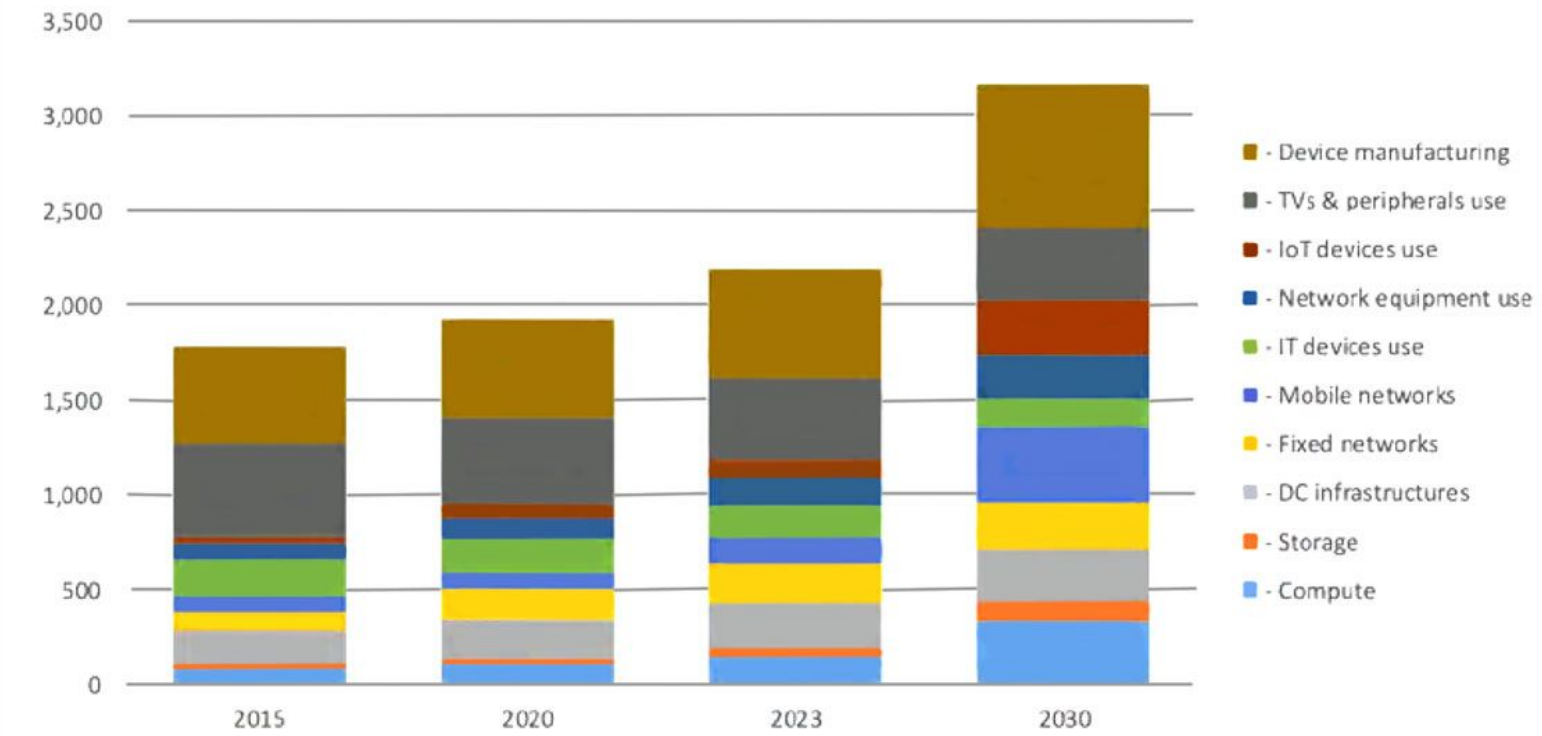
A Flexible Operational Framework for Energy Profiling of Programs

Authors: Roblex NANA TCHAKOUTE, Claude TADONKI, Petr DOKLADAL and Youssef MESRI

Mines Paris - PSL (CRI/CMM/CEMEF - France)

