Energetically Efficient Acceleration EEA-Aware For Scientific Applications of Large-Scale On Heterogeneous Architectures

John A. García. H.[1,2,4], Esteban Hernandez B.[1,3], Philippe O. A. Navaux[2], Carlos. J. Barrios H.[1]

[1,2,4]jgarciahenao@acm.org, [1,3]ejhernandezb@udistrital.edu.co, [2]navaux@inf.ufrgs.br, [1]cbarrios@uis.edu.co

1. High Performance and Scientific Computing Center, SC3 Universidad Industrial de Santander, UIS - Bucaramanga, Colombia
2. Parallel and Distributed Processing Group, GPPD Universidade Federal do Rio Grande do Sul, UFRGS - Porto Alegre, Brasil
3. Interoperability of Academic Networks Group - GIIRA, Universidad Distrital Francisco Jose de Caldas, UD - Bogota, Colombia
4. Laboratoire d'Informatique, Signaux et Systèmes de Sophia Antipolis - I3S, Université Nice Sophia Antipolis, UNS - Nice, France

Challenge

Heterogeneous parallel programming has two main problems on large computation systems:

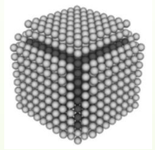
- The increase of power consumption on supercomputers in proportion to the amount of computational resources used to obtain high performance.
- The underuse these resources by scientific applications with improper distribution of tasks.

"Select the optimal computational resources and make a good mapping of task granularity is the fundamental challenge for build the next generation of Exascale Systems"

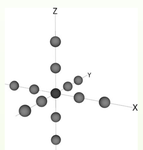
Analysis of Energy Efficiency and Portability of large Scale Scientific Application:

Case Study: Ondes3D©

Simulation of Seismic Wave Propagation: It's a crucial tool for analyzing geophysical strong movements.



3D Seismic Model with blocks discretization



Spatial Data Dependencies Stencil

Unknowns:

Velocity and Stress
On each point of Space and Time.

$$\frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial j} + F_i \quad (3.6)$$

Seismic Wave Equation for a solid medium.

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda \delta_{ij} \left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}\right) + \mu \left(\frac{\partial v_i}{\partial j} + \frac{\partial v_j}{\partial i}\right) \quad (3.7)$$

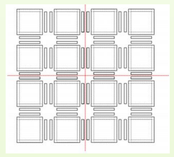
Constitutive relation in the case of an isotropic medium.

Ref John A. García H. et al. Energetically Efficient Acceleration EEA-Aware. Degree work to obtain the title of Master of Science in Systems Engineering and Informatics at UIS 2016.
Ref. Victor Martinez et al. Towards Seismic Wave Modeling on Heterogeneous Many-Core Architectures Using Task-Based Runtime System. SBAC-PAD 2015.

Planning workload experimental on Ondes3D©

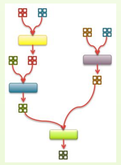
StarPU© task programming library for hybrid architectures.

7.2 Millions Grid (300×300×80) - 625 MB
Block Size 150x150

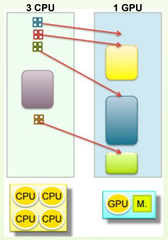


2D seismic model discretized in stencil blocks of primary grid.

36 Task processing (20 Speed; 16, Effort)
144 Task of communication



Task dependency: Directed Acyclic Graph (DAG)



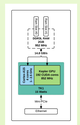
StarPU© used to task mapping over CPU and GPU

Ref John A. García H. et al. Energetically Efficient Acceleration EEA-Aware. Degree work to obtain the title of Master of Science in Systems Engineering and Informatics at UIS 2016.
Ref. Victor Martinez et al. Task-based programming on Low-power Manycore Architectures: Seismic Simulation on Jetson TK1. CARLA 2015.
Ref. Olivier Aumage. Porting BRGM's Ondes3D on top of StarPU. Task Parallelism on Heterogeneous platforms. INRIA – University of Bordeaux. March 2013.

