Software-based energy measurement as first step for power-aware application optimization

(Matinée numérique éco responsable à Paris 13 le 28 Juin 2024)

Roblex NANA TCHAKOUTE
PhD Student
Centre de recherche en informatique (CRI)
Mines Paris - PSL University
PhD supervised by: Claude TADONKI (CRI), Petr DOKLADAL (CMM) and Youssef MESRI (CEMEF)

Date : 28/06/2024

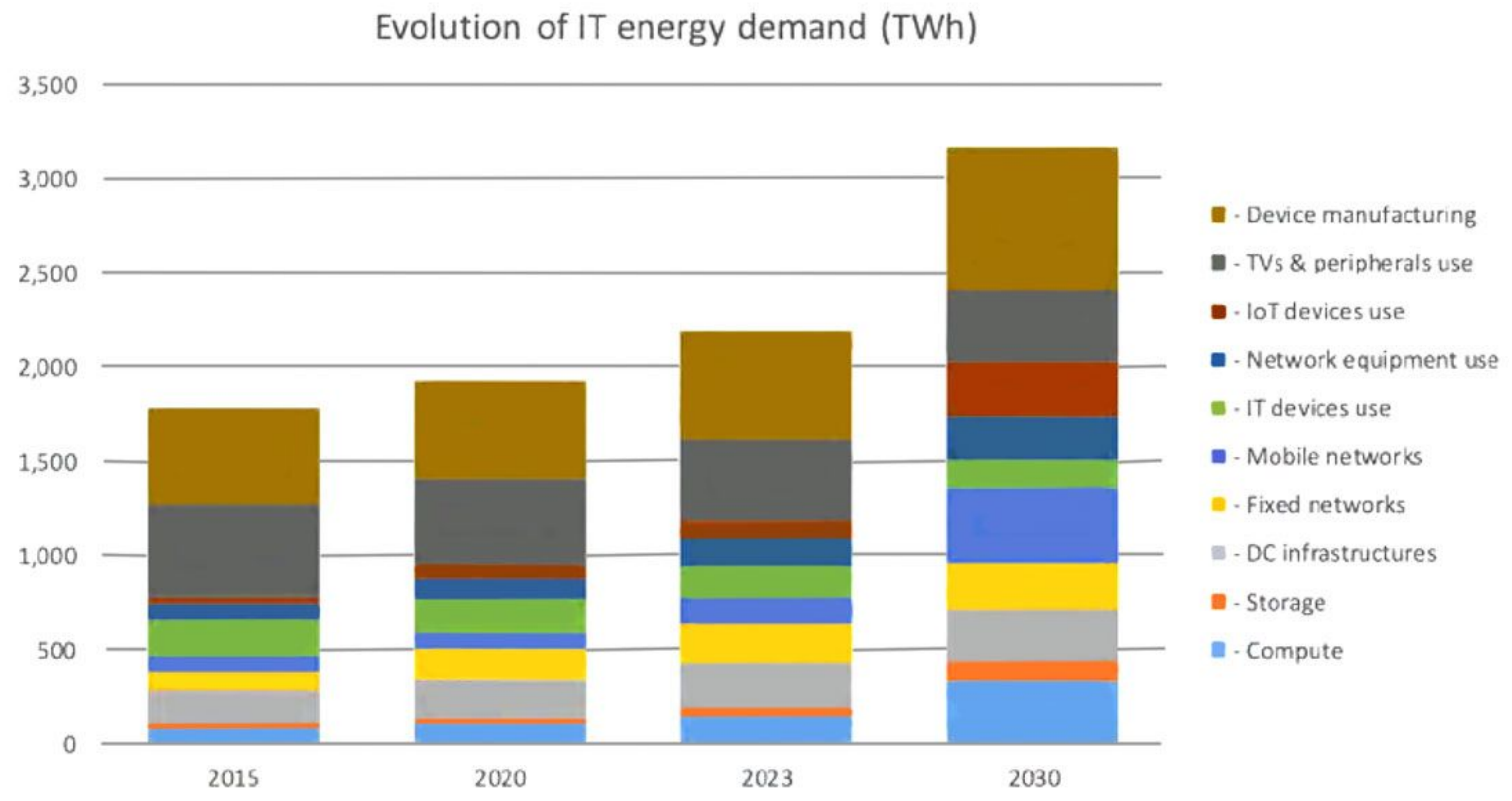
MINES PARIS | PSL | TTI.5° The Transition Institute

