enerGyPU and enerGyPhi Monitor for Power Consumption and Performance Evaluation on Nvidia Tesla GPU and Intel Xeon Phi

John A. García H.[1,2], Esteban Hernandez B.[1,3], Carlos E. Montenegro[3], Philippe O. Navaux[2], Carlos J. Barrios H.[1]

[1]High Performance and Scientific Computing Center - SC3, Universidad Industrial de Santander - UIS
[2]Parallel and Distributed Processing Group - GPPD, Universidade Federal do Rio Grande do Sul - UFRGS
[3]Interoperability of Academic Networks Group - GIIRA, Universidad Distrital Francisco José de Caldas - UD
[1,2]john.garcia1@correo.uis.edu.co, {[1,3]ejhernandezb, [3]cemontenegrom}@udistrital.edu.co, [2]navaux@inf.ufrgs.br, [1]cbarrios@uis.edu.co

***Abstract*—The evaluation of performance and power consumption is a key step in the design of applications for large computational systems as supercomputers and clusters (multicore and accelerator nodes, multicore and coprocessor nodes, manycore and accelerator nodes). In these systems the developers must design several experiments for workload characterization observing the architectural implications when using different combinations of computational resources such as number of GPU, number of cores for processing, number of cores for administration of GPU, number of MPI processes and thread affinity policy. It should also engage factors as the clock frequency and memory usage as well select the combination of computational resources that increases the performance and minimizes the power consumption. This research proposes an integrated energy-aware scheme called efficiently energetic acceleration (EEA) for large-scale scientific applications running on heterogeneous architectures. This paper shows the use of a monitoring tool with two components called enerGyPU and enerGyPhi to recording EEA control factors in runtime on two environments: one cluster with multicore and accelerator nodes (2-CPU/8-GPU) and one server with multiple cores and one coprocessor (2-CPU/1-MIC). These monitors allow to analyze multiple testing results under different parameter combinations to observe the EEA control factors that determine the energy efficiency.**

***Index Terms*—Energy efficiency, Energy-aware EEA scheme enerGyPU, enerGyPhi, Power capping technique, Performance evaluation.**

I. Introduction

Heterogeneous parallel programming has two problems on large computation systems: one is the increase of power consumption on supercomputers in proportion to the amount of computational resources used to obtain high performance, and the other problem is the underuse these resources by scientifics applications with a inexact distribution of tasks. Select the optimal computational resources and make a good mapping of task granularity is the main challenge for build the next generation of Exascale Systems [1].

The power consumption increases when the heterogeneous architectures increase the number of cores inside a chip, such as Intel Xeon Phi or Nvidia Tesla GPU. Depending on the circuit design (power gating, clock gating, memory bandwidth usage), the gradient of power consumption also varies [2]. However, mapping of task granularity of an application to achieve high performance with a lower power consumption is even more difficult. The heterogeneous parallel programming required different scheduling algorithms and number of threads to use Simple Instruction Multiple Thread (SIMT) for Nvidia Tesla GPU or Single Instruction Multiple Data (SIMD) for Intel Xeon Phi [3]. On Intel Xeon Phi, the number of Cores, threads per core and level of affinity is a critical aspect to achieve better level of performance and know the factors where can be minimized the power consumption. On Nvidia Tesla GPU, the kernel grid and number of threads/blocks, streaming multiprocessor clock frequency, memory clock frequency, memory usage, bandwidth usage is a key aspect.

This paper shows the use of monitor called enerGyPU and enerGyPhi to centralize and automate the capture of EEA control factors in runtime while is executed in parallel with the scientific application and displays information via sequence plots, statistical tables, bar graphs and shows results in terms of energy efficiency. The aim of this research is analyze the power capping technique on multiGPU with different combinations of computational resources to obtain the GPU power levels and build the GPU power cost function for that the programmer or a load balancing framework can select of computational resources at static time to mapping parallel task granularity. This work uses the Highly-Parallel LINPACK (HPL) Benchmark a general dense matrix problem as case of study for solving systems of linear equations in large scientific applications to evaluate the performance and the power consumption on machines with multicore and accelerator/coprocessor. The HPL is widely used by Top500 and Green500 in the evaluation of performance and power consumption on supercomputers [4].

This paper is organized as follows: the section II presents the related work. The section III shows the energy-aware scheme proposed. The section IV defines the computing resources and selected energy metrics. The section V describes the HPL benchmark as case of study workload. The section VI presents the EEA parameters used by enerGyPU and enerGyPhi. The section VII describes the experimental procedures with power

consumption and performance analysis. Finally we conclude in the section VIII.

II. Related Work

A. Power Capping and Load Balance for CPU and GPU

Power capping is a technique widely used to save power consumption from the supercomputers through Dynamic Voltage and Frequency Scaling (DVFS), this technique allows adjust manual or automatically the frequency and voltage in CPU and recently in the GPU. Komoda et al. [5] uses the power capping to manage device frequencies and task mapping at the beginning of the execution and presents a empirical model with a small number of profiles to predict the execution time and the power consumption using the frequency of CPU and GPU as main parameters. The GreenGPU framework proposes an energy management and load balance in two-tier, in the first tier dynamically splits and distributes workloads to GPU and CPU trying can finish approximately at the same time, and the second tier, GreenGPU dynamically throttles the frequencies of GPU cores and memory in a coordinated manner, based on their utilizations [6]. These schemes of power capping and load balance work well when is used a CPU and a GPU, nevertheless, when having many nodes with multi-GPUs (4, 6 and 8 GPUs) the problem is transferred to know the amount of computational resources that are needed to task mapping for a specific workload. Hogn and Wang [2] proposes an integrated power and performance (IPP) prediction model for a GPU architecture. IPP predicts performance per watt and also the optimal number of cores to achieve energy savings in a GPU. The number of active cores predicted by IPP only involves the number of blocks inside an application, unless they change the hardware or the thread scheduler.

