# Reducing HPC energy footprint for large scale GPU accelerated workloads

Gabriel Hautreux
HPC Department
CINES
France
hautreux@cines.fr

Etienne Malaboeuf
HPC Department
CINES
France
malaboeuf@cines.fr

## ABSTRACT

As the energy cost continues to rise, High Performance Computing (HPC) centers may seek to reduce their energy footprint. This could be implemented as a temporary production shutdown or in a more scientific production friendly way where the machine is able to increase its viability through increased efficiency. In this document we examine the second approach using a French, production ready machine hosted at Centre Informatique National de l'Enseignement Supérieur (CINES) in Montpellier. This machine, Adastra, is based on the AMD MI250X GPU architecture and currently #3 in Green500. Adastra is used by hundreds of French researchers, representing dozens of different applications from different scientific fields. As a base for the study, we define a set of applications representative of our current HPC and AI production workload. In this parametric study, we characterize our very diverse workload by applying a range of frequency capping or power capping policies at the node level in order to build an efficiency profile of each application. Based on the collected results, we produce guidelines trading between pure energy savings (energy to solution) to pure performance (time to solution) for each applications and, more importantly, for the production workload as a whole. We hope the results of this study will be of help to accelerators enabled HPC centers seeking to reduce their energy footprint by applying policies on either accelerators frequency or power capping at the node level.

## KEYWORDS

HPC, Energy efficiency, GPU, frequency caping, power management

## 1 Introduction

As energy costs continue to rise, High Performance Computing (HPC) centers are increasingly looking to reduce their energy footprint. In this document, we aim to address this issue by developing guidelines for more efficient management of energy consumption at the HPC center level, specifically by optimizing the handling of HPC workloads.

The study is performed on Adastra, a French supercomputer hosted at CINES, a Tier 1 computing center located in Montpellier, France.

The Adastra [1] supercomputer is an HPE-Cray EX system, ranked #11 in TOP500 and #3 in GREEN500 in Nov. 22.

Adastra is based on two main partitions, an accelerated one and a scalar one. The study is performed only on the accelerated partition of the machine.

Adastra accelerated nodes are similar to those of Frontier [2] (#1 TOP500 Nov. 22) and LUMI [3] (#3 TOP500 Nov. 22), known as « Trento nodes » :

- 1 AMD Trento EPYC 7A53 64-Core@3.5 GHz processor with 256 Gio of DDR4-3200
- 4 AMD MI250X@1.7GHz with 512 Gio of HBM2 per node
- 4 Slingshot 200 Gb/s NICs

In this parametric study, we define an HPC workload based on widely-used codes within

1 https://www.top500.org/system/180051/

2 https://www.top500.org/system/180047/

3 https://www.top500.org/system/180048/

the French research community, which are representative of the targeted production on the Adastra system. This workload encompasses applications from various fields, such as chemistry, astrophysics, plasma physics, and others, and exhibits diverse behaviors, including compute-intensive and memory bandwidth-intensive characteristics. While these applications are not covered in detail in this document, they form our workload, assessed with different policies for energy efficiency.

**This workload can be run on the entire system N times a year, consuming an energy of E for each run, i.e. our global energy consumption is equal to N*E per year.**

Results for time to solution are reported by using the time command while results for energy are reported using Slurm, based on energy consumption at node level. Hence, energy results do not include the network and services of the cluster (storage, login nodes, admin nodes, ..) but we do know that those energy costs are constant during the exploitation of the system, and only represent a fraction of the total power consumption.

Three studies were performed on our system for the given workload :

1. Impact of the CPU turbo mode usage
2. Impact of frequency capping
3. Impact of power capping

## 2 Impact of the CPU turbo mode usage

In order to start our studies, we wanted to make sure in which mode we would operate the Adastra GPU partition. In that perspective, we noticed that the turbo mode, used on the CPUs of the Trento nodes, could be activated in order to obtain better performances on our workload.

The first test was to run all our benchmarks using turbo mode, activated at runtime by the slurm prolog available in appendix 1, or not activating it.

Relative speed-up for using turbo on each application and relative energy consumed are computed and displayed in Figure 1.

