Focus on the energy-efficiency of components of the IFS on Grace Hopper GH200 superchips

Andrew Beggs (ECMWF)
Samuel Hatfield (ECMWF)
Mathieu Stoffel (Eviden)

January 28th 2026

EUPEX

EuroHPC Joint Undertaking

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101033975. The JU receives support from the European Union's Horizon 2020 research and innovation programme and France, Germany, Italy, Greece, United Kingdom, Czech Republic, Croatia.

What are we going to talk about?

- What is the IFS, and more specifically ecTrans?

- Coarse-grain efficiency profiling of ecTrans on x86, aarch64 and GPU executions
  - Motivation [ECMWF]: get some insight on the most suited HW target for ecTrans

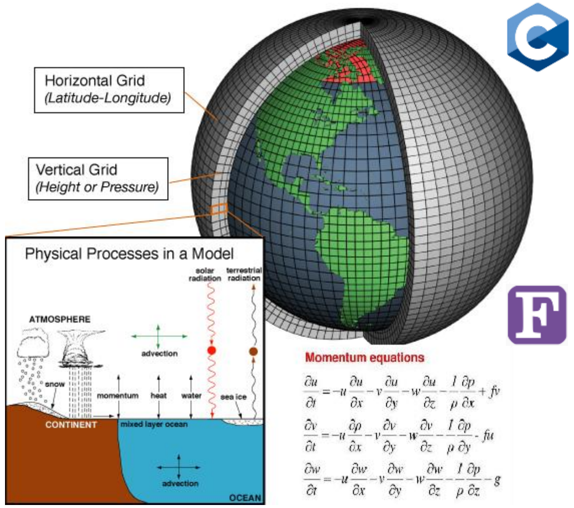
- Impact of GPU frequency scaling on ecTrans when executed on GPU
  - Motivation [Eviden]: efficiency improvement potential with (static) GPU frequency scaling while agnostic of the application (no prior knowledge, no source code modification, no recompilation)

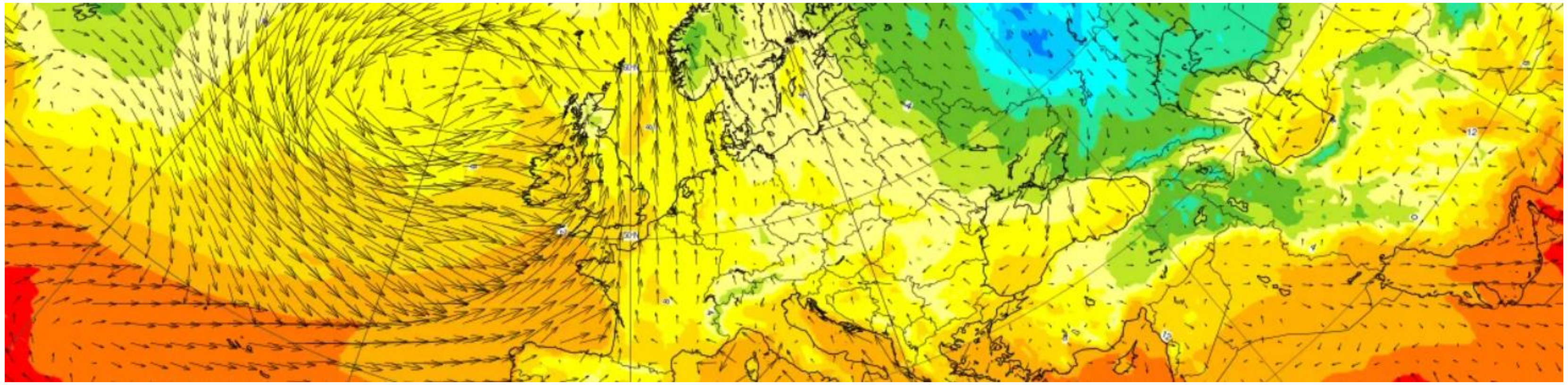
What are the IFS and ecTrans?

What is the IFS?

- The Integrated Forecasting System (IFS) is a sophisticated system which aims at forecasting weather-related physical quantities (e.g. temperature, pressure, rainfall, …) over Europe on a 15-day time scale
- The forecasts built with the IFS are highly precise (if they say it will rain, you should take an umbrella) and have a high resolution (one side of a cell of the computation grid represents between 8 and 9 km)
- This sophisticated system notably comprises data assimilation facilities, complex numerical models, and computing infrastructures

What is the IFS?





Focus on ecTrans

- ecTrans is “a high-performance numerical library for transforming meteorological fields between global grid-point space representation and a spectral representation based on spherical harmonics” (wiki)

- ecTrans belonged to the source code of the IFS, and was extracted from it and open-sourced in 2022 (source code)

- I am 2 years older than ecTrans – more than 30 years of development and optimization!

- Since ecTrans exhibits compute-intensive and communication-intensive phases and implements computational kernels quite common in weather-related codes, it would be a great HPC benchmark … Good news, one benchmark was built from it and is packaged with its source code

- One of the input parameters of ecTrans is the Truncation Cubic Octahedral (TCO). It sets the overall resolution of the computational grid, and hence of the benchmark. The higher the TCO, the larger the problem size. ECMWF production forecasts are made with TCO = 1279

Coarse-grain efficiency profiling of ecTrans on x86, aarch64 and GPU executions

Architectures of the nodes used for experiments

x86 node	aarch64 + GPU node
XH3150 (TINO) Eviden R&D system (RHEL97) GCC-15 + OpenMPI-5.0 2x Intel Granite Rapids 6952P 792 GB DDR5 (6400 MT/s) Infiniband NDR200	XH3515-MP (NEBA Max-P) Hosted in the SDV (RHEL95) NVHPC-25.7 4x NVIDIA GH200 modules • Grace CPU - 128 GB LPDDR5 • Hopper GPU - 96 GB HB3 • Infiniband NDR200