Heterogeneous Architectures

Results of running Ondes3D© with StarPU© on different machines.

Jetson TK1

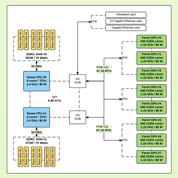


Mean Power Consumption on Jetson TK1(15 Watts):
10 Watts by GPU use.
5 Watts by CPU use.

Desktop: Salmon

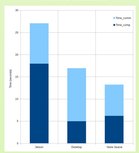
Mean Power Consumption of Salomon Desktop (135.6 Watts):
Limite Superior de la GPU: 191.35 Watts (Processing)
Nivel medio de GPU: 80 Watts (Transfers)

Node: GUANE



Mean Power Consumption on GUANE Node(987 Watts)
Upper limit of Node: 987 Watts (Processing)
Mean level of Node: 413 Watts (Transfers)

Time to Solution

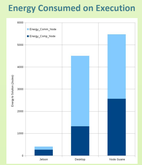


Jetson(27.06 seconds):
17.97 seconds in processing tasks.
9.09 seconds in transfer tasks.

Desktop(16.94 seconds):
4.03 seconds in processing tasks.
11.96 seconds in transfer tasks.

Node (13.23 seconds):
6.20 seconds in processing tasks.
7.03 seconds in transfer tasks.

Energy Consumed on Execution



Jetson(405.9 Joules):
270.6 Joules by using GPU.
135.3 Joules by using CPU.
10W (GPU) + 5W (CPU) = 15 Watts

Desktop(4500.5 Julios):
2298.3 Joules by using GPU.
2202.2 Joules by using CPU.
135.6W (GPU) + 130W (CPU) = 265.6 Watts

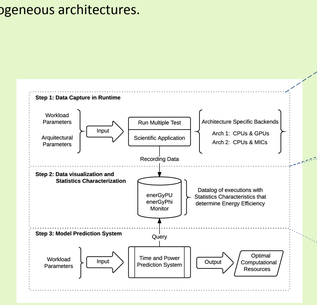
Node (5470.4 Julios):
2692.1 Joules by using CPU y GPU.
2778.3 Joules by component in an idle state.
203.4W (N_run) + 210W (N_idle) = 413 Watts

Which is the task distribution to be mapping on heterogeneous architectures CPU-GPU, that determine an efficient computational acceleration and low power consumption during the time of scientific application execution?

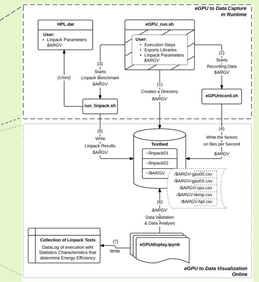
Ref John A. García H. et al. Energetically Efficient Acceleration EEA-Aware. Degree work to obtain the title of Master of Science in Systems Engineering and Informatics at UIS 2016.

EEA-Aware Structure

This research proposes an integrated energy-aware scheme called Efficiently Energetic Acceleration (EEA) for scientific applications of large-scale on heterogeneous architectures. The EEA energy-aware scheme has a workflow of three steps as shown in EEA-Aware Scheme. In the first step, the data is captured in runtime and 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. A deep description of monitor structure and utilization is present by García John et al. in [2], [3]. Finally using the data collected by monitors cost functions are build and model prediction system running for obtaining the optimal computational resources in a static time to mapping parallel task granularity of scientific applications on heterogeneous architectures.



EEA-Aware Scheme



enerGyPU Monitor Structure

Step 1: Experimental Procedures and Results

At the first level the global parameters have chosen the workload and computational architecture according to the combination of resources that will be used. The experimental procedures were executed with a set of tests of HPL code variants using different workload and architectural parameters on Cluster nodes GUANE. The experimental procedures was chosen following the fractional factorial design principle proposed by Raj Jain [4].