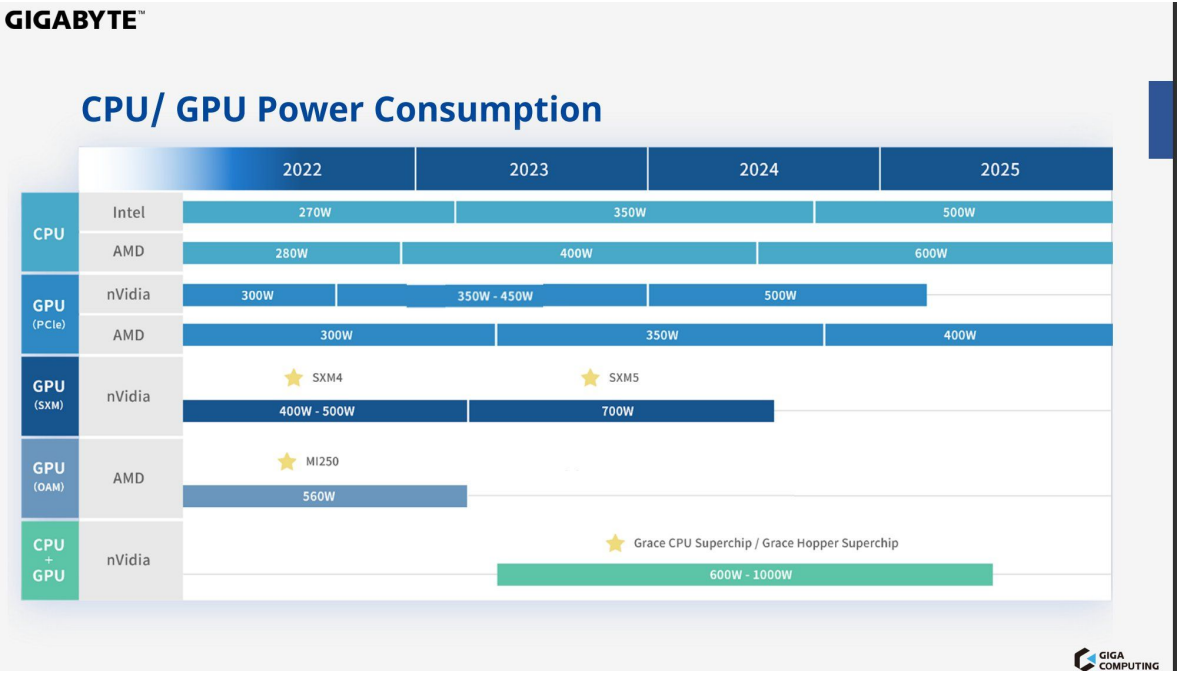
15th Workshop on Applications for Multi-Core Architectures, in conjunction with the
36th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2024)
Hawaii, USA, November 13-15, 2024

Evolution of IT energy demand (in TWh)

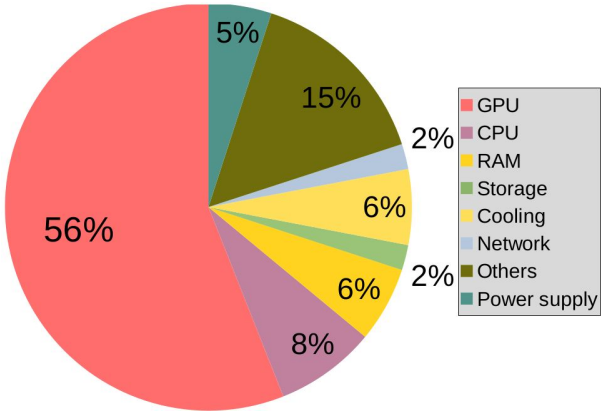


Schneider Electric estimates that IT sector electricity demand will grow by 50 percent by 2030, reaching 3,200TWh, equivalent to 5 percent Compound Annual Growth Rate (CAGR) over the next decade. | © Image: Schneider Electric

Power breakdown in modern computing systems



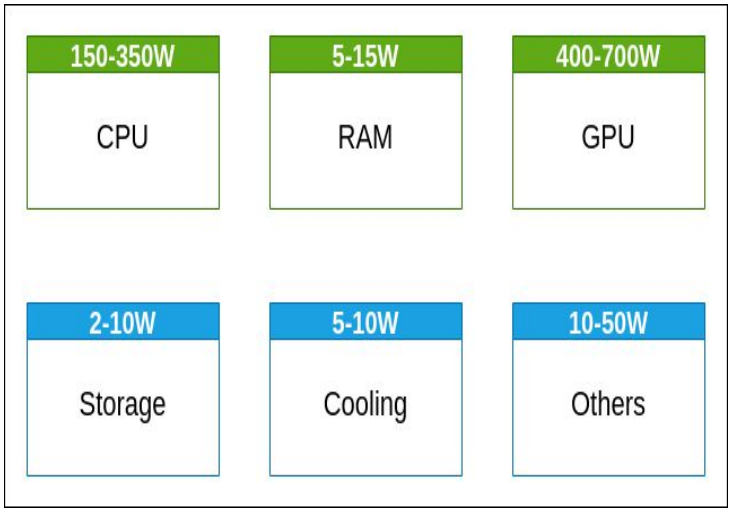
Next-Gen server CPUs & GPUs could consume 1000W+ power by 2025.



Nvidia DGX H100 server node, with 8 H100 GPUs, require 10.2 kW for power supply. Thus we can derive the breakdown.

Motivations for software profiling

- Energy can be measured through Power Meters or sensors
- The most accurate way, but less helpful for optimization
- We need fine grained measurements to understand the behavior devices and then optimize accordingly
- We can also optimize programs



Mains energy hungry part within a modern computer server

News devices provide integrated sensors for fine grained energy/power measurements

Key characteristics expected from of a profiling tool

- **Programmability**
  Should provides fine-grained control over energy profiling and allows developers to focus on specific parts of the codebase to optimize energy efficiency and performance (instrumentation and APIs)
- **Flexibility**
  To measure specific parts of the computer, allowing configurations, auto target hardware detection, and porting to other architectures.
- **Standalone**
  Easy to install, few dependences on others library and tools, minimum privileged rights for access
- **Portability**
  Compatibility across device generations, even within the same manufacturer (facilitate maintenance)
- **Accuracy**
  The tool does indeed measure the desired behavior and should be consistent across workloads

