ESCAPE 824064

LAPP
Laboratoire d'Annecy de Physique des Particules

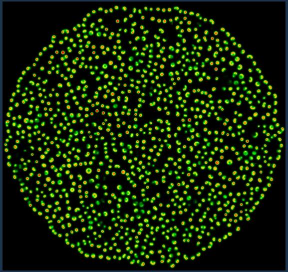
Use NVidia HPC SDK on MUST

Pierre Aubert

ESCAPE
cta cherenkov telescope array
cnrs
UNIVERSITÉ SAVOIE MONT BLANC
MUST
Mésocentre de l'Université de Savoie

School computing example

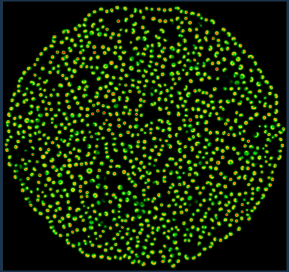
Gray Scott reaction (a chemistry game of life)
(for **CNRS 2023 Computing School**)



School computing example

Gray Scott reaction (a chemistry game of life)
(for **CNRS 2023 Computing School**)

$$\begin{aligned} U + 2V &\longrightarrow 3V \\ V &\longrightarrow P \end{aligned}$$



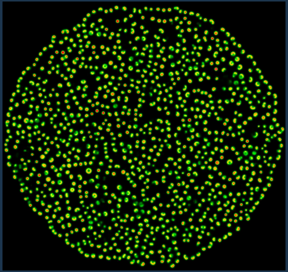
School computing example

Gray Scott reaction (a chemistry game of life)
(for **CNRS 2023 Computing School**)

Computing :

$$U + 2V \longrightarrow 3V$$
$$V \longrightarrow P$$

$$\frac{\partial u}{\partial t} = r_u \nabla^2 u - uv^2 + f_r \times (1 - u)$$
$$\frac{\partial v}{\partial t} = r_v \nabla^2 v + uv^2 - (f_r - k_r) \times v$$



- u and v are concentration of product **U** and **V**
- r_u and r_v diffusion rate of **U** and **V**
- k_r (**Kill Rate**), conversion rate from **V** to **P**
- f_r (**Feed Rate**), speed of process which feed **U** and kills **V** and **P**
- $\nabla^2 u$ and $\nabla^2 v$ are différence of space concentration between current cell and its neighbours

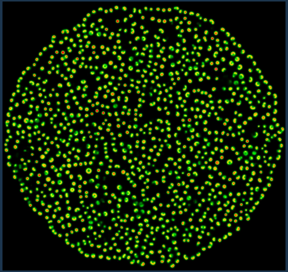
School computing example

Gray Scott reaction (a chemistry game of life)
(for **CNRS 2023 Computing School**)

Computing :

$$U + 2V \longrightarrow 3V$$
$$V \longrightarrow P$$

$$\frac{\partial u}{\partial t} = r_u \nabla^2 u - uv^2 + f_r \times (1 - u)$$
$$\frac{\partial v}{\partial t} = r_v \nabla^2 v + uv^2 - (f_r - k_r) \times v$$



- u and v are concentration of product **U** and **V**
- r_u and r_v diffusion rate of **U** and **V**
- k_r (**Kill Rate**), conversion rate from **V** to **P**
- f_r (**Feed Rate**), speed of process which feed **U** and kills **V** and **P**
- $\nabla^2 u$ and $\nabla^2 v$ are différence of space concentration between current cell and its neighbours

- Easy to understand
- Not so easy for the compiler
- Possibility of high speed up

Computation exercices

- Compute $1000 \times 34 = 34\,000$ images 1920×1080 **float**, store 1000 in **HDF5** file (8.3 GB).
- Evaluate full computation with **time**

Computation exercices

- Compute $1000 \times 34 = 34\,000$ images 1920×1080 **float**, store 1000 in **HDF5** file (8.3 GB).
- Evaluate full computation with **time**

Lectures :

- **C++20** : `https://cta-lapp.pages.in2p3.fr/COURS/PERFORMANCE_WITH_STENCIL/index.html`
- **Cuda/nvc++** : `https://cta-lapp.pages.in2p3.fr/COURS/PERFORMANCE_WITH_STENCIL_GPU/index.html`

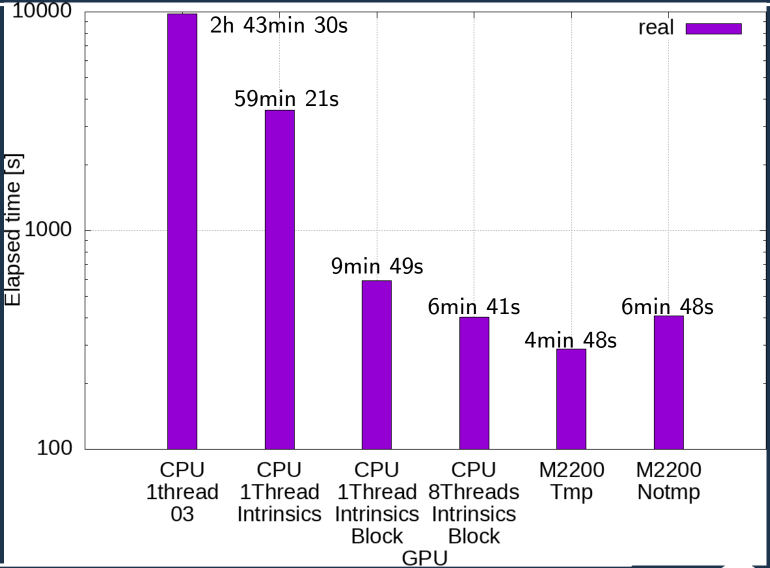
Computation exercices

CPU :

- Intel® Core™ i7-7820HQ
- 2.9 GHz
- 8 cores
- 32 GB RAM

GPU :

- M2200
- 1.04 GHz
- 1 024 cores
- 4 GB DRAM



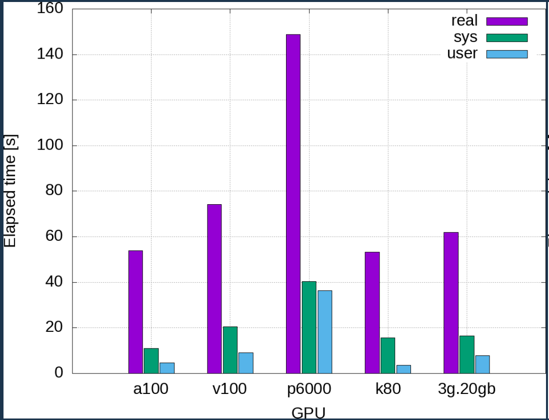
GPU MUST

	K80	P6000	T4	V100	A100	3G.20GB
TFlops (float)	8.73 (boost)	12.6	8.1	14	19.5	9.75
Memory (GB)	11.441 (24)	24	15	16	40	20
Nb Cuda Cores	2496 (4992)	3840	2560	5120	6912	2688
Clock rate (GHz)	0.824	1.645	1.590	1.380	1.410	1.410
Generation	3.7	6.1	7.5	7.0	8.0	8.0

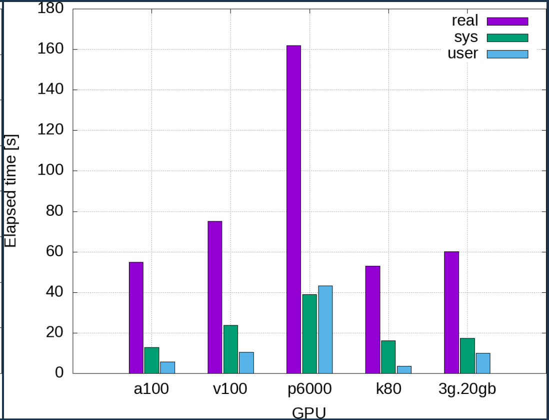
CUDA : $1\,000 \times 34 = 34\,000$ images

Result with 100 tests per **GPU**

With temporary



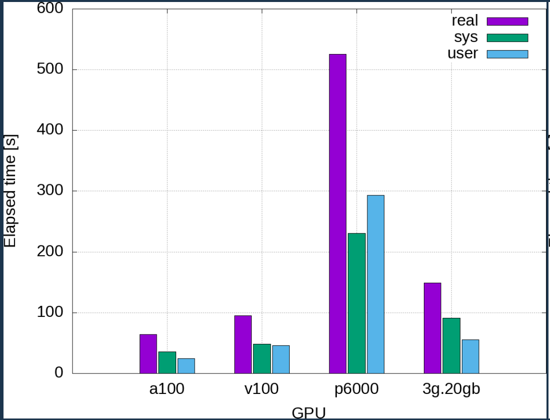
Without temporary



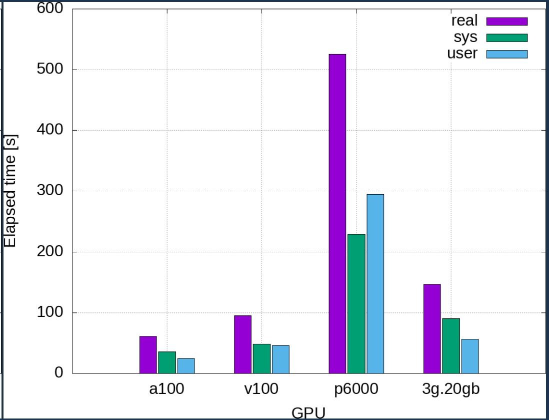
CUDA : $\underset{I/O}{5} \times 68\,000 = 340\,000$ images