The computational resources used: 1 GUANE Node on 'A' settings.

Configuraciónes	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
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

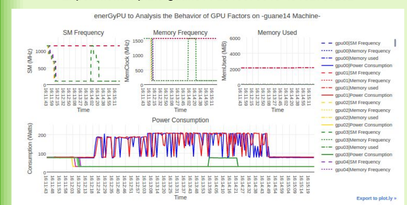
The Linpack used: HPL.2.0 version configured for Tesla GPUs.

Ref. Massimiliano Fatica. Accelerating linpack with CUDA on heterogenous clusters. ACM, 2009.

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
$Core_{KGPU}$	2	1	12	3	1	12	3	1	12
MP_{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
$S_{Solution}$ (KJ)	**61.52**	**131.10**	**179.13**	**96.7**	**243.13**	**312.38**	**142.77**	**341.04**	**469.25**

Representative results of enerGyPU to analyze **the best Energetic to Eolution (ES)**, **the worst Power Consumption (PC)** and **the worst Execution Time (ET)** for each matrix problem.

Experiment: hpl2.0-guane14-2015073016-49152768262



Export to plot.ly »

***** Global Workload and Architectural Parameters | hpl2.0 Application *****

	1) Matrix size	2) Block size	3) GPU Used	4) Cores per GPU	5) Process MPI
0	49152	768	2	6	2.1

***** enerGyPU Results to Execute 'hpl2.0' on 'guane14' Machine *****

	1-Time_to_Solution	2-Performance_Rmax	3-Power_Consumption	4-Performance_per_Watt	5-Energy_to_Solution
0	149.85	528.3	488.51	1081.46	73.202714

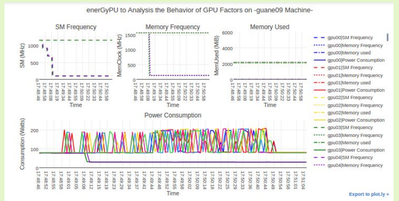
The units are: TimeTS(sec), Rmax(GFLOPS), Power_C(Watts), PPW(MFLOPS/W) and Energy_C(kJ)

***** enerGyPU Measurement: Average Factors for each GPU to Execute 'hpl2.0' on 'guane14' Machine *****

	SM Frequency	Memory Frequency	Memory Used	Power Consumption	Energy Consumption
gpu00	1147.000000	1566.000000	2136.723301	147.893333	22.161816
gpu01	1147.000000	1566.000000	2136.723301	155.880467	23.356808
gpu02	176.951456	273.932039	10.000000	29.613000	4.437508
gpu03	288.441748	461.490291	10.000000	36.414600	5.456728
gpu04	206.300971	315.611650	10.000000	29.683067	4.448008
gpu05	185.990291	287.825243	10.000000	29.812600	4.467418
gpu06	185.990291	287.825243	10.000000	29.523067	4.424032
gpu07	191.067961	294.771845	10.000000	29.686467	4.448517

The units are: SM_F(MHz), Memory_F(MHz), Memory_U(MiB), Power_C(Watts) and Energy_C(kJ)

Experiment: hpl2.0-guane09-2015073117-49152768626



Export to plot.ly »

***** Global Workload and Architectural Parameters | hpl2.0 Application *****

	1) Matrix size	2) Block size	3) GPU Used	4) Cores per GPU	5) Process MPI
0	49152	768	6	2	6.1

***** enerGyPU Results to Execute 'hpl2.0' on 'guane09' Machine *****

	1-Time_to_Solution	2-Performance_Rmax	3-Power_Consumption	4-Performance_per_Watt	5-Energy_to_Solution
0	113.96	694.7	539.96	1286.58	61.533842

The units are: TimeTS(sec), Rmax(GFLOPS), Power_C(Watts), PPW(MFLOPS/W) and Energy_C(kJ)