B. Energy-aware Scheme and Power Modeling for Clusters

Huo et al. [7] presents an energy efficient task scheduling scheme for heterogeneous GPU clusters can switch between load balancing and dynamic resource scaling according to the node selection policy based on GPU utilization of the particular task, it can decrease the static energy consumption of GPU in idle status. This scheme uses the evaluation method of suitable GPU core number for particular task based on performance per watt has been proposed by Hogn and Wang [2]. Zhang et al. [8] proposes WSLSA greedly heuristic which can involve the weights of the processor to approximately optimal mapping between tasks and computational resources. They also partition all task to six classification based on the degree of parallelism, workload and input data size of task. The input data is a key parameter for determining the total number of threads and the workload is a key parameter for task sheduling. The WSLSA algorithm may not efficient when the tasks with big workloads are too much according to the task assigning rule, however, it is possible to tune WSLSA algorithm. Sîrbu and Babaoglu [9] present a recent study of power consumption modeling and prediction in a hybrid CPU-GPU-MIC Supercomputer. Using data from a dedicated monitoring framework, they build a data-driven model of power consumption for each user in the system and use it to predict the power requirements of future jobs. They used the *CountVectorizer* class in the *scikit-learn* python package and using the frequencies as main power parameters for predicting power usage per job into 5 regression analyses, one for each component type, and then sum the predicted component powers to obtain an estimate for the global job power.

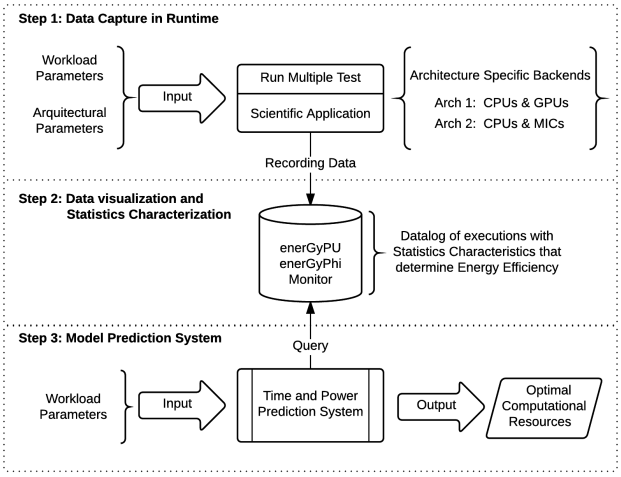

Fig. 1: Efficiently Energetic Acceleration (EEA) Scheme

III. EEA Energy-aware Scheme

This paper proposes an integrated energy-aware scheme called efficiently energetic acceleration (EEA) for scientific applications of large-scale on heterogeneous architectures. The EEA scheme has a workflow of three steps as shows in the Figure 1. The data capture in runtime, in this step is executed the enerGyPU or enerGPhi monitor tool in parallel with the scientific application using different combinations of computational resources applying the power capping technique for nodes with multi-GPUs. In the second step, the data visualization and statistics characterization used a separate level of enerGyPU and enerGPhi monitor tool for analysis the key factors by results of each experiment in terms of energy efficiency and estimated the power levels. Finally using data from enerGyPU and enerGPhi monitor is built the cost functions and model prediction system for obtain the optimal computational resources, at stactic time to mapping parallel task granularity of scientific applications on heterogeneous architectures.

A. Data Capture in Runtime

The first step is very important because is the workload and architectural parameter selection for data capture on environments with CPU, GPU and MIC. Researchers must design full or fractional factorial experiments [10] with different combinations of computational resources and different size as input data for recording of factors, such as Streaming Multiprocessor frequency, GPU memory frequency and usage with diferents number of devices, using the power

capping technique for saving power consumption in data transfer times. Input data and number of devices are key parameters for determining the total number of threads inside a GPU and the kind workload determines the task scheduling.

The task scheduling algorithms uses a directed acyclic graph (DAG), therefore, task mapping in different problems as LU decomposition, laplace equation solver, stencil algorithms and other, varied the task graph granularities, by varying the communication-to-computation ratio. [11]. This work uses the HPL, which is an implementation of the LU decomposition to solve a dense $N * N$ system of linear equations as a case study of one workload. The parameter selection for enerGyPU and enerGPhi monitor is shows in the section VI.

B. Data visualization and statistics characterization

The second step used enerGyPU or enerGPhi at post-processing for data visualization and statistics characterization. It displays information via sequence plots, statistical tables, histograms and shows results in terms of energy efficiency. This displays information of tests through the IPython Notebook Viewer in a program called *enerGyPUdisplay.ipynb* that contains predefined code routines written in Python, where the researchers have to type the argument $ARGV$ to identify the unique test of the Linpack benchmark. The statistics characterization is used for analysis the key factors by results of each experiment in terms of energy efficiency and estimated the power levels.

