Introduction à la consommation d'énergie des GPUs

Statistiques au sommet de Rochebrune 2026

Hugo Gangloff
Université Paris-Saclay, AgroParisTech, INRAE, MIA Paris-Saclay

23 - 27 mars 2026

RÉPUBLIQUE FRANÇAISE
Liberté Égalité Fraternité

INRAE

1. Introduction

2. Principes de base

3. Tendances

4. Conclusion

1. Introduction

2. Principes de base

3. Tendances

4. Conclusion

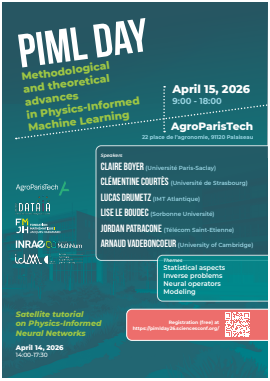
CATI SOBRE

- Animateur avec Sophie Schbath
- Collectif (d'ingénieurs) INRAE motivés par les questions de sobriété numérique, l'étude des impacts du numérique de notre recherche et la recherche d'alternatives
- Quelques futurs groupes de travail
  - Prolonger les durées de vie (sobriété matérielle via le don d'équipements par exemple)
  - Promotion et formation de la science ouverte et au logiciel libre (Windows → Linux)
  - Étude de l'écoconception de logiciel
  - Se former et communiquer sur les impacts environnementaux et aux méthodes de calculs d'empreinte environnentale
  - **Étude des datacenters** avec GT Éthique et Environnement (Groupe Calcul CNRS)

14 - 15 avril 2026 à AgroParisTech

- **Methodological and theoretical advances in Physics-Informed Machine Learning**
  - **14 avril 2026** : Après-midi tutoriel PINNs + `jinns`
    → Demander sa place !
  - **15 avril 2026** : 6 présentations autour du PIML (problèmes inverses, aspects statistiques, opérateurs neuronaux, ...) + posters
    → Complet en présentiel, possibilité de visio !

https://pimlday26.sciencesconf.org/

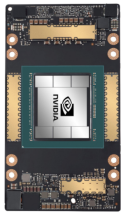


NVIDIA A100 GPU (2020)

Composants :
- puce GPU
- SRAM
- Voltage regulators
- PCIe
- Power connector

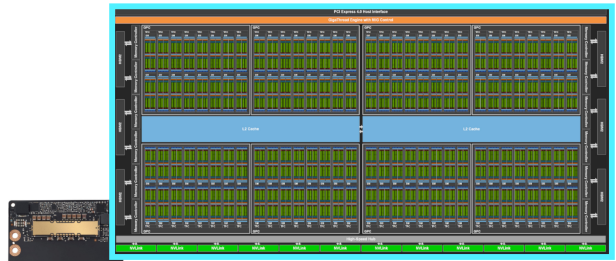
(- Heat sink)
(- Casing)
- **TDP = 400W**

Source : NVIDIA A100 WhitePaper



NVIDIA A100 GPU (2020)

Zoom sur la puce GPU : GA100
→Streaming Multiprocessors
→L2 cache
→Communication



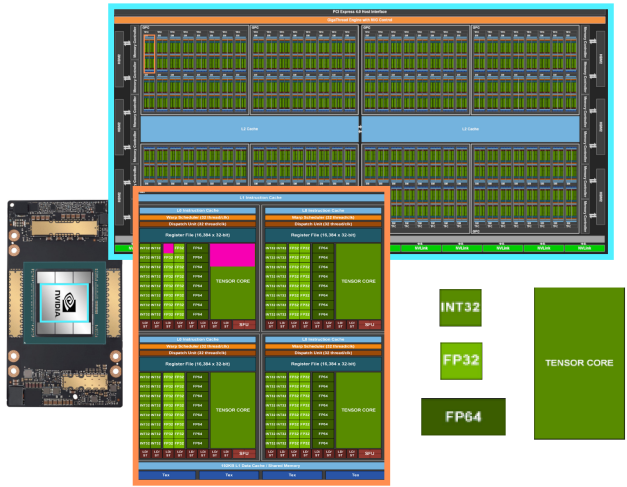
NVIDIA A100 GPU (2020)

Streaming Multiprocessors :
→ Cores
→ L1 cache

NVIDIA A100 GPU (2020)

→ Coeurs de calculs :
- 8192 *CUDA cores*
- 512 *Tensor cores*

→ Warp Scheduler → L0 cache



NVIDIA A100 GPU (2020)

Des cœurs de calcul pour l'IA, les simulations, le rendu graphique, opèrent sur différents types (FP64, FP32, FP16, etc.) [Jarmusch et al., 2025]

- CUDA core : pour le calcul parallèle généraliste
- Tensor core : optimisé pour les calculs matriciels

Des milliards de transistors composent ces cœurs de calcul
$\rightarrow$ 54.2 milliards de transistors
(TSMC 7 nm FinFET)

Sources : NVIDIA A100 WhitePaper

NVIDIA DGX A100

SYSTEM SPECIFICATIONS

	NVIDIA DGX A100 640GB
GPUs	**8x NVIDIA A100 80GB Tensor Core GPUs**
GPU Memory	**640GB total**
Performance	**5 petaFLOPS AI** **10 petaOPS INT8**
NVIDIA NVSwitches	**6**
System Power Usage	**6.5 kW max**
CPU	**Dual AMD Rome 7742, 128 cores total, 2.25 GHz (base), 3.4 GHz (max boost)**

1. Introduction

2. Principes de base

3. Tendances

4. Conclusion

Expérience

- $A, B \in \mathbb{R}^{n \times n}$
- Mesurons la consommation d'énergie et le temps pour le calcul de AB
- Deux scénarios d'initialisation pour A et B
- Le reste des paramètres est fixé (même *hardware*, nombre de FLOP, lignes de code, etc.)

Initialisation	A and $B \sim \mathcal{N}(0, I_n)$	$A = B = \mathbf{0}_{R^{n \times n}}$
Énergie, E, (J)		
Temps, t, (s)	1,13	1,13

→ **Quel scénario coûte le plus d'énergie ? D'où vient la différence ?**

(détails sur l'expérience)

Expérience

- $A, B \in \mathbb{R}^{n \times n}$
- Mesurons la consommation d'énergie et le temps pour le calcul de AB
- Deux scénarios d'initialisation pour A et B
- Le reste des paramètres est fixé (même *hardware*, nombre de FLOP, lignes de code, etc.)

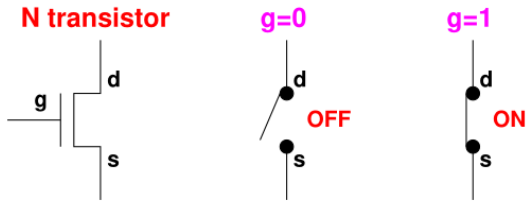
Initialisation	A and $B \sim \mathcal{N}(0, I_n)$	$A = B = \mathbf{0}_{R^{n \times n}}$
Énergie, E, (J)	**38,17**	**28,97**
Temps, t, (s)	1,13	1,13

→ **Quel scénario coûte le plus d'énergie ? D'où vient la différence ?**

(détails sur l'expérience)

Consommation énergétique d'un circuit

Simple logic behavior (≈ switch)



Adapté de [Tisserand, 2010]