Contents

Introduction

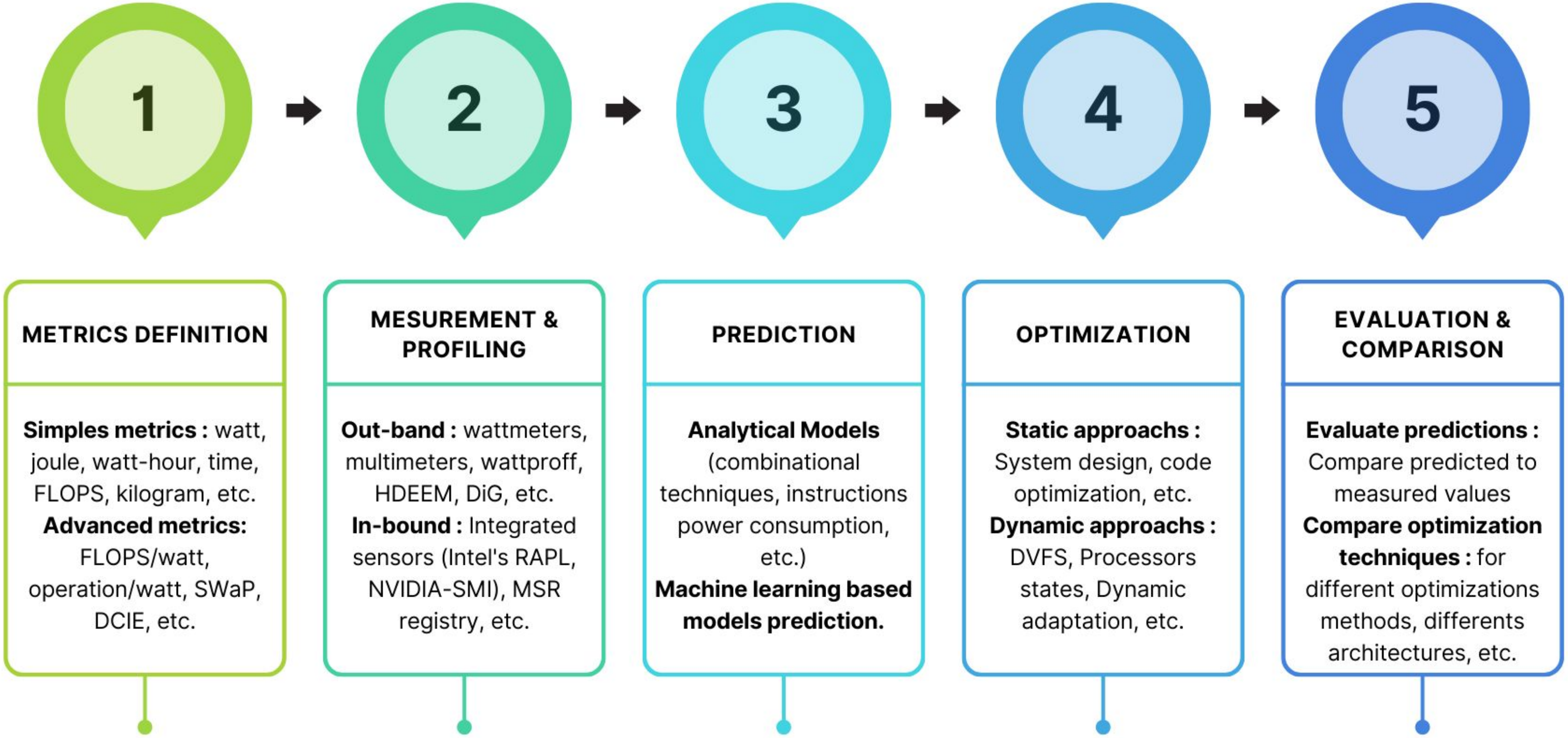
Context

- **Applications performance:**
  - Computational demand is rising every year, necessitating continuous growth in computer performance to meet these demands.
  - The performance of computers is continuously increasing to serve the growing demands of applications.
- **Energy efficiency challenge:**
  - Energy is becoming critical, making energy efficiency a challenging aspect for applications.
  - Balancing the increasing performance demands with the need for energy efficiency poses a significant challenge.
- **Understanding performance:**
  - We'll explore the dichotomy between architecture performance and application performance.
  - What exactly do we mean by applications performance? And how does it relate to architecture performance? Moreover, how does energy efficiency play a role in this dynamic?



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

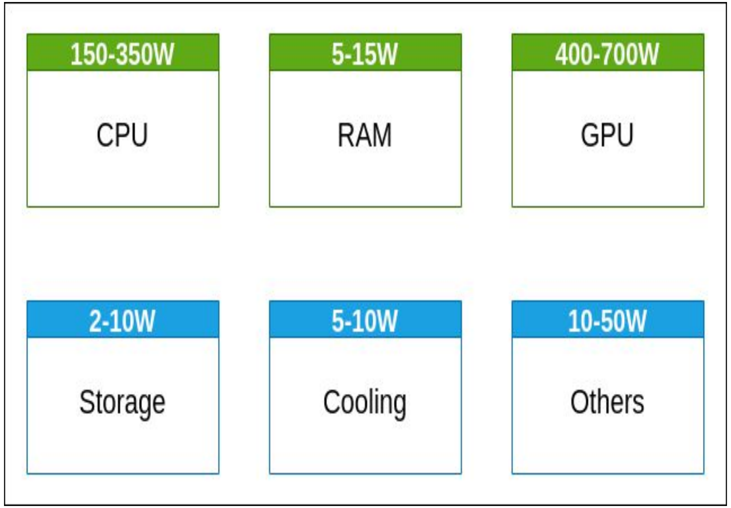
Taxonomy of energy related aspects of computing



Part I : Software based approach of energy measurement

Motivations for software profiling (In-bound)

