and energy profiles of the NAS parallel benchmarks (NPB) [10] is presented. In another work [13], a functional and component-level study of the HPCC benchmarks [7] by using PowerPack [5] 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 HPC systems for executing benchmarks such as HPL and NPB is reported in [9]. In this paper we focus on power profile of the system at different workload to understand its trends which is not addressed in [3, 9]. In [13], the focus is on analyzing energy and power profiles of the HPCC benchmarks. [8] present a study of existing benchmark metrics for evaluating energy-efficiency. Metrics like Energy Delay Product (EDP) and Performance/Power ratio are analyzed for suitability. In this paper, we focus on creating a new benchmark to evaluate the energy efficiency in HPC.

7 Conclusion

In this paper, we proposed GBench and created a load-varying benchmark from HPL to demonstrate the efficacy of our methodology. We first identified the parameters that influence the performance of HPL and presented LV-LINPACK via a calibration of parameters such as $P \times Q$ and NB. 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.

References

1. Barroso LA, Hölzle U (2007) The case for energy-proportional computing. Computer 40(12):33–37
2. Bergman K, Borkar S, Campbell D, Carlson W, Dally W, Denneau M, Franzon P, Harrod W, Hill K, Hiller J, Karp S, Keckler S, Klein D, Lucas R, Richards M, Scarpelli A, Scott S, Snavely A, Sterling T, Williams RS, Yelick K, Kogge P (2008) Exascale computing study: technology challenges in achieving exascale systems
3. Feng X, Ge R, Cameron KW (2005) Power and energy profiling of scientific applications on distributed systems. In: IEEE IPDPS. doi:10.1109/IPDPS.2005.346
4. Fast Correlation-Based Filter (FCBF) Solution Software (2003) Available at http://www.public.asu.edu/~huanliu/FCBF/FCBFsoftware.html
5. Ge R, Feng X, Song S, Chang H, Li D, Cameron KW (2010) PowerPack: energy profiling and analysis of High-Performance systems and applications. IEEE Trans Parallel Distrib Syst 99(2). doi:10.1109/TPDS.2009.76
6. High performance LINPACK (HPL) (2008) Available at http://www.netlib.org/benchmark/hpl
7. HPC Challenge Benchmarks (2003) Available at http://icl.cs.utk.edu/hpcc
8. Hsu C, Feng W, Archuleta JS (2005) Towards efficient supercomputing: a quest for the right metric. In: IEEE IPDPS HPPAC workshop
9. Kamil S, Shalf J, Strohmaier E (2008) Power efficiency in high performance computing. In: 2008 IEEE international symposium on parallel and distributed processing, Miami, FL, USA, pp 1–8

10. NAS parallel benchmarks (1992) Available at http://www.nas.nasa.gov/Resources/Software/npb.html
11. Performance Application Programming Interface (PAPI) (2011) Available at http://icl.cs.utk.edu/papi
12. Press WH, Flannery BP, Teukolsky SA, Vetterling WT (1992) Numerical recipes in C: the art of scientific computing, 2nd edn. Cambridge University Press, Cambridge
13. Song S, Ge R, Feng X, Cameron KW (2009) Energy profiling and analysis of the HPC challenge benchmarks. Int J High Perform Comput Appl 23(3):265–276
14. SPECpower benchmark (2008) Available at http://www.spec.org/power_ssj2008
15. Tan TZ, Goh RSM, March V, See S (2009) Data mining analysis to validate performance tuning practices for HPL. In: 2009 IEEE international conference on cluster computing and workshops
16. The Top500 list (1993) Available at http://top500.org
17. Yu L, Liu H (2003) Feature selection for high-dimensional data: a fast Correlation-Based filter solution. In: The twentieth international conference on machine learning

Balaji Subramaniam is a Ph.D. student in the Department of Computer Science at Virginia Tech (VT), where he is a member of the Synergy Lab. His research interests include energy-aware computing, power modeling and prediction, hardware- and software-controlled power management, and benchmarking. He received a B.E. in Computer Science and Engineering from Anna University at Chennai in 2009.

Wu-chun Feng became an Associate Professor in the Department of Computer Science and Department of Electrical & Computing Engineering at Virginia Tech (VT) in January 2006. He leads the Synergy Lab and serves as site co-director of the NSF Center for High-Performance Reconfigurable Computing at VT. He received B.S. degrees in Computer Engineering and Music (Honors) and M.S. degree in Computer Engineering at Penn State University in 1988 and 1990, respectively. He then earned his Ph.D. in Computer Science at the University of Illinois at Urbana-Champaign in 1996. His research interests encompass a broad range of topics in efficient parallel computing, including high-performance computing and networking, energy-efficient (or green) supercomputing, accelerator-based computing, cloud computing, grid computing, bioinformatics, and computer science pedagogy for K-12.