C. Model prediction System

The three step used data from enerGyPU and enerGPhi monitor is built the cost functions and model prediction system for obtain de optimal computational resources, at stactic time to mapping parallel task granularity of scientific applications on heterogeneous architectures. This module will be present on future works, because it's required a detailed evaluation process to obtain accurate results.

IV. Computational Resources and Energy Metrics

A. Cluster for Nvidia Tesla GPUs

The computational resources used is GUANE-1[1] cluster, that consists in 16 ProLiant SL390s computing nodes, each node with 2 CPU and 8 GPU for a total of 128 GPU at GUANE. The architecture is detailed in Table I.

B. Server for Intel Xeon Phi

The computational resources used to run the mpHPL is composed by one server with Intel Xeon Phi coprocessor such as shown in Table II.

[1]GUANE-1 (GpU AdvaNced Environment for scientific computing) is a heterogeneous cluster in the High Performance and Scientific Computing Center (SC3-UIS), located at technological camp of Industrial University of Santander at Piedecuesta, a municipality of the metropolitan area of Bucaramanga in Santander, Colombia. See http://www.sc3.uis.edu.co/

Setting GUANE	**A**	**B**	**C**
Node type	SL390s	SL390s	SL390s
Number of nodes	8	3	5
Processor Intel	Xeon	Xeon	Xeon
Processor Model	E5645	E5645	E5640
Microarchitecture	Sandy Bridge	Sandy Bridge	Sandy Bridge
Processor by node (#)	2	2	2
Clock frequency (GHz)	2.40	2.40	2.670
Core/Processor (#)	6	6	4
Thread/Core (#)	2	2	2
GPUS Nvidia	Tesla	Tesla	Tesla
GPUS Model	M2075	M2050	M2050
Microarchitecture	Fermi	Fermi	Fermi
GPUs by node (#)	8	8	8
CUDA core (#)	448	448	448
Memory DDR4 (GB)	104	104	104
SAS disk (GB)	200	200	200
Gigabit Ethernet (Gbps)	10	10	10
InfiniBand (IB)	1	1	1

TABLE I: Settings GUANE Cluster Nodes.

Setting PhiServer	**A**
Node type	Server
Number of nodes	1
Processor Intel	Xeon
Processor Model	E5-2630 v3
Microarchitecture	Sandy Bridge
Processor by node (#)	2
Clock frequency (GHz)	2.4 GHz
Core/Processor (#)	8
Thread/Core (#)	2 with HT
Device Intel	Xeon Phi
Device Model	3120A
Microarchitecture	Knights Corner
Device by node (#)	1
Cores per Device	57
Threads per Core	4 with MT
Max Frecuency	1.1Ghz
DeviceMemory	6 GB GDDR5
Memory DDR4 (GB)	64

TABLE II: Settings PhiServer Machine.

C. Energy Metrics Used by EEA Scheme

Instantaneous power over time estimates the overall energy consumption of a system. Where the amount of energy used to achieve the solution, is called *energy-to-solution* in GreenHPC [12]. Energy to solution integrates instantaneous power over time as equation 1 ilustrates.

$$Energy = \int Power(t)\ dt \tag{1}$$

This work proposes the equation 2 to calculate the *energy-to-solution* used in the execution of applications on GUANE Cluster Nodes. The $E_{Solution}$ used the discrete time equivalent of the integral using the power and time samples, for each sample i there is an instantaneous power consumption per node $Power_{Node}(i)$ and a time interval Δt. The total execution time is split in the time used by a group of CPU t_{CPU} and a group of GPU t_{GPU} to proces a percentage of workload the same application. Where N is a maximun execution time of

a application finishes on a group of CPU or a group of GPU.

$$E_{Solution} = \sum_{i=1}^{N} Power_{Node}(i) * max\ (\Delta t_{CPU}(i), \Delta t_{GPU}(i)) \quad (2)$$

The instantaneous power consumption is the throughput of energy delivered on a specific instant [12]. This work estimates the instantaneous power consumption per node $Power_{Node}$ as the sum of each measure of power consumption by component (CPU, GPU and RAM), as represented at equation 3. Where the P^j_{CPU} is the sum of total power consumption by each CPU homogeneous component and nc is the total of CPU. P^j_{GPU} is the sum of total power consumption by each GPU homogeneous component and ng is the total of GPU. P^j_{RAM} is the sum of total power consumption by each main memory and and nm is the total of memories.

$$Power_{Node}(i) = \sum_{j=1}^{nc} P^j_{CPU}(i) + \sum_{j=1}^{ng} P^j_{GPU}(i) + \sum_{j=1}^{nm} P^j_{RAM}(i) \quad (3)$$

For example, to calculate the *energy-to-solution* used to execute a subprogram BLAS3 a matrix multiplication $GEMM$: $C = \alpha * C + \beta * A * B$ using only one CPU (Xeon E5645 2.40 GHz) of 'A' settings GUANE cluster node. The total execution time of GPU t_{GPU} is zero, because they will not be used in this example. The total power per node $Power_{Node}$ must be calculated wiht all components regardless if the application is executed on one CPU or eight GPU and the total execution time at CPU should be calculated at equation 4. The total execution time in a CPU assume overlapping between at the time to move the matrix from memory to cache $time_{comm}$ and the time to perform computation the subprogram $time_{comp}$.