Consommation énergétique d'un circuit

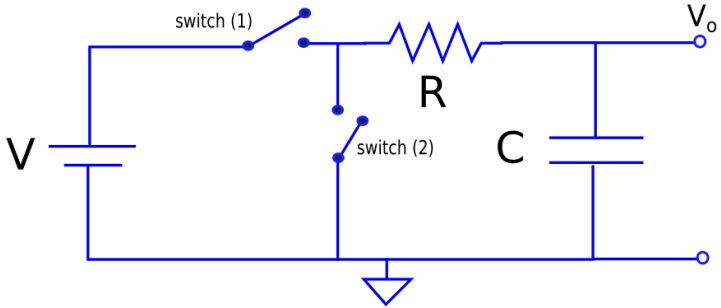


Figure 11.2: A simple RC circuit

Adapté de [Sarangi, 2023]

Consommation énergétique d'un circuit

Deux sources de consommation d'énergie [Sarangi, 2023]

$$P_{tot} = P_{static} + P_{dyn}$$

- P_{dyn} : liée à énergie dissipée lorsque les transistors du circuit changent d'état
  - Énergie des transistors qui changent d'état
  - ($P_{short-circuit}$: Énergie dissipée par un courant qui se crée quand deux transistors changent d'état au même moment)
- P_{static} : liée à énergie dissipée lorsque le circuit est en mode oisif, dûe principalement aux courants de fuite

Consommation énergétique d'un circuit

Pour un circuit, on peut montrer que (détails) :

$$P_{dyn} \propto \alpha C V^2 f$$

avec α la proportion de transistors qui changent d'état à chaque cycle, f la fréquence des changements de cycles, C la capacité du circuit et V la tension source du circuit. Et on sait que

$$E = P \times t$$

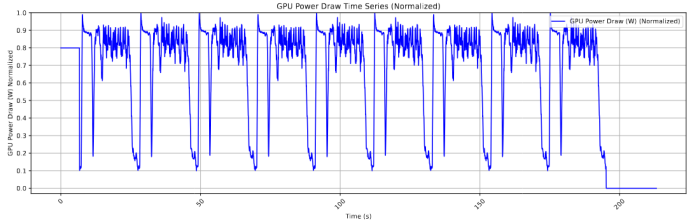
Retour à l'expérience

- Si on ne considère que P_{dyn} et si on considère C et V constant alors :
  $\rightarrow$ les différentes initialisations ont joué sur le facteur α
  $\rightarrow$ l'initialisation mathématique a influé directement sur les switchs des transistors
- La dépendance de la puissance nécessaire sur les données d'entrée est bien connue [Lucas and Juurlink, 2016] [1]
- On ne peut creuser plus à ce stade :
  - l'équation n'est qu'un modèle
  - C et V ne sont pas connues

1. Voir aussi une expérience plus détaillée : https://www.thonking.ai/p/strangely-matrix-multiplications

Retour à l'expérience

- Plus généralement, les différentes phases d'un process d'entrainement d'IA ne mobilisent pas la même puissance
  → Ces pics de consommation sont l'objet de nombreuses recherches [Choukse et al., 2025]



- On remarque aussi que le TDP est une indication maximale plus ou moins atteinte

P_{static} illustration

GPU Nvidia RTX A6000

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 550.107.02             Driver Version: 550.107.02     CUDA Version: 12.4     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  NVIDIA RTX A6000               Off |   00000000:01:00.0 Off |                  Off |
| 32%   61C    P2             78W /  300W |      93MiB /  49140MiB |      0%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
```

1. Introduction

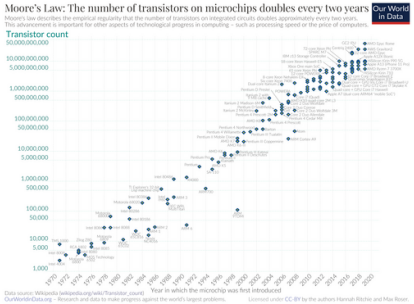
2. Principes de base

3. Tendances

4. Conclusion

Loi de Moore [Moore et al., 1965]

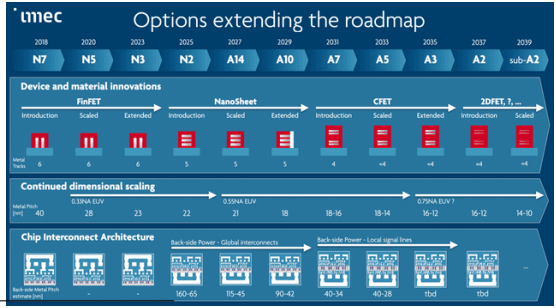
Moore's law is the observation that the number of transistors in an integrated circuit doubles about every two years, and thus projection of a historical trend.



https://en.wikipedia.org/wiki/Moore%27s_law

Loi de Moore [Moore et al., 1965]

- C'est une feuille de route de l'industrie du semi-conducteur
- Loi économique qui donne le rythme idéal de la miniaturisation des transistors (et donc la quantité de capacité de calculs à mettre sur le marché) [1]



1. Il faut que la quantité de calcul mise sur le marché soit consommée pour que les investissements de l'industrie soient rentables ! Pour qu'elle soit consommée il faut *réduire la qualité et l'efficacité des [logiciels]*...
https://gauthierroussilhe.com/en/articles/how-to-use-computing-power-faster

Dennard scaling [Dennard et al., 1974]

- D'après [Hennessy and Patterson, 2019] :

  as transistor density increased [size decreases], power consumption per transistor would drop [V decreases], so the power per mm2 of silicon would be near constant.

- Mais une réduction de la taille entraine aussi :

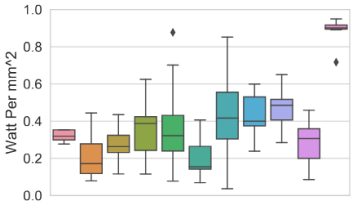
$$\downarrow C \implies \text{possibilité d' } \uparrow f$$

$\implies$ Il semblait que l'on pouvait gagner en vitesse de calcul (f) et devenir plus efficace énergétiquement sans limite... ! [2]

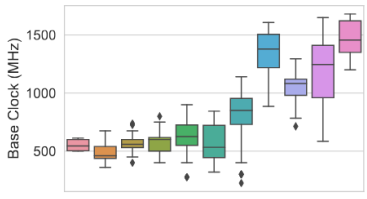
2. Voir aussi https://shape-of-code.com/2026/02/08/dennard-scaling-a-necessary-condition-for-moores-law/

Dennard scaling [Dennard et al., 1974]

- Or cette loi ne s'observe plus depuis le début des années 2000
- D'après [Sun et al., 2019] :
  - Explosion de la puissance par mm^2 sur la technologie 7nm
  - Stagnation de la fréquence horloge f



(b) Watt per mm^2 vs. process size.



(c) Base clock vs. process size.

Dennard scaling [Dennard et al., 1974]

Physiquement, que se passe-t-il ?

- D'une part P_{static} **augmente avec la température** et la diminution de la taille notamment [Sarangi, 2023, Section 11.1.2, *Subthreshold Leakage*]

Dennard scaling [Dennard et al., 1974]

Physiquement, que se passe-t-il ?

- D'une part P_{static} **augmente avec la température** et la diminution de la taille notamment [Sarangi, 2023, Section 11.1.2, *Subthreshold Leakage*]
- D'autre part, on a vu que :

$$\uparrow f \implies \uparrow P_{dyn} \implies \uparrow P_{tot}$$

Dennard scaling [Dennard et al., 1974]

Physiquement, que se passe-t-il ?

- D'une part P_{static} **augmente avec la température** et la diminution de la taille notamment [Sarangi, 2023, Section 11.1.2, *Subthreshold Leakage*]
- D'autre part, on a vu que :

$$\uparrow f \implies \uparrow P_{dyn} \implies \uparrow P_{tot}$$

- Or (détails)

$$\uparrow P_{tot} \implies \uparrow T$$

Dennard scaling [Dennard et al., 1974]

Physiquement, que se passe-t-il ?

- D'une part P_{static} **augmente avec la température** et la diminution de la taille notamment [Sarangi, 2023, Section 11.1.2, *Subthreshold Leakage*]
- D'autre part, on a vu que :

$$\uparrow f \implies \uparrow P_{dyn} \implies \uparrow P_{tot}$$

- Or (détails)

$$\uparrow P_{tot} \implies \uparrow T$$

- Mais donc

$$\uparrow T \implies \uparrow P_{static} \implies \uparrow P_{tot} \implies \uparrow T \implies \uparrow P_{static} \implies \uparrow P_{tot}$$
$$\implies \cdots \implies \text{Thermal Runaway}$$

Dennard scaling [Dennard et al., 1974]

Physiquement, que se passe-t-il ?

- D'une part P_{static} **augmente avec la température** et la diminution de la taille notamment [Sarangi, 2023, Section 11.1.2, *Subthreshold Leakage*]
- D'autre part, on a vu que :

$$\uparrow f \implies \uparrow P_{dyn} \implies \uparrow P_{tot}$$

- Or (détails)

$$\uparrow P_{tot} \implies \uparrow T$$

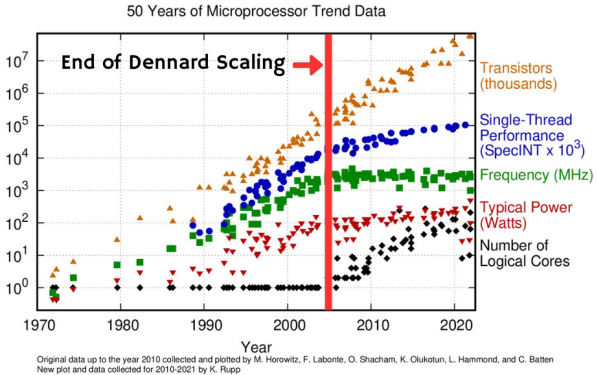
- Mais donc

$$\uparrow T \implies \uparrow P_{static} \implies \uparrow P_{tot} \implies \uparrow T \implies \uparrow P_{static} \implies \uparrow P_{tot}$$
$$\implies \cdots \implies \text{Thermal Runaway}$$

- **Il faut de contrôler** P_{tot} (via f notamment, mais aussi V) pour contrôler T

Dennard scaling [Dennard et al., 1974]

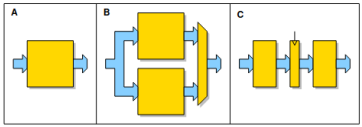
- Le phénomène P_{static} n'avait pas été pris en compte par Dennard !



Dennard scaling [Dennard et al., 1974]

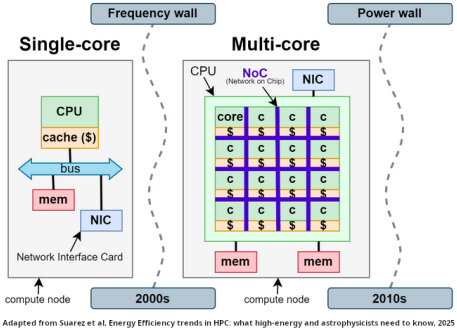
- Début du règne des approches parallèles / multicœurs et des GPUs (*General Purpose GPU*[2]) → Contrôler P_{tot}

Idea: reduce supply voltage V_{DD} without speed degradation



solution	total capa.	supply voltage	circuit freq.	power consumption
standard (A)	C	V	f	$P_A = CV^2f$
parallel (B)	$2.2\,C$	$0.6\,V$	$0.5\,f$	$P_B = 2.2C(0.6V)^2 0.5f = 0.396\,P_A$
pipeline (C)	$1.2\,C$	$0.6\,V$	f	$P_C = 1.2C(0.6V)^2 f = 0.432\,P_A$

Adapté de [Tisserand, 2010]



- Cela implique aussi un changement des techniques de programmation → parallélisation des algorithmes, etc.

2. https://developer.nvidia.com/blog/cuda-refresher-reviewing-the-origins-of-gpu-computing/

Efficacité énergétique des GPUs

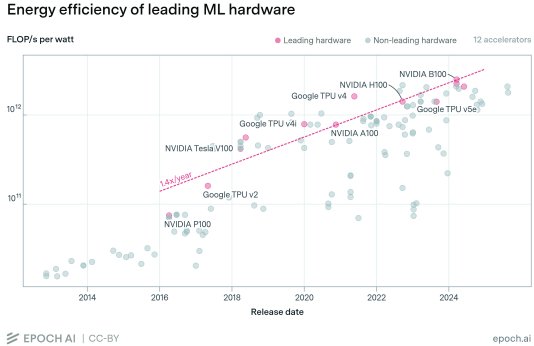
Des recherches soutenues améliorent les GPUs et la manière de les opérer. Par exemple [3] :

- Techniques de gestion de la puissance (*power gating*, *clock gating*, *Dynamic Voltage Frequency Scaling*) → Contrôler P_{tot}
- Spécialisations des composants (*tensor cores*, TPUs)
- Augmentation de la taille de la puce et amélioration des semi-conducteurs
- Gestion de la mémoire (augmentation de la bande passante mémoire)
- Types de faibles précisions (*tensor-FP16*, *tensor-INT8*, *tensor-INT4*)

3. Liste non exhaustive issue de https://padopado.org/2025/06/11/gpu-power-management-a-comprehensive-analysis/ et https://epoch.ai/blog/trends-in-machine-learning-hardware#trends-of-primary-performance-metrics

Efficacité énergétique des GPUs

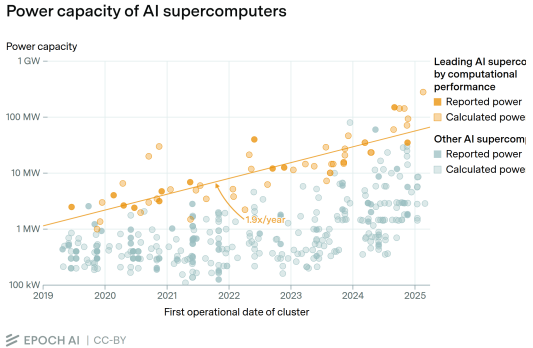
- Le rapport FLOP / Watt (efficacité énergétique) continue d'augmenter [3]

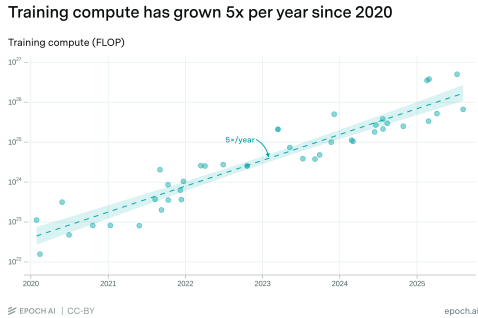


3. *Performance Per Watt Is The New Moore's Law* : https://newsroom.arm.com/blog/performance-per-watt

Effet rebond

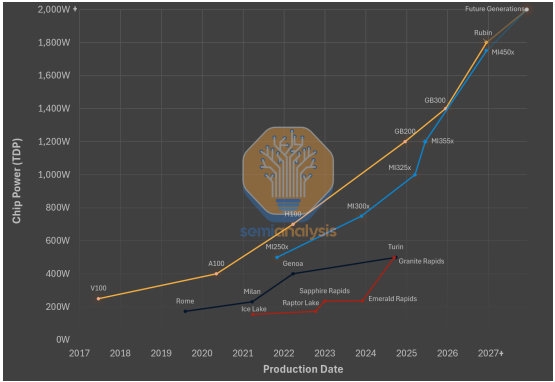
- Malgré tout, **la consommation énergétique explose** car les besoins en capacité de calculs explosent pour les applications d'IA (**IAg** surtout). Les gains en efficacité énergétique ne peuvent compenser la croissance des besoins en calculs [EPRI and epoch.ai, 2025].





Effet rebond

- La puissance maximale (TDP) des GPUs tend à exploser



https://newsletter.semianalysis.com/p/datacenter-anatomy-part-2-cooling-systems

Effet rebond

Les datacenters *hyperscale* se démultiplient : Stargate Abilene, Texas, 445ha, 1.5GW à terme

Source : https://openai.com/fr-FR/index/stargate-advances-with-partnership-with-oracle/

Effet rebond

Petit-Landau (68), France, 35ha

Effet rebond

Alouette des champs - Alauda arvensis

1. Introduction

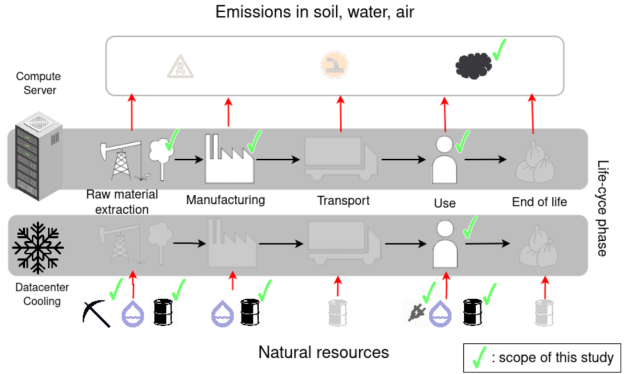
2. Principes de base

3. Tendances

4. Conclusion

Remise en perspective

L'utilisation ne représente qu'une fraction des impacts (énergie, eau, ressources abiotiques) [Morand et al., 2025]



Conclusion

- Les GPUs sont les équipements phares pour le calcul parallèle qui sont montés en puissance au milieu des années 2000s
- **L'efficacité énergétique des GPUs s'améliore rapidement** mais cela ne permet pas de compenser **l'énorme consommation énergétique des datacenters pour l'IA**
- Les GPUs font partie d'une **chaîne de production technologique parmi les plus complexes au monde** dont le développement est intimement lié au monde économique
- Les **impacts environnementaux** de toutes les phases de la vie des GPUs explosent

References I

Choukse, E., Warrier, B., Heath, S., Belmont, L., Zhao, A., Khan, H. A., Harry, B., Kappel, M., Hewett, R. J., Datta, K., et al. (2025).
Power stabilization for ai training datacenters.
arXiv preprint arXiv :2508.14318.

Dennard, R. H., Gaensslen, F. H., Yu, H.-N., Rideout, V. L., Bassous, E., and LeBlanc, A. R. (1974).
Design of ion-implanted mosfet's with very small physical dimensions.
IEEE Journal of solid-state circuits, 9(5) :256–268.

EPRI and epoch.ai (2025).
Scaling intelligence : The exponential growth of ai's power needs.
https://www.epri.com/research/products/000000003002033669.

Hennessy, J. L. and Patterson, D. A. (2019).
A new golden age for computer architecture.
Communications of the ACM, 62(2) :48–60.

Jarmusch, A., Graddon, N., and Chandrasekaran, S. (2025).
Dissecting the nvidia blackwell architecture with microbenchmarks.
arXiv preprint arXiv :2507.10789.

Lucas, J. and Juurlink, B. (2016).
Alupower : Data dependent power consumption in gpus.
In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 95–104. IEEE.

References II

Moore, G. E. et al. (1965).
Cramming more components onto integrated circuits.

Morand, C., Ligozat, A.-L., and Névéol, A. (2025).
The environmental impacts of machine learning training keep rising evidencing rebound effect.
arXiv preprint arXiv :2510.09022.

Sarangi, S. R. (2023).
Next-Gen Computer Architecture.
White Falcon, 1st edition edition.

Sun, Y., Agostini, N. B., Dong, S., and Kaeli, D. (2019).
Summarizing cpu and gpu design trends with product data.
arXiv preprint arXiv :1911.11313.

Tisserand, A. (2010).
Introduction to power consumption in digital integrated circuits.

Détails sur l'expérience *AB* I

Il vaut mieux réduire la fréquence d'horloge pour éviter la saturation du GPU :

```
$ sudo nvidia-smi -lgc 1000
```

undo this modification with :

```
$ sudo nvidia-smi --reset-gpu-clocks
```

Pour éviter des optimisations non voulues :

```
import os
os.environ["JAX_DISABLE_MOST_OPTIMIZATIONS"] = "True"
```

Détails sur l'expérience *AB* II

Imports, définitions et compilation de la fonction

```
import time
import jax
import jax.numpy as jnp
from zeus.monitor import ZeusMonitor

key = jax.random.PRNGKey(0)

@jax.jit
def f(A, B):
    return (A @ B)
s = 10000
# waste a computation for compilation
# Notably most of the time is spent here
key, subkey = jax.random.split(key)
A = jax.random.normal(subkey, ((s, s)), dtype=float)
key, subkey = jax.random.split(key)
B = jax.random.normal(subkey, ((s, s)), dtype=float)
_ = f(A, B) # compilation of f happens here
```

Détails sur l'expérience *AB* III

Initialisation selon des tirages de loi normale

```
print("Sampling matrices")
key, subkey = jax.random.split(key)
A = jax.random.normal(subkey, ((s, s)), dtype=float)
key, subkey = jax.random.split(key)
B = jax.random.normal(subkey, ((s, s)), dtype=float)
print("Start computations")
monitor = ZeusMonitor(gpu_indices=[0], sync_execution_with="jax")
monitor.begin_window("matmul")
start = time.time()
res = f(A, B)
jax.block_until_ready(res) # without that, python timer would give unreliable
# result (asynchroneous run)
end = time.time() # end - start identical to measure.time
measure = monitor.end_window("matmul")
print(res.shape, measure.gpu_energy[0], measure.time)
```

Détails sur l'expérience *AB* IV

Initialisation avec des inputs à 0

```
# Zeros initialization
print("Instanciating matrices")
A = jnp.zeros((s, s), dtype=float)
B = jnp.zeros((s, s), dtype=float)
print("Start computations")
monitor = ZeusMonitor(gpu_indices=[0], sync_execution_with="jax")
monitor.begin_window("matmul")
res = f(A, B)
measure = monitor.end_window("matmul")
print(res.shape, measure.gpu_energy[0], measure.time)
```

Puissance I

On peut approximer les circuits électroniques par des montages RC. Chaque composant est approximé par un modèle RC. La structure des transistors est modélisée par un condensateur. D'après [Sarangi, 2023] et [Tisserand, 2010] :

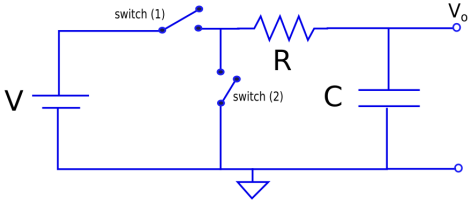
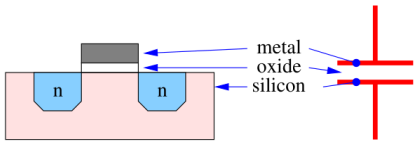


Figure 11.2: A simple RC circuit



Puissance II

Dans le circuit précédent le courant $I = C\frac{dV_0}{dt}$ et on a $V_0 = V - IR$. L'énergie totale à la charge est $\int VI = CV^2$ et donc $P_{tot} = CV^2 f$.
On peut montrer que l'énergie stockée dans un condensateur est $\frac{1}{2}CV^2$. Donc l'énergie dissipée en charleur à la charge est $\frac{1}{2}CV^2$. Le reste de l'énergie est dissipée à la décharge.