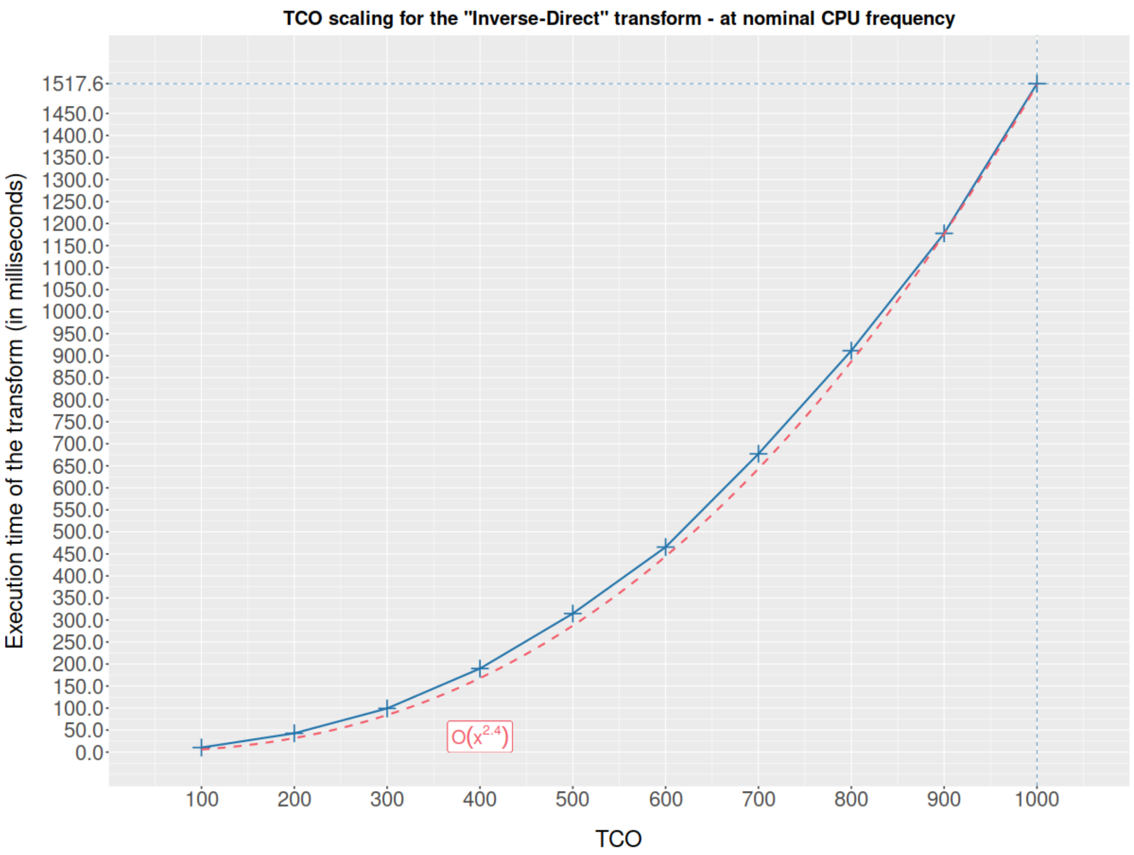
- Both nodes are state-of-the-art:
  - x86 – same model freshly installed at IT4I/VSB
  - aarch64 + GPU – twin model of JUPITER nodes
- In the following, “aarch64” means “only the Grace CPUs part of the node”

$$Efficiency = Eff = \frac{Sustained\ Performance}{Average\ Power\ Consumption} = \frac{Perf}{Pwr}$$

Experimental protocol

- Single-node study with the same experimental parameters on 3 different architectures (x86, aarch64, GPU) to compare the performance and efficiency of the ecTrans benchmark
- Scale the size of the workload by increasing TCO (multiples of 100) while monitoring (1 Hz out-of-band) the full node power consumption
- The ecTrans benchmark reports a performance metric – the duration of the "Inverse-Direct" transform
- The number of iterations of the computational kernel can be adjusted. With 500 iterations, the ecTrans benchmark lasts roughly 5 minutes for TCO=1000 on GPUs (plus it is enough repetitions for statistics to be relevant)
- Similarly, TCO=1000 was selected to compare architectures because that is the highest multiple of 100 for which the workload fits in the memory of the GPUs

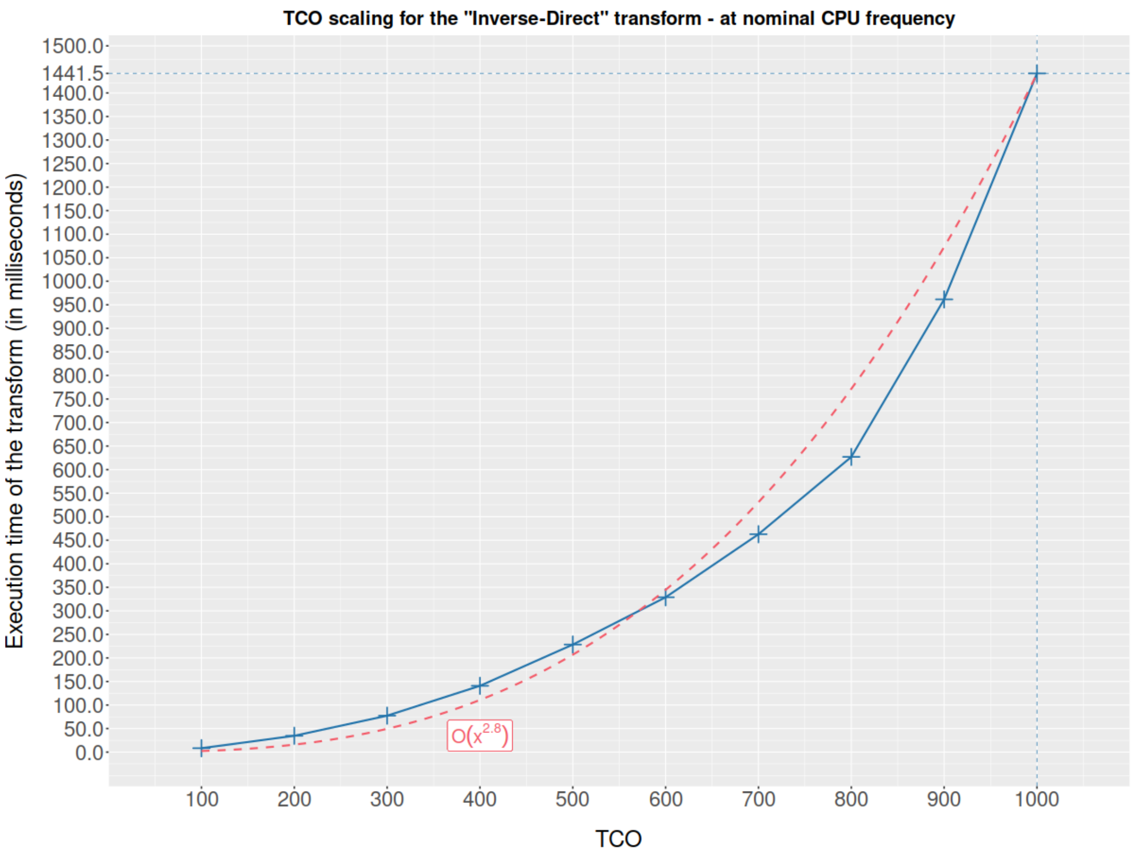
Coarse-grain profiling of ecTrans on x86



- Scaling the workload (i.e. the TCO) on a single Intel GNR compute node with execution of ecTrans on 2 x86 CPUs
- GCC-15 + OpenMPI-5.0
- Execution time is globally exponential ($\sim O(x^{2.4})$) of the TCO
- For TCO=1000 and 500 iterations:
  - Duration of “Inverse-Direct” transform = 1517.6 ms
  - Power consumption = 1014 W