- 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

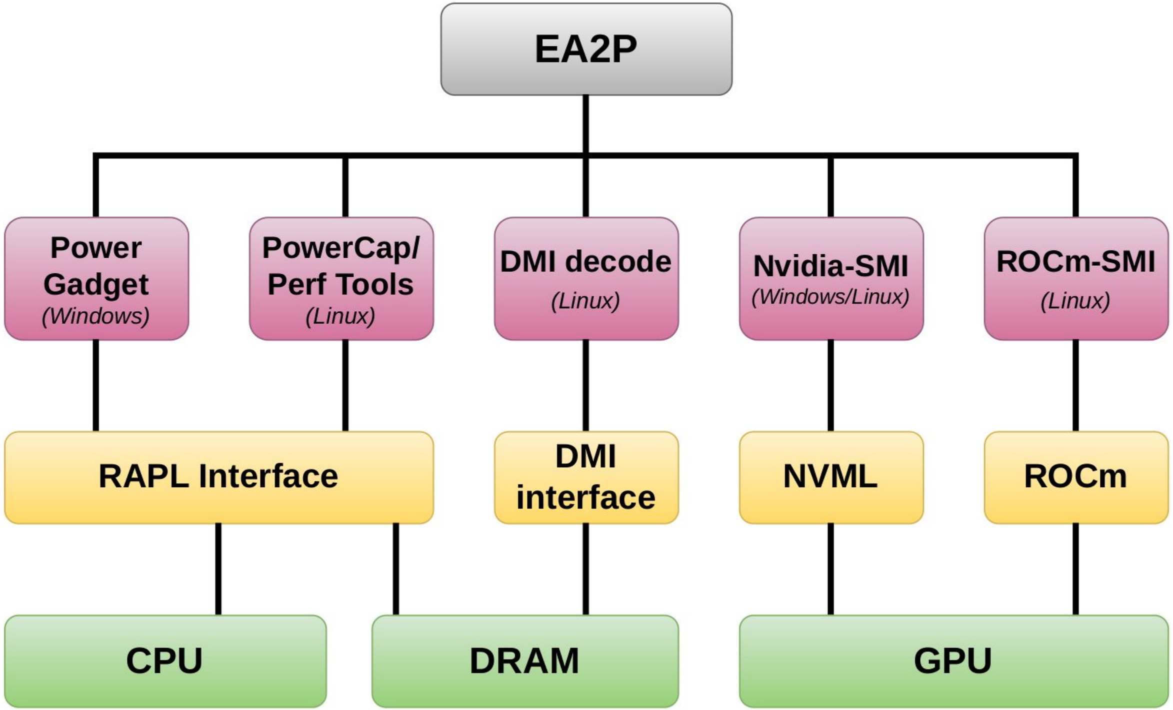
News devices provide integrated sensors for fine grained energy/power measurements

SOTA Energy Profiling tools

Support	CC	EIT	CT	EA	TB	PJ	Perf	LI	PAPI	PG	PT	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	✓	✓	✓	✓	✓	✓		✓	✓			✓

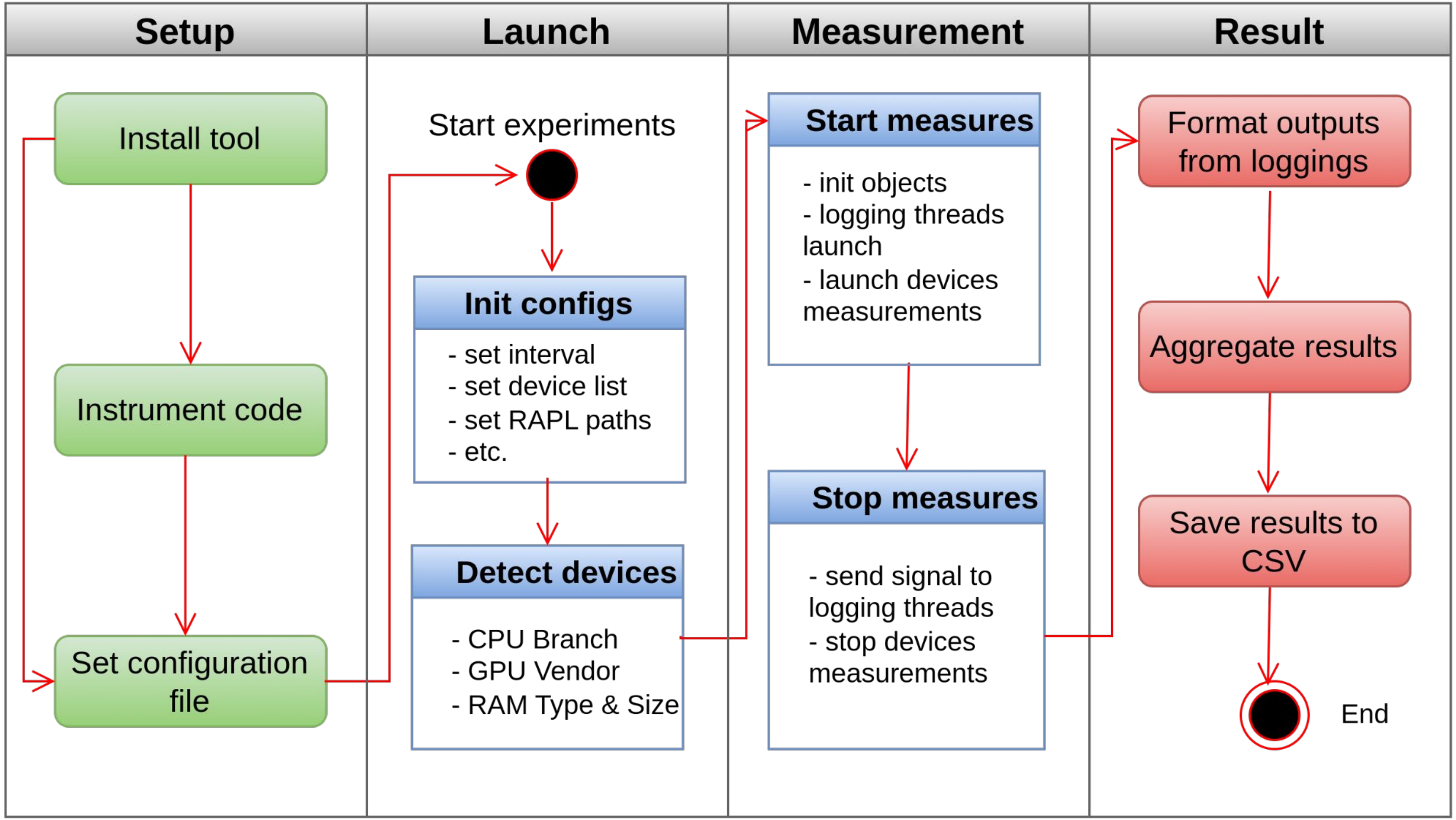
Design overview : Energy Aware Application Profiler (EA2P)