SOTA Energy measurement tools

Support	Perun	CodeC	EIT	CTracker	Eco2AI	TraC	PyJoules	Perf	LIKWID	PAPI	IntelPG	Powertop	Score-P	Variorum	CrayPat	DDT-MAP	EA2P
GPU support																	
Nvidia GPU	✓	✓	✓	✓	✓	✓	✓							✓	✓	✓	✓
AMD GPU	✓													✓		✓	✓
Intel GPU														✓			
CPU and RAM supports																	
Intel CPU	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
AMD CPU					✓			✓	✓			✓	✓	✓			✓
RAM	✓	✓	✓	✓	✓	✓			✓					✓			✓
OS support																	
Linux	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
Windows		✓									✓						✓
Mac OS		✓	✓			✓					✓						
Other important characteristics																	
Documentation	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
Configurable	✓	✓	✓									✓					✓
Code API	✓	✓	✓	✓	✓	✓	✓		✓	✓							✓
Open Source	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓			✓
Perf oriented								✓	✓	✓			✓		✓	✓	
Energy oriented	✓	✓	✓	✓	✓	✓	✓				✓	✓		✓			✓
Multi-Nodes	✓								✓				✓	✓	✓	✓	✓
Device details																	✓

Reality with existing tools (why a new tool ?)

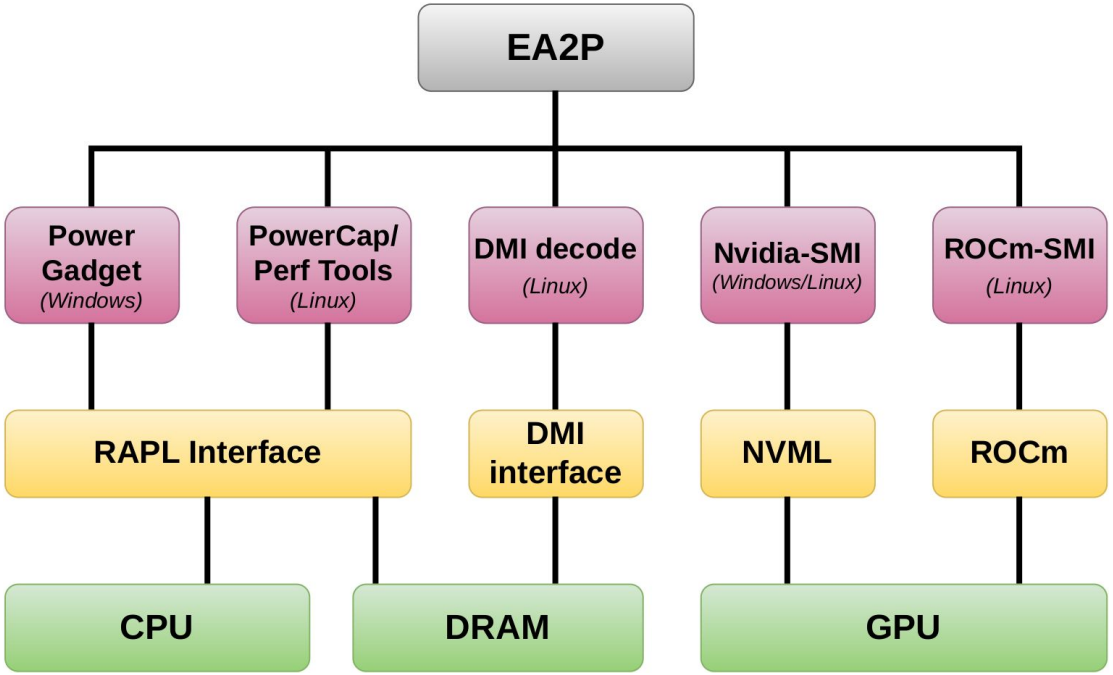
- Difficult to get, installed (*need for hand configuration*) and run
- Not reliable measurements (provide estimates - inconsistency)
- Lack of flexibility (device dependent - OS dependent)
- Lack of documentation (comprehension of the approach and outputs)

Thus our motivation to design a new energy measurement tool

MINES PARIS | PSL

Design overview : Energy Aware Application Profiler (EA2P)

Open Source available at :
https://github.com/HPC-CRI/EA2P
Documentation at :
https://hpc-cri.github.io/EA2P/



- written in Python
- we retrieve the values of the (power dedicated) registers through medium-level tools
- can be used in a standalone *(external call)* form or through an API for programmability (internal call)
- automatically detects needed subtools for its execution (e.g. perf, PowerCap, …)

Analytical Models

$$Power\_RAM = Power\_base \times nb\_slots \times mem\_use \quad (1)$$

$$Energy\_RAM = (\sum_{i=1}^{N} Power\_RAM) \times interval, \quad (2)$$

- Power_base is the TDP like value (or nominal power) associated to each memory type and capacity.
- nb_slots is the total number of memory slots within the entire motherboard
- mem_use is the percentage (i.e. footprint)