EUPEX

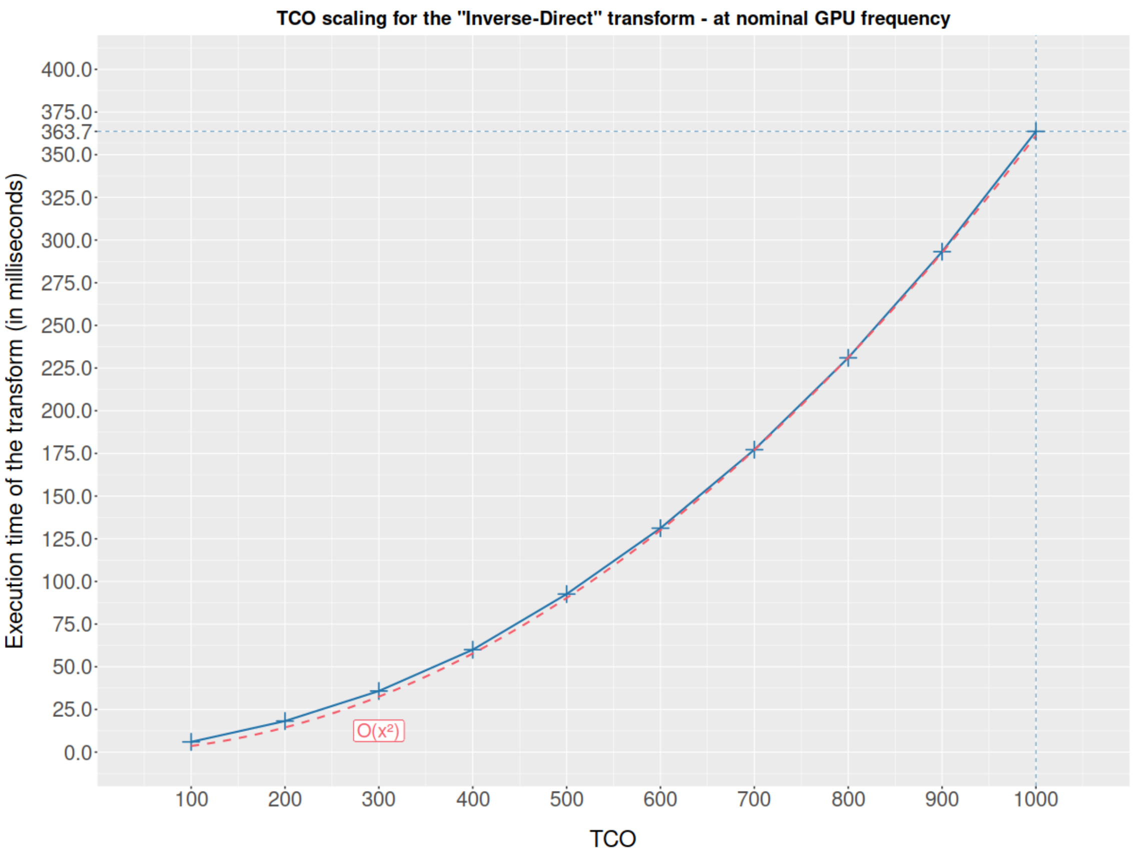
Coarse-grain profiling of ecTrans on aarch64



- Scaling the workload (i.e. the TCO) on a single GH200 compute node with execution of ecTrans on 4 Grace CPUs
- NVHPC-25.7
- Execution time is globally exponential ($\sim O(x^{2.8})$) of the TCO
- The power consumption of the GPUs was subtracted from the full node power consumption
- For TCO=1000 and 500 iterations:
  - Duration of “Inverse-Direct” transform = 1441.5 ms
  - Power consumption = 1053 W

EUpEX

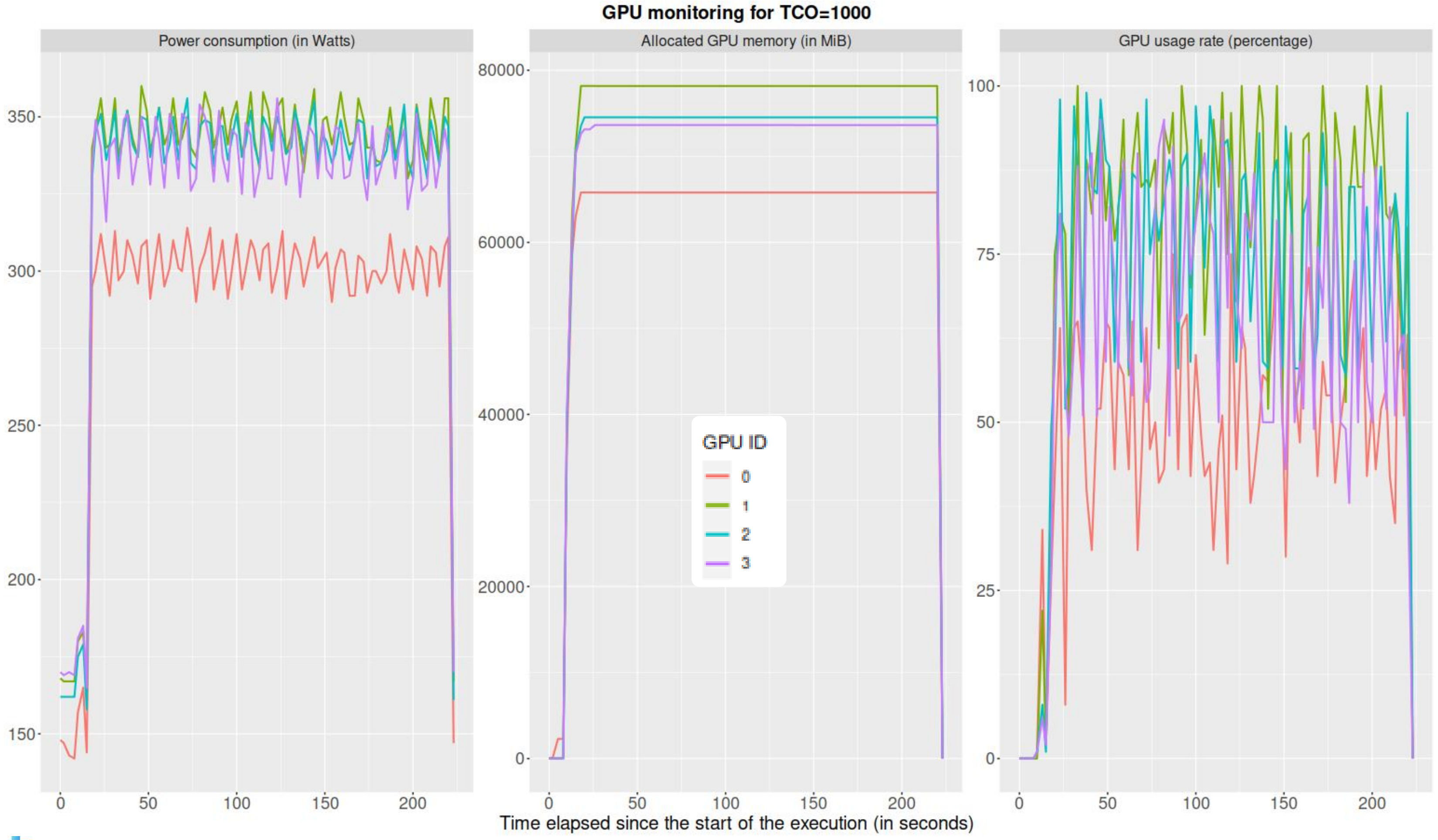
Coarse-grain profiling of ecTrans on GPU