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



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

Functional overview of EA2P



Experimental validations

		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"

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

Application	tool	CPU (Wh)	GPU (Wh)	time(sec)
sleep	CodeCarbon	0.305	0.987	181.931
	EA2P	0.204	0.824	180.706
VGG16 CIFAR-GPU	CodeCarbon	0.229	2.077	67.993
	EA2P	0.230	2.047	67.757

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

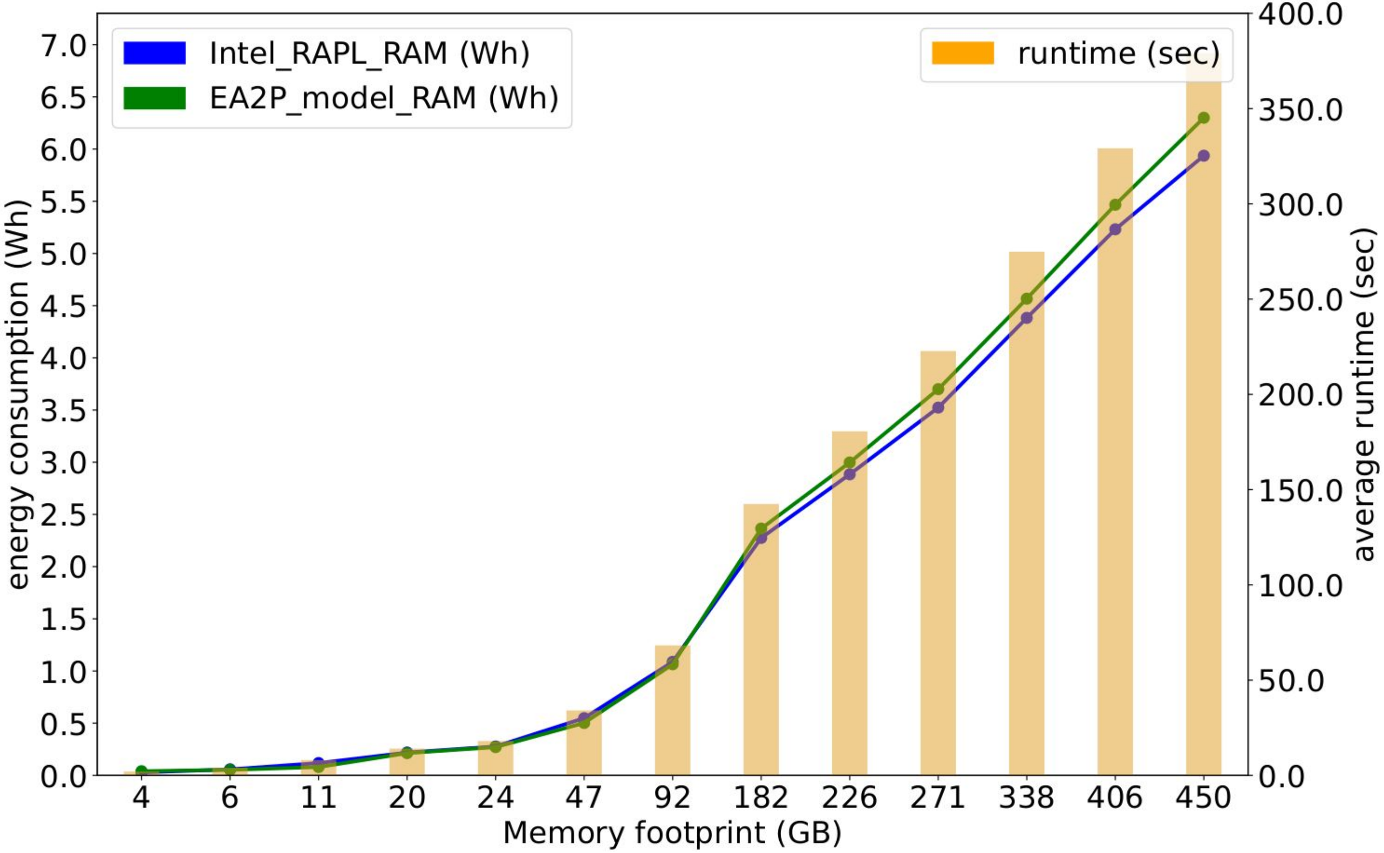
Application	pkgs(Wh)	ram (Wh)	GPU0 (Wh)	GPU1 (Wh)	GPU2 (Wh)	GPU3 (Wh)	GPU4 (Wh)	GPU5 (Wh)	GPU6 (Wh)	GPU7 (Wh)	time (sec)
Sleep	2.194	1.333	2.179	2.103	2.127	2.104	2.103	2.129	2.105	2.143	181.038
VGG16 DOG-CPU	**28.528**	5.40	5.633	5.419	5.505	5.413	5.399	5.490	5.418	5.524	495.096
VGG16 DOG-GPU	1.219	0.388	**2.519**	0.811	0.816	0.804	0.810	0.816	0.802	0.813	52.459

Multi GPU systems energy report 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.

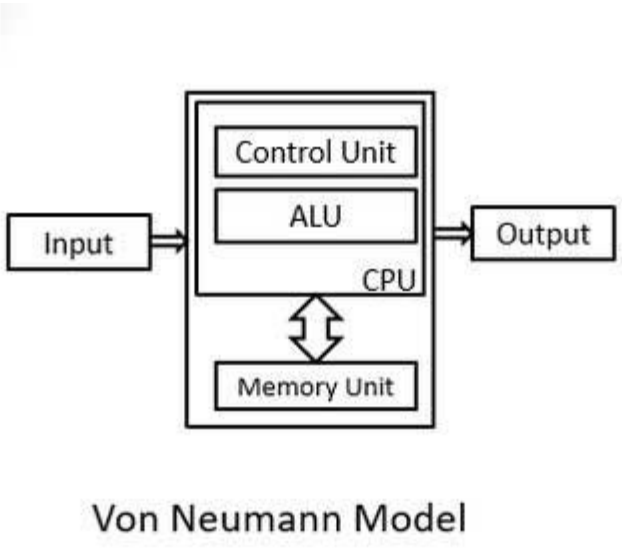
The same program using GPU computing consume around 10 Wh for less than a minute of execution on the same machine. So 10x faster and 8x energy efficient

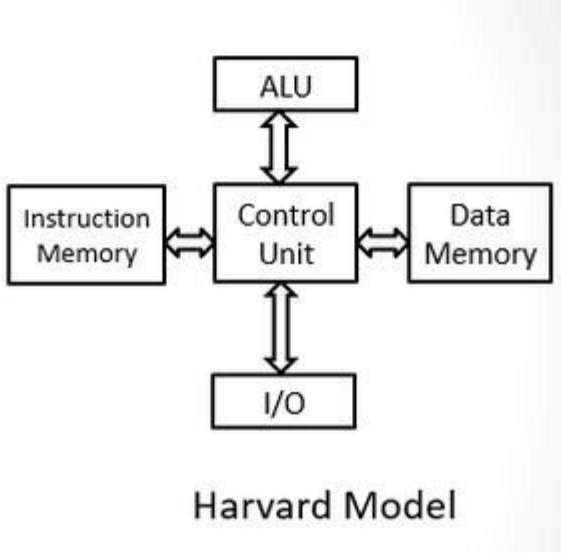
Experimental validations : (*RAM Model validation through intel RAPL*)



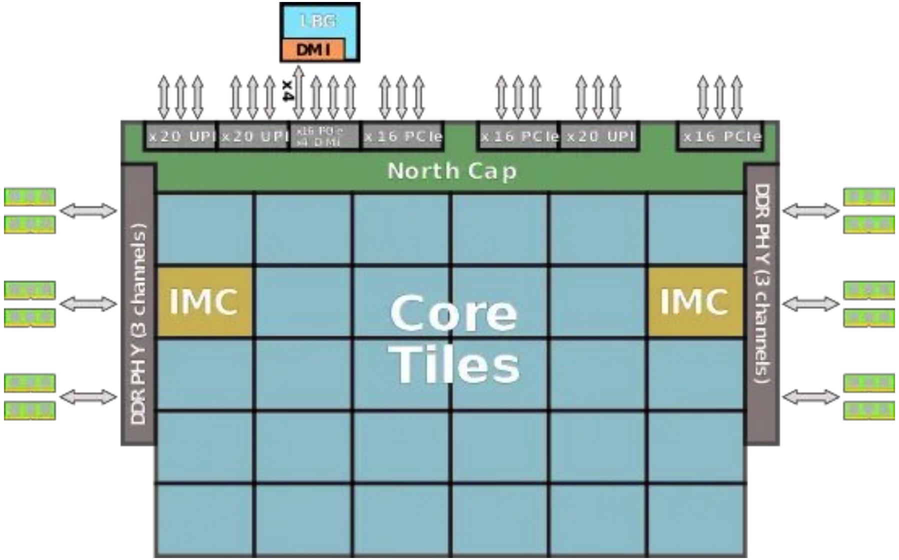
Part II : Performance vs energy trade-off on modern x86 CPU