$$t_{CPU} = max\ (time_{comm}, time_{comp}) \quad (4)$$

Therefore if the matrix A and B have a size of $n = 700$ the data size is 3.92 MBytes per matrix. The algorithm complexity to perform computation is $2n^3 FLOP$ and $3n^2$ in memory reference to move the three matrix. The time to move this matrix in chache is $0.367ms$ with $32GB/s$ of bandwith, the time to perform computation is the $11.9ms$ with $52.6GFLOP/s$ of peak and as result the total execution time without uses overlap is $12.267ms$.

The average of power consumption per CPU is represents by $80watts$ when the processor is operating at Base Frequency, the average of power consumption per GPU is represets by $87.21watts$ when the acelerators is operating at Base Frecuency and the average of power consumption per RAM is $70watts$. All those states is called Baseline. In this way, the energy to processes a subprogram is $11.53Kilojoules$ with $940watts$ the total power consumption per node, compared with $0.98Kilojoules$ the energy to processes a subprogram using $80watts$ of power consumption per CPU. This work analysis the power capping technique to manage the power consumption usage by each GPU, enable only the component that is used to perform computation and lowering frequencies to component that is not used.

V. HPL Benchmark as Case of Study

The Highly-Parallel LINPACK (HPL) benchmark solves dense systems $N * N$ of linear equations by partial Gaussian elimination with partial pivoting. The HPL benchmark has a workload of a general dense matrix problem $Ax = b$, using a dense random matrix A. The problem size varies to find the best floating-point execution rate, *Rmax* Maximal LINPACK performance achieved. In computing the execution rate, the number of operations should be $2n^3/3 + 2n^2$ independent of the actual method used [13]. Two types of HPL code are selected to exploit the architectures described on the section IV. The modified variants are described in the following subsections.

A. HPL-2.0 optimized for NVIDIA Tesla GPU

The HPL-2.0 uses in this study case was developed by Petitet [14] is setting for GPU Tesla 20-series and was designed to accelerate the Linpack benchmark on heterogeneous clusters, where CPU and GPU are used in synergy with minor modifications to the original source code. This implementation uses the Intel MKL and CUBLAS libraries to intercepts the calls to DGEMM and DTRSM and executes them simultaneously on both GPU and CPU cores. The HPL-2.0 implementation is described by Fatica [15].

B. HPL-2.1 optimized for Intel Xeon Phi

The HPL-2.1 uses in this study case is part of Intel parallel studio and uses binary accelerates execution by offloading computations to Intel Xeon Phi coprocessors if they are available on the system. This implementation uses the dynamic scheduling technique to highly optimized panel factorization and advanced offload DGEMM. The HPL-2.1 implementation is described by Heinecke [16].

VI. EEA Parameters Used by enerGyPU and enerGyPhi

The enerGyPU and enerGyPhi are a type of batch monitor that is formed by two levels: one level called data capture in runtime composed by three events for data centralization, recording data and start the application. Another separate level called data visualization is used to analyse the results of each experiment in terms of energy efficiency. A deep description of enerGyPU monitor structure and utilization is present by Garcia in [17]

The parameters of the EEA scheme is used by enerGyPU and enerGyPhi monitors to data capture in runtime while is executed in parallel with the HPL code variants on GUANE cluster nodes and Phi-Server. The global parameters are used to setting the workload of a general dense matrix problem size divided in blocks size to generate the total task for processing the $Task_{Number}$ and get the $Task_{size}$ on each experiment. The architectural parameters are the combination of computational resource and configurations used to processing the HPL code variants. The control factors are

monitoring to analizes the implications in power consumption and performance when using different combinations of computational resources to solve the HPL problem. The dependent parameters could be used to created a model that predict the time and power consumption that determine the energy efficiency by specific problem size, described in table III.

Global Workload Parameters	
Ms	Matriz size
Bs	Block size
Architectural Parameters for Nvidia GPU	
GPU_{Usage}	Number of GPU used to HPL
$Cores_{GPU}$	Number of cores per GPU to HPL
MPI_{GPU}	Number MPI process same of GPU used
Architectural Parameters for Intel Xeon Phi	
$Cores_{MIC}$	Number of cores used on MIC
$Threads_{MIC}$	Number of threads per core on MIC
Control Factors for Nvidia Tesla GPU	
SM_{Clock}	GPU SM clock frequency
Mem_{Clock}	GPU memory clock frequency
Control Factors for Intel Xeon Phi	
$Thread_{Affinity}$	Threads execution policy on MIC cores
$Thread_{Granularity}$	Hyperthreading used per core
Dependent Parameters	
$Task_{Size}$	Bytes allocated per task
$Task_{Number}$	Total task for processing
$GPUMem_{Usage}$	GPU memory usage
$MICMem_{Usage}$	MIC memory usage
$Time$	Total time execution of application
$Power_{GPU}$	Total power consumption by all GPU
$Power_{MIC}$	Total power consumption by MIC

TABLE III: EEA parameters used by enerGyPU and enerGyPhi

VII. Experimental Designs and Power Consumption and Performance Analysis

The experimental procedures executed a set test of HPL code variants using different workload and architectural parameters on GUANE cluster nodes and Phi-Server. These results collection of Linpack test allows created a data base with AEE parameters to characterize the power levels. The experimental procedures was chosen following the fractional factorial desing principle propoused by Raj Jain [10] and will described in the next subsections.

