Simulation of the Energy Consumption of GPU

Dorra Boughzala
@Greendays - Anglet

24 June 2019

Supervisors
Laurent LEFEVRE ~ Avalon/INRIA
Anne-Cécile ORGERIE ~ Myriads/CNRS
Outline

1. Introduction
2. Context: GPU architecture & CUDA execution model
3. Our macroscopic analysis of GPU power consumption
   a. State-of-art on GPU Power Analysis
   b. Our Methodology
   c. Experimental results & Analysis
4. Simulating the power consumption of High Performance GPU-based Applications with SimGrid
   a. State-of-art on GPU Power Modeling
   b. Our proposition: From SimGrid to “GPUSimGreen”
5. Conclusion & Future works
Why we need Exascale computing?

Exascale computing refers to computing with systems that deliver performance with the range of $10^{18}$ floating points operations per second (Flops).

Source: Huffingtonpost.com
Where are we from the Exascale?

- HPL performance: 148.6 PetaFlops
- Power consumption: 10,096 kw
- Power efficiency: 14.7 GFlops/watt
- 4,608 nodes: 2 IBM POWER9 CPUs & 6 NVIDIA volta GPUs

Aurora: coming soon in 2021
- Promise of > 1 ExaFlops
- USA or CHINA??
Challenges

- Parallelism increases orders of magnitude, power consumption as well.
- The “desired” objective of consuming power to reach exascale should not exceed 20 MW, equal to only 3-fold increase in energy efficiency of today most-energy efficient system in the [Green500].
- Therefore, energy has now become the leading concern for HPC system designs.
- It’s mandatory to understand and predict power and performance profiles of current and future HPC systems and applications in order to improve their Performance/Watt.
- Nodes are becoming highly heterogeneous and hierarchical, modeling the power and energy consumption of such systems is a challenging task.
GPU-based computing

- GPUs have become an integral part of today mainstream computing systems thanks to their high computational power and energy efficiency.
- The CPU-GPU heterogeneous computing is more energy-efficient than traditional many-core parallel computing.
- Nvidia GPUs are present in five of the top 10 of [Top500].
My research questions are:

- How to measure and analyse the power consumption of GPUs?
- How to predict the performance and power consumption of GPUs using simulation?
- How to improve the energy efficiency of GPUs?

My approach is to:

“Simulate the Energy Consumption of GPU-based systems”

1. Power and performance profiling with real measurements
2. Power modeling:
   - Implementation in a simulator
   - Validation with real workloads measurements
3. Integration of GPU DVFS in the model for example
NVIDIA GPU architecture

Example Fermi:

Source: NVIDIA Fermi whitepaper
CUDA Execution Model (1)

Abstractions

- CUDA (Compute Unified Device Architecture) is both the platform and the programming model built by NVIDIA for developing applications on NVIDIA GPUs cards.

- CUDA exposes an abstract view of the GPU parallel architecture.

- CUDA proposes 3 key logical abstractions:
  - Threads
  - Thread blocks
  - Grids
CUDA Execution Model (2)

Scheduling

**Notion**: A warp (a block of 32 consecutive threads) is the basic unit for scheduling work inside an SM.

We have two-level of scheduling provided by:

1. **The GigaThread scheduler (global scheduling)**:
   - Each SM can be scheduled to run one or more thread blocks, depending on how many resident threads and thread blocks an SM can support. **No guarantee of order of execution**.

2. **The SM warp schedulers (local scheduling)**:
   - The warp schedulers on an SM select **active warps** on every clock cycle and dispatch them to execution units.
Outline

1. Introduction
2. Context: GPU architecture & CUDA execution model
3. Our macroscopic analysis of GPU power consumption
   a. State-of-art on GPU Power Analysis
   b. Our Methodology
   c. Experimental results & Analysis
4. Simulating the power consumption of High Performance GPU-based Applications with SimGrid
   a. State-of-art on GPU Power Modeling
   b. Our proposition: From SimGrid to “GPUSimGreen”
5. Conclusion & Future works
State-of-art on Power Analysis

- [Collange2009] characterize power consumption of various GPU functional blocks (ALU, register file or memory) with real measurements on different NVIDIA GPUs.
- [Huang2009] propose an empirical study of the performance, the power and energy characteristics of GPUs for a GEM applications.
- [Cebri’n2012] analyze kernels taken from the CUDA SDK in order to discover resource underutilization.
- [Burtscher2014] discuss unexpected behaviors when measuring GPU power consumption, when working with k20 power samples.
Our Macroscopic Approach

● For a selected representative kernel vector addition, we did real measurements of the execution time and the power consumption of all phases in our code.
● Our methodology seeks to explore the execution configurations (data size, Number of threads/block, Number of Active SMs) and characterize their impact on the time and power of a compute kernel.
● This generates 3 study cases:
  1. Data size impact
  2. Number of Threads/ Block impact
  3. Number of blocks and Active SMs impact
Experimental Setup

- In this work, we rely on the Grid'5000* infrastructure in particular on the Orion cluster (Lyon), due to the availability of a GPU card and wattmeters.
- The orion node: 2 Intel Xeon E5-2630 with 6 physical cores per CPU, 32 GiB of RAM and an NVIDIA Tesla M2075 GPU (fermi architecture) card (installed in 2012).
- Idle power of the node (CPU+GPU): **156 W** (subtracted in the following): **Idle power of the targeted GPU: 57W**

<table>
<thead>
<tr>
<th>Resources</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of SMs</td>
<td>14</td>
</tr>
<tr>
<td>Number of cores/SM</td>
<td>32</td>
</tr>
<tr>
<td>Number of cores</td>
<td>448</td>
</tr>
<tr>
<td>Max Number of threads/SM</td>
<td>1536</td>
</tr>
<tr>
<td>Max number of threads/block</td>
<td>1024</td>
</tr>
<tr>
<td>Warp size</td>
<td>32</td>
</tr>
<tr>
<td>Global memory size</td>
<td>5301 Mbytes</td>
</tr>
<tr>
<td>Shared memory/Block</td>
<td>49152 bytes</td>
</tr>
<tr>
<td>Registers/Block</td>
<td>32768</td>
</tr>
<tr>
<td>Memory bus width</td>
<td>384-bit</td>
</tr>
<tr>
<td>Compute capability</td>
<td>2.0</td>
</tr>
</tbody>
</table>

Vector Addition execution flow:

1. Allocates arrays in the CPU memory (*Malloc*)
2. Initiates them with random floats.
3. Allocates arrays in the GPU memory (*cudaMalloc*)
4. Copies those arrays from the CPU memory to the GPU memory (*CopyC2G*)
5. Launches the kernel by the CPU to be executed on the GPU (*VectAdd*)
6. Copies the result from the GPU memory to the CPU memory (*CopyG2C*)
7. Frees arrays from the GPU memory (*CudaFree*)
8. Frees arrays from the CPU memory (*Free*)
Case study 1: data size impact

- We use 1024 threads/block = maximum.
- We vary the data size from $5 \times 10^5$ to $5 \times 10^7$

<table>
<thead>
<tr>
<th></th>
<th>Malloc(C)</th>
<th>Malloc(G)</th>
<th>CopyC2G</th>
<th>AddVec(G)</th>
<th>CopyG2C</th>
<th>Free(G)</th>
<th>Free(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=500000</td>
<td>0.012</td>
<td>2218.67</td>
<td>1.664</td>
<td>23880.14</td>
<td>1.73</td>
<td>0.289</td>
<td>0.381</td>
</tr>
<tr>
<td>N=5000000</td>
<td>0.011</td>
<td>2216.51</td>
<td>12.768</td>
<td>235425.28</td>
<td>14.85</td>
<td>0.345</td>
<td>2.599</td>
</tr>
<tr>
<td>N=50000000</td>
<td>0.012</td>
<td>2216.95</td>
<td>123.75</td>
<td>2368287</td>
<td>144.216</td>
<td>0.613</td>
<td>22.921</td>
</tr>
</tbody>
</table>

Table 2: Execution time characterization in milliseconds

<table>
<thead>
<tr>
<th></th>
<th>Malloc(C)</th>
<th>Malloc(G)</th>
<th>CopyC2G</th>
<th>AddVec(G)</th>
<th>CopyG2C</th>
<th>Free(G)</th>
<th>Free(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N=500000</td>
<td>-</td>
<td>37.9</td>
<td>-</td>
<td>146.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>N=5000000</td>
<td>-</td>
<td>37.2</td>
<td>-</td>
<td>146.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>N=50000000</td>
<td>-</td>
<td>41.2</td>
<td>-</td>
<td>146.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: Dynamic power consumption characterization in Average Watt

→ No impact on power consumption for the kernel execution.
Case study 2: number of Threads/block impact

- For a longer run time, we used a constant data size $N=5*10^6$.
- We vary the number of threads per block (multiple of 32) from 128 to max 1024.

<table>
<thead>
<tr>
<th></th>
<th>Malloc(C)</th>
<th>Malloc(G)</th>
<th>CopyC2G</th>
<th>AddVec(G)</th>
<th>CopyG2C</th>
<th>Free(G)</th>
<th>Free(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T/B= 128</td>
<td>0.01</td>
<td>2217.81</td>
<td>12.80</td>
<td>234380.4</td>
<td>14.60</td>
<td>0.35</td>
<td>2.43</td>
</tr>
<tr>
<td>T/B= 256</td>
<td>0.01</td>
<td>2214.74</td>
<td>12.76</td>
<td>233362.03</td>
<td>14.82</td>
<td>0.36</td>
<td>2.76</td>
</tr>
<tr>
<td>T/B= 512</td>
<td>0.01</td>
<td>2214.60</td>
<td>12.77</td>
<td>242002.4</td>
<td>14.64</td>
<td>0.32</td>
<td>2.53</td>
</tr>
<tr>
<td>T/B= 1024</td>
<td>0.01</td>
<td>2214.65</td>
<td>12.92</td>
<td>235320.53</td>
<td>14.28</td>
<td>0.34</td>
<td>2.59</td>
</tr>
</tbody>
</table>

Table 4: Execution time characterization in milliseconds

<table>
<thead>
<tr>
<th></th>
<th>Malloc(C)</th>
<th>Malloc(G)</th>
<th>CopyC2G</th>
<th>AddVec(G)</th>
<th>CopyG2C</th>
<th>Free(G)</th>
<th>Free(C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T/B= 128</td>
<td>-</td>
<td>44.3</td>
<td>-</td>
<td>141.6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T/B= 256</td>
<td>-</td>
<td>31.5</td>
<td>-</td>
<td>150</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T/B= 512</td>
<td>-</td>
<td>39.8</td>
<td>-</td>
<td>153.3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>T/B= 1024</td>
<td>-</td>
<td>39.4</td>
<td>-</td>
<td>142</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5: Dynamic power consumption characterization in Average watt

- A slight impact on the execution time and the dynamic power consumption.
- keeping the GPU busy, does not increase the power consumption further.
- Focus on the energy consumption and the energy efficiency!
Case study 3: number of blocks & Active SMs impact

- We vary the number of active multiprocessors from 1 to the maximum (for Tesla M2075, 14 SMs).
- Indeed, we vary the number of blocks such that only one block can be executed in each SM.
Conclusions & Future works

- Investigate the irregular behavior in the power profile when having 14 blocks distributed to 14 SMs, more precisely the scheduling process proposed by NVIDIA.
1. Introduction
2. Context: GPU architecture & CUDA execution model
3. Our macroscopic analysis of GPU power consumption
   a. State-of-art on GPU Power Analysis
   b. Our Methodology
   c. Experimental results & Analysis
4. Simulating the power consumption of High Performance GPU-based Applications with SimGrid
   a. State-of-art on GPU Power Modeling
   b. Our proposition: From SimGrid to GPUSimGreen
5. Conclusion & Future works
State-of-art (1)

GPU Power models and Simulators

- [Sheaffer2005] propose a functional performance, power and temperature simulator: **Qsilver**: the first microarchitectural simulator for GPUs.
- [Lucas2013] and [Leng2013] propose respectively **GPUSimPow** and **GPUWattch**. Two power models build on the GPU performance simulator **GPGPU-Sim**. Both models rely on the **McPAT** tool to model GPU microarchitectural components.

Limitations:

- Such models require a deep knowledge of the architecture.
- GPU architecture is evolving very fast.
- Detailed product specifications are not usually public.
Our Proposition

- A GPU power model that is simpler, more flexible and portable through different generations of GPUs.
- **Simulation is an excellent approach to study HPC applications behavior in time and power.**
- Our proposition is then to simulate the performance and power consumption of HPC applications for GPUs using an open-source toolkit SimGrid.
- Inspiration: work done by [Heinrich2017] for CPUs in SimGrid.
Why SimGrid?

How simulate it inside?

- A free scientific tool for simulating different distributed systems such as grids, clouds, HPC or P2P systems = **reproducible**
- Provides accurate yet **fast** simulation models.
- Offers off-line and on-line simulation.

---

![Diagram of SimGrid components](attachment:image.png)
How CPU is modeled?
What we propose for GPUs?

**CPU model in SimGrid**
- Computational resources: cores with capacity C (Flops).
- Resource sharing algorithm
- Power is a linear to CPU usage

**Our GPU model in SimGrid**
- Computational resources: SMs with capacity C (Flops) as a black box
- Round Robin algorithm (as Nvidia says) on blocks!!

Source: SimGrid tutorial
Source: [https://users.ices.utexas.edu/~sreepai/fermi-tbs/](https://users.ices.utexas.edu/~sreepai/fermi-tbs/)
On going (slowly but surely..)

- Developing our GPU performance and power models in SimGrid.
- Validating our model with real measurements of applications from the SHOC benchmark suite, NAS, and machine learning applications.
- Extend our power model to support GPU DVFS in SimGrid.
Thank you for your attention!

Questions?
References

- **[Top500]**: Top500 website, URL = [https://www.top500.org/](https://www.top500.org/)
- **[Green500]**: online, URL = [https://www.top500.org/green500/](https://www.top500.org/green500/)
- **[Collange2009]**: “Power consumption from a software perspective”, in International Conference on computational Science, 2009
- **[Huang2009]**: “On the energy efficiency of graphics processing units for scientific computing”, IPDPS 2009
- **[Cebri'n2012]**: “Energy efficiency analysis of GPUs”, International Parallel and Distributed Processing symposium workshops PhD forums, 2012
- **[Sheaffer2005]**: “Fine-grained graphics architectural simulation with Qsilver”, ACM SIGGRAPH, 2005
- **[Leng2013]**: “GPUWattch: Enabling energy optimizations in GPGPUs, ISCA, 2013
- **[Heinrich2017]**: “Predicting the energy consumption of MPI applications at scale using a single node”, CLUSTER, 2017
How CPU power is modeled in SimGrid?

- The power consumption is a **linear function** of the CPU usage (proved by real experiments).
- For this example, 100W is the idle power, and 200W is the power when the CPU is fully loaded.
- So when the CPU is 50%, we have 150W
State-of-art (1)

GPU counter-based Power models

\[ \text{Power} = \text{Dynamic}_{\text{power}} + \text{Static}_{\text{power}}. \]

- Rely on performance counters to yield correlation to power consumption
- Methods used are linear (like SLR support linear regression etc,) or non-linear (RF random forest, ANN artificial neural network, K-means etc,)
- **Limitations:**
  - The number and type of counters are not uniform across hardware.
  - Some architectures only allow counters for a whole SM.

<table>
<thead>
<tr>
<th>Method</th>
<th>Device</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVR</td>
<td>Nvidia 8800GT</td>
<td>2009</td>
</tr>
<tr>
<td>SLR</td>
<td>Nvidia Tesla GTX285</td>
<td>2010</td>
</tr>
<tr>
<td>RF</td>
<td>AMD Radeon HD5870</td>
<td>2011</td>
</tr>
<tr>
<td></td>
<td>/Tesla CTX280</td>
<td></td>
</tr>
<tr>
<td>ANN</td>
<td>Nvidia fermi C2075</td>
<td>2013</td>
</tr>
<tr>
<td>K-means</td>
<td>AMD Radeon HD7970</td>
<td>2015</td>
</tr>
</tbody>
</table>