- Scaling the workload (i.e. the TCO) on a single GH200 compute node with execution of ecTrans on 4 GPUs
- NVHPC-25.7 (including OpenACC)
- Execution time is globally quadratic ($\sim O(x^2)$) of the TCO
- TCO=1000 fills 80+% of the memory of the GPUs ("limiting factor" as stated previously)

- For TCO=1000 and 500 iterations:
  - Duration of "Inverse-Direct" transform = 363.7 ms
  - Power consumption = … small issue here, let's focus on it

Coarse-grain profiling of ecTrans on GPU



- Power consumption, allocated memory and GPU usage rate during the execution of ecTrans for TCO=1000
- Imbalance between the GPUs
- Erratic and limited usage rate of the GPUs (~65%)
- Low usage rate of the GPUs implies lower power consumption when compared to expectations (700+ W per GPU sustained for HPL)

Coarse-grain profiling of ecTrans on GPU

- We believe the low usage rate of GPUs stems from the fact that memory allocations and transfers of input data from host to accelerator are done in the main computational loop, not in bulk before the latter
- (Very) roughly, while moving data between host and accelerator, the GPUs are not computing - hence the low usage rate
- Consequently, the full node power consumption associated with ecTrans benchmark is not representative of the full node power consumption of ecTrans in the production context of the IFS (while performance are)

E.g. link to source code (L544-617) TRGTOL is a “leaf” function called in the main loop of the benchmark.
It transposes grid point data from column structure to latitudinal and it reorganizes data between grid point calculations and direct Fourier Transform.
In this function, like in all the other key “leaf” functions, OpenACC “COPYIN” directives are called just before “PARALLEL LOOP” directives.
Thus, data movement between host and accelerators happens at each iteration of the benchmark, just before the offloading of the nested computational loops.

Working around the power monitoring issue

- Since measuring full node power consumption for ecTrans benchmark is not relevant to estimate full node power consumption for production ecTrans, we can **build a worst-case scenario** for it.
- Idea: use GPU-Burn (link to source code) as a surrogate to build an **upper bound** on the power consumption of production ecTrans (and thus a **lower bound** on its efficiency)
- Indeed, GPU-Burn induces a 100% rate usage with very high and steady power consumption on the GPUs (it is quite unlikely that production ecTrans would consume more power than GPU-Burn).
- Executing GPU-Burn with an 80% memory occupancy for nominal/maximal GPU frequency draws 3684 W (more details about experiments with GPU-Burn in the last section)

Wrap up the efficiency comparison between x86, aarch64 and GPU

TCO = 1000

x86 node	
Inverse-Direct transform lasts:	1518 ms
Mean power consumption is:	1014 W
Relative efficiency w.r.t. aarch64:	- 1.35 %
^Relative efficiency w.r.t. GPU:	- 12.9 %

aarch64 node	
Inverse-Direct transform lasts:	1442 ms
Mean power consumption is:	1053 W
Relative efficiency w.r.t. x86:	+ 1.37 %
^Relative efficiency w.r.t. GPU:	- 11.7 %

GPU node	
Inverse-Direct transform lasts:	364 ms
Mean power consumption is:	< 3684 W
*Relative efficiency w.r.t. x86:	+ 14.8 %
*Relative efficiency w.r.t. aarch64:	+ 13.2 %

- Execution of ecTrans benchmark is significantly more efficient on GPU, even in a worst-case scenario
- There is a 4x speed up for single-node performance of ecTrans when going from CPUs to GPUs
- For ecTrans, GPU is also better than CPU from the density POV, since there is usually 3 CPU nodes per 1U blade, versus 1 GPU node per 1U blade

EUPEX

[*] Lower/ [^] Upper bounds based on the power consumption of GPU-Burn

Impact of GPU frequency scaling on ecTrans

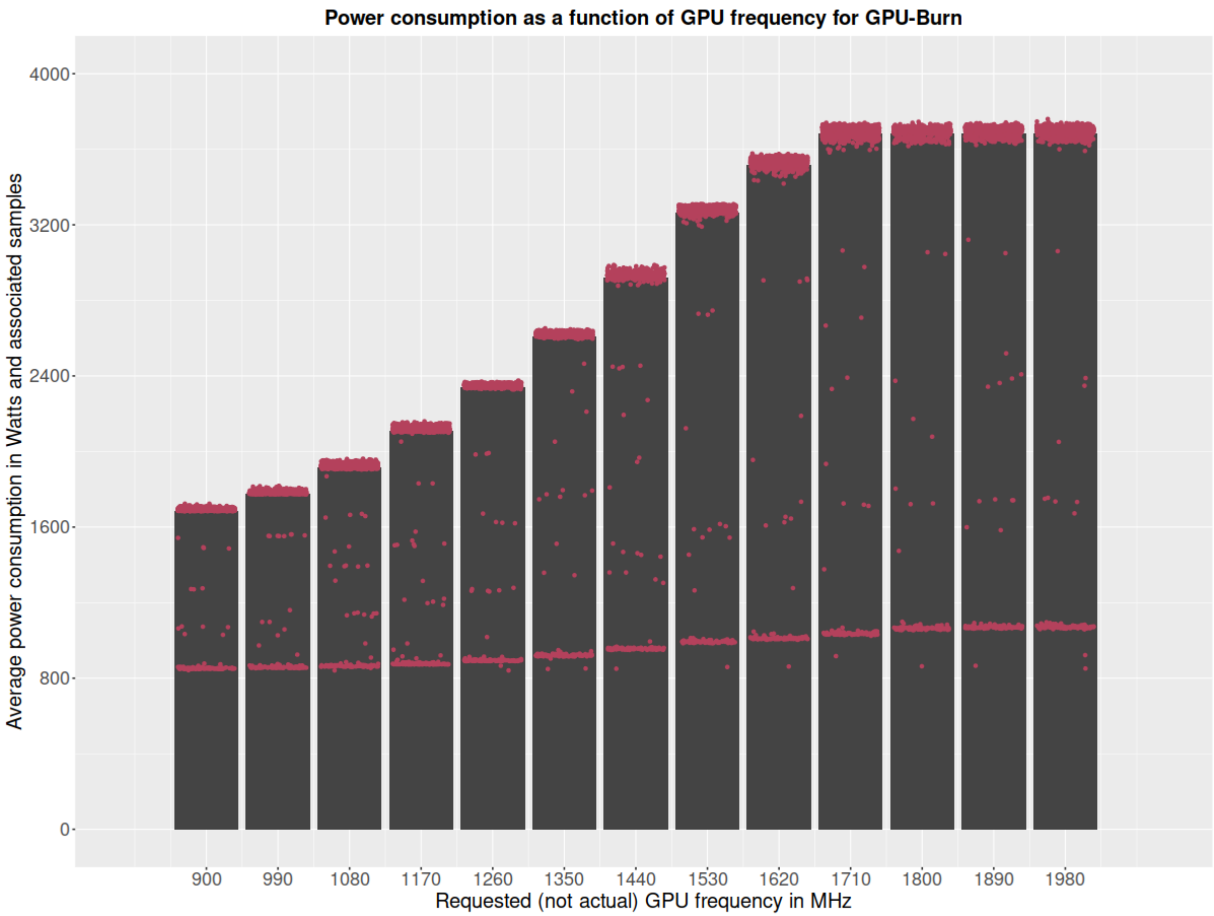
Goal of GPU frequency scaling

- GPU cores consume power when they are active, and the amount of power notably depends linearly on the operating frequency of the GPUs ($P \propto A \times f \times V^2$, with Activity, Frequency and Voltage)
- Scaling down the frequency decreases the power consumption and peak reachable computational capability supplied by the GPU …
- … but not necessarily application effective performance! Think of all the "wasted" cycles waiting for data to be moved from the memory of the GPUs to their cores
- If scaling down the frequency reduces the power consumption more than the induced performance degradation, you have an efficiency gain
- This is precisely our goal: determine if we can improve the energy efficiency of ecTrans by using a lower GPU frequency

Experimental protocol

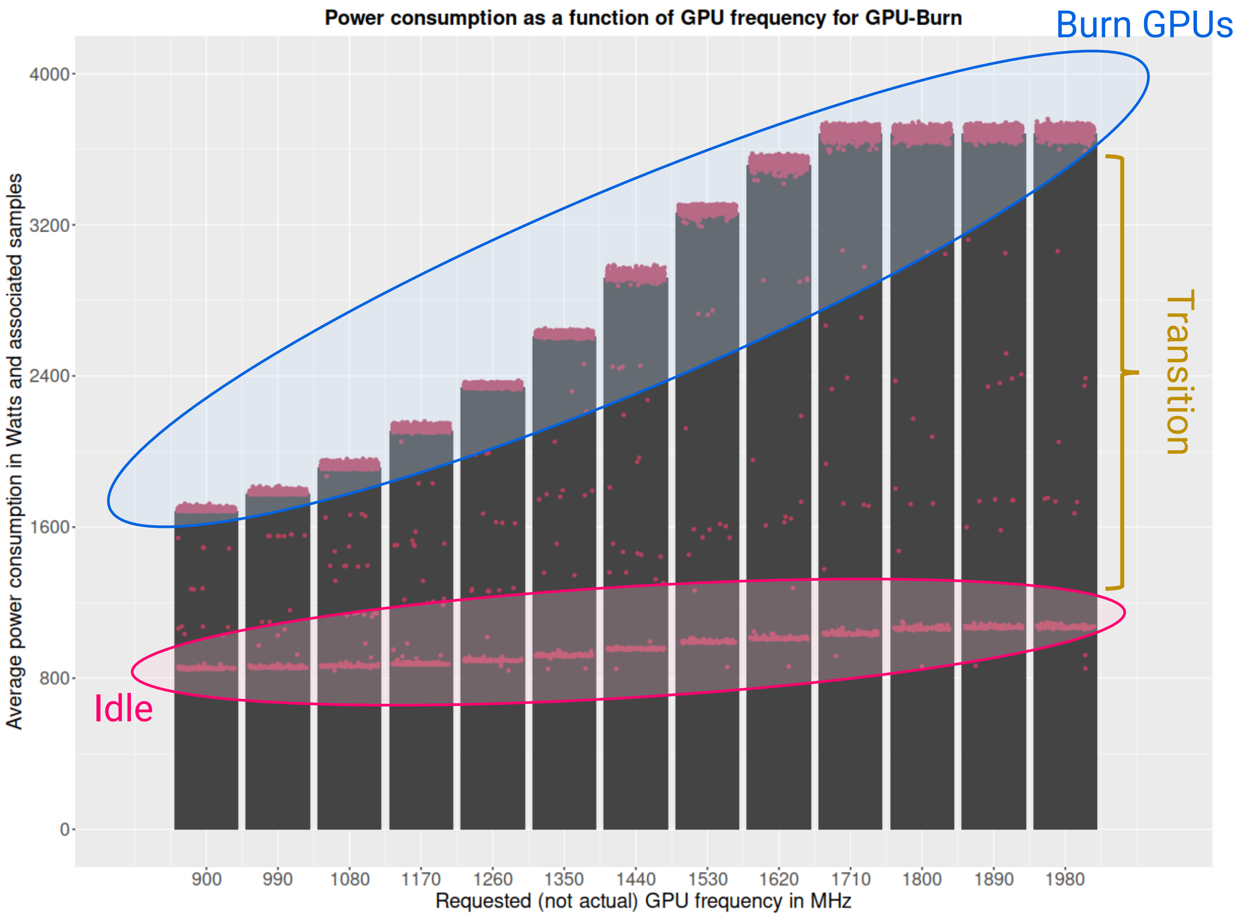
- Goal: evaluate the impact of GPU frequency scaling on the efficiency of ecTrans – is the nominal/maximal frequency the best from the efficiency POV?
- Execute ecTrans benchmark (TCO=1000, 500 iterations) for a relevant subset of the available GPU frequencies $(900\ MHz \xrightarrow[90\ MHz\ step]{} 1980\ MHz)$ while monitoring power consumption (1 Hz out-of-band)
- Also execute GPU-Burn with the same protocol to build a reference table between “GPU frequency” and “steady power consumption” for an extremely intense workload
- This reference table will be called an “abacus” in the following, and will help us estimate the efficiency of production ecTrans by substituting the power consumption of GPU-Burn instead of the one of ecTrans benchmark
- Each data point is the average of 3 repetitions of the concerned experiment sample

GPU-Burn and the power consumption “abacus”