A. Experimental Designs to HPL-2.0 on GUANE

The experimental consisting in: 12 different workload parameters using three matriz size $\{49152, 61440, 73728\}$ with four block size $\{768, 1024, 1536, 2048\}$ to divide these matrices. 8 different combinations of architectural parameters such as number of GPU, number of cores for processing, number of cores for administration of GPU and number of MPI processes. 2 control factors (SM and memory clock frequency) that represent tree power levels for GPU worker, GPU idle and GPU baseline or base state.

B. Experimental Designs to HPL-2.1 on Phi-Server

The experimental consisting: 12 experiments with different workload parameters using four matriz size $\{20480, 30702, 40960, 56320\}$ with four block size $\{512, 768, 1024\}$ to divide these matrices. 6 different combinations of architectural parameters such Number of cores used on MIC and Number of threads per core on MIC. 2 control factors as threads execution policy on MIC cores and hyperthreading used per core.

C. Results to HPL-2.0 on GUANE

The arquitectural combination {6 GPU, 2 cores per GPU and 6 MPI process} is the best energy efficient computational resource used in these experiments to solve a matrix size of 49142 and divide with block size of 768, that generates 48 task with $384Mbytes$ of task granularity as shows in the Figure2. This experiment executed the HPL-2.0 in $113.96sec$ with an averange $539.87Watts$ of power cosumption to obtained $694.7GFLOPS$ and also spend $61.52KJoules$ as energy to solution.

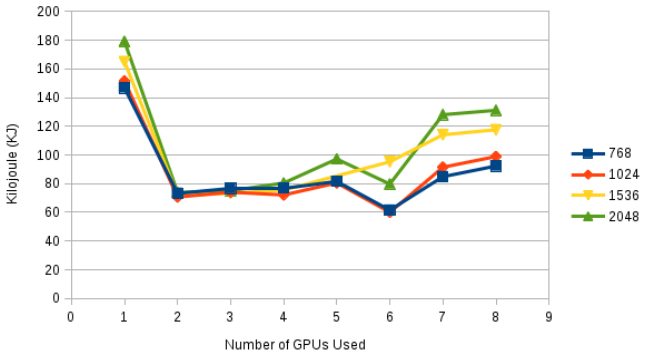

Fig. 2: Ms: 49142 for Resolve the DGEMM on GUANE

The arquitectural combination {4 GPU, 3 cores per GPU and 4 MPI process} is the best energy efficient computational resource used in these experiments to solve a matrix size of 61440 and divide with block size of 768, that generates 80 task with $360Mbytes$ of task granularity as shows in the Figure3. This experiment executed the HPL-2.0 in $222.47sec$ with an averange $435.06Watts$ of power cosumption to obtained $695GFLOPS$ and also spend $96.7KJoules$ as energy to solution.

The arquitectural combination {4 GPU, 3 cores per GPU and 4 MPI process} is the best energy efficient computational resource used in these experiments to solve a matrix size of 73728 and divide with block size of 1024, that generates

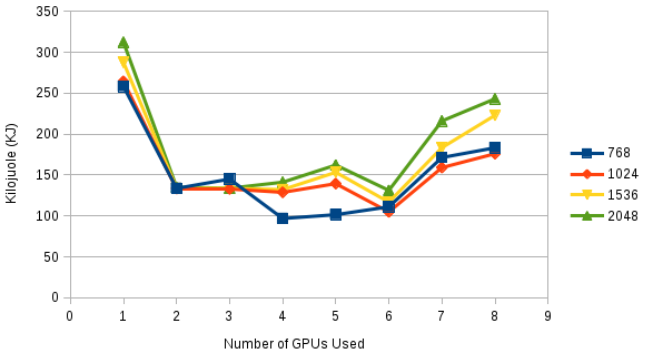

Fig. 3: Ms: 61440 for Resolve the DGEMM on GUANE

72 task with $576Mbytes$ of task granularity as shows in the Figure4. This experiment executed the HPL-2.0 in $327.9sec$ with an averange $435.43Watts$ of power cosumption to obtained $814.8GFLOPS$ and also spend $142.77KJoules$ as energy to solution.

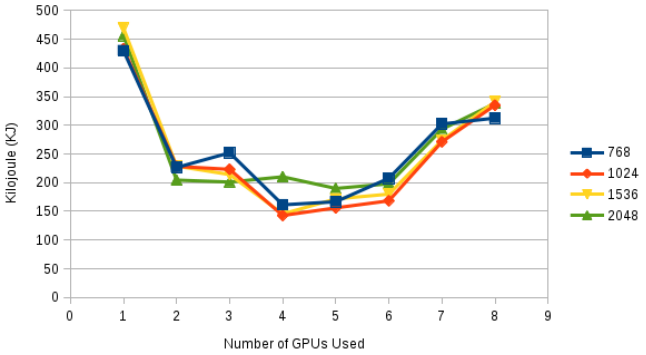

Fig. 4: Ms: 73728 for Resolve the DGEMM on GUANE

D. Power levels on GUANE

The most representative enerGyPU results to solve different matrix problem HPL-2.0 on GUANE are showing at Table IV. This Table is classified in: the best energy to solution (ES), the worst power consumption (PC) and the worst execution time (ET) for each matrix problem. The last two classifications represent the worst energy to solution divided on two behaviors: the first is given for high power consumption determined by the use of all GPU and the second is given for long execution time determined by the use of minimal computational resources.

As a result of using enerGyPU for monitoring performance and power consumption of executed the set test of HPL-2.0 using different workload and architectural parameters on GUANE. Allows obtained the power GPU levels by each determined number of GPU used to process the workload parameters as shows in the Figure 5. The figure represents the total power comsumption for all GPU when are used the power capping in two states GPU worker and GPU idle.

GPU worker: Represents the number of GPU that developer add at architectural parameters, which uses the maximun level of power capping wiht a constant frequencies of 1145 MHz for SM clock and 1566 Mhzz for memmory clock, with a constant memory usage of 2128 MiB, thas generating an averange of power 1120 Watts for each working GPU.

GPU idle: Represents the number of GPU that are inactive, which uses the minimun level of power capping with a constant frequencies of 212 MhZ for SM clock and 340 MHz for memory clock, wich 0% of memory usage to reduce power comsumption to 29.46 watts.

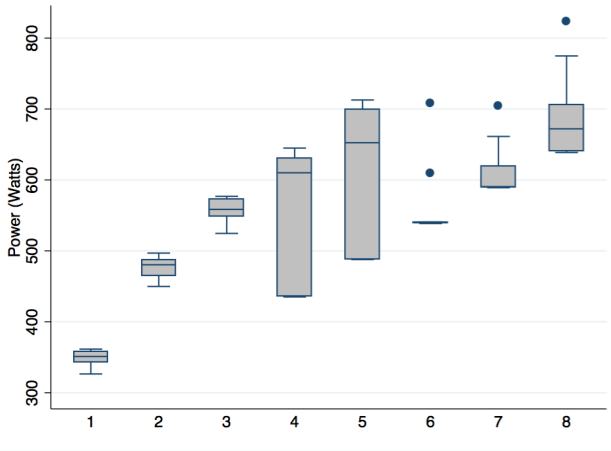

Fig. 5: Power GPU levels

E. Results to HPL-2.1 on Phi-Server

The affinity compact get the best performance and energy efficient computational resource used in these experiments to solve a matrix size of 20480,30720,40960 and divide with block size of 1024the performance is higher, the time to finish is shorter because avoid transfer and synchronization time. as shows in the Figures 6, 7 and 8.

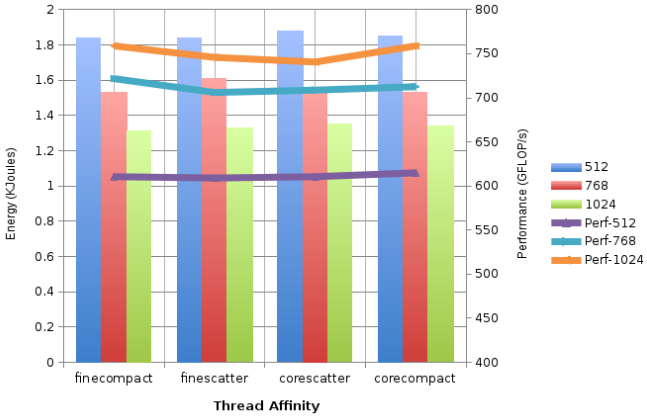

Fig. 6: Ms: 20480 for Resolve the DGEMM on Phi-Server

Experiments to solve HPL2.0	ES	PC	ET	ES	PC	ET	ES	PC	ET
Global Workload Parameters									
Ms	49152			61440			73728		
Bs	768	2048	2048	768	2048	2048	1024	1536	1536
Dependent Parameters									
$Task_{Size}$ *(MBytes)*	384	768	768	360	960	960	576	864	864
$Task_{Number}$	48	24	24	80	30	30	72	48	48
Architectural Parameters for Nvidia GPU									
GPU_{Usage}	6	8	1	4	8	1	4	8	1
$Cores_{GPU}$	2	1	12	3	1	12	3	1	12
MPI_{GPU}	6	8	1	4	8	1	4	8	1
enerGyPU Results									
$enerGyPU_{Time}$ *(sec)*	120	211	537	228	369	915	333	493	1350
$enerGyPU_{Latency}$ *(sec)*	6	6	5	6	6	6	6	6	6
$HPL2.0_{Time}$ *(sec)*	113.96	205.23	531.31	222.47	363.28	909.34	327.9	486.76	1344.09
$Rmax$ *(GFLOPS)*	694.7	131.10	149	695	425.6	170	814.8	548.9	198.8
$Power_{GPU}$ *(Watts)*	539.87	638	337.16	435.06	669.28	337.16	435.43	700.63	337.16
PPW *(MFLOPS/W)*	1286.79	603.93	441.92	1597.44	635.90	494.85	1871.22	783.43	569.37
$E_{Solution}$ ***(KJ)***	**61.52**	**131.10**	**179.13**	**96.7**	**243.13**	**312.38**	**142.77**	**341.04**	**469.25**

TABLE IV: Representative results of enerGyPU to analyze the best energetic to solution (ES), the worst power consumption (PC) and the worst execution time (ET) for each matrix problem.