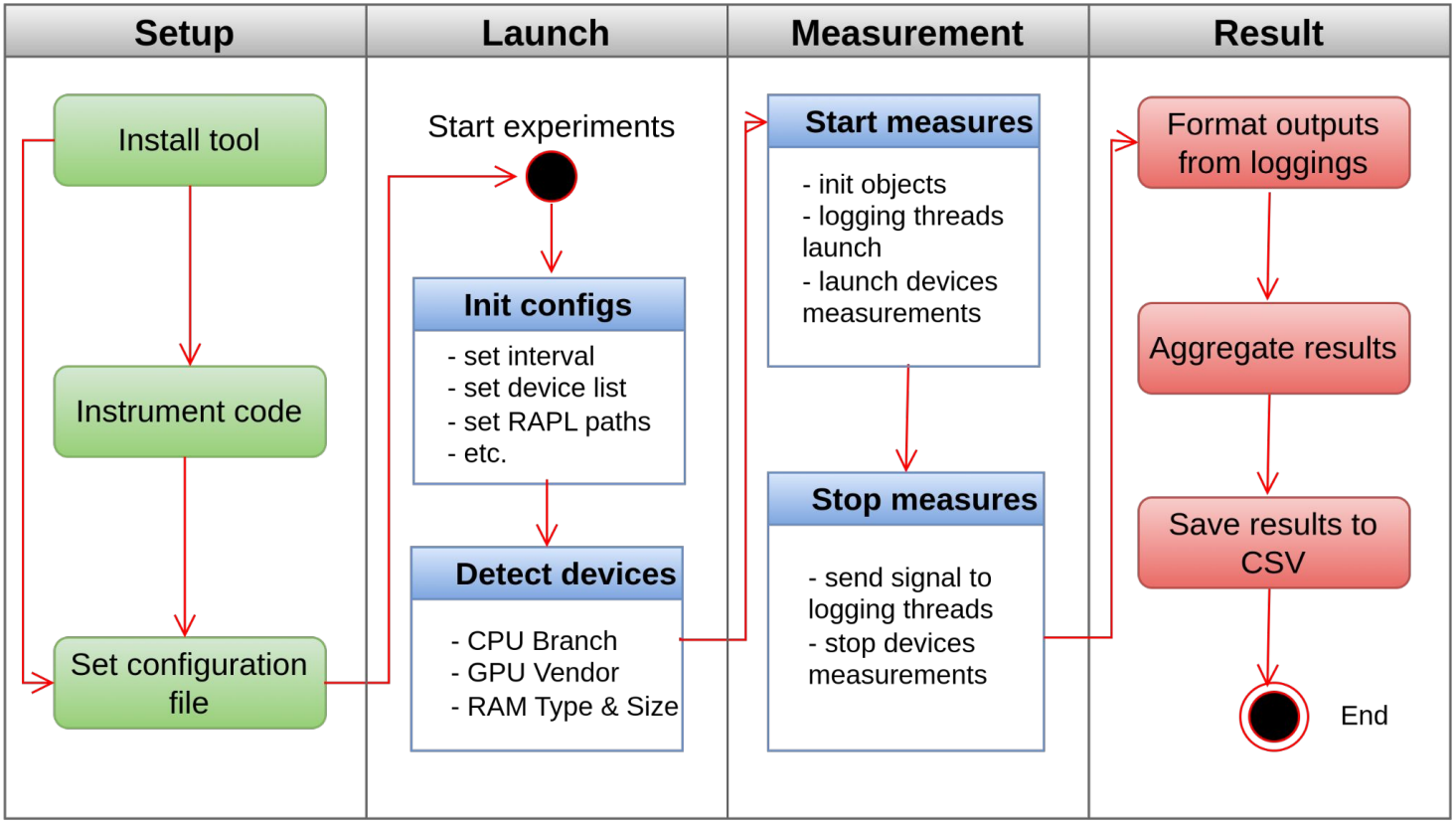
$$Energy_{GPU} = \sum_{i=1}^{N} P_i \times \Delta T_i, \quad (3)$$

where Pi (resp. ΔTi) is the power value of the GPU (resp. the ith time interval during which we assume a constant power Pi) at measurement step i

$$E_{CPU_{dom}} = 0$$
$$e_{current} = 0$$
$$\textbf{if } (E_i > e_{current})$$
$$E_{CPU_{dom}} = E_{CPU_{dom}} + (E_i - e_{current})$$
$$e_{current} = E_i$$

Update of the overall CPU energy for Intel RAPL. For AMD CPU, we use perf as wrapper.

Functional overview of EA2P



Experimental evaluation : Goals

- **Tool Accuracy Assessment:** Validate the accuracy and precision of the energy profiling tool in measuring power consumption across different hardware components, including CPU, RAM, and GPU.
- **Energy Profiling Consistency:** Ensure the consistency of energy profiling results across multiple hardware platforms (AMD, Intel, and Nvidia).
- **Workload Characterization:** Profile various computational workloads, including CPU-intensive, GPU-intensive, and heterogeneous computing tasks, to evaluate the tool's ability to capture energy usage patterns accurately.
- **Cross-Platform Compatibility:** Assess the tool's compatibility with different hardware components (AMD and Intel CPUs, AMD and Nvidia GPUs) to ensure its versatility.

The testbed used

Platforms :

name	Intel-1	AMD-1	AMD-2	Intel-2
CPU name	i9 12950HX	EPYC 7452	EPYC 7513	E5-2698v4
GPU name	RTX 3080Ti	A100 SXM4	A100 SXM4	Tesla V100
CPU TDP	55W	155W (x2)	200W(x1)	135W (x2)
GPU TDP	150W	400W (x2)	400W (x4)	300W (x8)
CPU threads	24	64 (x2)	64 (x1)	40 (x2)
GPU VRAM	16GB	40GB (x2)	40GB (x4)	32GB (x8)
CPU RAM	32 GB	128 GB	512 GB	512 GB
Multi-SoC	No	Yes	No	Yes

Applications:

- System sleep test
- VGG16 with cifar10 TensorFlow dataset
- VGG16 with Stanford dogs TensorFlow dataset
- Parallel OpenMP multiplication with matrix size 8000x8000, varying number of threads
- Parallel OpenMP Streaming vector TRIAD, varying data size (for RAM model validation)