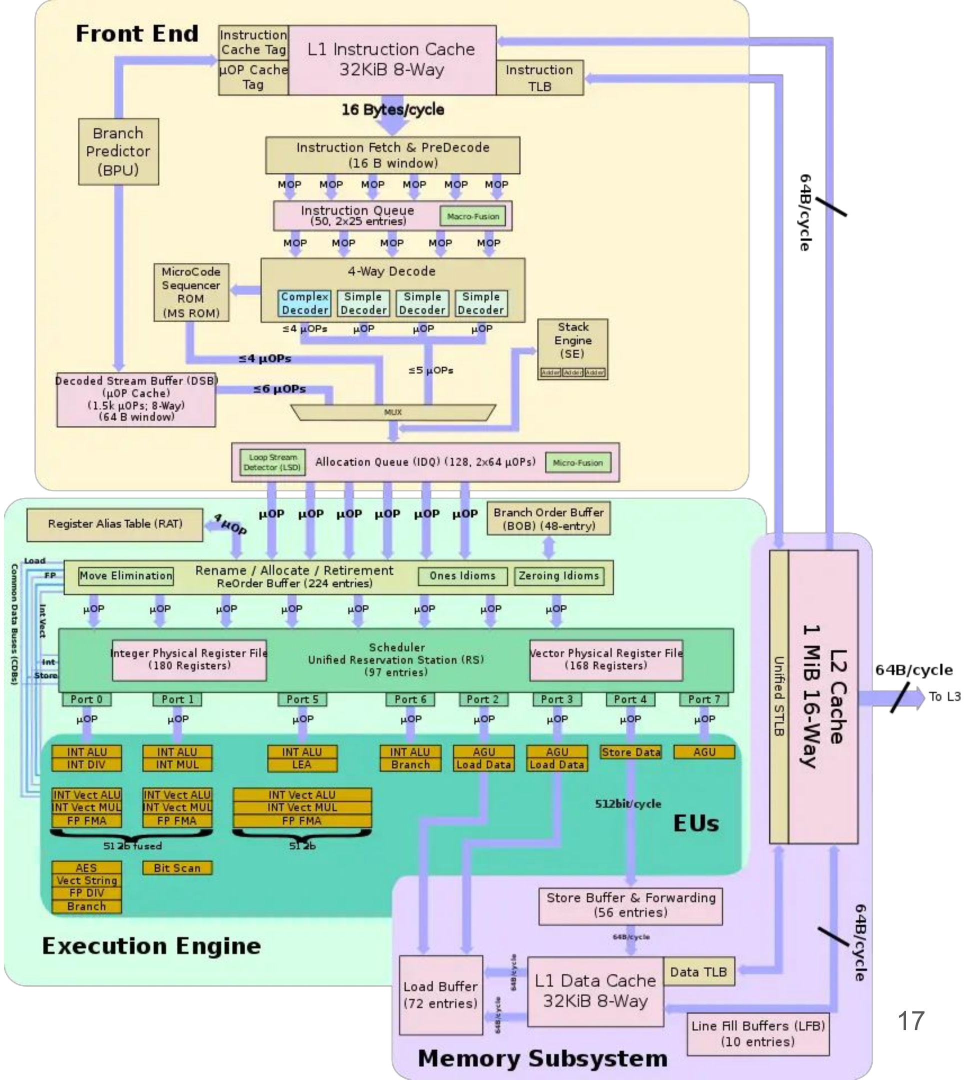
The two most popular classic architectures of computer systems.





An example of modern architecture For intel skylake server (Below is the full chip and right is the microarchitecture of one CPU core). Source : wikichip.org





Observations

- Nowadays, computers architecture are increasingly complex.
- Applications design must consider this complexity.
- The theoretical complexity of a good algorithm design may be constrained by the target computer architecture for certain applications.
- Specific architectures are necessary for optimal performance with some applications.
- Analyzing hardware bottlenecks is crucial for the performance and efficiency of programs.

Well known bottleneck analysis for code performance

- **Caches Utilisation**
  - Miss analysis, Hit analysis, Request rate, data locality, data reuse, ….
- **Vectorization efficiency and pipeline**
  - Vectorization ratio (Packed FLOPS:DP:SP:AVX),
  - pressure on L/S Unit, pipeline efficiency (Stalls, cycles, IPC, ….), ….
- **Multi-core/Multi-threads efficiency**
  - workload/task dispatching, offloading, core/uncore frequencies, etc….
- **Memory efficiency**
  - Bandwidth, bus saturation, latency, data volume, memory request, ….

Research questions

- Is execution time correlate with energy consumption ?
- Do caches misses indicate more/less energy consumption ?
- How data traffic on memory can affect DRAM energy ?
- Does SIMD use more energy than scalar implementation ?
  - energy & time scalability
- And what about parallel multithreaded versions ?
  - Threads scalability for energy & time correlation
  - usage of full power of the CPU (SIMD+OpenMP all core)

Methodology

1. Select representatives compute bound and memory bound kernels
2. for each kernel, implement naive version, vectorized version (SSE, AVX, AVX512), and parallelized (OpenMP)
3. Use LIKWID MAKER API for code instrumentation in implementation
4. Define representative size of dataset to be generated
5. Make sure to use appropriate compilations flags to avoid unexpected compiler optimization and compile all versions
6. Define LIKWID events group to count in regard of metrics to analyse and commands to run with LIKWID
7. Repeat experiments for each event group, each code version and save results for analysis
8. Preprocess results and make discussions.

Workloads selection

- **TRIAD (saxpy):** This operation involves multiplying vector A by the scalar α, then adding vector B. It's a common operation in linear algebra and has various applications in mathematics, physics, computer science, and many other fields.

$$C=\alpha \cdot A+B$$

- **gemm:** This formula represents the calculation of each element of the resulting matrix C based on the elements of matrices A and B.

$$C_{ij} = \sum_{k=0}^{N-1} A_{ik} \cdot B_{kj}$$

- **Distance:** generalized form of the Euclidean distance, where instead of computing the distance between two points (x, y), you're summing the distances between the origin (0, 0) and each point (xi, yi)

$$\sum_{i=1}^{n} \sqrt{(x_i)^2 + (y_i)^2}$$

- **spmv :** The loop iterates over the nonzero elements of the sparse matrix (indexed by i) and performs the multiplication and addition operation for each nonzero element. COO format:

$$y_{\text{rowind}[i]} += \text{val}[i] \cdot x_{\text{colind}[i]}$$

Experimental evaluation : metrics (Events Groups)

- **MEM : Memory performance**
  - Metrics : Memory Bandwidth, Memory data volume, vectorization ratio, Operational Intensity
- **ENERGY : Energy efficiency**
  - Metrics : Power & energy CPU, Power & energy DRAM, SoC Temperature
- **L2CACHE : Analysis for cache efficiency**
  - Metrics : L2 request rate, L2 Miss Ratio, L2 Miss rate
- **FLOPS_AVX : Computing performance**
  - Metrics : time, Cores frequency, Uncore frequency, Packet FLOPS SP,