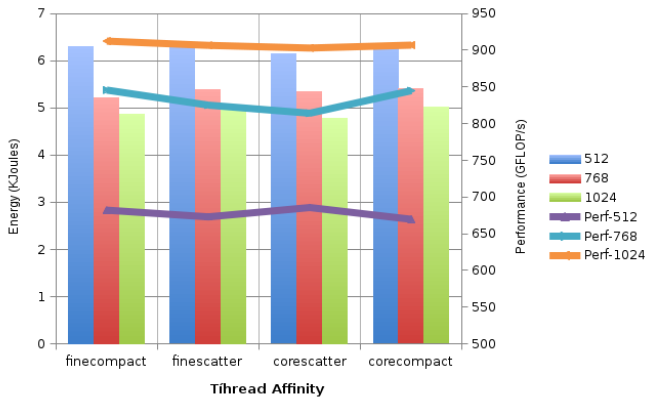

Fig. 7: Ms: 30720 for Resolve the DGEMM on Phi-Server

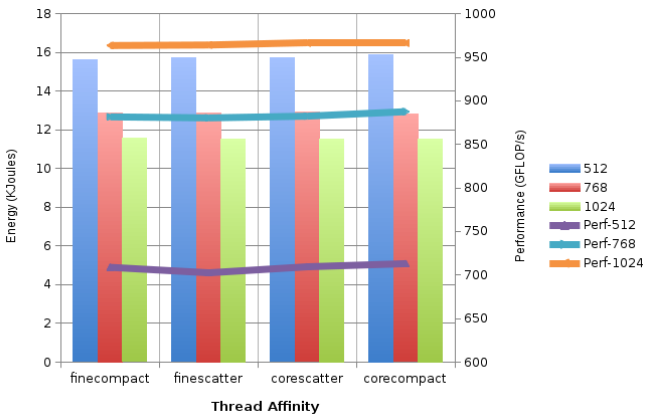

Fig. 8: Ms: 40960 for Resolve the DGEMM on Phi-Server

On high matrices (56320) we observe a benefits of use Hyper-threading (fine) for best performance and energy consumption based on a minor execution time. as shows in the Figures 9.

Thread affinity allows software thread (OpenMP Thread for this test) to execute within in the scope of specific processing resources. In this experiment we using 2 types of thread affinity (compact,scatter) with Hyper-threading enable and disable(fine,core).

When the compact affinity is used each new thread is as close as possible to the thread context, to gain in aspect how cache L2 access on NUMA configuration, the effort of schedule the closer available thread is compensate by the data dependence aspect. For a independent data access the scatter distributes the threads as evenly as possible across the entire system avoid time on schedule process.

The Intel Xeon Phi has UMA architecture (Intel Xeon Phi Coprocessor High-Performance Programming [18] and can use 4 Threads of execution SMT, accepting HT activation.

The Shared Memory (L3) on MIC is 6GB to be used by 57 cores (3120A family), the cost of go to L2 cache is near of 30 clock cycles against 250 clock cycles of use the shared memory (L3), then the thread affinity has tremendous impact on high computations process(Empirical study of Intel Xeon Phi).

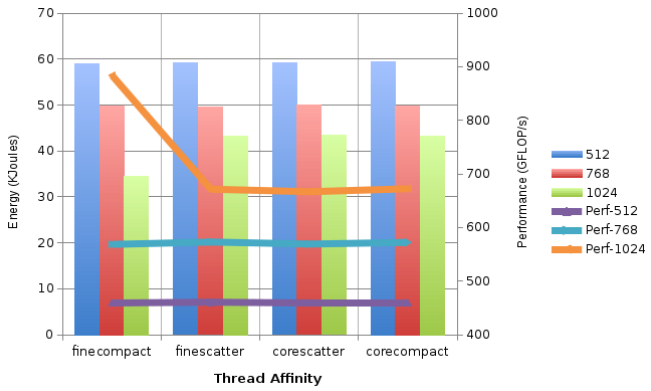

Fig. 9: Ms: 56320 for Resolve the DGEMM on Phi-Server

VIII. Conclusions

We constructed enerGyPU and enerGyPhi to facilitate the collection and visualization of data to analyze many experiments under different combinations of EEA parameters and observe the granularity of the control factors that determine energy efficiency in clusters with multiGPU and server with coprocesadores IntelXeon Phi. The method used in this paper is focused only to analyze previously compiled applications, where researchers do not need to orchestrate the code to execute enerGyPU and enerGyPhi, ensuring the integrity of the results. enerGyPU and enerGyPhi are a software level monitor, therefore can be complemented with other energy monitors that are designed to be plugged-in directly into the power supply to make holistic measures on heterogeneous arquitectures.

Based on the experiment procedures and results presented, the energy-aware EEA scheme is a good alternative to analyze the energy efficiency of applications that runing on arquitectures heterogeneous. The current approach to energy-aware EEA scheme under task-level programming will be use at design on next generation of heterogeneous architecture for large systems, to characterize each specific workload using the best combination of computational resources to optimize the time and power consumption according to task granularity.