Sample instrumentation

- VGG16 fine tuning (just train the last layer)
- Example of annotation for power measurement
- Main call for training

```python
def build_model(num_classes):
    inputs = tf.keras.layers.Input(shape=(IMG_SIZE, IMG_SIZE, 3))
    model = VGG16(include_top=False, input_tensor=inputs, weights="imagenet")

    # Freeze the pretrained weights
    model.trainable = False

    # Rebuild top
    x = tf.keras.layers.GlobalAveragePooling2D(name="avg_pool")(model.output)
    x = tf.keras.layers.BatchNormalization()(x)

    top_dropout_rate = 0.2
    x = tf.keras.layers.Dropout(top_dropout_rate, name="top_dropout")(x)
    outputs = tf.keras.layers.Dense(num_classes, activation="softmax", name="pred")(x)

    # Compile
    model = tf.keras.Model(inputs, outputs, name="VGG16")
    optimizer = tf.keras.optimizers.Adam(learning_rate=1e-2)
    model.compile(
        optimizer=optimizer, loss="categorical_crossentropy", metrics=["accuracy"]
    )
    return model
```

```python
model = build_model(num_classes=NUM_CLASSES)

@power_meter.measure_power(
    package="tensorflow",
    algorithm="VGG16",
    data_type="images",
    data_shape="(32,32,60000)",
    algorithm_params="batch_size=64,epochs=10,optimizer=Adam,loss='categorical_crossentropy'"
)
def train_model():
    model.fit(ds_train, epochs=epochs, batch_size=batch_size, validation_data=ds_test)

if __name__ == '__main__':
    train_model()
```

Experimental validations : measurements overhead EA2P vs perf

Application	tool	Pkg (std)	RAM (std)	time(s) (std)
sleep	perf	2.254(0.004)	1.290(0.002)	183.508(0.02)
	EA2P	2.181(0.02)	1.270(0.001)	180.272(0.002)
	gap (perf-EA2P)	**0.073**	**0.02**	**3.236**
CIFAR-CPU	perf	28.592(0.32)	4.899(0.10)	445.236(5.36)
	EA2P	28.252(0.40)	4.825(0.12)	438.33(6.81)
	gap (perf-EA2P)	**0.34**	**0.074**	**6.906**

Overhead validation on intel client "Laptop" (Intel Core i9 12950 HX)

- *Each of the experiments was repeated 5 times to assess the consistency through the standard deviation which turned to be tiny.*
- *For simplicity, we only report the average values of the 5 runs and we report standard deviation to illustrate the variability which is due to the operating system and the machine state.*

Experimental validations : multi RAPL power domains EA2P vs perf

		Energy(Wh)					time(s)
Application	**Tool**	**cores**	**uncore**	**pkg**	**psys**	**RAM**	
Sleep	perf	0.008	0.000	0.149	0.520	/	180.029
	EA2P	0.008	0.000	0.149	0.520	0.031	180.192
CIFAR-GPU	perf	0.089	0.001	0.274	2.78	/	72.626
	EA2P	0.056	0.001	0.229	2.672	0.014	66.903
CIFAR-CPU	perf	3.715	0.007	5.949	12.001	/	1476.905
	EA2P	3.696	0.007	5.952	13.488	0.295	1478.121

CPU and DRAM validation on intel client "Laptop" (Intel Core i9 12950 HX)

The energy of the whole system when no program is running can be non negligible. So it should be taken into account in measurement as we can see with sleep test.

Experimental validations : EA2P VS CodeCarbon and multiGPU report

Application	tool	CPU (Wh)	GPU (Wh)	time(sec)
sleep	CodeCarbon	0.30538	0.98752	181.931
	EA2P	0.20417	0.82411	180.706
VGG16 CIFAR-GPU	CodeCarbon	0.22944	2.07726	67.993
	EA2P	0.23011	2.04792	67.757