Preliminary results

Code version	Runtime [s]	Time accel.	Energy CPU [J]	Energy CPU ratio
SPMV				
seq	14,26	1	1 108,50	1
sse	15,22	0,94	1 157,10	0,96
avx	11,32	1,26	897,72	1,23
omp_2	5,78	2,47	544,11	2,04
omp_4	3,22	4,43	338,55	3,27
omp_8	3,07	4,64	327,58	3,38
omp_16	1,87	7,65	227,27	4,88
omp_32	1,63	8,72	229,85	4,82
GEMM				
seq	124,83	1	9 160,88	1
sse	46,95	2,66	3 469,80	2,64
avx	29,02	4,3	2 151,76	4,26
omp_2	64,15	1,95	5 248,30	1,75
omp_4	32,22	3,87	3 039,53	3,01
omp_8	25,44	4,91	2 468,35	3,71
omp_16	12,98	9,62	1 562,17	5,86
omp_32	7,14	17,49	1 136,23	8,06

Tab (a.): Time acceleration vs energy gain ratio on AMD EPYC 7513 (Milan Microarchitecture)

Code version	Runtime [s]	Time accel.	Energy CPU [J]	Energy CPU ratio	Energy RAM [J]	Energy RAM ratio
SPMV						
seq	9,91	1	991,37	1	164,17	1
sse	10,74	0,92	1 070,30	0,93	175,74	0,93
avx	9,17	1,08	920,92	1,08	153,05	1,07
omp_2	5,63	1,76	597,85	1,66	105,59	1,55
omp_4	2,96	3,35	349,97	2,83	66,35	2,47
omp_8	1,66	5,97	230,2	4,31	46,37	3,54
omp_16	0,99	9,98	175,81	5,64	34,74	4,73
omp_32	0,71	14,03	159,36	6,22	29,4	5,58
GEMM						
seq	116,56	1	11 943,84	1	1 606,94	1
sse	56,08	2,08	5 662,58	2,11	774,71	2,07
avx	29,14	4	2 920,22	4,09	401,29	4
omp_2	56,84	2,05	6 162,93	1,94	786,53	2,04
omp_4	34,26	3,4	4 092,08	2,92	473,73	3,39
omp_8	17,14	6,8	2 471,08	4,83	236,93	6,78
omp_16	8,61	13,54	1 670,17	7,15	119,88	13,41
omp_32	5,56	20,97	1 330,47	8,98	77,29	20,79

Tab (b.): Time acceleration vs energy gain ratio on Intel Platinum 8358 (Ice Lake microarchitecture)

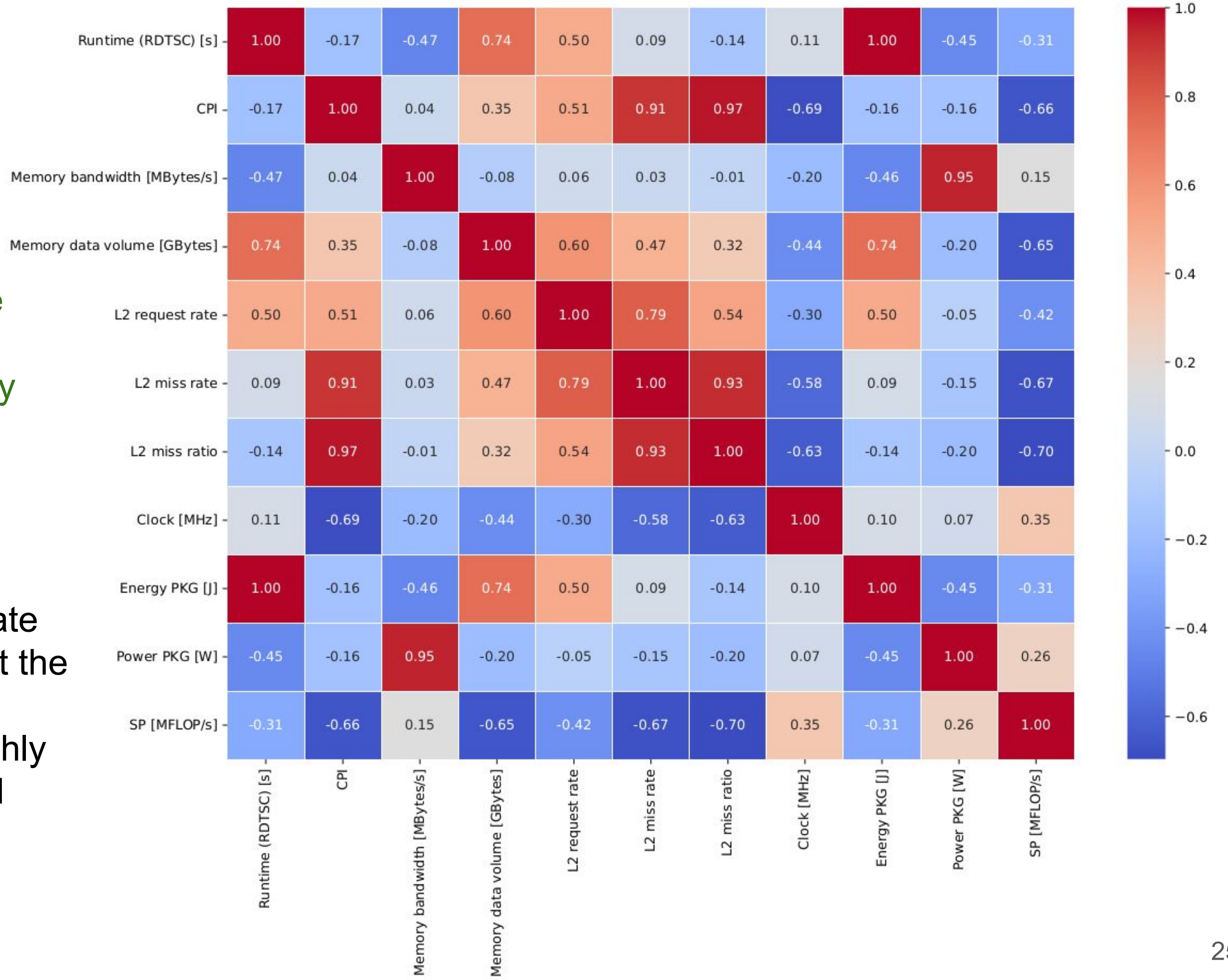
Preliminary results

(Single threads correlation on AMD)