***** enerGyPU Measurement: Average Factors for each GPU to Execute 'hpl2.0' on 'guane09' Machine *****

	SM Frequency	Memory Frequency	Memory Used	Power Consumption	Energy Consumption
gpu00	1147.000000	1566.000000	2125.915888	79.400333	9.049374
gpu01	1147.000000	1566.000000	2125.915888	80.581667	9.183087
gpu02	1147.000000	1566.000000	2125.915888	81.620000	9.301415
gpu03	241.728972	389.102804	11.000000	30.620000	3.489455
gpu04	251.504673	402.476636	11.000000	29.613333	3.374735
gpu05	1147.000000	1566.000000	2125.915888	79.040000	9.007398
gpu06	1147.000000	1566.000000	2125.915888	79.86333	9.101225
gpu07	1147.000000	1566.000000	2125.915888	79.213333	9.027151

The units are: SM_F(MHz), Memory_F(MHz), Memory_U(MiB), Power_C(Watts) and Energy_C(kJ)

Step 2: Architectural Characterization

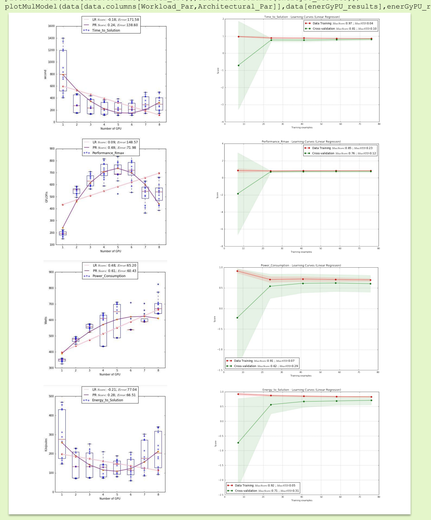
The second level uses the enerGyPU monitor in the post-processing for data visualization and statistical characterization of each architecture. In which the Figure below shows that the accuracy using the key factors is much more accurate than just using the number of GPUs. In addition, this monitor displays information via sequence data, statistics, histograms and tables showing results in terms of energy efficiency, for each experiment.

Minimization model with Task and Architectural Parameters

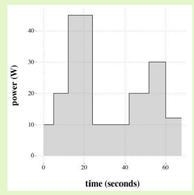
`GPUPowerRuntime(GPU_worker[GPU_Factors],GPU_idle[GPU_Factors])`

Accuracy of Linear and Polynomial Models Among Learning Curves Multivariable Model

```
plotL&PModel(data[data.columns[Number_GPU]],data[data.columns[enerGyPU_results]])
plotMulModel(data[data.columns[Workload_Par,Architectural_Par]],data[enerGyPU_results],enerGyPU_results)
```



Discretization Power Consumption per Unit Time:



Ref. Gustavo Rostirolla et al. GreenHPC: A Novel Framework to Measure Energy Consumption on HPC applications. 2015

Equations to Calculate the Energy Consumption:

$$Energy = \int Power(t)\, dt \quad (3.1)$$

$$Energy = \sum_{i=1}^{N} Power(i) * \Delta t(i) \quad (3.2)$$

$$E_{Solution} = \sum_{i=1}^{N} Power_{Node}(i) * max\left(\Delta t_{CPU}(i), \Delta t_{GPU}(i)\right) \quad (3.3)$$

$$Power_{Node}(i) = \sum_{j=1}^{nc} P^j_{CPU}(i) + \sum_{j=1}^{ng} P^j_{GPU}(i) + \sum_{j=1}^{nm} P^j_{RAM}(i) \quad (3.4)$$

$$\Delta t_{CPU}(N) = max\left(time_{comm}, time_{comp}\right) \quad (3.5)$$

Ref. john A. García H. et al. Energetically Efficient Acceleration EEA-Aware. Degree work to obtain the title of Master of Science in Systems Engineering and Informatics at UIS 2016.
Ref. Jack Dongarra. Algorithmic and Software Challenges For Numerical Libraries at Exascale, 2013.