IX. Project Repository

To use enerGyPU and enerGyPhi monitors and validate the results, you can download the source code and the data experiments used in this research on project repository url *http://forge.sc3.uis.edu.co/redmine/projects/eea-uis*. To obtain more information about general project you can visit the web site *www.sc3.uis.edu.co*.

X. Acknowledgment

We express gracefull to SuperComputer Center and Scientific Calculation SC3 of Universidad Industrial de Santander for gave access to GUANE GPU cluster and High Performance Computing Center CECAD of Universidad Distrital Francisco Jose de Caldas for offer access to server with Intel Xeon Phi accelerator. The project is developed in collaboration with the High Performance and Scientific Computing Center, SC3UIS of University Industrial of Santander and the Parallel and Distributed Processing Group, GPPD of Federal University of Rio Grande do Sul.

References

[1] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, J. Hiller, S. Karp, S. Keckler, D. Klein, R. Lucas, M. Richards, A. Scarpelli, S. Scott, A. Snavely, T. Sterling, R. Williams, and K. Yelick. Exascale Computing Study: Technology Challenges in Achieving Exascale Systems. Peter Kogge, Editor & Study Lead. Technical report, 2008.

[2] Sunpyo Hong and Hyesoon Kim. An integrated GPU power and performance model. In André Seznec, Uri C. Weiser, and Ronny Ronen, editors, *ISCA*, pages 280–289. ACM, 2010.

[3] Hadi Esmaeilzadeh, Emily R. Blem, Renée St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Power challenges may end the multicore era. *Commun. ACM*, 56(2):93–102, 2013.

[4] Run Rules, Green500. Energy Efficient High Performance Computing Power Measurement Methodology. Version: 1.2RC2, 2014.

[5] Toshiya Komoda, Shingo Hayashi, Takashi Nakada, Shinobu Miwa, and Hiroshi Nakamura. Power capping of CPU-GPU heterogeneous systems through coordinating DVFS and task mapping. In *ICCD*, pages 349–356. IEEE Computer Society, 2013.

[6] Kai Ma, Xue Li, Wei Chen, Chi Zhang, and Xiaorui Wang. GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures. In *ICPP*, pages 48–57. IEEE Computer Society, 2012.

[7] Hongpeng Huo, Chongchong Sheng, Xinming Hu, Baifeng Wu. An Energy Efficient Task Scheduling Scheme for Heterogeneous GPU-Enhanced Clusters. In ICSAI, pages 623-6277. IEEE Computer Society, 2012.

[8] Keliang Zhang and Baifeng Wu. Task Scheduling Greedy Heuristics for GPU Heterogeneous Cluster Involving the Weights of the Processor. In *IPDPS Workshops*, pages 1817–1827. IEEE, 2013.

[9] Alina Sîrbu and Özalp Babaoglu. Power Consumption Modeling and Prediction in a Hybrid CPU-GPU-MIC Supercomputer (preliminary version). *CoRR*, abs/1601.05961, 2016.

[10] Raj Jain. *The art of computer systems performance analysis - techniques for experimental design, measurement, simulation, and modeling*. Wiley professional computing. Wiley, 1991.

[11] Andrei Radulescu and Arjan J. C. van Gemund. Fast and Effective Task Scheduling in Heterogeneous Systems. In *Heterogeneous Computing Workshop*, pages 229–238, 2000.

[12] Gustavo Rostirolla, Rodrigo da Rosa Righi, Vinicius Facco Rodrigues, Pedro Velho, and Edson Luiz Padoin. GreenHPC: a novel framework to measure energy consumption on HPC applications. In *SustainIT*, pages 1–8. IEEE, 2015.

[13] Jack Dongarra, Piotr Luszczek, and Antoine Petitet. The LINPACK Benchmark: past, present and future. *Concurrency and Computation: Practice and Experience*, 15(9):803–820, 2003.

[14] Antoine P. Petitet. High Performance Computing Linpack Benchmark, HPL - 1.0a. University of Tennessee, Knoxville. Innovative Computing Laboratories. (C) 2000 - 2004.

[15] Massimiliano Fatica. Accelerating linpack with CUDA on heterogenous clusters. In David R. Kaeli and Miriam Leeser, editors, *GPGPU*, volume 383 of *ACM International Conference Proceeding Series*, pages 46–51. ACM, 2009.

[16] Alexander Heinecke, Karthikeyan Vaidyanathan, Mikhail Smelyanskiy, Alexander Kobotov, Roman Dubtsov, Greg Henry, Aniruddha G. Shet, George Chrysos, and Pradeep Dubey. Design and Implementation of the Linpack Benchmark for Single and Multi-node Systems Based on Intel Xeon Phi Coprocessor. In *IPDPS*, pages 126–137. IEEE Computer Society, 2013.

[17] John A. G. Henao, Victor M. Abaunza, Philippe O. A. Navaux, Carlos J. B. Hernandez. eGPU for Monitoring Performance and Power Consumption on Multi-GPUs. University Industrial of Santander. In WSPPD, pages 5-8. IEEE Computer Society, 2015.

[18] James Jeffers and James Reinders. *Intel Xeon Phi Coprocessor High Performance Programming*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2013.