- CPU Power is highly correlate with Mem Bandwidth
- Runtime and energy are highly correlated

Notice for Intel:

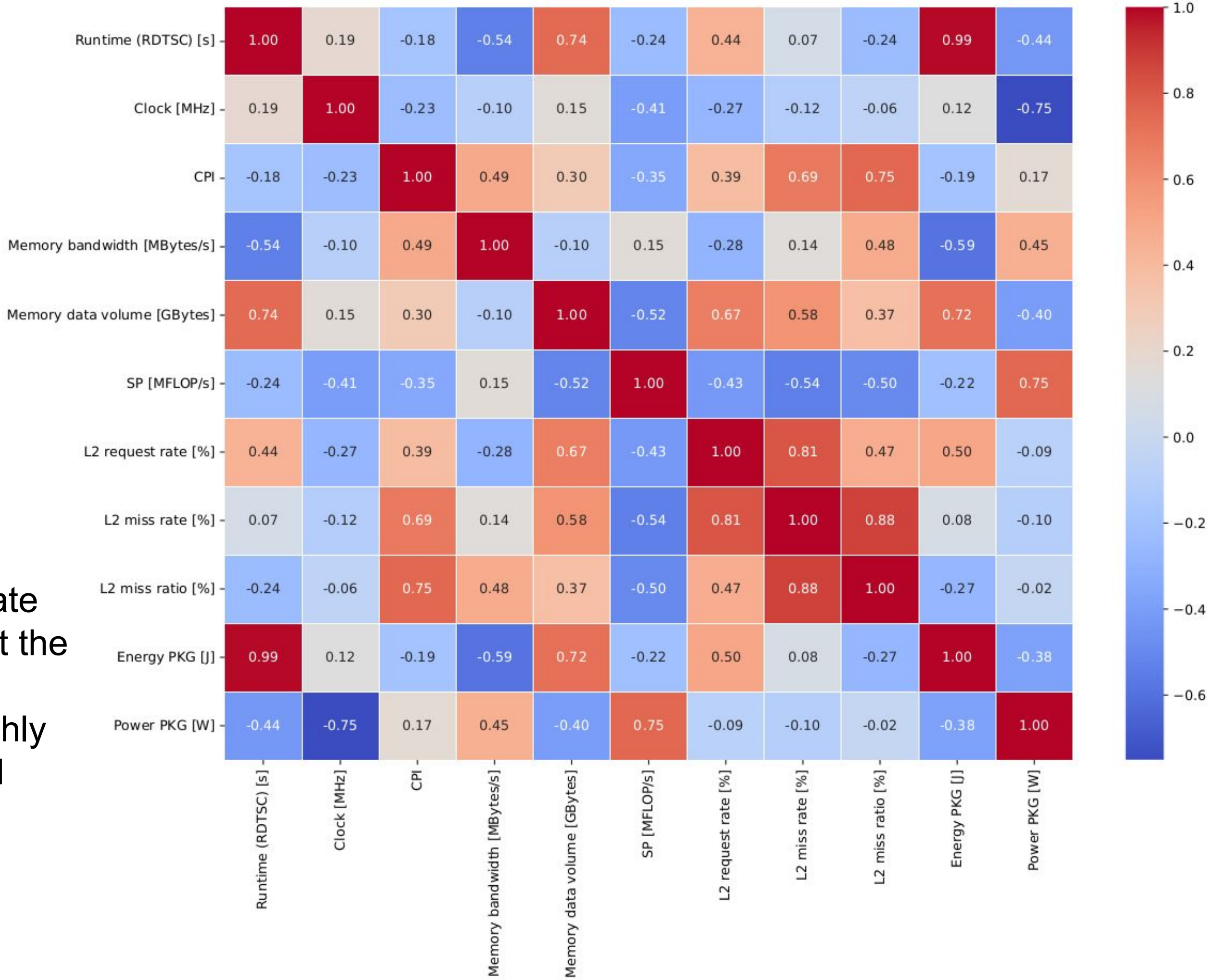
- RAM Power is highly correlate with Mem Bandwidth but not the case for CPU power
- Runtime and energy are highly correlated for both CPU and RAM energy



Preliminary results

(Multi-threads correlation on AMD)

- No correlation between CPU Power and Mem Bandwidth for multi-threading run
- Runtime and energy are highly correlated



Notice for Intel:

- RAM Power is highly correlate with Mem Bandwidth but not the case for CPU power
- Runtime and energy are highly correlated for both CPU and RAM energy

Conclusion and future works

- Conclusion:
  - Design and implementation of multi-devices energy profiler that works for the major cases.
  - Validation of our analytical model for RAM energy measurement on platforms that do not integrate RAPL DRAM.
  - Preliminary results for time vs energy correlation analysis
  - Preliminary results for performance metrics correlation with energy/power consumption
- Future works
  - Establish analytical models for energy vs performance using non correlated metrics
  - Apply methodology for GPU

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.