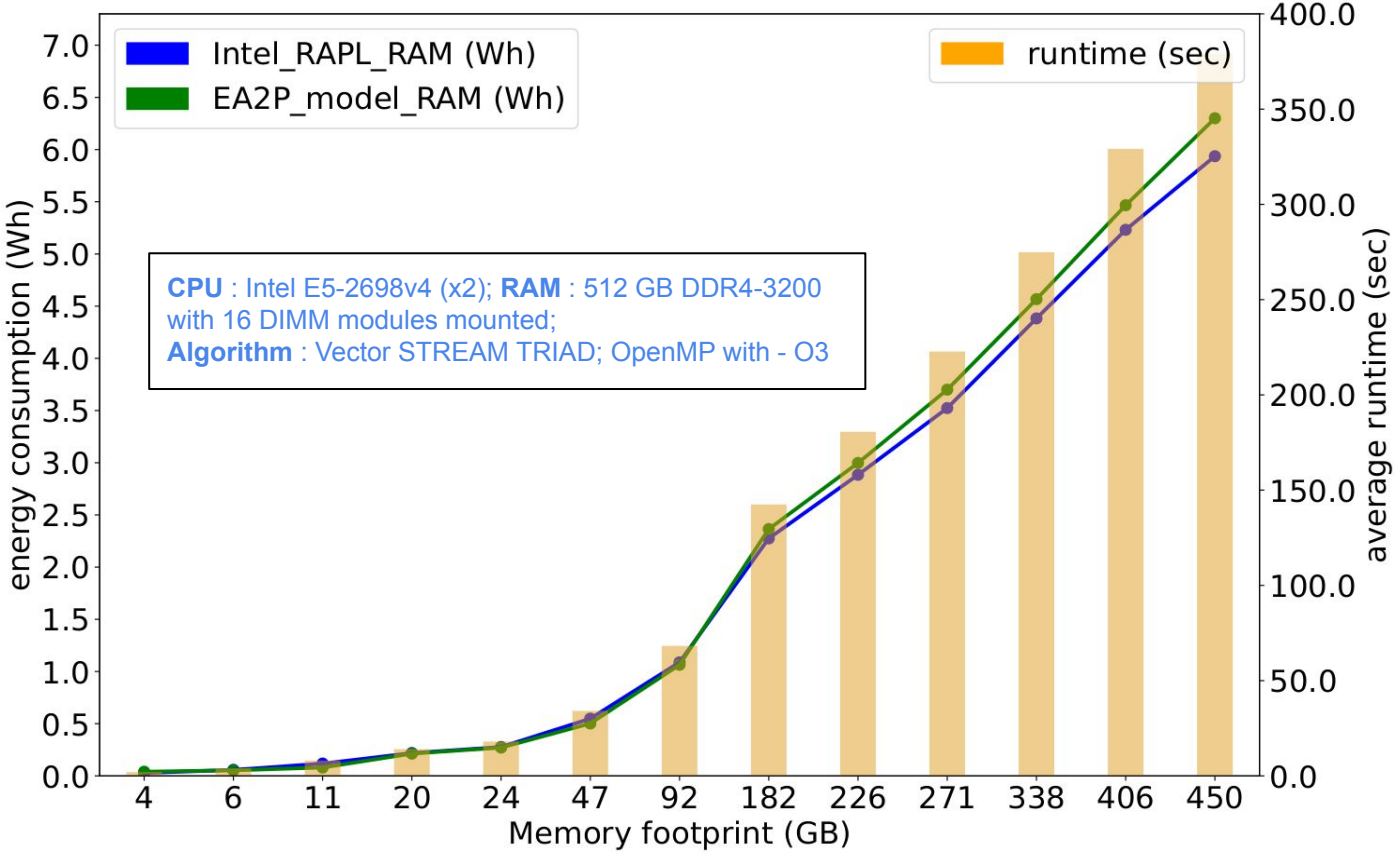
GPU validation on Nvidia ("Laptop"). ***CPU is the energy of package domain***

Appli	pkgs(Wh)	ram (Wh)	GPU0 (Wh)	GPU1 (Wh)	GPU2 (Wh)	GPU3 (Wh)	GPU4 (Wh)	GPU5 (Wh)	GPU6 (Wh)	GPU7 (Wh)	time (sec)
Sleep	2.19481	1.33308	2.17964	2.10399	2.12799	2.10432	2.10385	2.12957	2.10584	2.14317	181.038
VGG16 DOG-CPU	**28.52879**	5.4077	5.63378	5.41911	5.50514	5.41387	5.39961	5.49029	5.41896	5.52412	495.096
VGG16 DOG-GPU	1.21921	0.38869	**2.51989**	0.81177	0.81666	0.80432	0.81027	0.81626	0.80222	0.81376	52.459

Multi GPU systems energy report "gemini" EA2P

Fine tuning VGG16 with Stanford dog dataset consume a total of more than 77 Wh for more than 9 minutes running on 80 threads Intel Xeon server with 8 Nvidia V100 GPU mounted.
The same program using GPU computing consume around 10 Wh for less than a minute of execution on the same machine. So 10x faster and 8x energy efficient

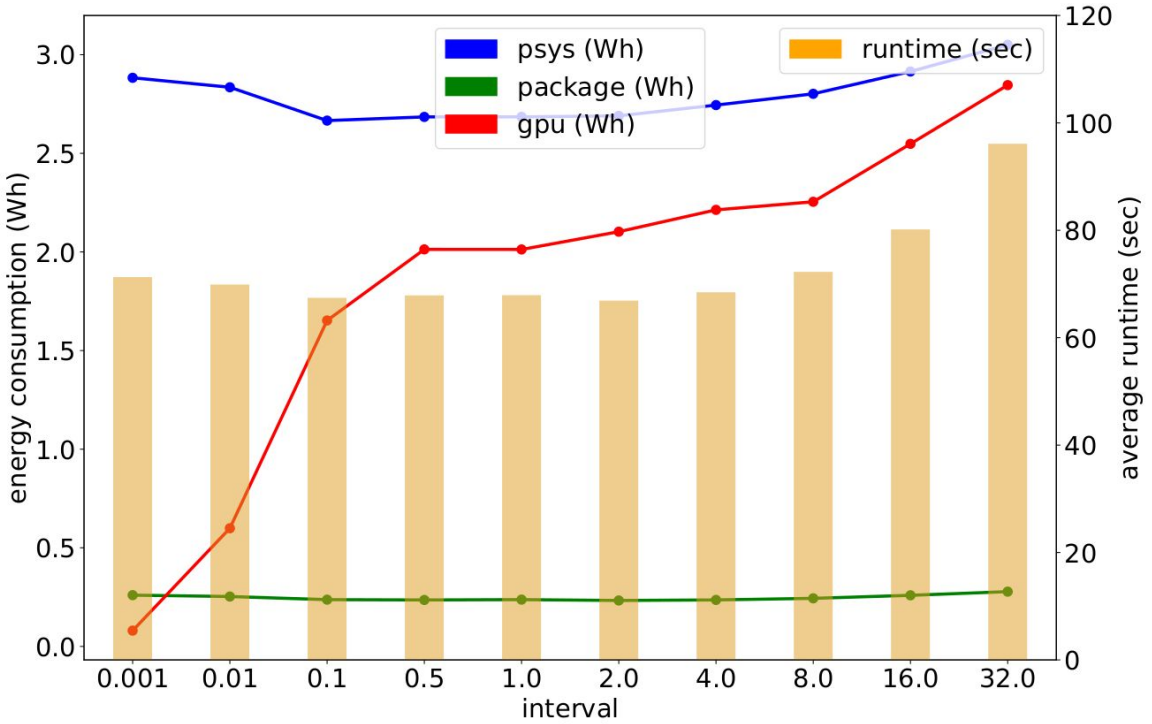
Experimental validations : RAM Model validation through intel RAPL