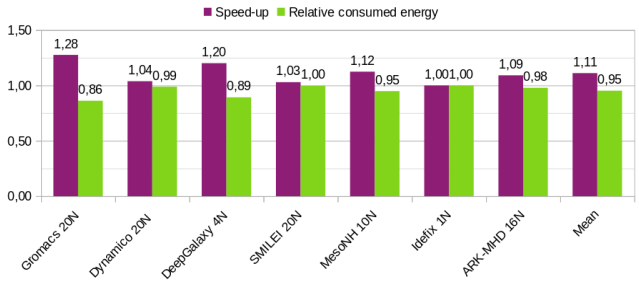


**Figure 1:** Impact of CPU turbo mode on our workload

The impact of turbo on CPU for time to solution is significant for some applications (Gromacs, DeepGalaxy, MesoNH, ARK-MHD). Those same applications consume less energy using turbo mode.

Each GPU can consume up to 560W [4], i.e. 2 240W in total per node, while the CPU part only consumes 280W[5], meaning 2 520W in total for the full node.

The CPU is only 11 % of the total TDP, increasing the CPU consumption by enabling turbo mode should not impact the energy performance of the full node.

Given the results provided in Figure 1, we validate the assumption that the power increase of the CPU part while using turbo is negligeable at the node level as the relative energy consumption is always equal or lower than 1.

On the other hand, we see that using turbo mode on the CPU enables to run the workload with a mean speed-up of 1.1 and with a relative energy cost of 0.95.

Over the year, we will then run 1.1*N times our workload, using each time 0.95*E of energy.

The global cost will be :

$$1.1 * 0.95 * N * E = 1.045 * N * E$$

meaning **an increase of 4.5 % of our energy bill for using the turbo mode on the Trento nodes.**

However, we have seen that the impact was significant for some applications (up to 1.28x speed-up) which is relevant enough to validate the usage of turbo mode on the system despite the raise of the energy cost.

**All the results presented below are using the turbo mode of the CPU on the Trento nodes.**

4 https://www.amd.com/en/products/server-accelerators/instinct-mi250x

5 https://icl.utk.edu/files/publications/2022/icl-utk-1570-2022.pdf

## 3 Frequency and power capping studies

Those studies are performed using the Energy Reporting tool In SLURM (ERIS, see appendix II). This script ensures that we are using the exact same nodelist for all the runs performed in order to minimize the noise in the output results. This script uses two spank plugins enabling the frequency and power capping.

Those plugins are using :

- rocm-smi[6] for FPU frenquency capping
- GPU device driver values (e.g. /sys/class/drm/card0/device/hwmon/hwmon0/power1_cap)

### 3.1 Frequency capping study

The study is performed by setting the maximum frequency from 0.8GHz to 1.7GHz. For each value, the normalized TTS (Time To Solution) and ETS (Energy To Solution) are computed, the 1.7GHz value is given as the reference value.

This study aimed at discovering a trend in our HPC workload which could lead us to find the most suitable frequency in order to run our workload using less energy without an important impact on the performance of our applications.

Once all the applications were run, we can define the best values of frequency for each application. This enables us to get the optimal consumed energy for the workload as each application is run at the GPU frequency enabling the less energy consumed for this given application.

Results are reported in the following table :

**Frequency capping study**

| Applications | Best relative energy | Corresponding frequency | Speed-up |
|---|---|---|---|
| Gromacs 20N | 0.85 | 0.8 | 0.99 |
| Dynamico 20N | 0.89 | 1.2 | 0.92 |
| SMILEI 20N | 1.00 | 1.7 | 1 |
| MesoNH 10N | 0.97 | 1.4 | 0.94 |
| Idefix 1N | 1.00 | 1.7 | 1 |
| ARK-MHD 16N | 0.89 | 1 | 0.98 |
| Namd 1N | 0.93 | 1.2 | 0.84 |
| **Mean** | **0.93** | | **0.95** |

In this specifc case, called the « Fine-Tuned case », the power consumption of our workload can be globally reduced by 7 %, while losing only 5 % of performance.

This case is the optimal one, but is not really realistic to implement at the scale of the supercomputer as it means we will need to investigate each application to find the best ETS value.

In that perspective, we can also plot the results concerning our workload for all the frequencies studied. Results are provided in Figure 2.

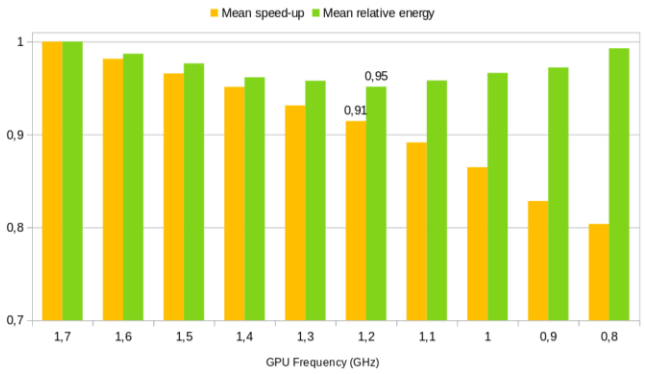


**Figure 2:** Speed-up and relative energy consumption of the workload depending on the GPU freq

In this scenario, we can clearly see that there is no gain in energy on our workload if we do not conceide to a larger decrease of the performance.

However, we can see a sweet spot at 1.2GHz for the GPU frequency where the relative energy is the lowest for our global workload. While this value of 0.95 is bigger than the 0.93 of the Fine-Tuned case, it is still an interesting energy improvement at the cost of 9 % of performance decrease.

By using the same approach as the one we used for the turbo mode, we can compute our yearly energy savings for each GPU frequency. We can also compute the « science loss », which is equal to the percentage of workloads we will not run on the system (i.e. science performed on the system). The results are displayed in Figure 3.

This graph summarize the results and computed savings for our workload, which start form 0 (at standard 1.7GHz per GPU) up to 20 % of energy savings (by reducing the GPU frequency down to 800MHz). In this

6 https://docs.amd.com/bundle/ROCm-SMI-v5.3/page/md__home_aelwazir_Downloads_rocm_smi_lib_README.html

graph, we can see that the science loss is always less than the energy savings, however sticking to a value of 1.2GHz, as mentionned before, could be a good balance between the savings over the year (13%) and the science loss (8.5 %, which may be reasonable).

The Fine-Tuned case enables slightly less energy savings (11.2%) but implies 4 % less science loss over the year.

## 3.2 Power capping study

The study is performed by setting the power cap for the following values : [300, 350, 400, 450, 500, 560]. For each value of power capping, the TTS and ETS are recorded.

This study aimed at discovering the impact of power capping on our HPC workload. Ideally, power capping could be a good option to reduce our power bill at the end of the year.

While performing runs using the power capping tools, we discovered that the values of average power consumed by the GPUs may be above the powercap set using the rocm-smi tools. In that perspective, we can not be totally sure that the results provided are fully in line with wath we would expect (i.e. hard power cap).

You can see in Figure 4 that power capping is set at 300W where the average power is up to 356W for GPU4, which is a raise of more than 15 %.

```
====================== ROCm System Management Interface ======================
================================ Concise Info ================================
GPU  Temp   AvgPwr  SCLK    MCLK     Fan  Perf    PwrCap  VRAM%  GPU%
0    54.0c  328.0W  770Mhz  1600Mhz  0%   manual  300.0W   13%   100%
1    50.0c  N/A     755Mhz  1600Mhz  0%   manual  0.0W     13%   100%
2    59.0c  292.0W  610Mhz  1600Mhz  0%   manual  300.0W   13%   100%
3    54.0c  N/A     620Mhz  1600Mhz  0%   manual  0.0W     13%   100%
4    51.0c  356.0W  695Mhz  1600Mhz  0%   manual  300.0W   13%   100%
5    50.0c  N/A     680Mhz  1600Mhz  0%   manual  0.0W     13%   100%
6    59.0c  338.0W  600Mhz  1600Mhz  0%   manual  300.0W   13%   100%
7    53.0c  N/A     585Mhz  1600Mhz  0%   manual  0.0W     13%   100%
==============================================================================
========================= End of ROCm SMI Log ================================
```

**Figure 4:** Example of power capping

However, we still decided to try and got the results for the workload, reported in the following table :

| Power capping study | | | |
|---|---|---|---|
| **Applications** | **Best relative energy** | **Corresponding Power Cap** | **Speed-up** |
| Gromacs 20N | 0.99 | 350W | 1.01 |
| Dynamico 20N | 0.91 | 350W | 0.87 |
| SMILEI 20N | 1.00 | 500W | 1.00 |
| MesoNH 10N | 1.00 | 500W | 1.00 |
| Idefix 1N | 1.00 | 560W | 1.00 |
| ARK-MHD 16N | 0.94 | 300W | 0.99 |
| Namd 1N | 0.95 | 300W | 0.79 |
| **Mean** | **0.97** | | **0.98** |

The best energy savings we could get while power capping was only 9 % for a given application, and 3 % as a global mean value, with very limited impact on the speed-up (2 % of performance decrease).

This power capping method was giving less energy gain than the frequency capping one, and was not sure enough in our perspective due to the frequency spikes we have seen while running with power capping enabled.

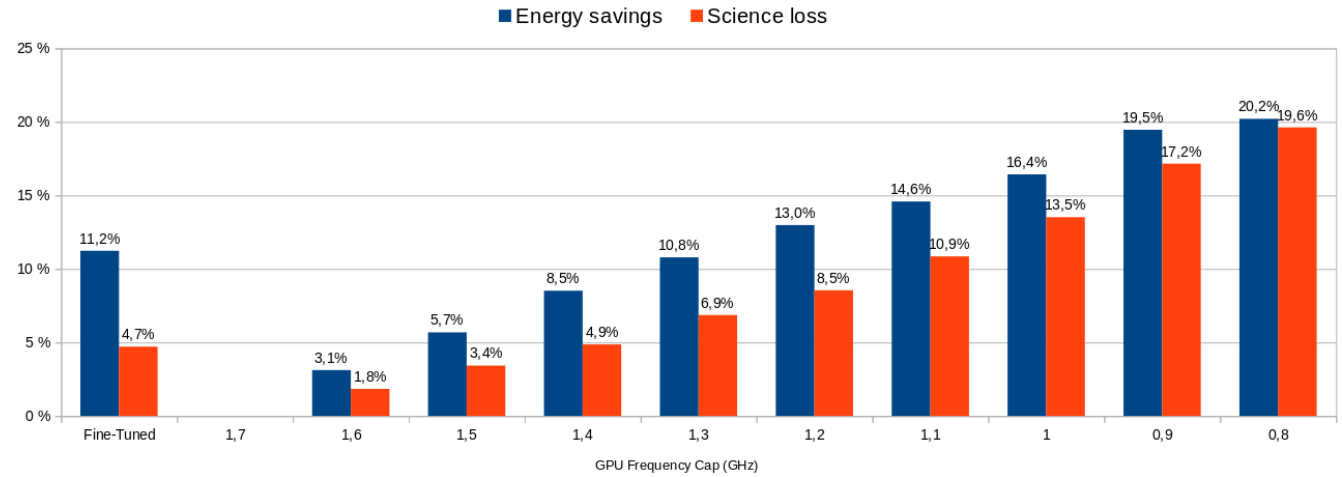


Figure 3: Energy savings for running the workload over a year depending on GPU frequency

We decided to drop this study for now and investigate into the power capping tools to better understand why the cap could not be respected at runtime.

## Conclusion

These studies have deepened our understanding of energy consumption issues in our HPC center. Initially, we anticipated that the study would reveal trends or sweet spots for optimizing the production system, but it led us to a new approach to the problem.

Indeed, it might seem that only TTS and ETS were the key in determining the optimal operation of an HPC system. But this study shows that, if you have a need to save the most energy out of your system, you must reduce the science produced on it.

By activating the turbo mode on our system, we achieved an 11 % increase in scientific output while raising energy consumption by only 4.5 %. This decision has been implemented and is now active on the Adastra system.

The energy increase has to be compensated somehow, as this increase has an important cost for any HPC center that may not be able to pay their bill at the end of the year. The frequency and power capping studies where designed in order to answer that specific issue.

The impact of frequency on our workload offers valuable insights that can be utilized to reduce energy consumption or costs when needed, even temporarily.

We hope that the results and scripts presented in this document will aid other HPC centers in conducting similar studies, providing them with the necessary tools to find the optimal balance between scientific output and energy consumption.

The fine-tuned approach remains intriguing in terms of energy savings and science performed, and should be considered if feasible for a given workload. Further investigations could be conducted to classify applications and identify the most suitable frequency for each category.

## APPENDIX

### I. Turbo Slurm prolog

```
#### Enabling the CPU Turbo Mode ####
if [ $(echo $SLURM_JOB_CONSTRAINTS | grep -c "turbo") -gt 0 ] ; then

   # Setting the acpi-cpufreq scaling driver
   test -e /sys/devices/system/cpu/cpufreq/boost && echo 1 > /sys/devices/system/cpu/cpufreq/boost

   # Setting the acpi-cpufreq scaling driver specific to AMD EPYC
   if [ -e /sys/devices/system/cpu/cpu0/cpufreq/cpb ]; then
      for f in `ls -d /sys/devices/system/cpu/cpu*/cpufreq/cpb`; do
         echo 1 > $f
      done
   fi

else # Ensuring the Turbo Mode is disabled
   # Setting the acpi-cpufreq scaling driver
   test -e /sys/devices/system/cpu/cpufreq/boost && echo 0 > /sys/devices/system/cpu/cpufreq/boost

   # Setting the acpi-cpufreq scaling driver specific to AMD EPYC
   if [ -e /sys/devices/system/cpu/cpu0/cpufreq/cpb ]; then
      for f in `ls -d /sys/devices/system/cpu/cpu*/cpufreq/cpb`; do
         echo 0 > $f
      done
   fi

fi
```

### II. ERIS script

```
#!/bin/bash
set -eu

echo "Welcome to the Energy Reporting tool In SLURM (ERIS)."

# 800 - 1 to void requested GPU min sclk value '800' is greater than requested
# max '800'
GPU_FREQUENCY_CAP_MINIMUM=799
GPU_FREQUENCY_CAPS=(1700 1600 1500 1400 1300 1200 1100 1000 900 800)
GPU_POWER_CAPS=(560 500 450 400 350 300)

function SACCTLogging() {

            STEP_ROW="$(sacct --format=JobID,ElapsedRaw,ConsumedEnergyRaw,NodeList --parsable --jobs=${1} | tail -n 1)"

   STEP_INDEX="$(echo -n ${STEP_ROW} | cut -d '|' -f 1)"
   STEP_DURATION="$(echo -n ${STEP_ROW} | cut -d '|' -f 2)"
   STEP_ENERGY="$(echo -n ${STEP_ROW} | cut -d '|' -f 3)"

   # We overwrite NODE_LIST.
   JOB_NODE_LIST="$(echo -n ${STEP_ROW} | cut -d '|' -f 4)"

   printf '%s;%s\n' "${STEP_DURATION}" "${STEP_ENERGY}"
   # printf '%s;%s;%s\n' "${STEP_INDEX}" "${STEP_DURATION}" "${STEP_ENERGY}"
}

# No restriction for the first run.
NODE_LIST_RESTRICTION=""

echo "Frequency capping:"
for A_GPU_FREQUENCY_CAP in ${GPU_FREQUENCY_CAPS[@]}; do
 CURRENT_JOB_ID="$(sbatch --wait ${NODE_LIST_RESTRICTION} --gpu-srange=${GPU_FREQUENCY_CAP_MINIMUM}-${A_GPU_FREQUENCY_CAP} "${@}" | cut -d ' ' -f 4)"

   if [ "${CURRENT_JOB_ID}" == "" ]; then
      echo "The returned job id is incorrect."
      exit 1
   fi
```

```
    SACCTLogging "${CURRENT_JOB_ID}"
    # JOB_NODE_LIST is returned by SACCTLogging
    NODE_LIST_RESTRICTION="--nodelist=${JOB_NODE_LIST}"
done

echo "Power capping:"
for A_GPU_POWER_CAP in ${GPU_POWER_CAPS[@]};
do
 CURRENT_JOB_ID="$(sbatch --wait ${NODE_LIST_RESTRICTION}
--gpu-power-cap=${A_GPU_POWER_CAP} "${@}" | cut -d ' ' -f 4)"

    if [ "${CURRENT_JOB_ID}" == "" ]; then
        echo "The returned job id is incorrect."
        exit 1
    fi

    SACCTLogging "${CURRENT_JOB_ID}"
    # JOB_NODE_LIST is returned by SACCTLogging
    NODE_LIST_RESTRICTION="--nodelist=${JOB_NODE_LIST}"
done
```