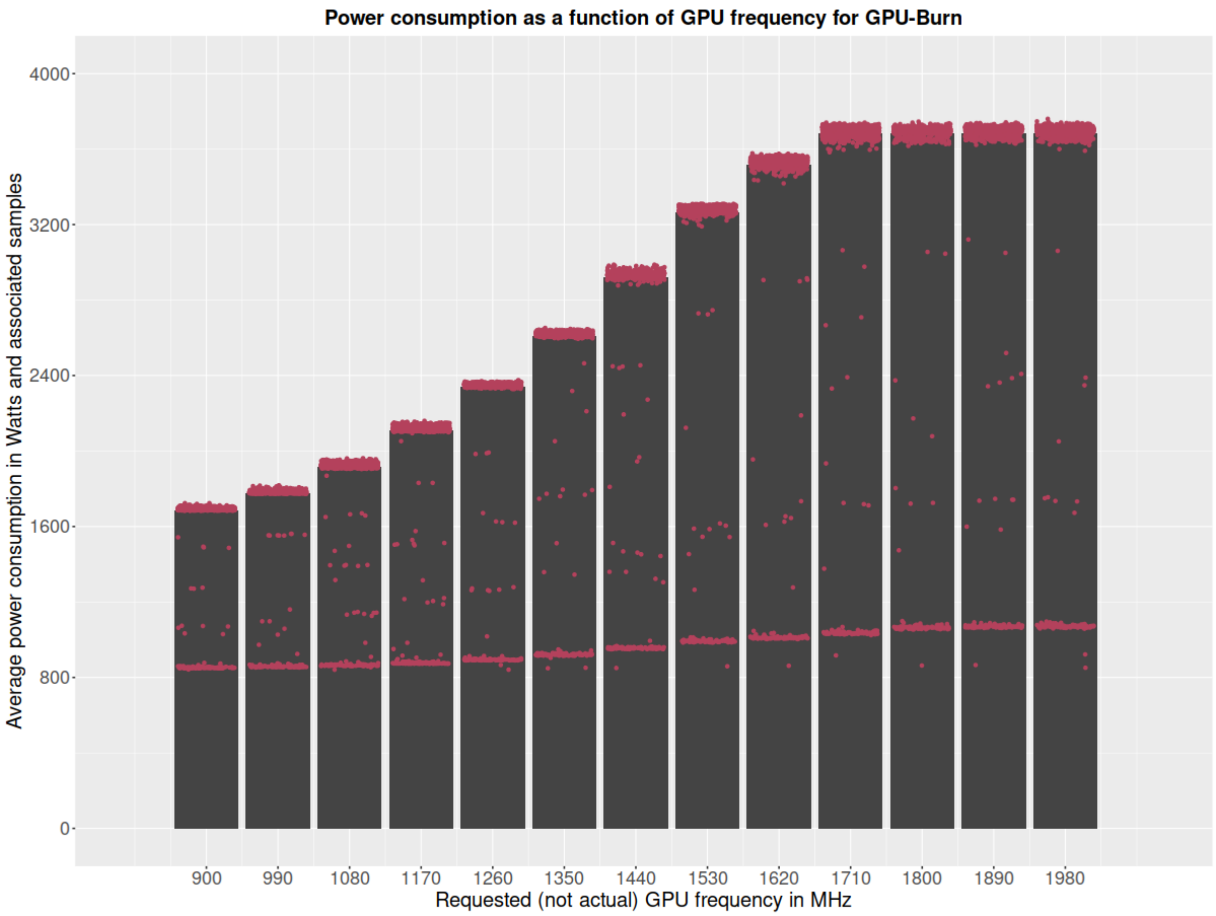
- For each selected GPU frequency, the following sequence was executed 3 times:
  - 30 seconds of “idleness” (rather “very low activity”)
  - 5 minutes of burning GPUs
  - 30 seconds of “idleness”
- (Almost) identical results for each repetition so the samples were considered altogether
- Red dots are 1Hz power samples
- Bars are average power consumptions for the 5 minutes of burning GPUs – this is the abacus

GPU-Burn and the power consumption “abacus”



- For each selected GPU frequency, the following sequence was executed 3 times:
  - 30 seconds of “idleness” (rather “very low activity”)
  - 5 minutes of burning GPUs
  - 30 seconds of “idleness”
- (Almost) identical results for each repetition so the samples were considered altogether
- Red dots are 1Hz power samples
- Bars are average power consumptions for the 5 minutes of burning GPUs – this is the abacus

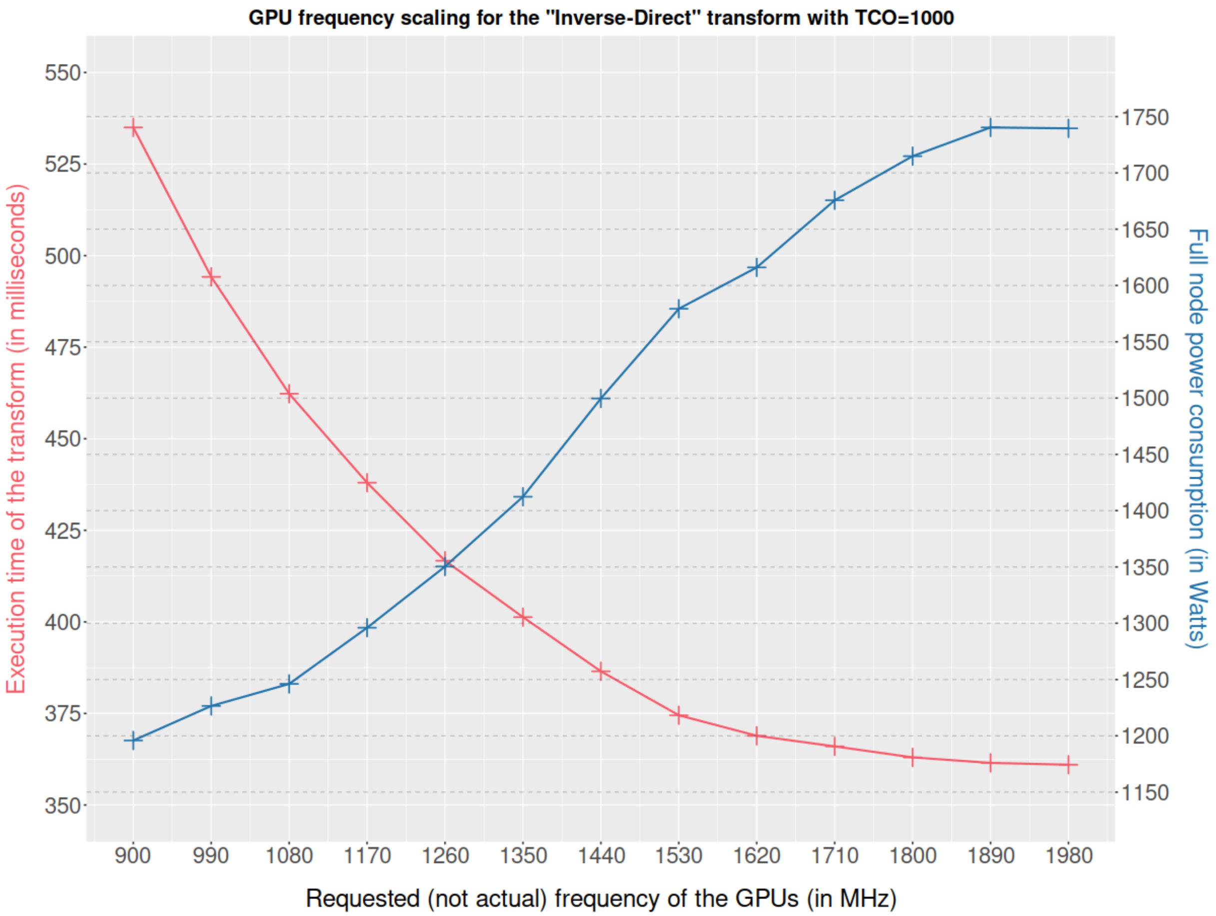
GPU-Burn and the power consumption “abacus”



- Focus on the plateau of power consumption from 1710 MHz and above (best hypothesis – proving it is future work):
  - Maximum consumption for the node is 4kW (enforced through a system power cap)
  - Power consumption for a given frequency notably depends on the precision of silicon etching
  - To fully exploit frequencies above 1710 MHz, the GPUs and hence the node would need to consume more power than 4kW
  - On this node, the highest enforcable frequency for a compute-intensive workload with 100% usage rate seems to be 1710 MHz

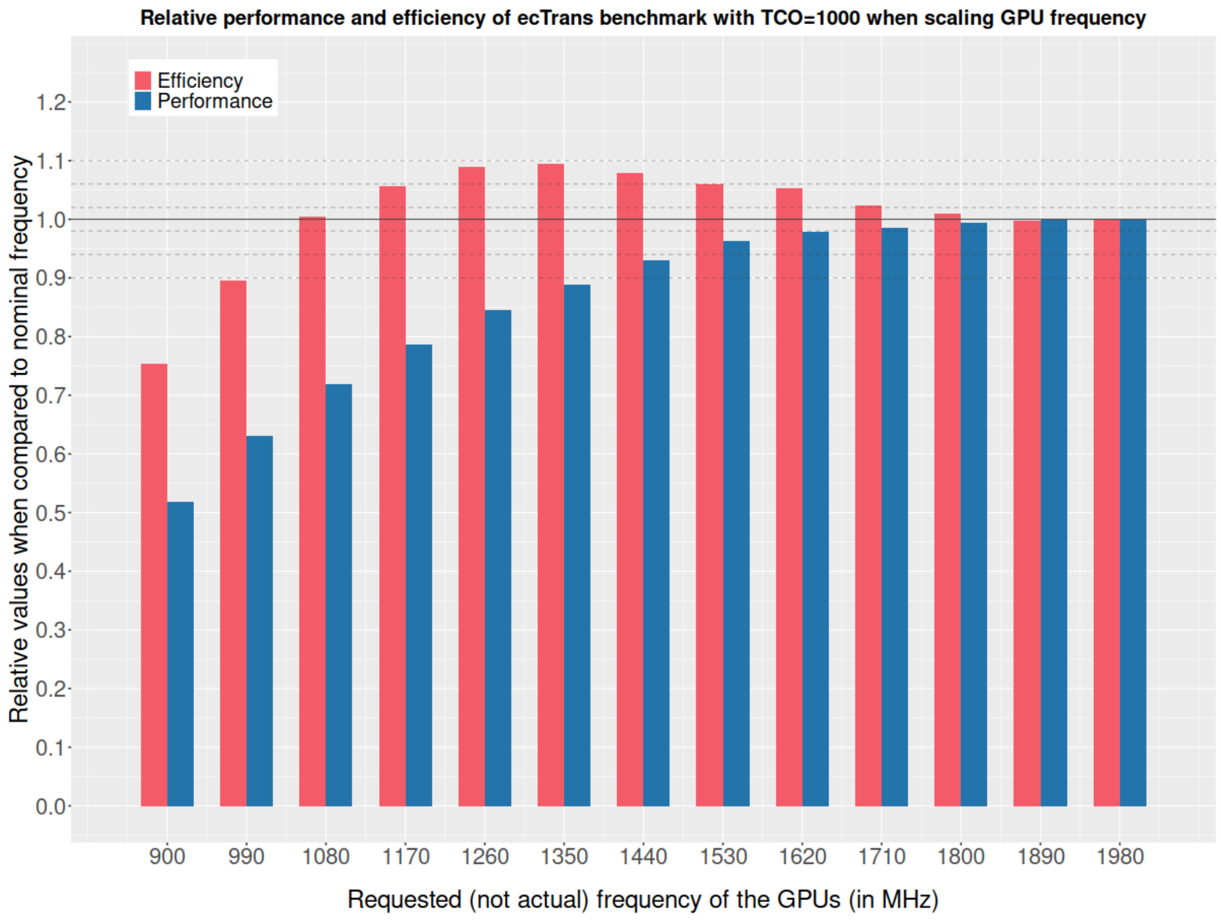
EUPEX

Scaling GPU frequency for ecTrans



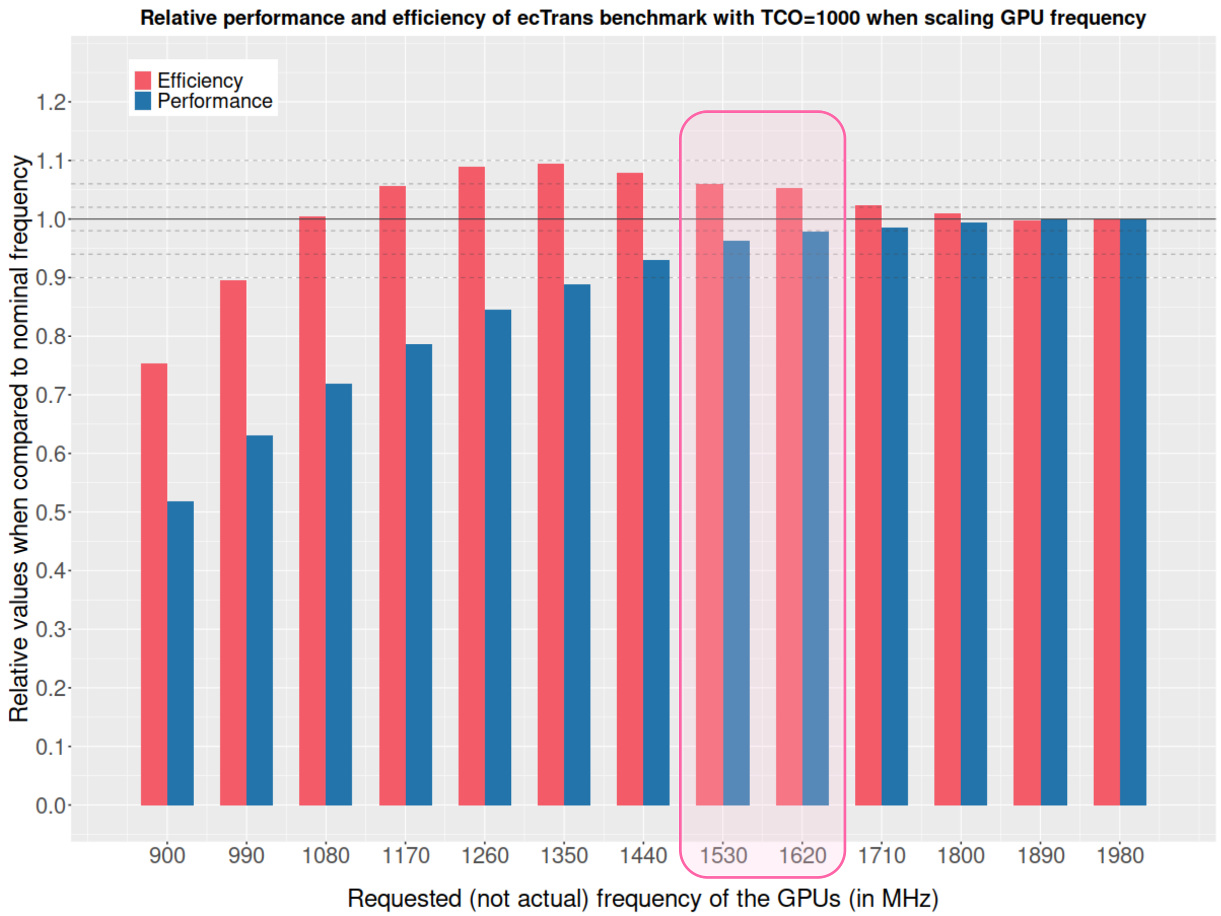
- Performance and full node power consumption as functions of the GPU frequency when executing ecTrans benchmark with TCO=1000
- Plateaux of performance and power consumption with highest frequencies
- Probably an efficiency sweet spot for a GPU frequency between 1500 MHz and 1700 MHz
- Performance degradation skyrockets when GPU frequency is scaled down below 1350 MHz
- Reminder: not 100% usage rate of GPUs, power consumption not relevant for projections for ecTrans in the context of the IFS

Then, what about the efficiency of ecTrans benchmark?