Result with 100 tests per **GPU**

With temporary



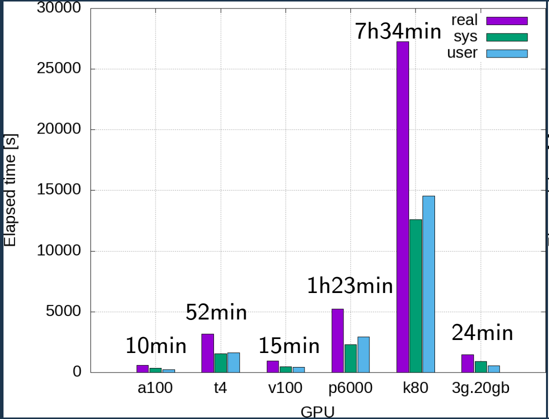
Without temporary



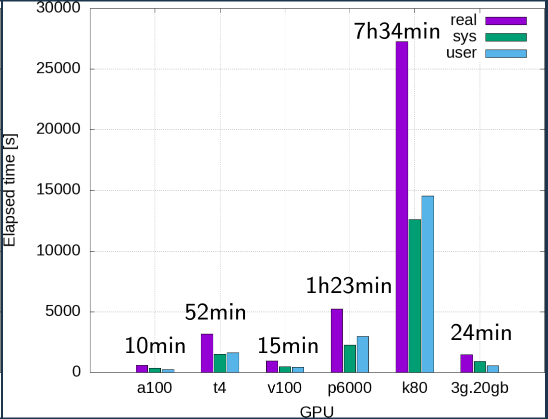
CUDA : $\underbrace{5}_{I/O} \times 680\,000 = 3\,400\,000$ images

Result with 100 tests per **GPU**

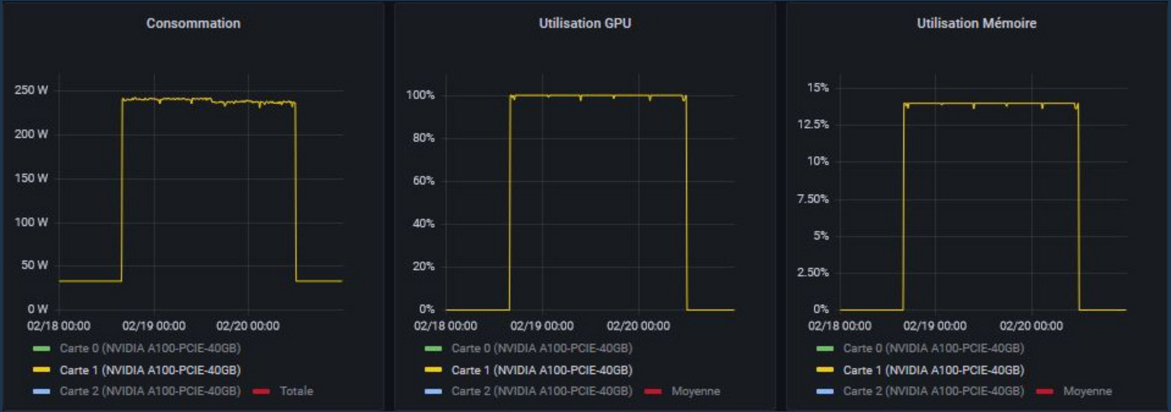
With temporary



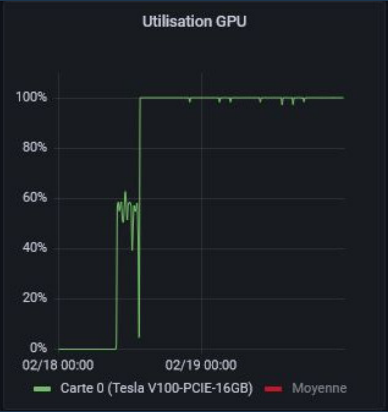
Without temporary

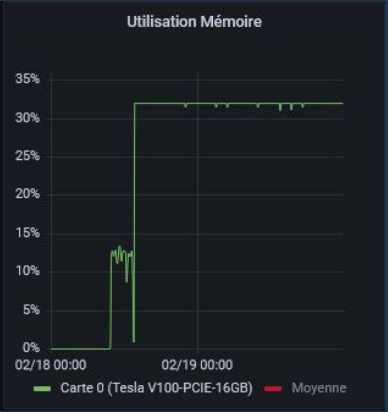


CUDA : Grafana perf A100 for 3 400 000 images

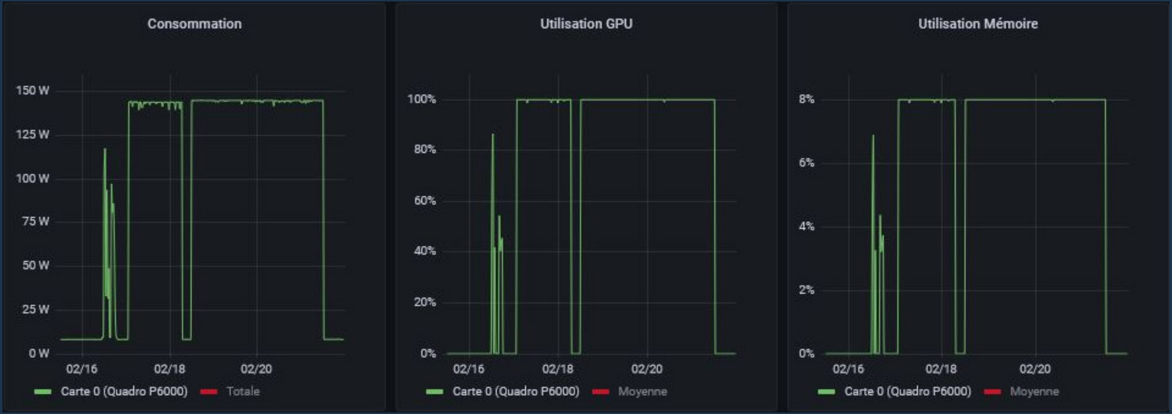


CUDA : Grafana perf V100 for 3 400 000 images

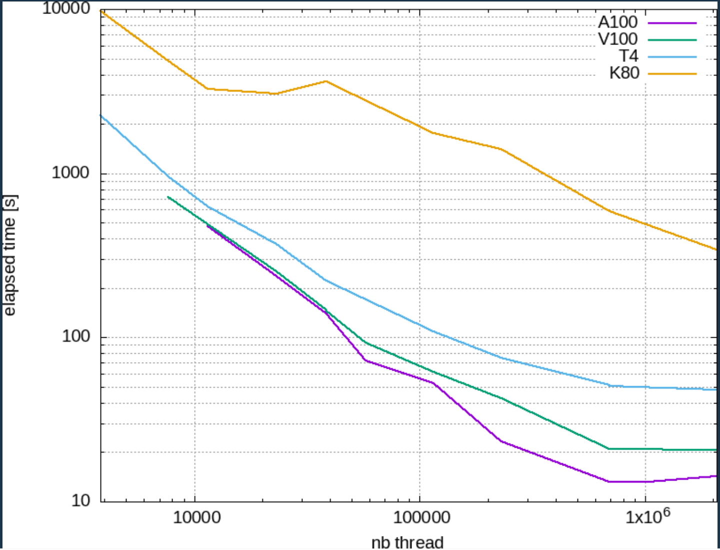




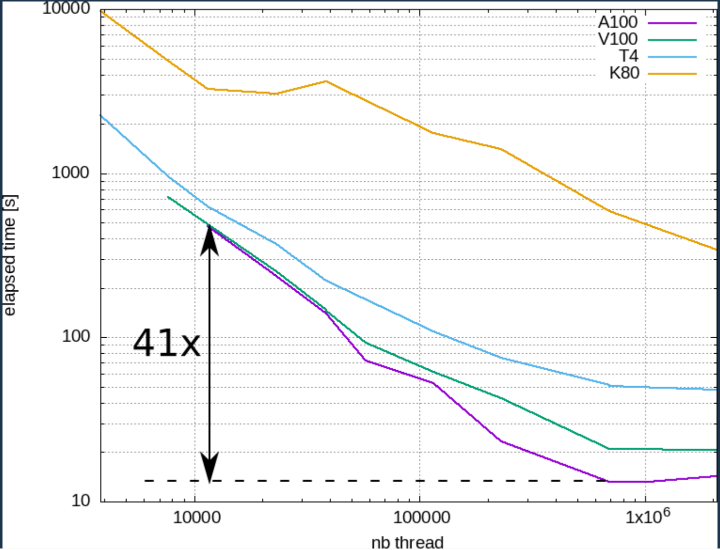
CUDA : Grafana perf P6000 for 3 400 000 images



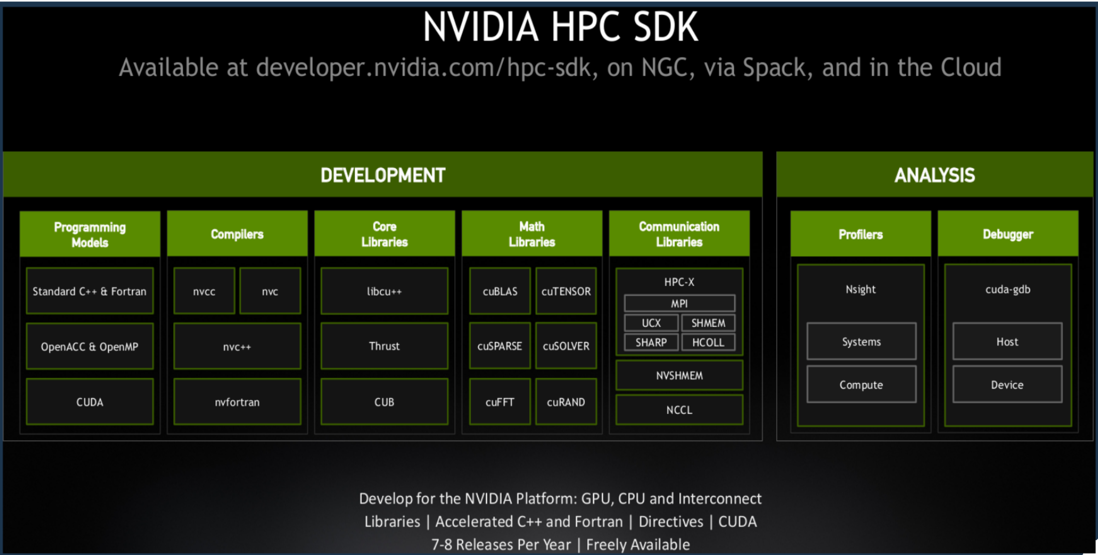
CUDA Perf VS nb threads for 340 000 images



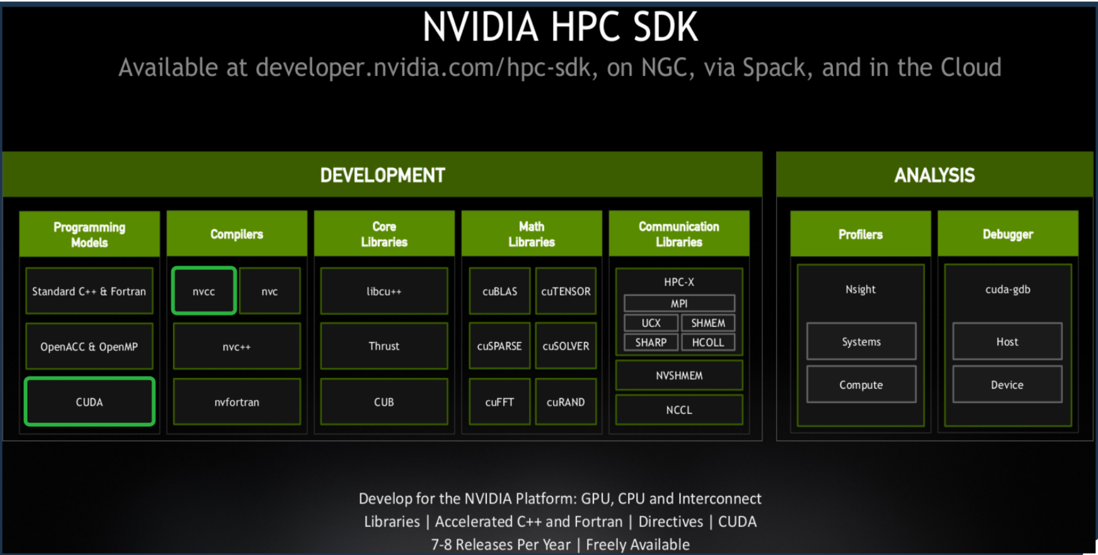
CUDA Perf VS nb threads for 340 000 images