Step 3: EEA Prediction System

The third level uses the projected multivariable regression model results to see metrics such as time, performance, power consumption, power consumption and performance per watt, which are used to execute the HPL for each combination of computational resources and and calculate the best combination of computational resources, as shown in Figure.

***Efficiently Energetic Acceleration* | EEA Prediction System**

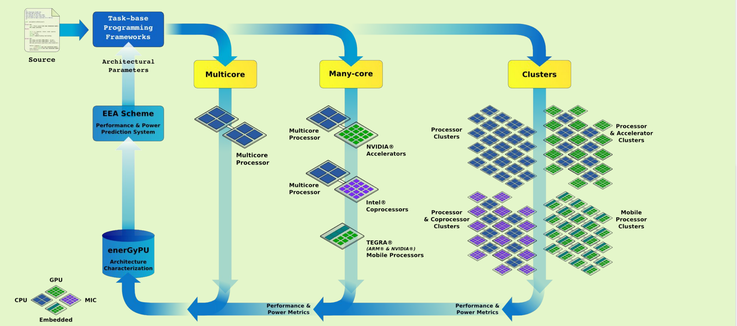
```
In [25]: eeaPredictionSystem(49152)
```

```
+++++ EEA Prediction System of Computational Resources for Matrix 49152 to Execute HPL-2.0 on GUANE +++++
```

Out[25]:

	Best Prediction	Time_to_Solution	Performance_Rmax	Power_Consumption	Energy_to_Solution	Performance_per_Watt
block_size	768.000000	1024.000000	768.000000	2048.000000	1024.000000	768.000000
Number_task	64.000000	48.000000	64.000000	24.000000	48.000000	64.000000
Task_size	288.000000	384.000000	288.000000	768.000000	384.000000	288.000000
GPU	3.000000	5.000000	5.000000	1.000000	3.000000	3.000000
Cores	4.000000	2.000000	2.000000	12.000000	4.000000	4.000000
Time_to_Solution	93.730042	13.185518	56.792579	751.901756	50.122982	93.730042
Performance_Rmax	662.169353	640.145940	674.374812	118.395452	627.940482	662.169353
Power_Consumption	546.218050	609.774292	607.899581	356.528464	548.092760	546.218050
Energy_to_Solution	55.832045	41.343681	65.305453	241.895380	31.870272	55.832045
Performance_per_Watt	1.218755	1.080187	1.151112	0.406359	1.147830	1.218755

EEA-Aware Scheme for Scalability and Portability of Large Scale Applications on Heterogeneous Architectures:



EEA-Aware Scheme Adaptation. **James R.** Intel Xeon processors and Intel Xeon Phi coprocessors, 2012

More Information:
www.sc3.uis.edu.co/
www.inf.ufrgs.br/gppd

Project Repository:
Stable:
http://forge.sc3.uis.edu.co/redmine/projects/eea-uis
Testing:
https://github.com/jagh/enerGyPU

References

[1] **John A. García H. et al.** Energetically Efficient Acceleration EEA-Aware. Degree work to obtain the title of Master of Science in Systems Engineering and Informatics at UIS 2016.

[2] **John A. García H. et al.** enerGyPU and enerGyPhi Monitor for Power Consumption and Performance Evaluation on Nvidia Tesla GPU and Intel Xeon Phi. CCGrid 2016 - IEEE/ACM.

[3] **John A. García H. et al.** eGPU for Monitoring Performance and Power Consumption on Multi-GPUs. XIII Workshop de Processamento Paralelo e Distribuído, WSPPD 2015 - IEEE.

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

[5] **K. Bergman et al.** Exascale Computing Study: Technology Challenges in Achieving Exascale Systems. Peter Kogge, Editor & Study Lead. Technical report, 2008.