Experimental validations : Sampling frequency influence

Application : VGG16 training on CIFAR10 with TensorFlow with batch size 64 and 10 epochs
CPU : Intel Core i9 12950HX (24 Threads)
RAM : 32 GB DDR5-4800
GPU : RTX 3080Ti, 16GB, GDDR6

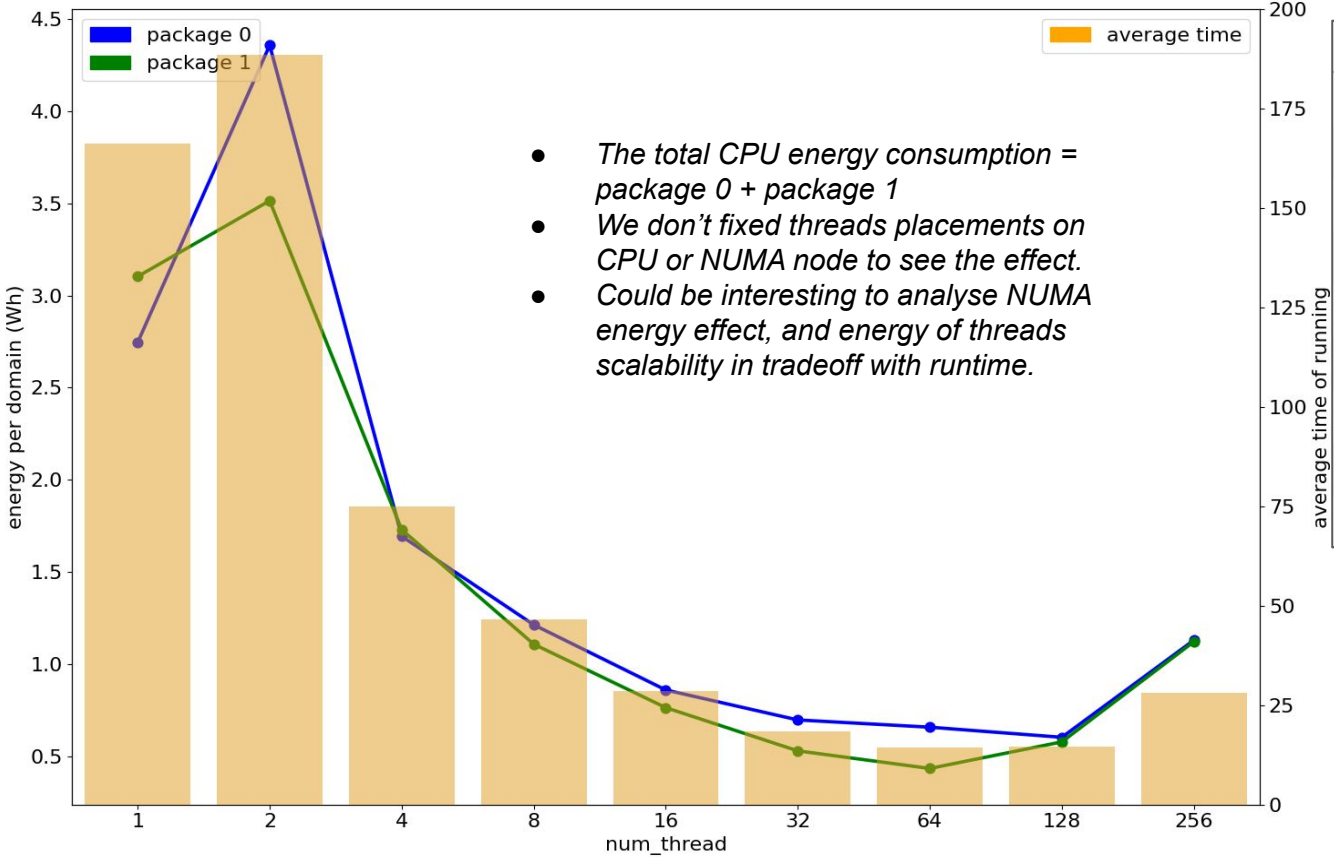
psys	package	gpu	time	interval
2.88300	0.25986	0.08007	71.28928	0.001
2.83491	0.25284	0.59910	69.90146	0.010
2.66597	0.23706	1.65259	67.43378	0.100
2.68465	0.23551	2.01294	67.88068	0.500
2.68499	0.23720	2.01260	67.91608	1.000
2.69010	0.23293	2.10269	66.90635	2.000
2.74465	0.23574	2.21340	68.46401	4.000
2.80177	0.24374	2.25440	72.26552	8.000
2.91496	0.25907	2.54813	80.16185	16.000
3.05029	0.27769	2.84596	96.12139	32.000