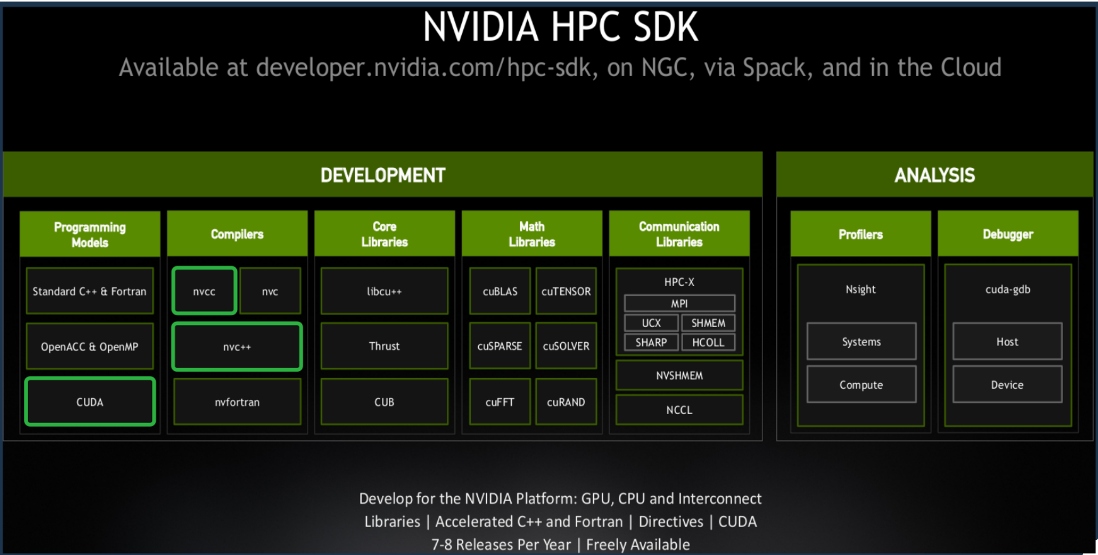
NVidia HPC SDK Introduction



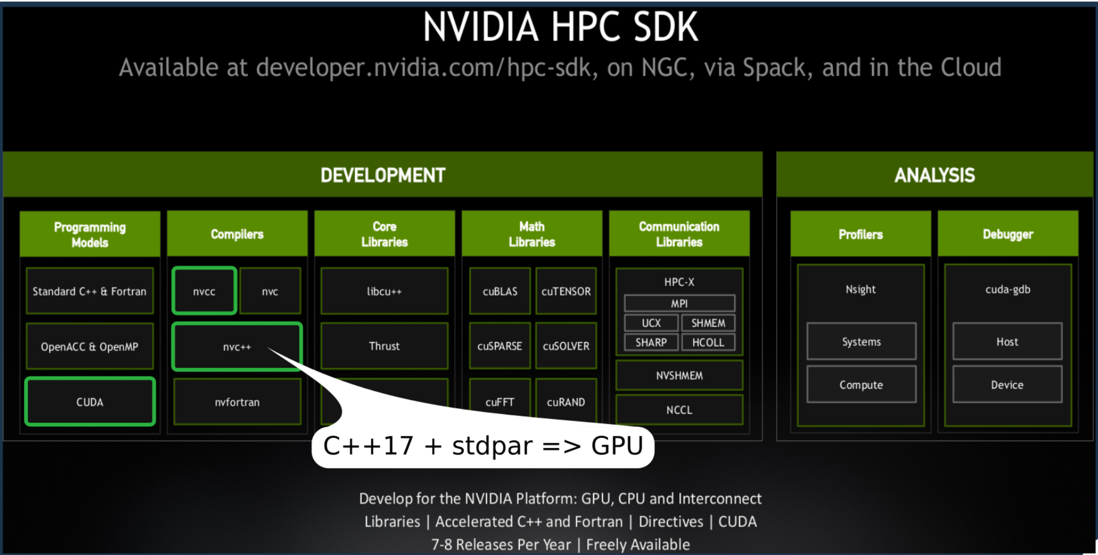
NVidia HPC SDK Introduction



NVidia HPC SDK Introduction



NVidia HPC SDK Introduction

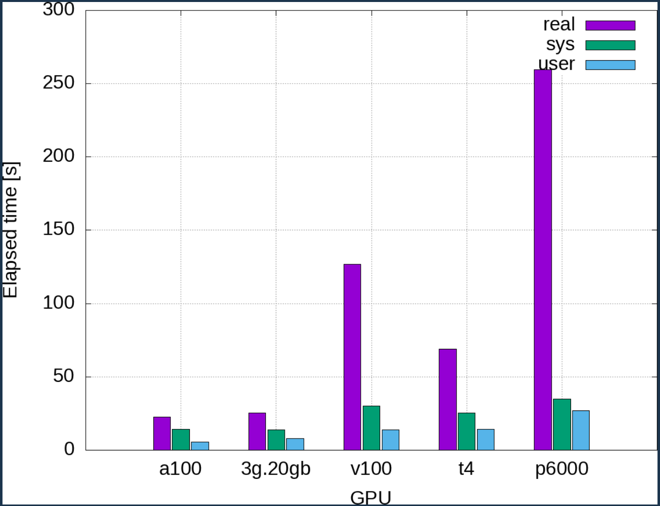


GPU MUST and HPC SDK

Use directly **C++17** to use **GPU** with **NVC++** :

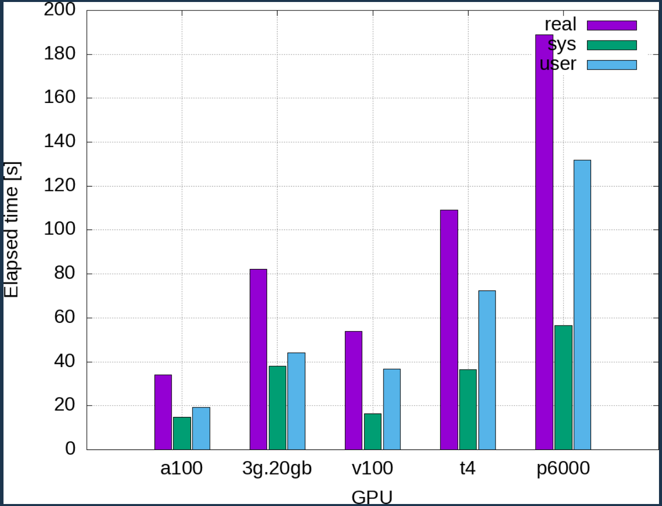
- Only for **compute capabilities** ≥ 6.0
- Can specify only one compute capability at compilation time
- Only for **C++17**/**C++20** (working with **G++-9**/**G++-11** or newer)
- Parallelism only with **TBB 2018** or newer (not on **CentOS 7**)

Result with 100 tests per **GPU**



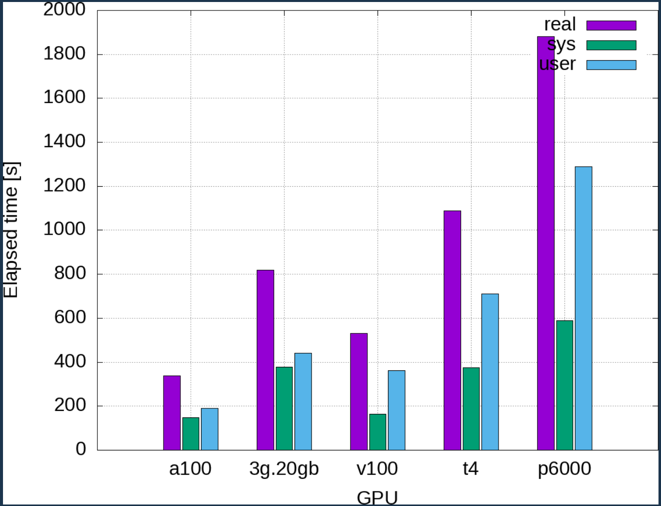
Gray Scott nvc++ : 5 $\times$ 68 000 = 340 000 images I/O

Result with 100 tests per **GPU**

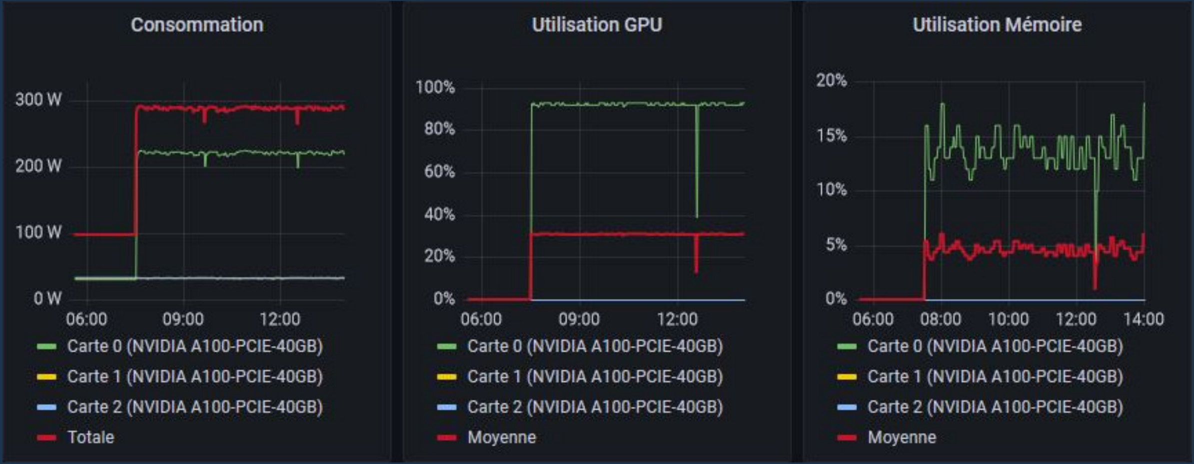


Gray Scott `nvc++` : $\frac{5}{I/O} \times 680\,000 = 3\,400\,000$ images

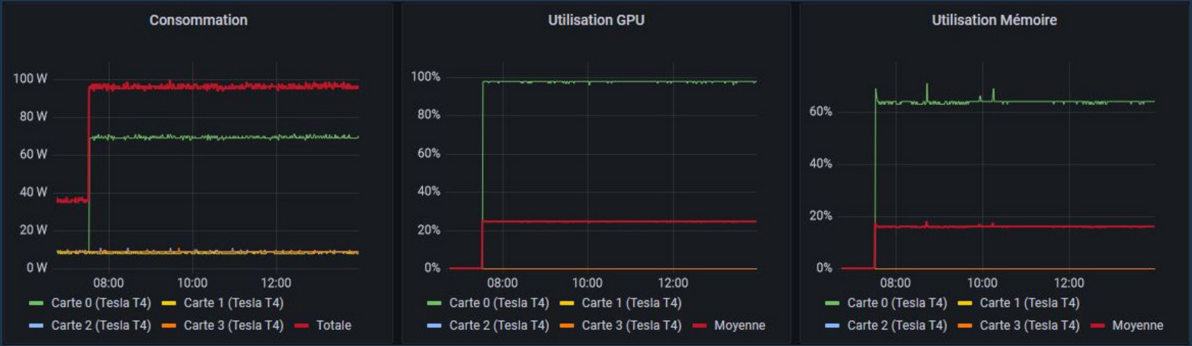
Result with 100 tests per **GPU**



Grafana A100 for Gray Scott `nvc++`



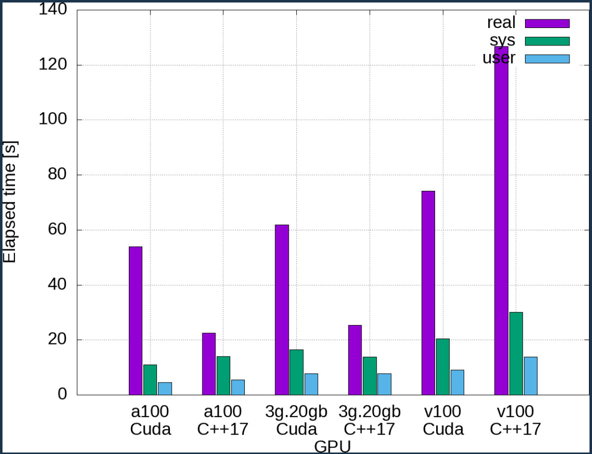
Grafana T4 for Gray Scott `nvc++`

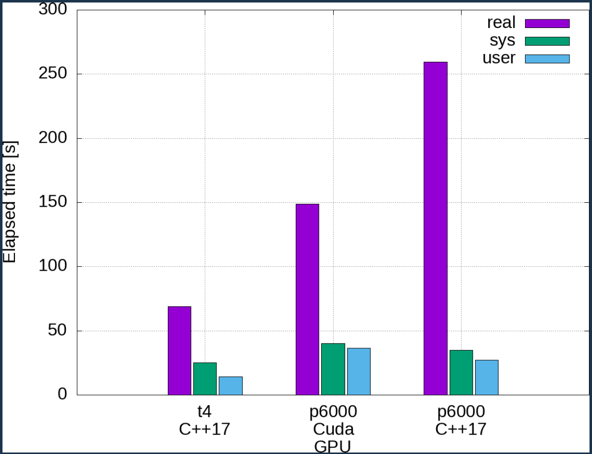


Gray Scott `nvc++` : $1\,000 \times 34 = 34\,000$ images

I/O

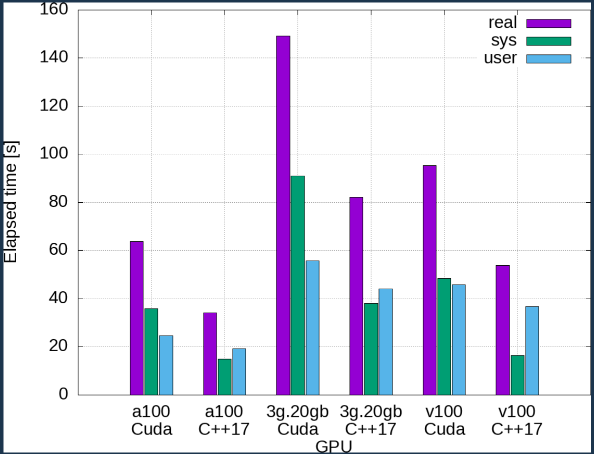
Result with 100 tests per **GPU**

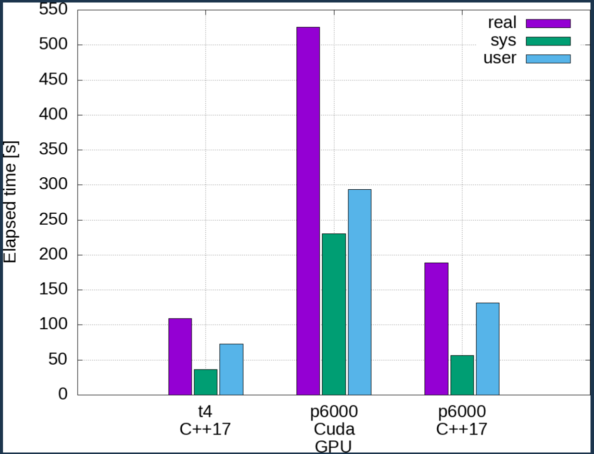




Gray Scott nvc++ : 5 I/O × 68 000 = 340 000 images

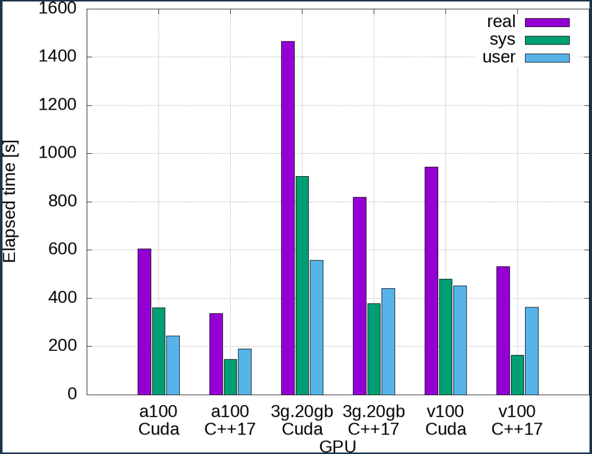
Result with 100 tests per **GPU**

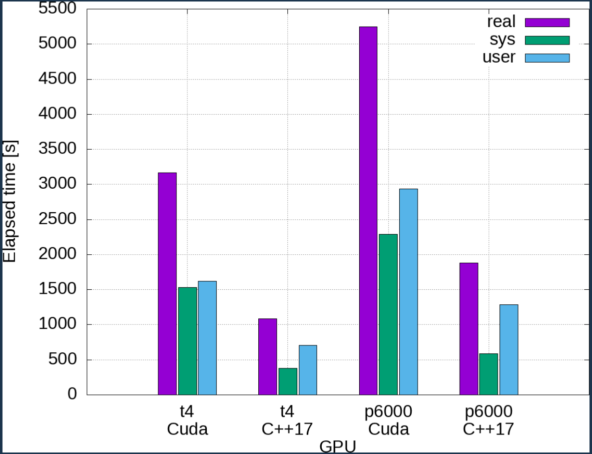




Gray Scott `nvc++` : $5_{I/O} \times 680\,000 = 3\,400\,000$ images

Result with 100 tests per **GPU**





Conclusion

- Good performances on **GPUs**
  - With **nvcc** (**CUDA**)
  - With **nvc++** (**C++17** / **C++20**)
- **HPC SDK** installed on **MUST** (version **21.09**/**22.09**)
- Compiler **nvc++** powerful and easy to use with **C++17**
  - **No explicit linking**
  - Automatic **GPU** Targeting or with **CUDA_VISIBLE_DEVICES**
  - **Avoid static allocation**
- Warning about industrial software
  - Will to drive for update
  - Old **GPUs** become **obsolete** :
    - **nvc++** : compute capabilities ≥ 6 (no **K80**)
    - **nvcc** : compute capabilities ≥ 3.5 (no **K80** in **CUDA 12**)
  - Need to save binaries to ensure long usability of **GPUs**