- Relative performance and efficiency of ecTrans benchmark for TCO=1000 while scaling GPU frequency, when compared to nominal frequency
- Monitored power consumption of ecTrans benchmark – lesser than the projected power consumption of production ecTrans (see previous discussion)
- Once again, plateaux of performance and efficiency for highest frequencies
- The most efficient GPU frequency for ecTrans benchmark is 1350 MHz, far from nominal frequency

Then, what about the efficiency of ecTrans benchmark?

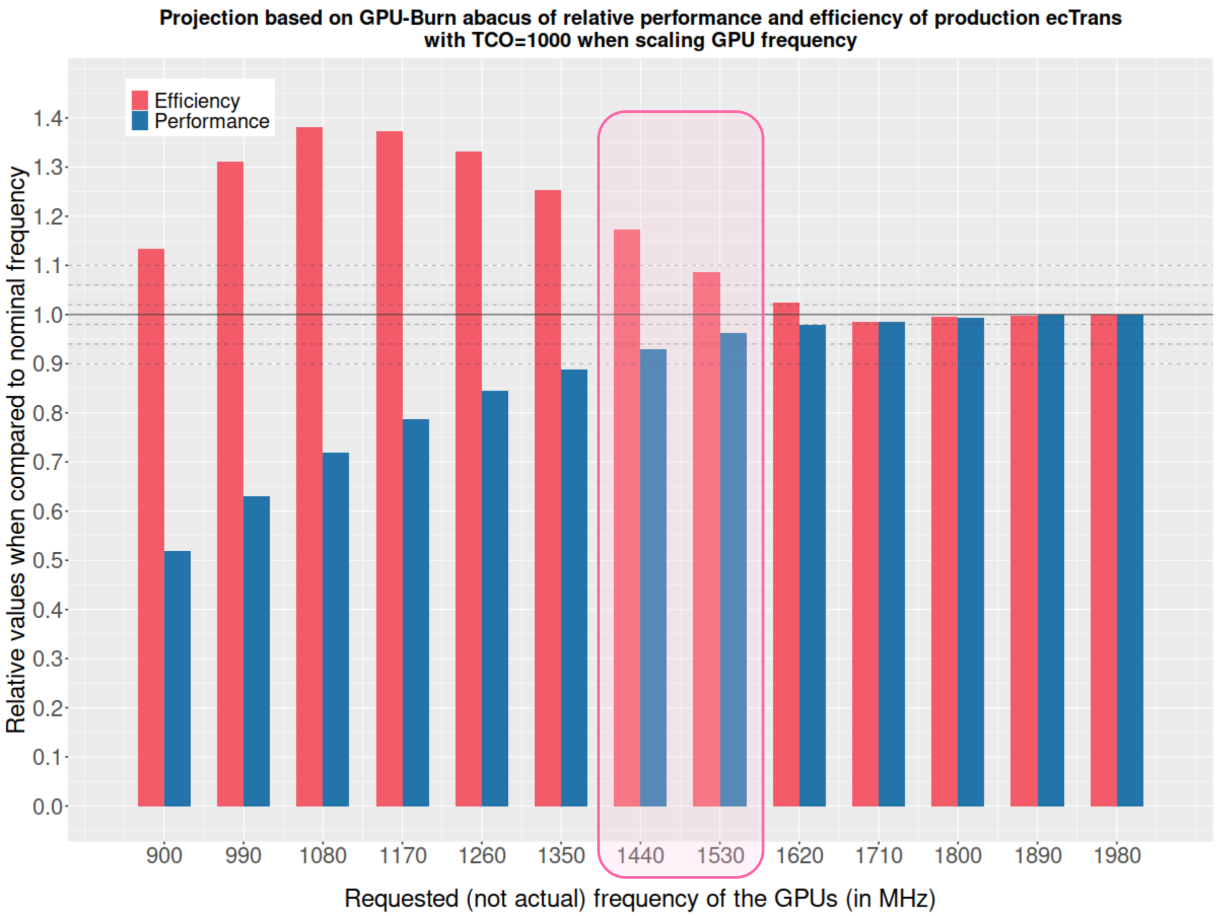


- Two GPU frequencies seem particularly promising since they induce efficiency gain with a controlled impact on the performance:
  - 1620 MHz: +5.29% efficiency for -2.19% performance
  - 1530 MHz: +6.03% efficiency for -3.74% performance
- A higher GPU usage rate would result in greater relative efficiency gain through frequency scaling – so those results can be considered as kind of a lower bound
- Let's get a look into that !

Projecting efficiency of production ecTrans with GPU-Burn

- Projection of relative performance and efficiency of production ecTrans for TCO=1000 while scaling GPU frequency, when compared to nominal frequency
- Using the GPU-Burn abacus built previously as a substitute for power consumption of ecTrans benchmark to build a projection of the efficiency of production ecTrans
- Once again, plateaux of performance and efficiency for highest frequencies
- The most efficient GPU frequency for ecTrans benchmark is 1080 MHz, even farer from nominal frequency

Projecting efficiency of production ecTrans with GPU-Burn



- Since frequency scaling has an impact on power consumption only when the core are active, the higher the GPU usage rate, the higher the potential efficiency gain
- That is why efficiency gain could reach 38%
- Efficiency gains moved toward lower frequencies when compared with previous experiment (to be explored)
- Two GPU frequencies seem particularly promising since they induce efficiency gain with a controlled impact on the performance:
  - 1530 MHz: +8.66% efficiency for -3.74% performance
  - 1440 MHz: +17.3% efficiency for -7.06% performance

EUPEX

Conclusions + Q&A

Conclusions - Efficiency comparison and GPU frequency scaling

TCO = 1000

x86 node	
Inverse-Direct transform lasts:	1518 ms
Mean power consumption is:	1014 W
Relative efficiency w.r.t. aarch64:	- 1.35 %
^Relative efficiency w.r.t. GPU:	- 12.9 %

aarch64 node	
Inverse-Direct transform lasts:	1442 ms
Mean power consumption is:	1053 W
Relative efficiency w.r.t. x86:	+ 1.37 %
^Relative efficiency w.r.t. GPU:	- 11.7 %

GPU node	
Inverse-Direct transform lasts:	364 ms
Mean power consumption is:	< 3684 W
*Relative efficiency w.r.t. x86:	+ 14.8 %
*Relative efficiency w.r.t. aarch64:	+ 13.2 %

- The nominal (and maximal) GPU frequency is not (always) the best from the efficiency POV
- For ecTrans with TCO=1000, setting the GPU frequency to 1620 MHz seems like a good idea: +5.29 % (increase) of efficiency and only -2.19 % (decrease) of performance
- If ecTrans scales to a 100% usage rate of the GPUs when executed in the context of the IFS, the efficiency gain projected thanks to the "GPU-burn abacus" could be substantial

[*] Lower/ [^] Upper bounds based on the power consumption of GPU-Burn