- Sampling frequency is the time between two query of energy values
- CPU (psys and package) energy and time are more correlated with sampling interval
- Normally, psys >= package+gpu since it's the entire board value
- GPU depend on Nvidia-smi which report the power and not the energy. So we notice consistency problem with low sampling intervals.
- Threads join from logging process is the problem of time overhead for big intervals

Multi-threading example

CPU : AMD EPYC 7452 (x2); **Threads** : 64 (x2), **CPU TDP** : 155W (x2) **RAM** : 128 GB;
Algorithm : Matrix Multiplication; **Matrix size** : 8000x8000; OpenMP with - O3



- *The total CPU energy consumption = package 0 + package 1*
- *We don't fixed threads placements on CPU or NUMA node to see the effect.*
- *Could be interesting to analyse NUMA energy effect, and energy of threads scalability in tradeoff with runtime.*

package 0	package 1	time	num_thread
1.12894	1.12073	28.06818	256
0.60182	0.57692	14.53785	128
0.65672	0.43274	14.38633	64
0.69608	0.52816	18.34158	32
0.85876	0.76281	28.47718	16
1.21250	1.10733	46.66632	8
1.69303	1.72893	75.04626	4
4.35776	3.51411	188.61890	2
2.74486	3.10269	166.27935	1

Multi-nodes measurement *(Numpy Matmul)*

MEASUREMENTS ON AMD-2 WITH 10240 × 10240 MATRICES, using send-Receive communication (2 first nodes compute)

Node	Pkg(J)	GPU 0 (J)	GPU 1 (J)	GPU 2 (J)	GPU 3 (J)	RAM (J)	Time (s)
0	536.67	255.75	261.8	258.92	271.46	75.6	4.34
1	555.88	380.09	344.25	331.31	357.12	100.8	5.85
2	94.84	113.97	123.36	128.24	134.06	33.6	1.24
3	94.09	112.98	121.65	135.57	147.30	33.6	1.24
Total	1281.48	862.78	851.09	854.02	909.93	243.6	12.67

MEASUREMENTS ON AMD-2 WITH $2^{14} \times 2^{14}$ MATRICES, using broadcast (all nodes compute)

Node	Pkg(J)	GPU 0 (J)	GPU 1 (J)	GPU 2 (J)	GPU 3 (J)	RAM (J)	Time (s)
0	1493.59	648.52	680.59	649.10	666.36	201.6	13.25
1	1419.99	714.36	633.82	625.94	670.95	193.2	12.54
2	1457.43	664.03	666.67	652.40	719.44	201.6	13.03
3	1472.65	667.21	653.88	659.93	689.50	201.0	13.11
Total	5843.66	2694.11	2634.95	2587.35	2746.25	596.4	51.93

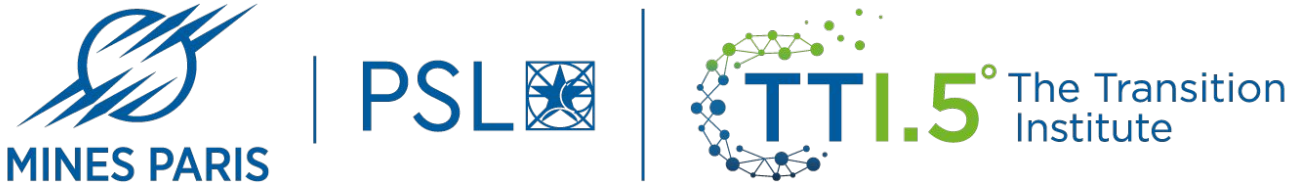
MEASUREMENTS ON AMD-2 WITH $2^{15} \times 2^{15}$ MATRICES, using broadcast (all nodes compute)

Node	Pkg(J)	GPU 0 (J)	GPU 1 (J)	GPU 2 (J)	GPU 3 (J)	RAM (J)	Time (s)
0	9109.88	3374.76	3530.46	3319.83	3425.45	1058.4	74.66
1	8777.08	3674.04	3255.40	3208.25	3430.79	999.6	70.65
2	9012.43	3353.60	3363.65	3291.97	3638.53	1041.6	73.26
3	9090.88	3408.30	3347.95	3435.98	3442.57	1058.4	74.43
Total	35990.27	13810.70	13497.44	13256.03	13937.32	4158.0	293.00

Conclusion and future works

- Conclusion:
  - EA2P provide small overhead compared to Linux perf and codeCarbon tools
  - provide fine grained results per device & power domains (Intel)
  - Measurement for RAM, AMD GPU & CPU, Nvidia GPU, and Intel CPU
  - Code Instrumentation API and CLI usages
  - Provide Sampling frequency option to users.
  - Automatic detection of device vendors and commands to use
  - Possibility to select specific devices measurement (Only subset of the system)
- Future works
  - Implement robust error handling mechanisms and logging functionalities to help handling troubleshooting

Thank you for your Attention !



Email : {roblex.nana_tchakoute, claude.tadonki, petr.dokladal, youssef.mesri}@minesparis.psl.eu

This research was supported by The Transition Institute 1.5 driven by École des Mines de Paris - PSL

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations.